Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

PRISCILA RODRIGUES DA SILVA

IMPLEMENTAÇÃO DE SOLUÇÃO ESTRATÉGICA PARA STUDENT RELATIONSHIP MANAGEMENT (SRM) BASEADA EM JAVA EE E SOCIAL MEDIA

Assis
2013

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

PRISCILA RODRIGUES DA SILVA

IMPLEMENTAÇÃO DE SOLUÇÃO ESTRATÉGICA PARA STUDENT RELATIONSHIP MANAGEMENT (SRM) BASEADA EM JAVA EE E SOCIAL MEDIA

Projeto de pesquisa apresentado ao Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientanda: Priscila Rodrigues da Silva.

Orientador: Prof. Esp. Guilherme de Cleva Farto.

Assis
2013

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

PRISCILA RODRIGUES DA SILVA

IMPLEMENTAÇÃO DE SOLUÇÃO ESTRATÉGICA PARA STUDENT RELATIONSHIP MANAGEMENT (SRM) BASEADA EM JAVA EE E SOCIAL MEDIA

Orientador: Prof. Esp. Guilherme de Cleva Farto

Nota do orientador:	Nota do avaliador:

Assis
2013

AGRADECIMENTOS

Primeiramente gostaria de agradecer a minha mãe, Enecir Rodrigues da Silva, pelo constante esforço para me manter durante minha formação, muitas vezes abdicando suas vontades e prioridades.

Aos professores Luiz Carlos Begosso, pela oportunidade de ser sua orientanda durante a iniciação científica; Luiz Ricardo Begosso, por me conceder a oportunidade de expandir meus conhecimentos além da sala de aula; Almir Camolesi, Domingos Villela e Alex Poletto por sempre estarem dispostos a incrementar meus conhecimentos técnicos, além de todos os outros professores, que de certa forma, foram fonte de inspiração para mim.

Ao meu orientador Guilherme de Cleva Farto pela constante disposição, encorajamento e enorme paciência.

E por fim, aqueles que direta ou indiretamente também de certa forma colaboraram com minha formação.

FICHA CATALOGRÁFICA

SILVA, Priscila Rodrigues da
Implementação de solução estratégia para Student Relationship Management (SRM) baseada em Java EE e Social Media / Priscila Rodrigues da Silva. Fundação Educacional do Município de Assis, 2013.
93 p.

Orientador: Prof. Esp. Guilherme de Cleva Farto
Trabalho de Conclusão de Curso
Instituto Municipal de Educação Superior de Assis – IMESA

1. Sistema SRM 2. Java 3. Social Media

CDD: 001.61
Biblioteca da FEMA

“You can't jump for the stars if your feet hurt”.

– Dan Brown, Digital Fortress

RESUMO

Atualmente, com a alta competitividade das instituições de ensino por novos alunos, é necessário descobrir e elaborar novas e mais adequadas técnicas a fim de atrair, manter e melhorar o relacionamento com possíveis novos ingressantes. O objetivo principal deste trabalho é desenvolver de um sistema de informação composto por módulos para a retenção de informações oriundas de potenciais futuros estudantes. O sistema também se utiliza por meio de estratégias de *Student Relationship Management* (SRM), como a utilização de mídias sociais tais como Facebook e Twitter, para que um melhor aprimoramento da comunicação seja alcançado. O sistema foi desenvolvido por intermédio da plataforma Java EE, juntamente às especificações *Java Server Faces* (JSF) com a API de componentes *PrimeFaces*; *Java Persistence* API (JPA) e, por fim, as APIs de desenvolvimento para redes sociais, Facebook4J e Twitter4J.

Palavras-chave: Student Relationship Management, SRM, Social Media, Java.

ABSTRACT

Nowadays, with the high competitiveness of the educational institutions for new students, it is necessary to discover and develop new and more appropriate techniques to attract, maintain and improve relationships with potential new students. The main objective of this work is to develop an information system composed of modules for withholding information from potential future students. The system is also used by Student Relationship Management strategies, such as the use of social media as Facebook and Twitter, to better improvement their communication is achieved. The system was developed through the Java EE platform, along with the specifications Java Server Faces (JSF) with API components *PrimeFaces*; Java Persistence API (JPA) and, finally, the APIs development for social networking, Facebook4J and Twitter4J.

Keywords: Student Relationship Management, SRM, Social Media, Java.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Atualmente, o crescimento exponencial de instituições de ensino privadas no Brasil tem feito com que o ponto de vista em relação ao aluno seja alterado, pois, consequentemente, a concorrência é maior e, portanto, torna-se necessária a adoção de conceitos e metodologias, principalmente as de *marketing* já consagradas, utilizadas em diversos outros segmentos de mercado. (L3 CRM, 2012).

Existem algumas formas de análise que visam estudar potenciais alunos com o intuito de trazê-los para a instituição e, entre as possíveis estratégias, pode-se utilizar o *Student Relationship Management* (SRM), uma vertente do *Customer Relationship Management* (CRM), que faz uso de conceitos que buscam minimizar a distância entre a instituição de ensino e o aluno, assim como melhorar e manter saudável a relação existente essas entidades (L3 CRM, 2012).

Visando alcançar e implementar os principais conceitos de *SRM*, será desenvolvida uma aplicação baseada na tecnologia Java *Enterprise Edition* (JEE) com o objetivo de centralizar dados referentes a alunos, ex-alunos, professores e participantes de eventos, bem como melhorar os mecanismos disponíveis para comunicação entre instituição de ensino, estudantes e possíveis estudantes assim como a qualidade de processos de ouvidoria com as pessoas relacionadas ao instituto.

A aplicação a ser desenvolvida fará uso dos conceitos base para uma aplicação *SRM* e será implementada utilizando-se tecnologias da plataforma *JEE* como *Java Persistence API* (JPA) e *JavaServer Faces* (JSF). Com o objetivo de tornar a aplicação mais dinâmica e permitir uma comunicação mais rápida entre instituição de ensino e pessoas, serão implementados componentes baseados em *Social Media* de forma a possibilitar a integração do *software* desenvolvido com redes sociais, tais como *Facebook* e *Twitter*.

Segundo Breakenridge (2010), “a ideia de *Social Media* significa ouvir com atenção e aprender a compartilhar informações valiosas que unem as pessoas e constroem relacionamentos fortes”.

A partir desta concepção, espera-se facilitar a comunicação e atendimento de solicitações entre a instituição de ensino e as pessoas que a compõe como alunos,

ex-alunos, professores e funcionários além de buscar novas oportunidades em possíveis estudantes.

1.1 OBJETIVOS

O presente trabalho tem como principal objetivo o desenvolvimento de uma aplicação *Web* baseada nos conceitos de *SRM* e tecnologias *Java EE* e *Social Media* para o auxílio de instituições de ensino nas tarefas de gerenciamento de dados acadêmicos e promoção da relação com estudantes e demais pessoas relacionadas.

Os seguintes objetivos específicos podem ser destacados:

- Estudo dos conceitos de *SRM*
- Estudo das tecnologias *Java EE, JPA, JSF, PrimeFaces*.
- Estudo de tecnologias para integração com *Social Media* como *Facebook* e *Twitter*, com o intuito de facilitar a comunicação entre usuários do sistema acadêmico e alunos, ex-alunos, professores e funcionários.

1.2 JUSTIFICATIVAS

Com o crescimento da educação superior privada, a alta competitividade por futuros estudantes está cada vez maior. Por esse e outros motivos é preciso criar novas e melhores estratégias de comunicação.

Com a finalidade de atrair novos alunos, serão utilizados conceitos de *SRM*, pois, por meio deles, é possível definir e usufruir de formas de planejamento já consolidadas para que um melhor serviço por parte da instituição seja prestado.

A partir desses conceitos, uma aplicação acadêmica *Web* será desenvolvida com o objetivo de prestar suporte aos diversos departamentos de uma instituição de ensino e de prover métodos para que o trabalho dos mesmos seja facilitado, já que o *software* fornecerá diversas informações acerca de entidades acadêmicas, como alunos e ex-alunos, assim como a possibilidade de utilizar redes sociais para buscar

e manipular informações, postagens e comentários, contribuindo com uma maior agilidade nos processos de comunicação e ouvidoria.

Optou-se também por buscar meios de integração com redes sociais por conta de sua influência no cenário cotidiano. Atualmente, as pessoas permanecem conectadas a redes sociais por meio de *smartphones*, *notebooks* ou *tablets*, recebendo, durante o dia todo, recomendações de lugares a se visitar, notícias e se comunicando com outros usuários (SATUDI, 2013).

O destaque das redes sociais é tão expressivo que até mesmo empresas possuem contas como forma de participar ativamente para divulgar e promover produtos e serviços. As organizações também consultam perfis de usuário a fim de saber suas preferências, posições e personalidades (SATUDI, 2013).

Por fim, o estudo e desenvolvimento de uma aplicação *Web*, amparados pelos conceitos de *SRM* e baseada em *Java EE*, *Social Media* e outros tópicos que serão abordados neste trabalho, também irá contribuir de forma direta para a aquisição de novos conhecimentos e experiências nas etapas de análise, modelagem, arquitetura e implementação de *softwares* com tecnologias e conceitos modernos e emergentes, possibilitando uma excelente forma de evoluir nas áreas acadêmica e profissional.

1.3 ESTRUTURA DO TRABALHO

O presente trabalho está estruturado em sete capítulos, sendo o primeiro esta introdução.

O segundo capítulo faz uma exposição dos conceitos de *CRM* e *SRM*, que serão implementados no projeto.

O capítulo três apresenta as tecnologias que serão utilizadas para a criação do projeto; *frameworks*, linguagem de programação e o banco de dados estão contidos neste capítulo.

No quarto capítulo foram pontuados os requisitos do sistema que o sistema conterá e a modelagem do sistema, incluindo-se o mapa mental, os casos de uso, diagrama

de classe, principais diagramas de atividades e de sequência, além do diagrama entidade relacionamento.

O quinto capítulo trata sobre a estrutura do projeto em termos de atividades que foram realizadas, orçamento do projeto e sua estrutura utilizando o diagrama *Work Breakdown*.

No sexto capítulo está a implementação do projeto. Dentro deste capítulo serão exibidas as principais classes do projeto, bem como sua estrutura organizacional. Dentre as principais classes estão as classes com anotações *JPA* e as classes de integrações com as redes sociais, além disto as interfaces da aplicação Web serão expostas.

E finalmente, no capítulo sete são inclusas a conclusão obtida ao final do trabalho, bem como trabalhos que poderão ser desenvolvidos futuramente.

2. STUDENT RELATIONSHIP MANAGEMENT (SRM)

Conforme RAPP et al. (1999 apud ALCÂNTARA, 2003, p.118-124), nos anos 90 o consumidor passou a ser mais consciente dos seus direitos, e após um ano, em 1991 o Código de Defesa do Consumidor entrou em vigor, as empresas então criaram o SAC (Serviço de Atendimento ao Consumidor) para ouvi-los. Com o tempo o serviço de SAC foi ampliado, e hoje é possível encontrar informações sobre os produtos, garantias, pontos de vendas, vendedores, franqueados, entre outros.

Logo a concorrência por maior quantidade de consumidores entre as empresas aumentou, e foi preciso investir em estratégias criativas para aproximá-los. Uma destas estratégias é o *Marketing one-to-one*, mencionada por Don Peppers e Martha Rogers como uma abordagem de *CRM*. Esta forma estratégica possibilita personalizada relação com o cliente como forma de estímulo para que haja uma relação leal e como resultado alcançar um melhor retorno deste investimento (ROUSE, 2007).

O *SRM* é portanto uma aplicação estratégica de negócio de *CRM*, e está associado a conceitos de *Business Intelligence* e tecnologias utilizadas para obter conhecimento sobre os estudantes e apoiar processos de decisão (PIEDADE, 2010).

2.1 INTRODUÇÃO À CUSTOMER RELATIONSHIP MANAGEMENT (CRM)

Juntamente com a globalização, a concorrência entre empresas vem aumentando, deste modo é cada vez mais necessário que as corporações se diferenciem umas das outras, e que possuam consigo a capacidade de propiciar ao consumidor uma melhor e sofisticada experiência em todas as áreas do negócio.

Segundo Kostojohn et al. (2011), a fim de atender e solucionar as necessidades do cliente, a empresa deve executar suas tarefas de modo correto, mais rápido e eficiente que seus concorrentes. Para que isto possa ser obtido, as empresas precisam de informações sobre seu público alvo e futuros clientes, para que assim, seu produto ou serviço estejam dentro das expectativas previstas pelos clientes.

Visando atender o público alvo de modo eficaz, surgiram ferramentas que proveem à empresa perspectiva do ponto de vista do cliente, gestão de informações dos mesmos e ajuda na tomada de decisão, a este tipo de software dá-se o nome de *CRM*, do inglês *Customer Relationship Management* (KOSTOJOHN et al., 2011, p1).

Vale ainda destacar que na visão de PEPPERS et al. (2004, p. 98), *CRM* é "estabelecer relacionamento com os clientes de forma individual e depois usar as informações coletadas para tratar clientes diferentes de maneira diferente".

Segundo GREENBERG (2001), existem três distintos segmentos de tecnologia *CRM*, caracterizadas como operacional, analítico e colaborativo:

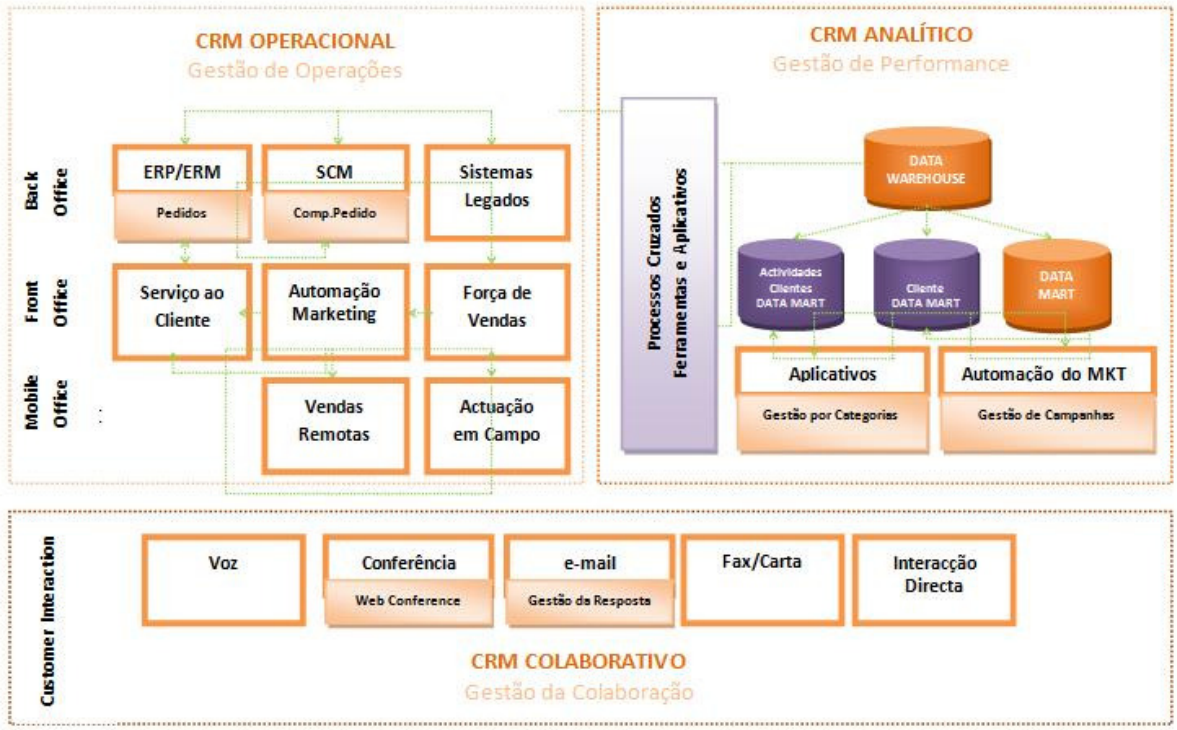


Figura 1 – Tipos de CRM. Fonte: (SCM & CRM, 2013)

CRM operacional engloba todos os serviços voltados diretamente para o cliente, neste segmento se incluem "pacotes para linha de frente de atendimento", ou seja, pessoas que lidarão de forma direta com o público; neste trabalho, será empregado de forma prática este conceito. Há também neste segmento outras funcionalidades como gerenciamento de faturamento e contabilidade, e por fim, automação de vendas e *marketing* (*help desk, call centers* e outros).

CRM analítico é o segmento focalizado na armazenagem dos dados captados. Deve-se primeiramente captar informações de variadas e de distintas fontes, e então armazená-las em um Data Warehouse. Segundo Kroenke et al. (1998, p.280):

> *Data Warehouse* é um depósito de dados corporativo que se destina a facilitar a tomada de decisões na empresa. Um *Data Warehousing* não inclui apenas dados, mas também ferramentas, procedimentos, treinamentos, pessoal e outros recursos que facilitam o acesso aos dados e os tornam mais adequados aos responsáveis por tomada de decisões.

Após isso, é preciso que algoritmos sejam executados para fazer a análise destes dados, que uma vez trabalhados, servirão como apontadores no qual se pode então criar estratégias de aproximação com o público-alvo.

CRM colaborativo diz respeito ao centro de comunicação com os clientes. Pode dar-se em forma de portal, centro de interação com os clientes (CIC), fax, cartas, ou até mesmo por meio de canais de comunicação por *Web, e-mail*, dentre outros canais de comunicação.

2.2 DEFINIÇÃO CONCEITUAL DE STUDENT RELATIONSHIP MANAGEMENT (SRM)

O *SRM* tem por objetivo fazer com que o aluno permaneça na instituição de ensino, tendo como meio o bom relacionamento com o aluno. Para que tais objetivos possam ser alcançados, algumas estratégias devem ser executadas, destacando-se como principais (PIEDADE, 2011):

- Centralizar informações em bases de dados acadêmicas;
- Melhorar a comunicação entre instituição de ensino e aluno com a finalidade de contribuir para a boa impressão do centro de estudos para futuros ingressantes;
- Estratégia de negócio centralizada no aluno;
- Melhorar a formação da imagem de qualidade e de excelência de processos;

- Validar informações.

A Figura 2 exemplifica em forma de pirâmide as camadas que se deve suportar a estratégia de *CRM*.

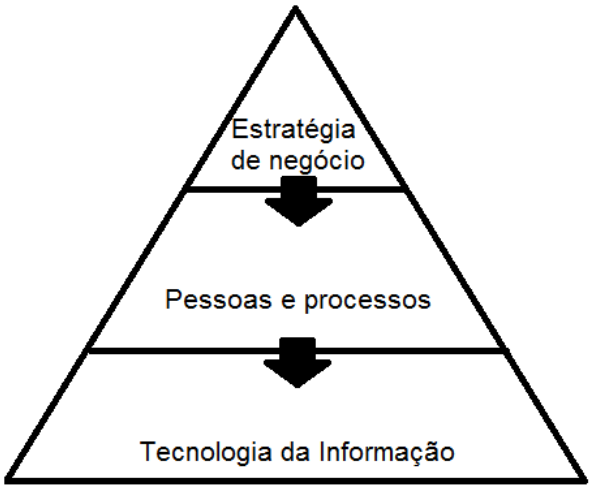


Figura 2 – Pirâmide CRM. Fonte: (GREENBERG, 2001)

Conforme apresentado na pirâmide, primeiramente é necessário se definir uma estratégia de negócio para chegar ao aluno, algumas destas estratégias já foram anteriormente citadas.

Logo após, se tem os processos e pessoas, nesta área é importante se definir e obter uma ideia clara de como serão executadas os métodos e os processos, a fim de alcançar clientes/alunos e mantê-los próximos à área na qual é centrada a empresa.

E por fim, como destaca GREENBERG (2001), deve-se levar em conta que o *CRM* e *SRM* não são simplesmente tecnologia ou aplicativos, mas principalmente uma forma de *marketing*, que possuem como objetivo melhorar a comunicação com seu público alvo, os sistemas são por sua vez o meio para alcançar tais objetivos.

3. CONCEITOS, TECNOLOGIAS E FERRAMENTAS DE DESENVOLVIMENTO

Neste capítulo serão apresentadas as tecnologias que serão utilizadas para a implementação do *software de SRM.*

O sistema *SRM* tem de ser capaz de obter dados oriundos de redes sociais, e posteriormente exibir estes dados para o usuário. Para que isto seja possível, o software se utiliza de *Java Archieves* – jars –, entre eles *Facebook4J* e *Twitter4J*, disponibilizados de forma gratuita, que por sua vez contém métodos no qual se faz possível este tipo de operação.

A tecnologia empregada é a *Java* em sua edição para *Web* – *JEE* –, esta possui vários *frameworks*, entre eles alguns que são utilizados no desenvolvimento deste sistema, tais como *JPA* e *JSF*, que ajudam ao desenvolvedor minimizando a complexidade do código.

Como ambiente de desenvolvimento, a *Integrated Development Environment* (IDE) Eclipse fora escolhida pois é um ambiente que está sempre em evolução, e também por ser totalmente personalizável, uma vez que de forma simples pode-se realizar alterações em *APIs* do projeto.

Por fim, é utilizado o servidor de aplicação *Tomcat*, por ser um servidor gratuito e robusto, servindo como contêiner para o banco de dados *PostgresSQL* no qual faz o uso de métodos providos pelo *framework JPA*, que é utilizado para realizar o mapeamento objeto/relacional, possibilitando a não codificação em linguagem *Structured Query Language* (SQL).

3.1 TECNOLOGIA JAVA

Java é uma tecnologia aberta e gratuita que utiliza o paradigma de orientação a objeto. É utilizada amplamente em sistemas de grande porte por ser uma tecnologia robusta, possuir extensa gama de *frameworks* e documentação, portabilidade, segurança, ser multi plataforma, possuir comunidade com milhares de usuários

ativos, e é por estes e outros motivos *Java* é atualmente uma das linguagens de programação mais utilizadas no mundo (SANTANA).

Segundo (JAVAFREE, 2012) a história do *Java* se iniciou em 1991, a partir de um projeto nomeado *Green Project*. Em 1992 a linguagem Oak foi criada contendo a primeira máquina virtual, porém sem sucesso de vendas.

Com o surgimento da *internet* em 1994, a *Sun* enxerga um novo uso para a Oak, a construção de uma nova linguagem para a Web baseados nela, a *Java*. Um ano após a linguagem *Java* é lançada (JAVAFREE, 2012).

A tecnologia *Java* oferece três edições de desenvolvimento são elas *JSE, JEE* e *JME* (DEVJR).

- *JSE – Java Standart Edition* é a plataforma para desenvolvimento *Java* para computadores pessoais, notebooks e arquiteturas com maior poder de processamento e memória. A maioria das aplicações é construída e executada com *JSE*. O *JSE* se divide em duas partes:
    - *Java Development Kit (JDK)* ou *Standart Development Kit (SDK)*: kits para desenvolvimento em *Java*.
    - *Java Runtime Edition (JRE)*: ambiente de execução *Java*, é a versão que executará sistemas construídos com *JDK* ou *SDK*.

A Figura 3 apresenta a visão geral da plataforma *JSE*:

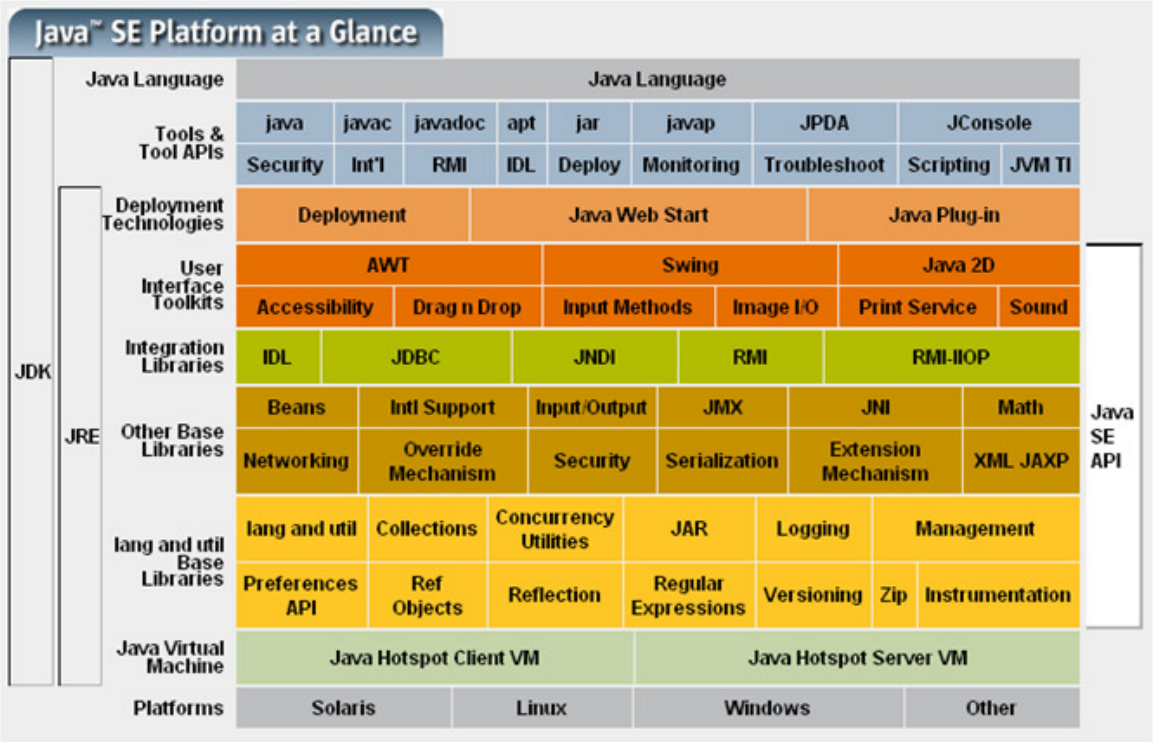


Figura 3 – Visão Geral Edição JSE. Fonte: (ORACLE, 2013)

- *JEE – Java Enterprise Edition* é a plataforma *Java* utilizada para aplicações corporativas que podem ou não estar na *internet* (DEVJR). Segundo (CAELUM), o *JEE* "consiste em uma série de especificações bem detalhadas, dando uma receita de como deve ser implementado um *software* que faz cada um desses serviços de infraestrutura".
- *JME – Java Micro Edition* é a edição reduzida da *JSE*, voltada para aparelhos móveis como celulares, *pagers* e outros dispositivos com baixa memória e processamento (DEVMEDIA).

A Figura 4 apresenta a visão geral da plataforma *JME*:

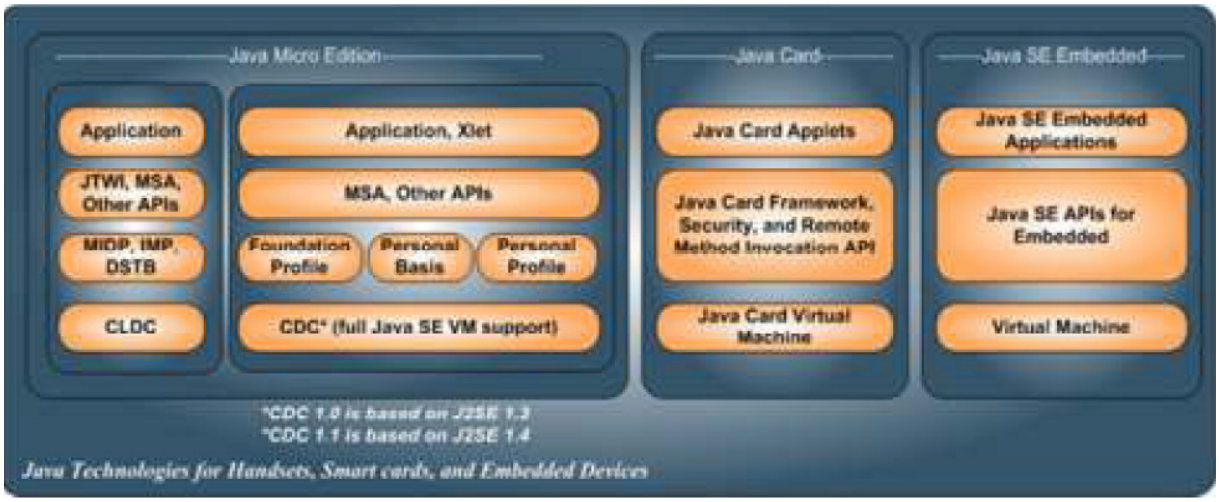


Figura 4 – Visão Geral Edição JME. Fonte: (SUN, 2011 apud SILVA, 2011)

3.2 PLATAFORMA JAVA ENTERPRISE EDITION (JEE)

Java EE é a plataforma *Java* voltada principalmente para o desenvolvimento *Web*, esta edição será utilizada para o desenvolvimento do sistema. Dentro desta modalidade de desenvolvimento existem várias *APIs* que facilitam a criação de sistemas algumas delas são *JSP, JSF e PrimeFaces* (CAELUM).

A Figura 5 apresenta a visão geral da plataforma JEE:

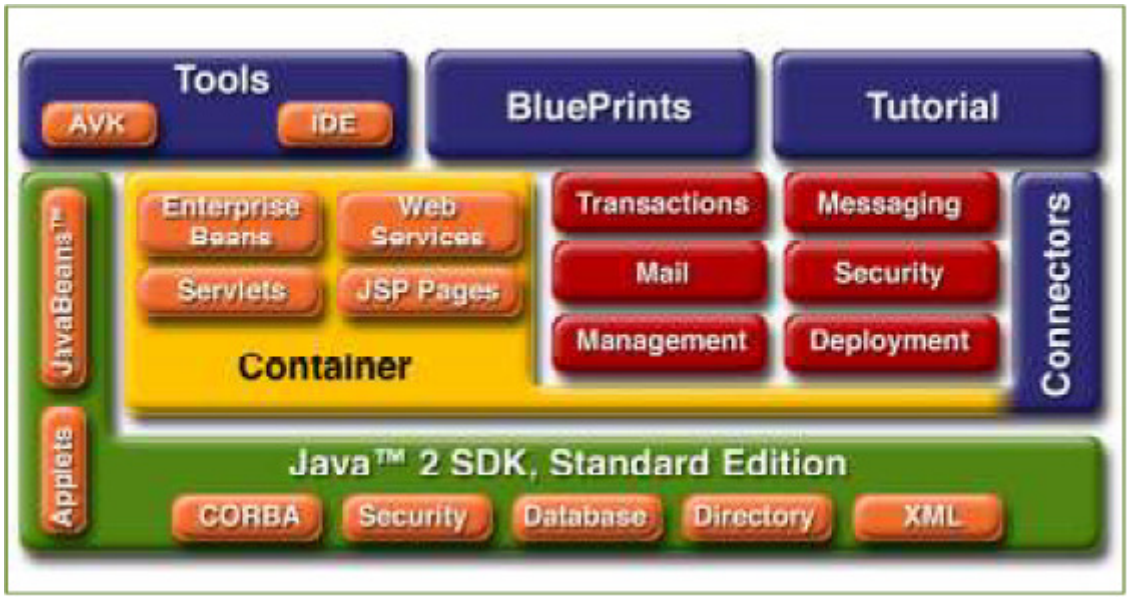


Figura 5 – Visão Geral Edição JEE. Fonte: (SUN, 2011 apud SILVA, 2011)

3.3 JAVASERVER FACES (JSF)

JavaServer Faces (JSF) é o *framework* oficial de aplicações *Java* para *Web*. Desenvolvido pela *Sun Microsystems*, o JSF possibilita a criação de interfaces amigáveis com o usuário (GUI) por meio dos conceitos de componentes (Gonçalves, 2008, p.11).

3.4 FRAMEWORK PRIMEFACES

Segundo ÇIVICI (2012), o *PrimeFaces*, é o "conjunto de componentes *open-source* que tem como objetivo propiciar uma interface mais amigável ao usuário".

O *PrimeFaces* portanto é um *framework* de componentes que pode ser empregado pelo *JSF*. Com o passar do tempo tornou-se bastante utilizado uma vez que contém extensa gama de componentes, seus componentes são estilizados e de fácil manipulação.

3.5 JAVA PERSISTENCE API (JPA)

Cada banco de dados possui sua própria sintaxe, o que dificulta a escrita do código. O *JPA* é um *framework* de persistência de dados que provê portabilidade de código uma vez que o utilizando não é preciso escrever em linguagem *SQL*. Seu mapeamento pode ser feito via *XML* ou *Annotation*.

3.6 BANCO DE DADOS POSTGRESQL

PostgreSQL é um Banco de Dados relacional e orientado a objetos, *open-source* de licença BSD. Pode ser executado em vários sistemas operacionais e possui extensa gama de compatibilidade com linguagens de programação, tais como: C/C++, *Java, Ruby, PHP, Python*, Lua, entre outras (MOMJIAN, 2000).

Além disto, o *PostgreSQL* é robusto, isto é, suportando grande volume de dados e de transações, seu custo é baixo, possui funcionalidades dos bancos de dados mais comundos, é seguro, há extensa documentação e grande suporte da comunidade e de seus mantedores (MOMJIAN, 2000).

É válido notar que grandes empresas, como: *Apple*, Cisco, Vivo e outras utilizam *PostgreSQL*.

Este projeto será feito para a plataforma *web*, portanto é necessário notar que haverá grande volume de dados sendo movimentados a todo o momento, sendo assim, o *PostgreSQL* atende as necessidades previstas devido suas características citadas anteriormente.

3.6.1 Um pouco da história do PostgreSQL

Inicialmente em 1977 a 1985 foi criado um banco chamado Ingres, agora pertencente à *Computer Associates*, no qual foi desenvolvido na Universidade da Califórnia em *Berkeley*.

Michael Stonebraker professor de ciência da computação, liderou um grupo de estudantes a desenvolver um banco de dados objeto-relacional chamado *Postgres* que foi um projeto de acompanhamento ao seu antecessor *Ingres* que perdurou de 1986 a 1994, também em *Berkeley*.

Uma empresa chamada Illustra assumiu o código do *Postgres* e então foi comercializado.

Nos anos de 1994 e 1995 Jolly Chen e Andrew Yu, estudantes em Berkeley, embutiram recursos de SQL ao *Postgres*, o produto gerado por este projeto foi chamado *Postgres95*.

Após deixarem *Berkeley*, Chen continuou mantendo o *Postgres95*, e em 1996 houve uma grande demanda por um banco de dados *SQL open source*, uma equipe então foi formada para continuar o desenvolvimento, neste ano alteraram o nome do banco de dados para *PostgreSQL* e distribuíram o código fonte para a comunidade (MOMJIAN, p.1-4).

4. ANÁLISE E ESPECIFICAÇÃO DO SISTEMA

Segundo SILVA et al. (2001), a análise do sistema consiste em levantar detalhadamente os problemas a serem solucionados pelo sistema e enumerá-los. A especificação é um documento que se utiliza da modelação dos comportamentos do sistema, no qual expressa as funções que serão embutidas no sistema e, por este motivo é o responsável por extinguir a imprecisão quanto às funções que o sistema deve ser capaz de realizar.

4.1 MAPA MENTAL

Com o propósito de melhorar o entendimento e a visualização acerca do sistema, fora construído um mapa mental. Desenvolvido pelo inglês Tony Buzan, mapas mentais são diagramas utilizados para conectar informações a uma ideia central.

Os mapas mentais têm como principal objetivo manter coisas organizadas e destacam palavras-chave de uma ideia principal que, a partir de seu refinamento, acrescentam novos tópicos em forma de ramificações (KOLIFRATH, 2013).

A Figura 6 ilustra o mapa mental referente às funcionalidades e responsabilidades da aplicação a ser desenvolvida neste trabalho de conclusão de curso:

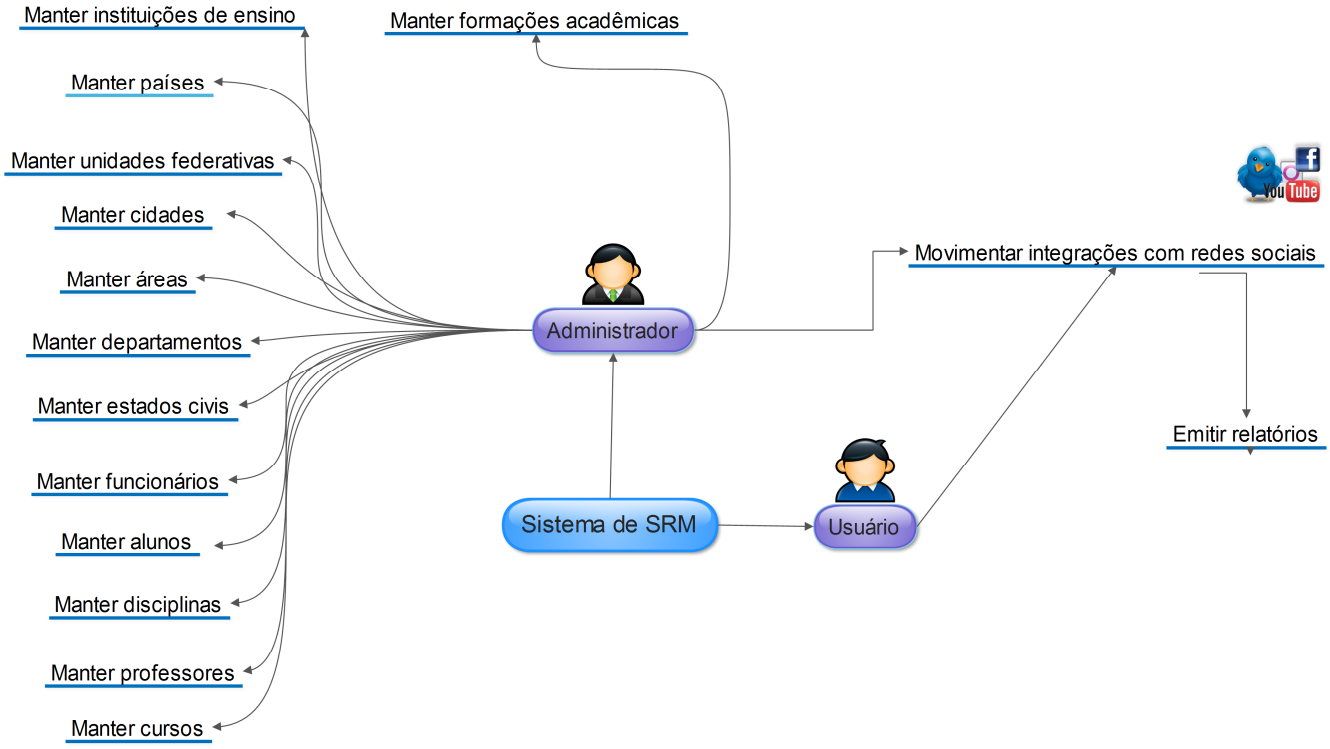


Figura 6 – Mapa Mental do Sistema SRM

4.2 LISTA DE REQUISITOS

- Manter instituições de ensino;
- Manter países;
- Manter unidades federativas;
- Manter municípios;
- Manter áreas;
- Manter departamentos;
- Manter estados civis;
- Manter funcionários (usuários da aplicação);
- Manter formações acadêmicas (de alunos e professores);
- Manter disciplinas;

- Manter turmas;
- Manter professores;
- Manter cursos;
- Manter alunos/vestibulandos;
- Movimentar integrações com redes sociais (*Twitter*, *Facebook* e outras);
- Emitir relatórios (postagem).

4.3 LINGUAGEM UML

A linguagem *Unified Modeling Language* (UML) surgiu em meados do ano de 1990 e é produto de três métodos de modelagem distintos, são eles: Booch, *Object Modeling Technique* (OMT) e *Object-Oriented Software Engineering* (OOSE). A UML é uma linguagem de modelagem de *software* voltada para o paradigma de orientação a objetos (GUEDES, 2009, p.19-20).

A *UML* contém vários diagramas, os diagramas são importantes para que o sistema seja visualizado em várias camadas distintas, podendo ter sua complexidade aumentada ou diminuída. Sendo assim, falhas e visualização de possíveis erros futuros poderão ser descobertas com maior facilidade (GUEDES, 2009, p.30).

Os principais diagramas da *UML* que serão documentados serão os diagramas de caso de uso, classe, atividade e sequência.

4.4 DIAGRAMA E ESPECIFICAÇÃO DE CASOS DE USO

O diagrama de Caso de Uso é baseado em requisitos solicitados pelo usuário, dentre os diagramas da *UML* este diagrama é o mais genérico e o mais simples de ser interpretado por pessoas leigas em desenvolvimento de *software*, a partir deste diagrama torna-se possível entender como o sistema funciona como um todo.

O diagrama de Caso de Uso utiliza-se basicamente de atores e casos de uso para ser construído.

4.4.1 Atores

Atores são papéis desempenhados normalmente pelo usuário do *software* e eventualmente por *hardware* e/ou *software*. Ator é tudo aquilo que troca informações com o sistema, interagindo portanto de forma direta com o ele. Atores são representados por um homem palito (GUEDES, 2009, p.56).

4.4.2 Casos de uso

Casos de uso são as funcionalidades do sistema, para que sejam identificados mais facilmente, primeiramente devemos focalizar os atores que iniciam ações do sistema, e então identificar os processos em que o ator participa que tem como fim um objetivo de negócio, o caso de uso é representado por um eclipse (LIMA, 2012, p.110).

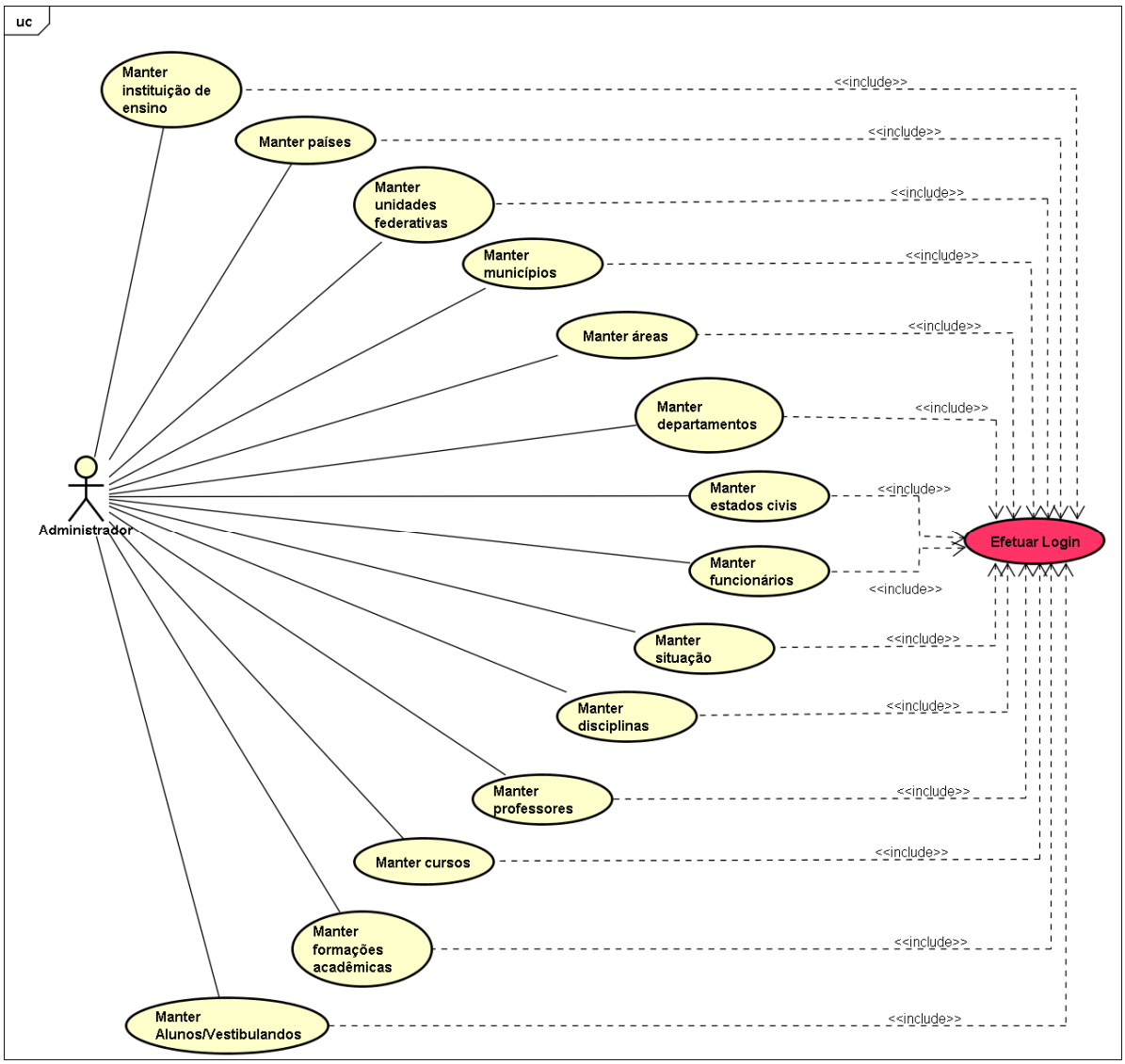


Figura 7 – Caso de uso do administrador

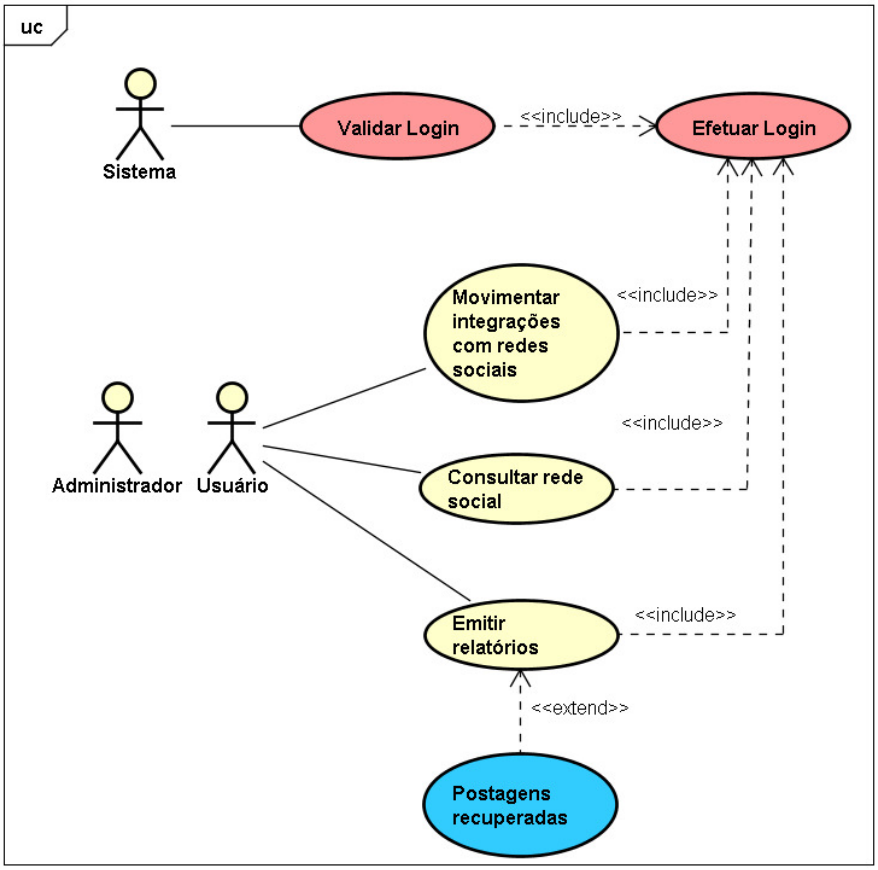


Figura 8 – Caso de uso geral comum

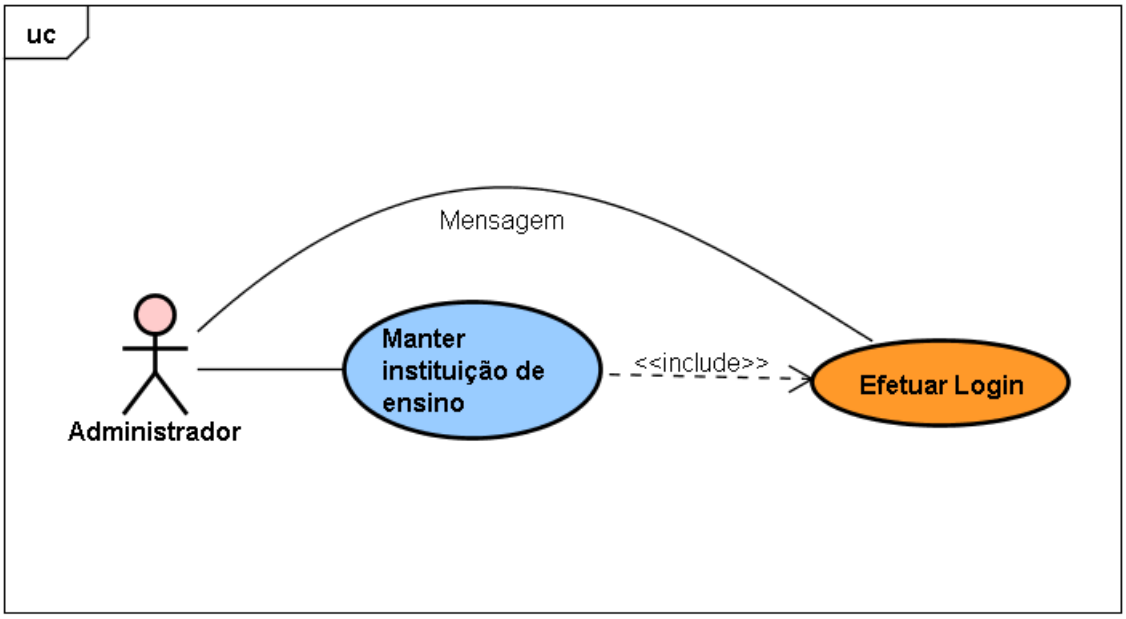


Figura 9 – UC 1 Diagrama de caso de uso Manter instituição de ensino

Nome do caso de uso 1	Manter instituição de ensino
Atores	Administrador
Pré-condição	Efetuar Login
Cenário principal	1. O administrador informa os dados da instituição de ensino. 2. O administrador informa o CNPJ da instituição de ensino. 3. O sistema valida os dados informados. 4. O administrador seleciona a opção "cadastrar". 5. O sistema emite mensagem de sucesso da gravação. 6. O sistema cadastra a instituição de ensino.
Cenário alternativo	Caso o sistema não valide os dados, o sistema enviará mensagem de alerta. O administrador poderá cancelar o cadastro durante a operação de cadastro.
Casos de teste	1.1 O sistema verifica se os dados foram inseridos corretamente. 1.2 O sistema emite mensagem de alerta informando que os dados não foram inseridos corretamente. 1.3 O sistema cancela a operação. 2.1 O sistema verifica se o CNPJ da instituição existe.

	2.2 O sistema emite mensagem de alerta informando que o CNPJ da instituição é inexistente. 2.3 O sistema não permite o cadastro da instituição.

Tabela 1 – Manter instituição de ensino

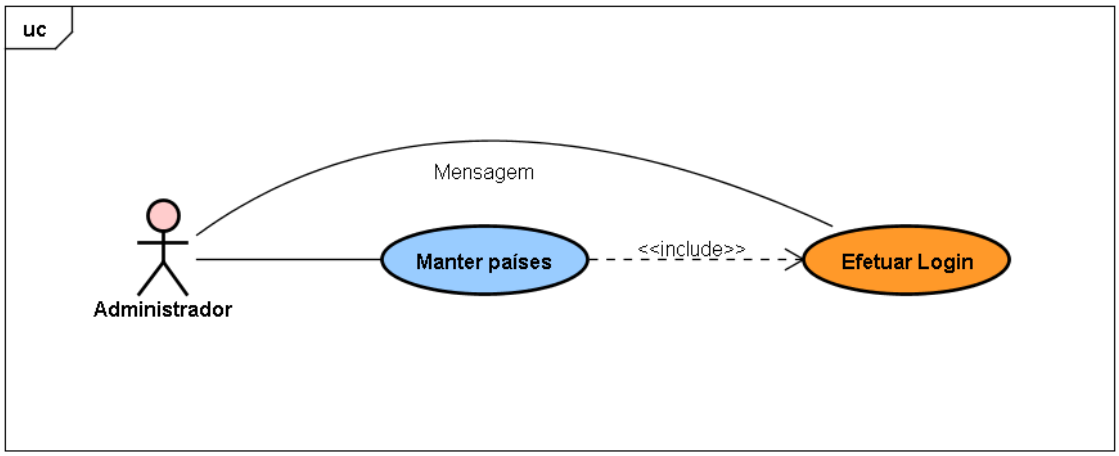


Figura 10 – UC 2 Diagrama de caso de uso Manter países

Nome do caso de uso 2	Manter países
Atores	Administrador
Pré-condição	Efetuar Login
Cenário principal	1. O administrador informa os dados do país. 2. O administrador informa o nome do país. 3. O sistema valida os dados informados. 4. O administrador seleciona a opção “cadastrar”. 5. O sistema emite mensagem de sucesso de gravação. 6. O sistema cadastra o país
Cenário alternativo	Caso o sistema não valide os dados, o sistema enviará mensagem de alerta. O administrador poderá cancelar o cadastro durante a operação de cadastro.
Casos de teste	1.1 O sistema verifica se os dados foram inseridos corretamente. 1.2 O sistema emite mensagem de alerta informando que os dados não foram inseridos corretamente. 1.3 O sistema cancela a operação.

Tabela 2 – Manter países

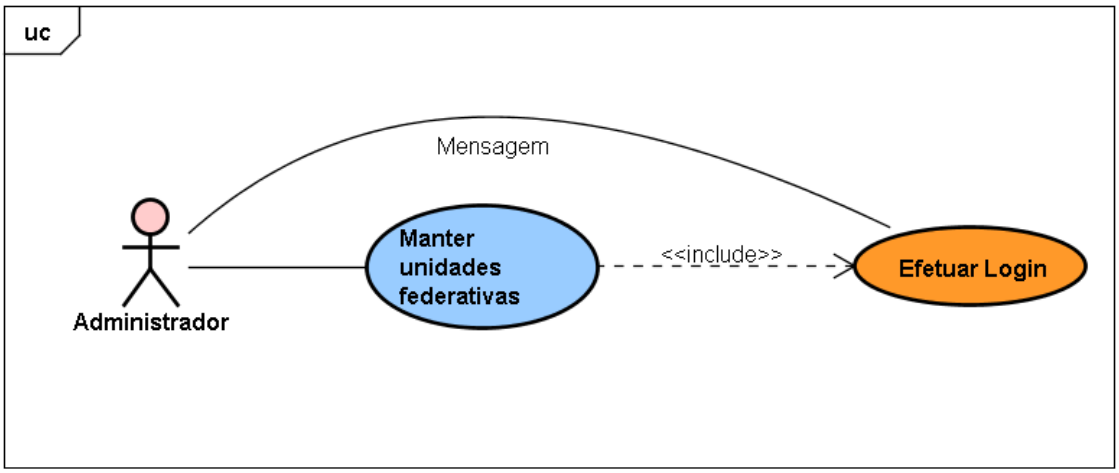


Figura 11 – UC 3 Diagrama de caso de uso Manter unidades federativas

Nome do caso de uso 3	Manter unidades federativas
Atores	Administrador
Pré-condição	Efetuar Login
Cenário principal	1. O administrador informa os dados da unidade federativa. 2. O administrador seleciona o país. 3. O administrador informa o nome da unidade federativa. 4. O sistema valida os dados informados. 5. O administrador seleciona a opção “cadastrar”. 6. O sistema emite mensagem de sucesso de gravação. 7. O sistema cadastra a unidade federativa.
Cenário alternativo	Caso o sistema não valide os dados, o sistema enviará mensagem de alerta. O administrador poderá cancelar o cadastro durante a operação de cadastro.
Casos de teste	1.1 O sistema verifica se os dados foram inseridos corretamente. 1.2 O sistema emite mensagem de alerta informando que os dados não foram inseridos corretamente. 1.3 O sistema cancela a operação. 2.1 O sistema verifica se o administrador selecionou um país.

Tabela 3 – Manter unidades federativas

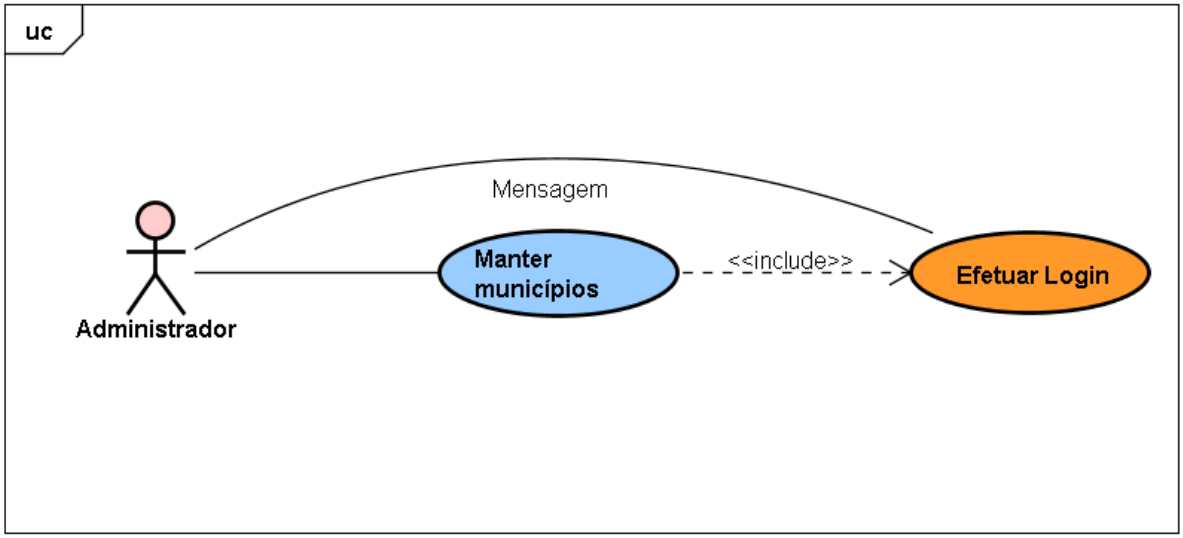


Figura 12 – UC 4 Diagrama de caso de uso Manter municípios

Nome do caso de uso 4	Manter municípios
Atores	Administrador
Pré-condição	Efetuar Login
Cenário principal	1. O administrador informa os dados do município. 2. O administrador informa o nome do município. 3. O administrador seleciona o país. 4. O administrador seleciona a unidade federativa. 5. O sistema valida os dados informados. 6. O administrador seleciona a opção “cadastrar”. 7. O sistema emite mensagem de sucesso de gravação. 8. O sistema cadastra o município.
Cenário alternativo	Caso o sistema não valide os dados, o sistema enviará mensagem de alerta. O administrador poderá cancelar o cadastro durante a operação de cadastro.
Casos de teste	1.1 O sistema verifica se os dados foram inseridos corretamente. 1.2 O sistema emite mensagem de alerta informando que os dados não foram inseridos corretamente. 1.3 O sistema cancela a operação. 2.1 O sistema verifica se o nome do município foi duplicado. 3.1 O sistema verifica se o administrador selecionou um país.

	4.1 O sistema verifica se o administrador selecionou uma unidade federativa.

Tabela 4 – Manter municípios

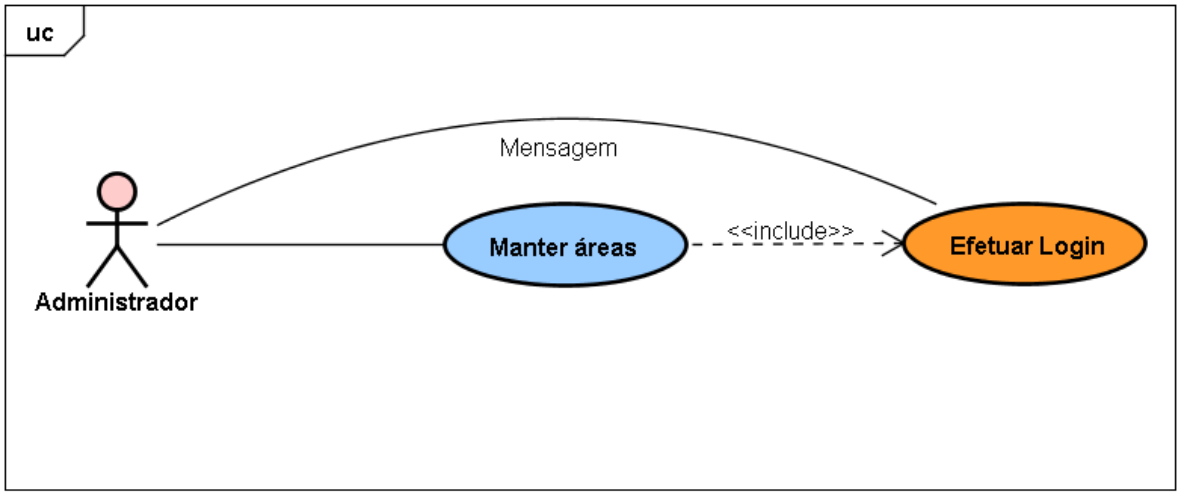


Figura 13 – UC 5 Diagrama de caso de uso Manter áreas

Nome do caso de uso 5	Manter áreas
Atores	Administrador
Pré-condição	Efetuar Login
Cenário principal	1. O administrador informa os dados da área. 2. O administrador informa o nome da área. 3. O sistema valida os dados informados. 4. O administrador seleciona a opção "cadastrar". 5. O sistema emite mensagem de sucesso da gravação. 6. O sistema cadastra a área.
Cenário alternativo	Caso o sistema não valide os dados, o sistema enviará mensagem de alerta. O administrador poderá cancelar o cadastro durante a operação de cadastro.

Casos de teste	1.1 O sistema verifica se os dados foram inseridos corretamente. 1.2 O sistema emite mensagem de alerta informando que os dados não foram inseridos corretamente. 1.3 O sistema cancela a operação. 2.1 O sistema verifica se o nome da área foi duplicado. 2.2 O sistema emite mensagem de alerta informando que a área já foi cadastrada previamente. 2.3 O sistema não permite cadastro da área.

Tabela 5 – Manter áreas

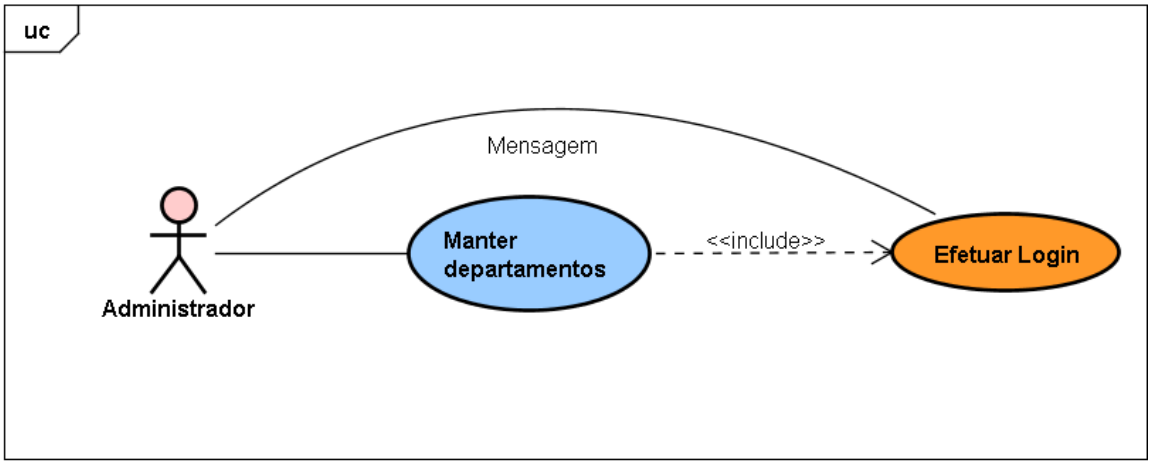


Figura 14 – UC 6 Diagrama de caso de uso Manter departamentos

Nome do caso de uso 6	Manter departamentos
Atores	Administrador
Pré-condição	Efetuar Login
Cenário principal	1. O administrador informa os dados do departamento. 2. O administrador informa o nome do departamento. 3. O administrador seleciona a área. 4. O sistema valida os dados informados. 5. O administrador seleciona a opção "cadastrar". 6. O sistema emite mensagem de sucesso da gravação. 7. O sistema cadastra o departamento.

Cenário alternativo	Caso o sistema não valide os dados, o sistema enviará mensagem de alerta. O administrador poderá cancelar o cadastro durante a operação de cadastro.
Casos de teste	1.1 O sistema verifica se os dados foram inseridos corretamente. 1.2 O sistema emite mensagem de alerta informando que os dados não foram inseridos corretamente. 1.3 O sistema cancela a operação.

Tabela 6 – Manter departamentos

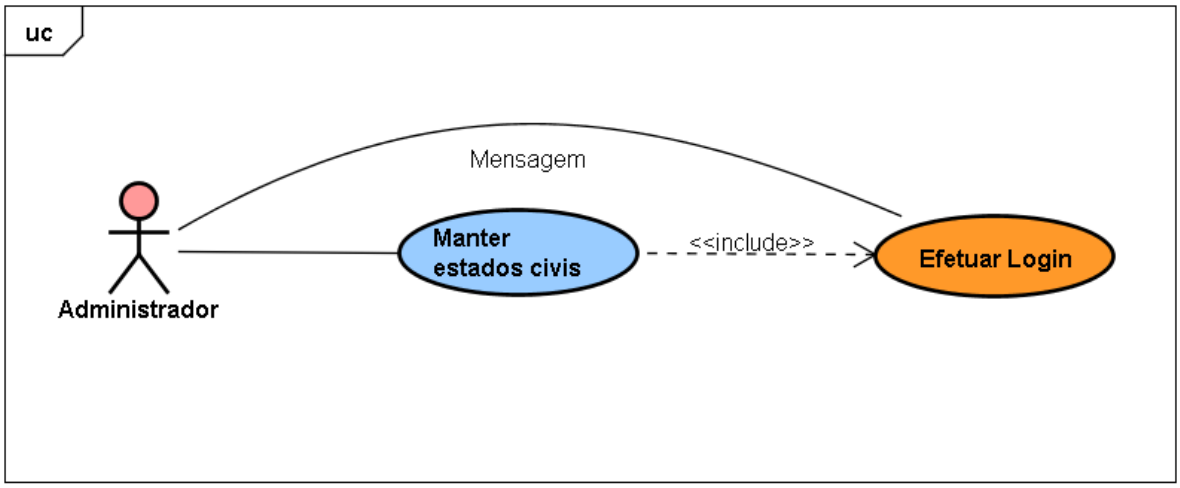


Figura 15 – UC 7 Diagrama de caso de uso Manter estados civis

Nome do caso de uso 7	Manter estados civis
Atores	Administrador
Pré-condição	Efetuar Login
Cenário principal	1. O administrador informa os dados do estado civil. 2 O administrador informa o nome do estado civil. 2. O sistema valida os dados informados. 3. O administrador seleciona a opção “cadastrar”. 4. O sistema emite mensagem de sucesso de gravação. 5. O sistema cadastra o estado civil.

Cenário alternativo	Caso o sistema não valide os dados, o sistema enviará mensagem de alerta. O administrador poderá cancelar o cadastro durante a operação de cadastro.
Casos de teste	1.1 O sistema verifica se os dados foram inseridos corretamente. 1.2 O sistema emite mensagem de alerta informando que os dados não foram inseridos corretamente. 1.3 O sistema cancela a operação.

Tabela 7 – Manter estados civis

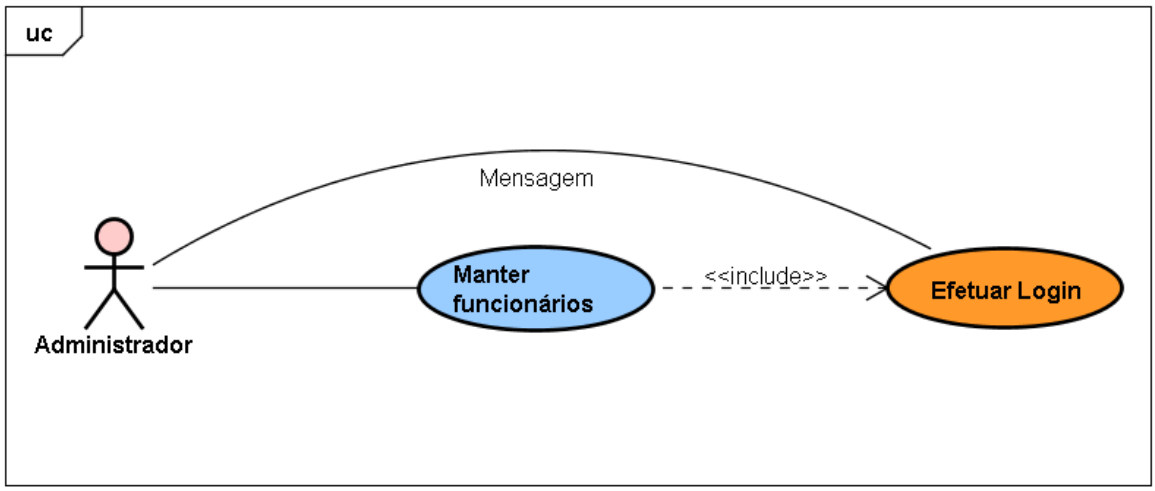


Figura 16 – UC 8 Diagrama de caso de uso Manter funcionários

Nome do caso de uso 8	Manter funcionários
Atores	Administrador
Pré-condição	Efetuar Login
Cenário principal	1. O administrador informa os dados do funcionário. 2. O administrador informa o CPF. 3. O sistema valida os dados informados. 4. O administrador seleciona a opção “cadastrar”. 5. O sistema emite mensagem de sucesso de gravação. 6. O sistema cadastra o funcionário.
Cenário alternativo	Caso o sistema não valide os dados,

	o sistema enviará mensagem de alerta. O administrador poderá cancelar o cadastro durante a operação de cadastro.
Casos de teste	1.1 O sistema verifica se os dados foram inseridos corretamente. 1.2 O sistema emite mensagem de alerta informando que os dados não foram inseridos corretamente. 1.3 O sistema cancela a operação.

Tabela 8 – Manter funcionários

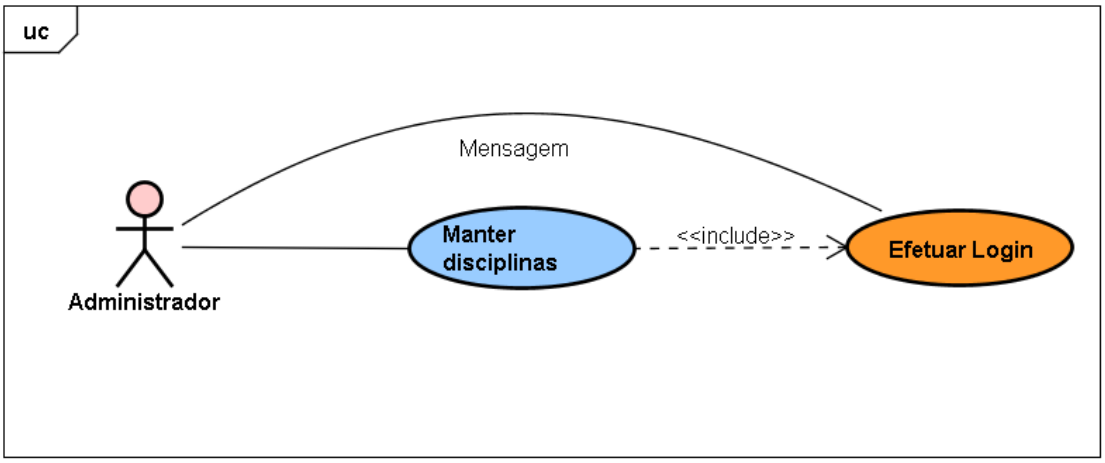


Figura 17 – UC 9 Diagrama de caso de uso Manter disciplina

Nome do caso de uso 9	Manter disciplinas
Atores	Administrador
Pré-condição	Efetuar Login
Cenário principal	1. O administrador informa os dados da disciplina. 2. O administrador informa o nome da disciplina. 3. O sistema valida os dados informados. 4. O administrador seleciona a opção "cadastrar". 5. O sistema emite mensagem de sucesso da gravação. 6. O sistema cadastra a disciplina.
Cenário alternativo	Caso o sistema não valide os dados, o sistema enviará mensagem de alerta. O administrador poderá cancelar o cadastro durante a operação de cadastro.

Casos de teste	1.1 O sistema verifica se os dados foram inseridos corretamente. 1.2 O sistema emite mensagem de alerta informando que os dados não foram inseridos corretamente. 1.3 O sistema cancela a operação.

Tabela 9 – Manter disciplinas

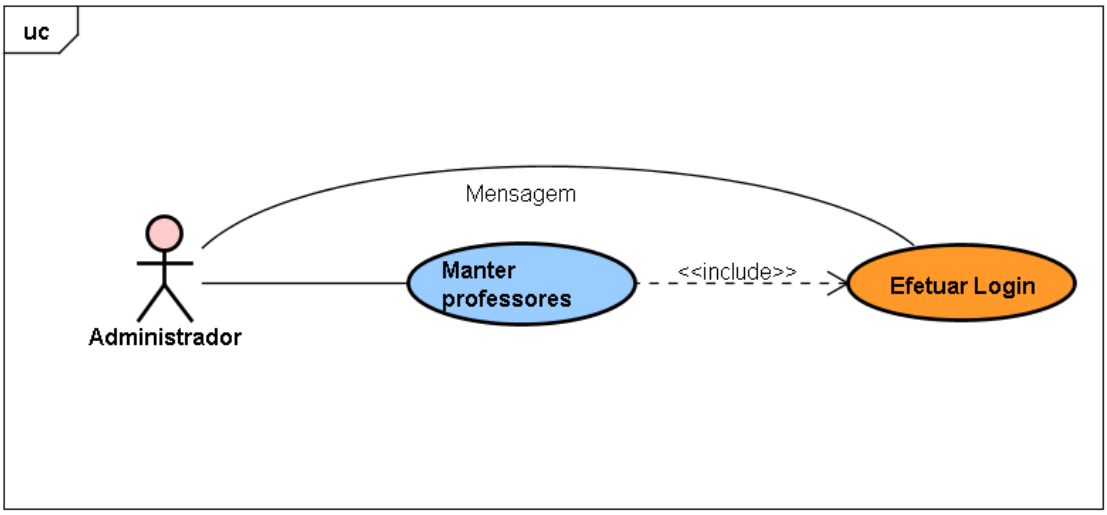


Figura 18 – UC 10 Diagrama de caso de uso Manter professores

Nome do caso de uso 10	Manter professores
Atores	Administrador
Pré-condição	Efetuar Login
Cenário principal	1. O administrador informa os dados do professor. 2. O administrador informa o CPF do professor. 3. O sistema valida os dados informados. 4. O administrador seleciona a opção “cadastrar”. 5. O sistema emite mensagem de sucesso de gravação. 6. O sistema cadastra o professor.
Cenário alternativo	Caso o sistema não valide os dados, o sistema enviará mensagem de alerta. O administrador poderá cancelar o cadastro durante a operação de cadastro.

Casos de teste	1.1 O sistema verifica se os dados foram inseridos corretamente. 1.2 O sistema emite mensagem de alerta informando que os dados não foram inseridos corretamente. 1.3 O sistema cancela a operação.

Tabela 10 – Manter professores

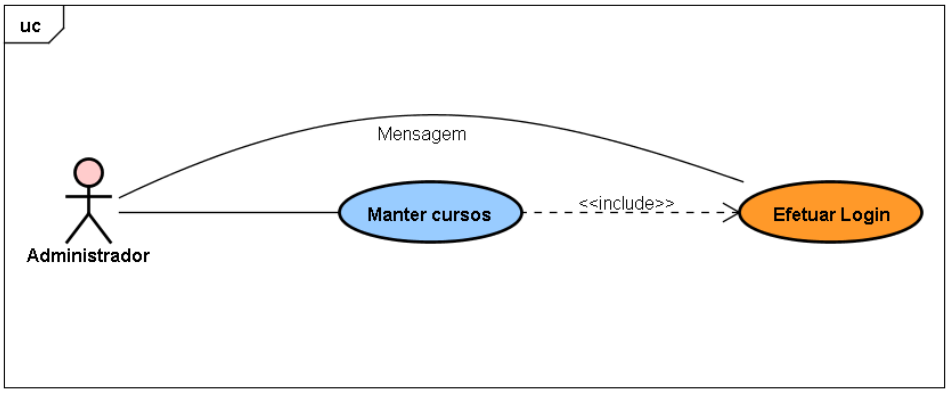


Figura 19 – UC 11 Diagrama de caso de uso Manter cursos

Nome do caso de uso 11	Manter cursos
Atores	Administrador
Pré-condição	Efetuar Login
Cenário principal	1. O administrador informa os dados do curso. 2. O administrador informa o nome do curso. 3. O sistema valida os dados informados. 4. O administrador seleciona a opção "cadastrar". 5. O sistema emite mensagem de sucesso da gravação. 6. O sistema cadastra o curso.
Cenário alternativo	Caso o sistema não valide os dados, o sistema enviará mensagem de alerta. O administrador poderá cancelar o cadastro durante a operação de cadastro.
Casos de teste	1.1 O sistema verifica se os dados foram inseridos corretamente. 1.2 O sistema emite mensagem de alerta informando que os dados não foram inseridos corretamente. 1.3 O sistema cancela a operação.

Tabela 11 – Manter cursos

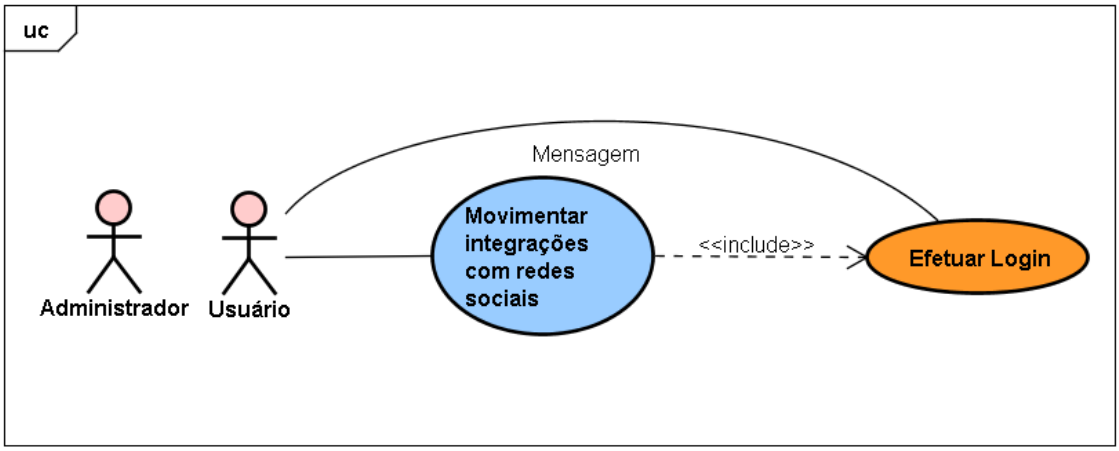


Figura 20 – UC 12 Diagrama de caso de uso Movimentar integrações com redes sociais

Nome do caso de uso 12	Movimentar integrações com redes sociais
Atores	Administrador e Usuário
Pré-condição	Efetuar Login
Cenário principal	1. O usuário escolhe a rede social que será utilizada. 2. O usuário preenche o campo para que a postagem seja feita.
Cenário alternativo	O usuário poderá cancelar o operação durante a qualquer momento do caso de uso.
Casos de teste	1.1 Verificar se o usuário escolheu uma rede social. 2.1 Verificar se o usuário digitou uma postagem.

Tabela 12 – Movimentar integrações com redes sociais

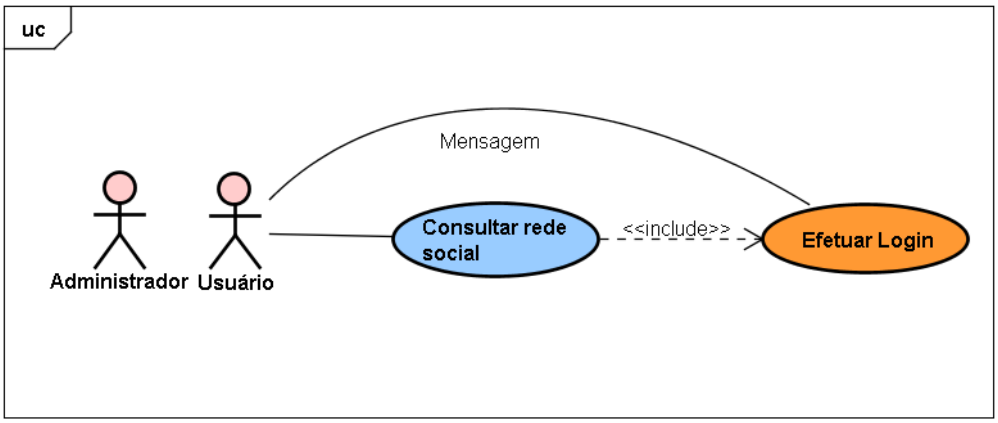


Figura 21 – UC 13 Diagrama de caso de uso Consultar rede social

Nome do caso de uso 13	Consultar rede social
Atores	Administrador e Usuário
Pré-condição	Efetuar Login
Cenário principal	1. O usuário escolhe a rede social que será consultada. 2. O usuário solicita a opção de visualização dos posts. 3. O usuário define os parâmetro de busca dos posts. 4. O usuário seleciona a opção “buscar” 5. O sistema retorna para o usuário os posts seguindo os parâmetros de busca.
Cenário alternativo	O usuário poderá cancelar o operação durante a qualquer momento da consulta.
Casos de teste	1.1 Verificar se o usuário escolheu uma rede social. 3.1 Verificar se o usuário escolheu um parâmetro de busca.

Tabela 13 – Consultar rede social

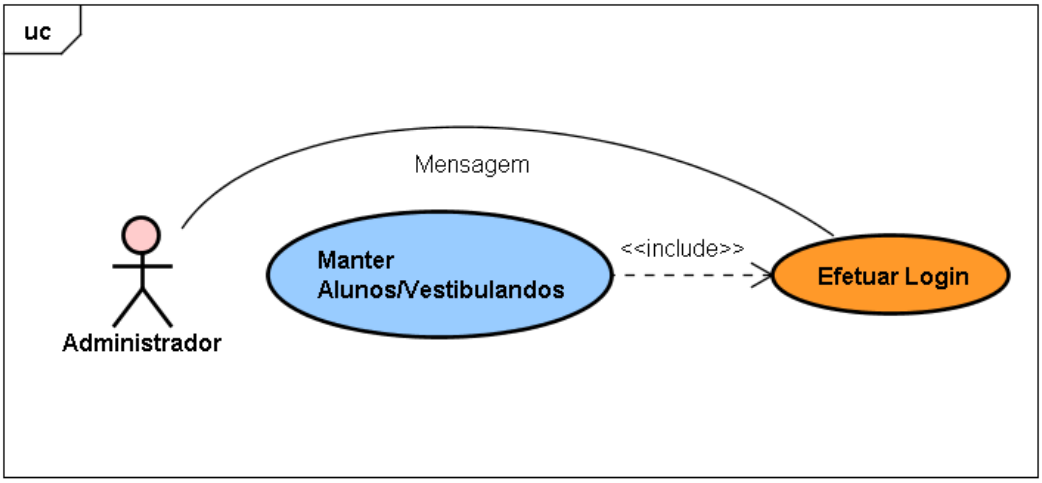


Figura 22 – UC 14 Diagrama de caso de uso Manter Alunos/vestibulandos

Nome do caso de uso 15	Manter alunos/vestibulandos
Atores	Administrador
Pré-condição	Efetuar Login
Cenário principal	1. O administrador e ou usuário insere os dados requeridos pelo sistema. 2. O administrador e ou usuário informa o CPF do vestibulando. 3. O sistema valida os dados do vestibulando. 4. 5. O administrador e ou usuário clique sobre o botão “cadastrar”. 5. O sistema emite mensagem de sucesso da gravação. 6. O sistema cadastra o vestibulando.
Cenário alternativo	Caso o sistema não valide os dados, o sistema enviará mensagem de alerta. O administrador poderá cancelar o cadastro durante a operação de cadastro.
Casos de teste	1.1 O sistema verifica se os dados foram inseridos corretamente. 1.2 O sistema emite mensagem de alerta informando que os dados não foram inseridos corretamente. 1.3 O sistema cancela a operação.

Tabela 14 – Manter alunos/vestibulandos

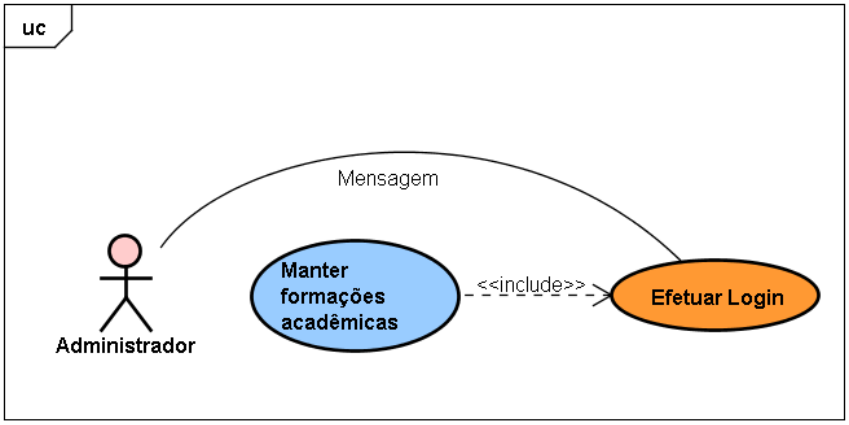


Figura 23 – UC 15 Diagrama de caso de uso Manter formações acadêmicas

Nome do caso de uso 15	Manter formações acadêmicas
Atores	Administrador
Pré-condição	Efetuar Login
Cenário principal	1. O administrador insere os dados requeridos pelo sistema. 2. O administrador informa o nome da formação acadêmica. 3. O sistema valida os dados da formação acadêmica. 4. 5. O administrador clica sobre o botão “cadastrar”. 5. O sistema emite mensagem de sucesso da gravação. 6. O sistema cadastra a formação acadêmica.
Cenário alternativo	Caso o sistema não valide os dados, o sistema enviará mensagem de alerta. O administrador poderá cancelar o cadastro durante a operação de cadastro.
Casos de teste	1.1 O sistema verifica se os dados foram inseridos corretamente. 1.2 O sistema emite mensagem de alerta informando que os dados não foram inseridos corretamente. 1.3 O sistema cancela a operação.

Tabela 15 – Manter formações acadêmicas

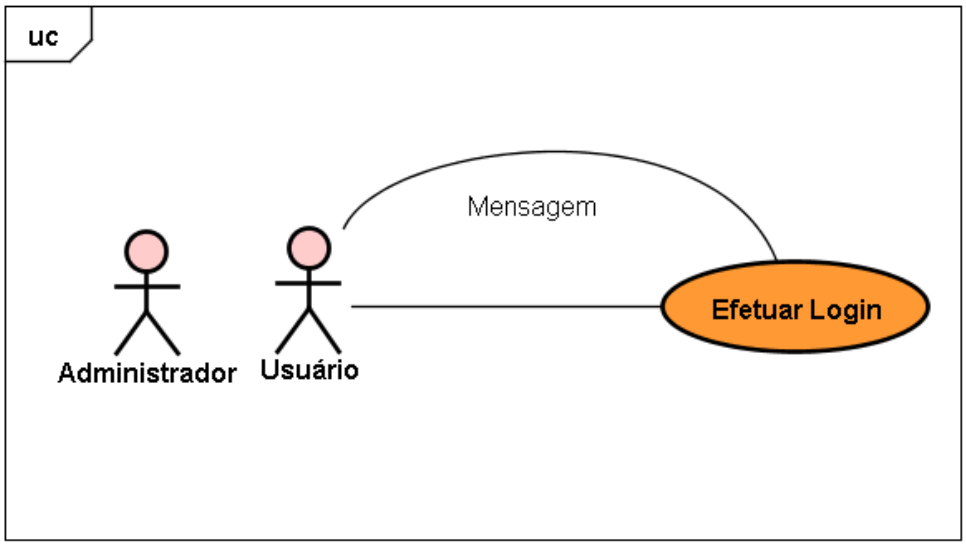


Figura 24 – UC 17 Diagrama de caso de uso Efetuar Login

Nome do caso de uso 17	Efetuar Login
Atores	Administrador e Usuário
Pré-condição	Não há
Cenário principal	1. O sistema solicita os dados para efetuar o login. 2. O usuário insere os dados. 3. O usuário confirma o login. 4. O sistema recupera os dados do usuário. 5. O sistema valida os dados. 6. O usuário entra no sistema.
Cenário alternativo	Não há.
Casos de teste	5.1 Caso os dados não estejam corretos, o sistema emite mensagem de alerta.

Tabela 16 – Efetuar Login

4.5 DIAGRAMA DE ATIVIDADES

Segundo SILVA et al. (2001), o diagrama de atividades representa o fluxo de controle entre as atividades que serão desempenhadas pelo sistema, e são utilizados para ilustrar o tempo de vida de um objeto.

Ainda segundo SILVA et al. (2001), os diagramas de atividade equivalem ao fluxograma, no qual é apresentado os processos de negócio de forma simplificada.

Para exemplificar como serão realizadas as atividades dentro do sistema foi escolhida a classe “Disciplina” para demonstrar serão procedidas as ações mais comuns que ocorrerão na aplicação. Além disto, os principais diagramas de atividades, aqueles que envolvem as redes sociais, estão contidos nesta mostra.

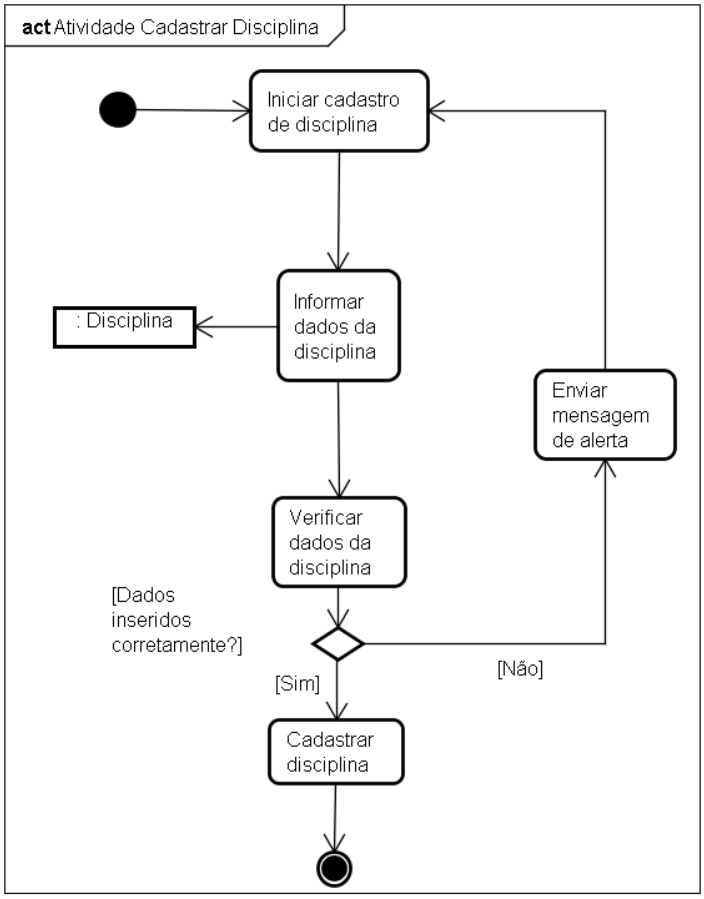


Figura 25 – Atividade Cadastrar Disciplina

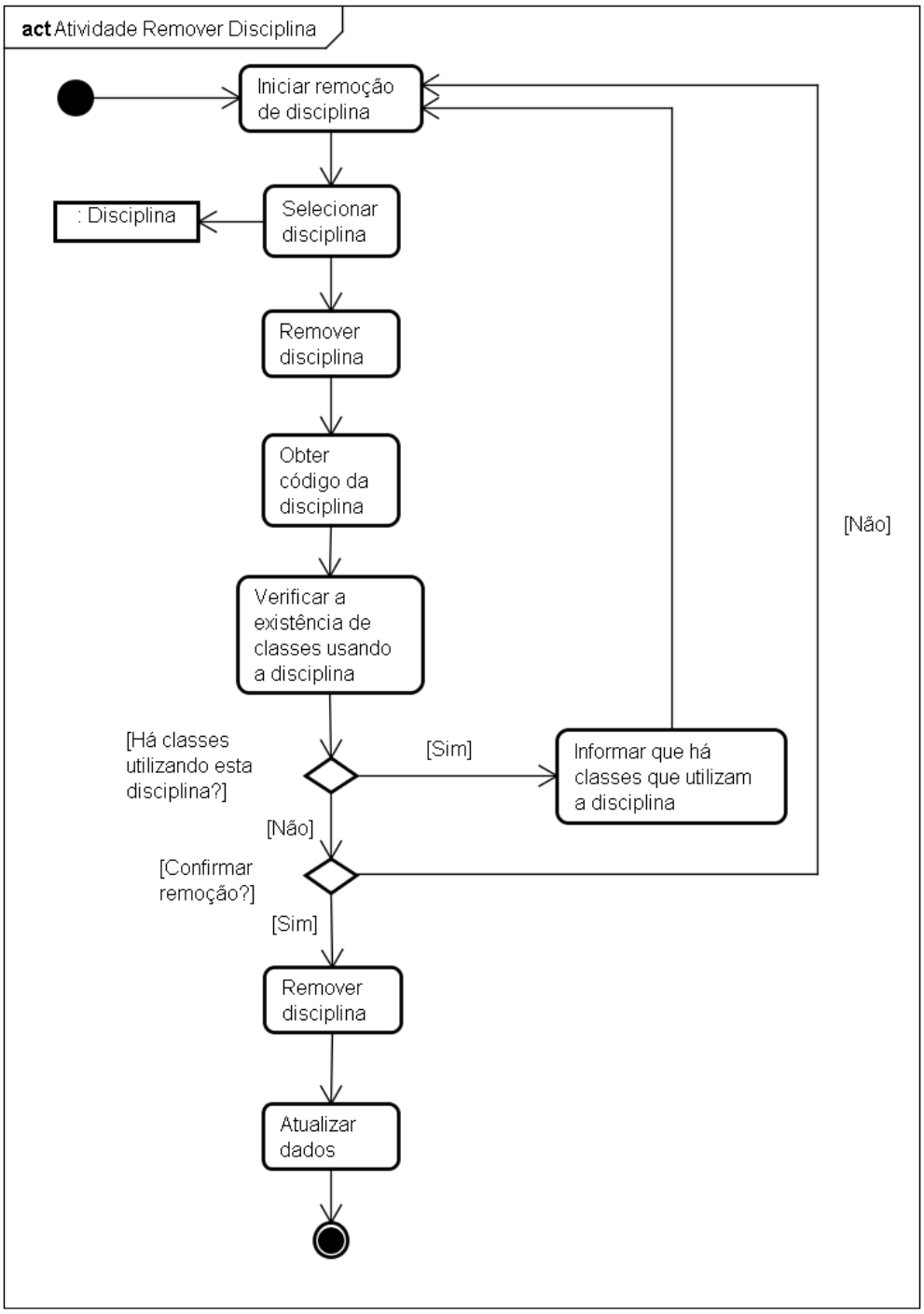


Figura 26 – Atividade Remover Disciplina

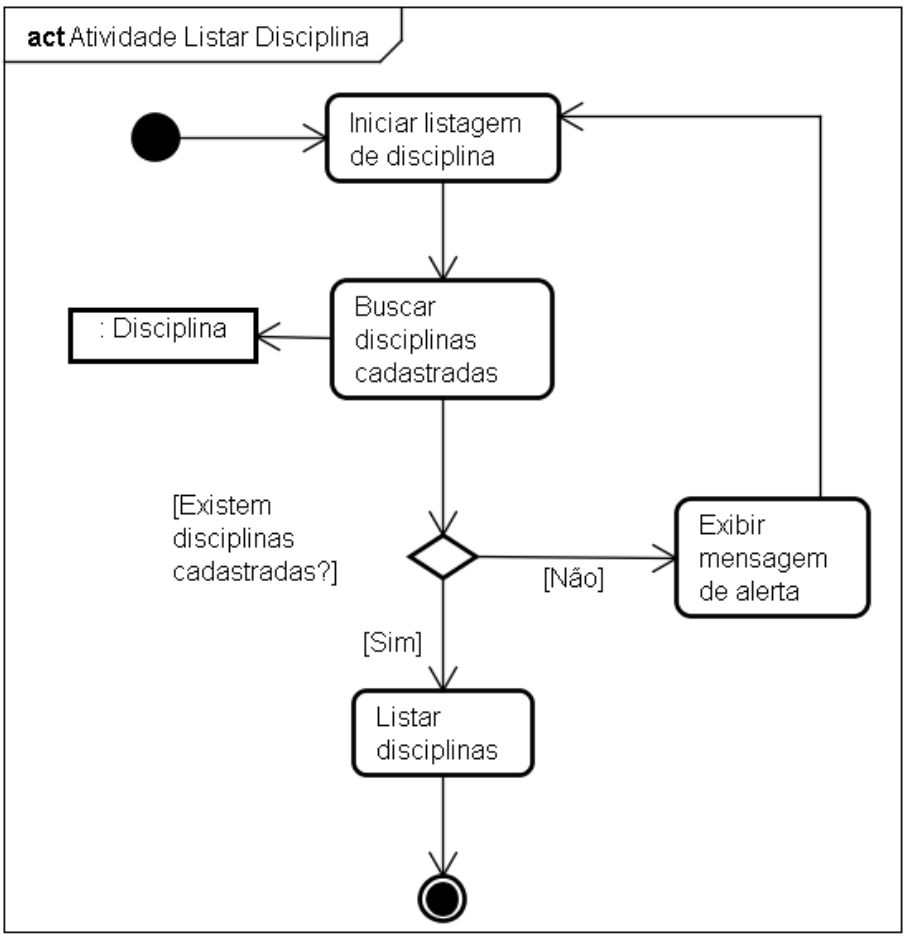


Figura 27 – Atividade Listar Disciplinas

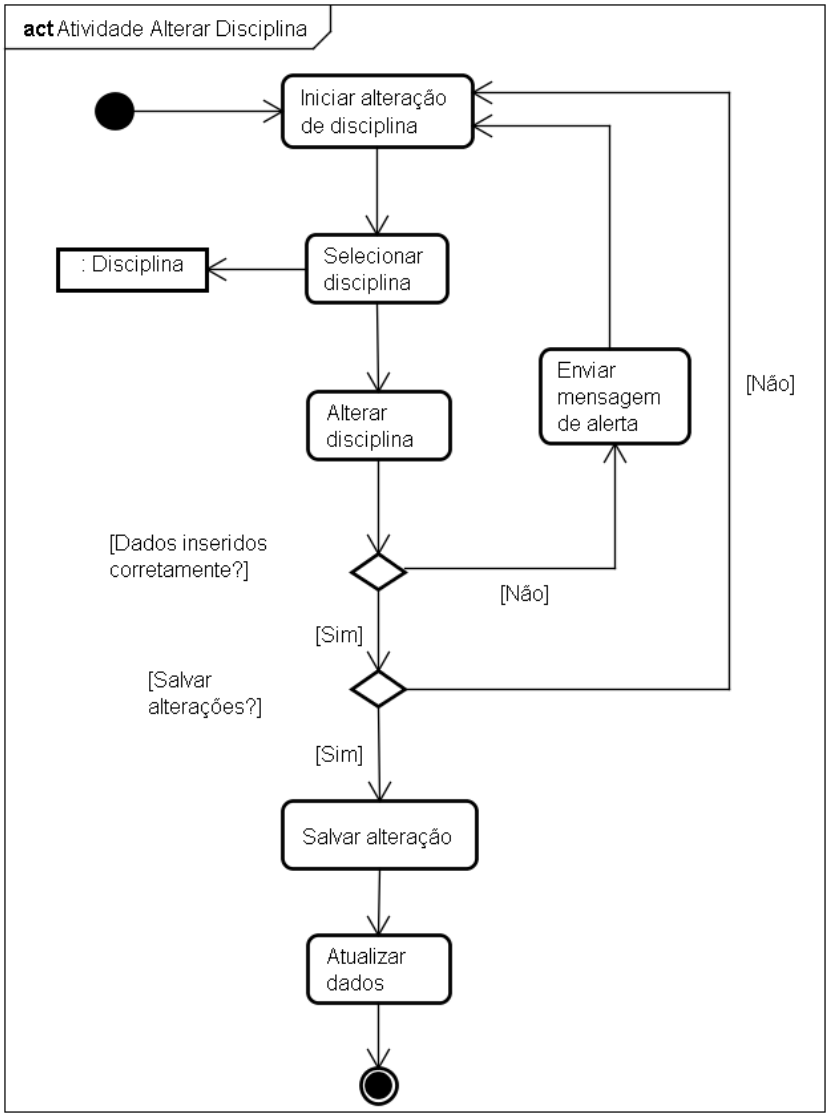


Figura 28 – Atividade Alterar Disciplina

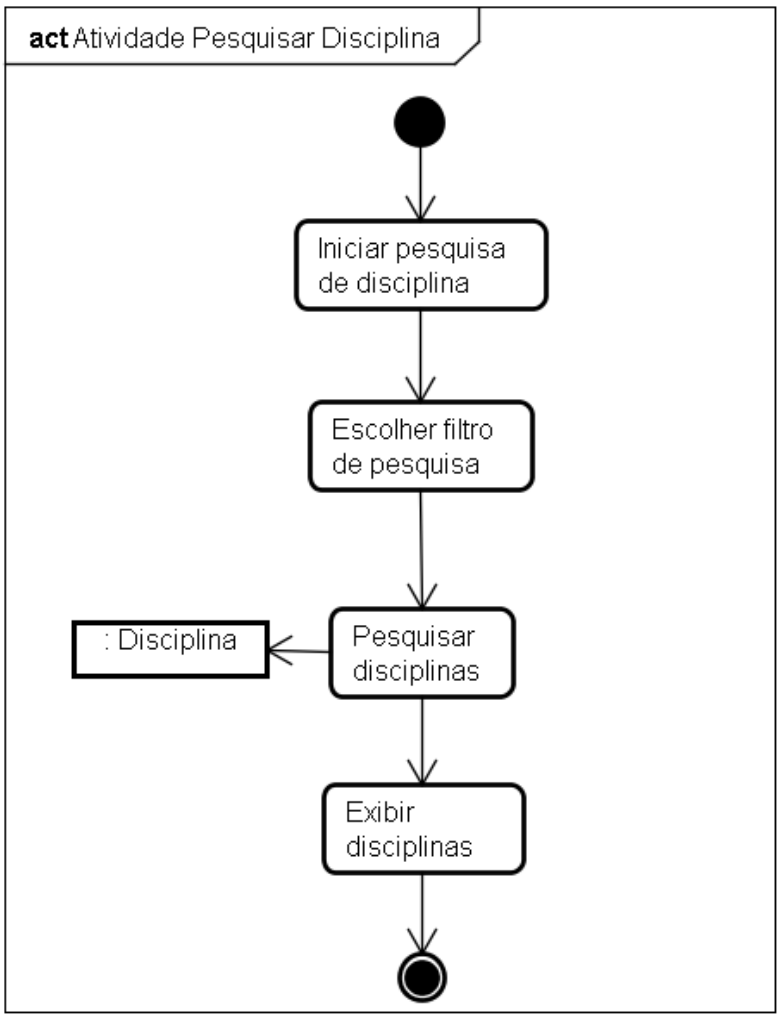


Figura 29 – Atividade Pesquisar Disciplina

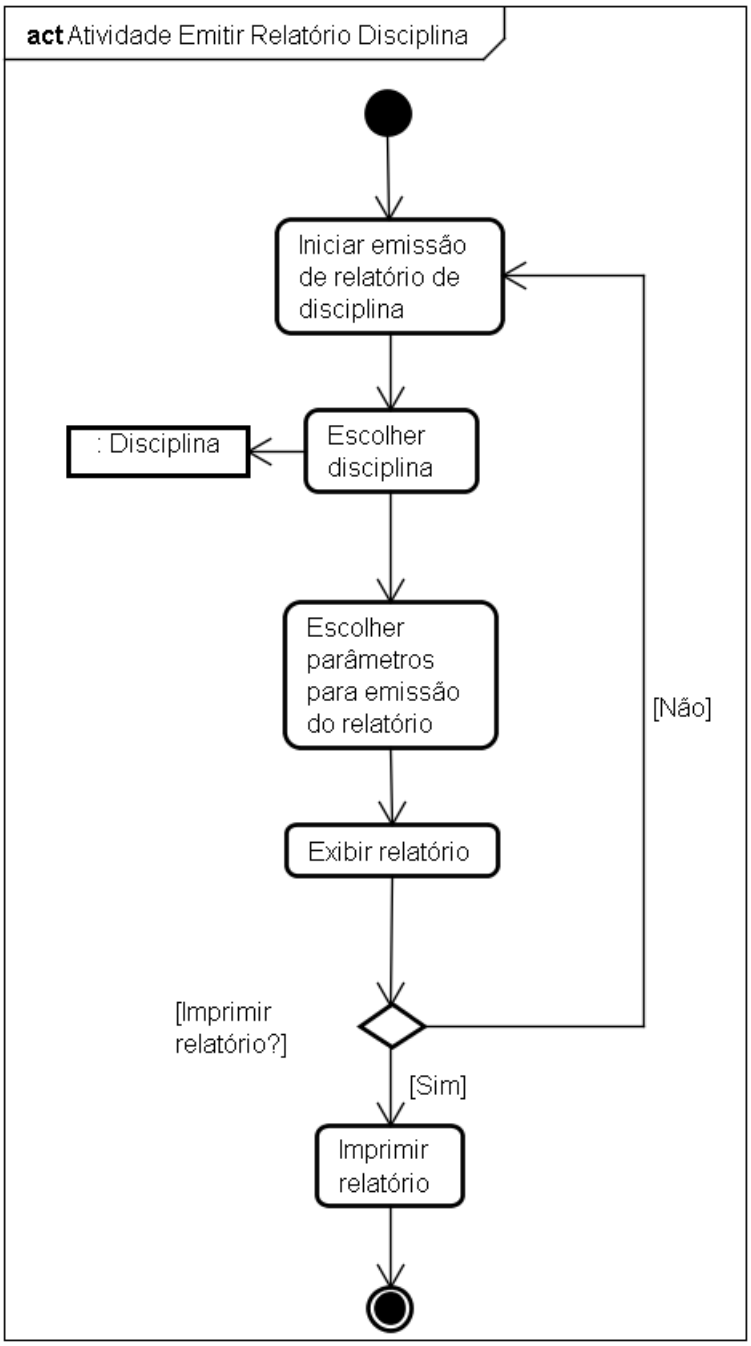


Figura 30 – Atividade Emitir Relatório de Disciplina

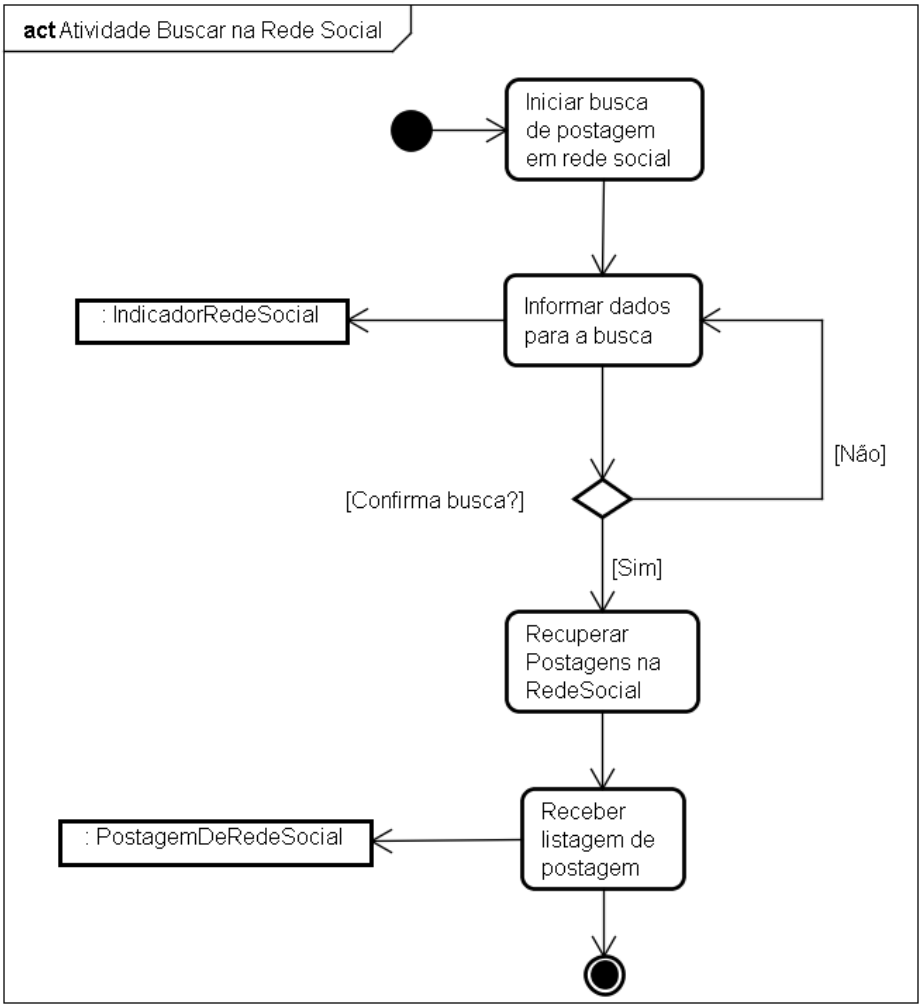


Figura 31 – Atividade Buscar na Rede Social

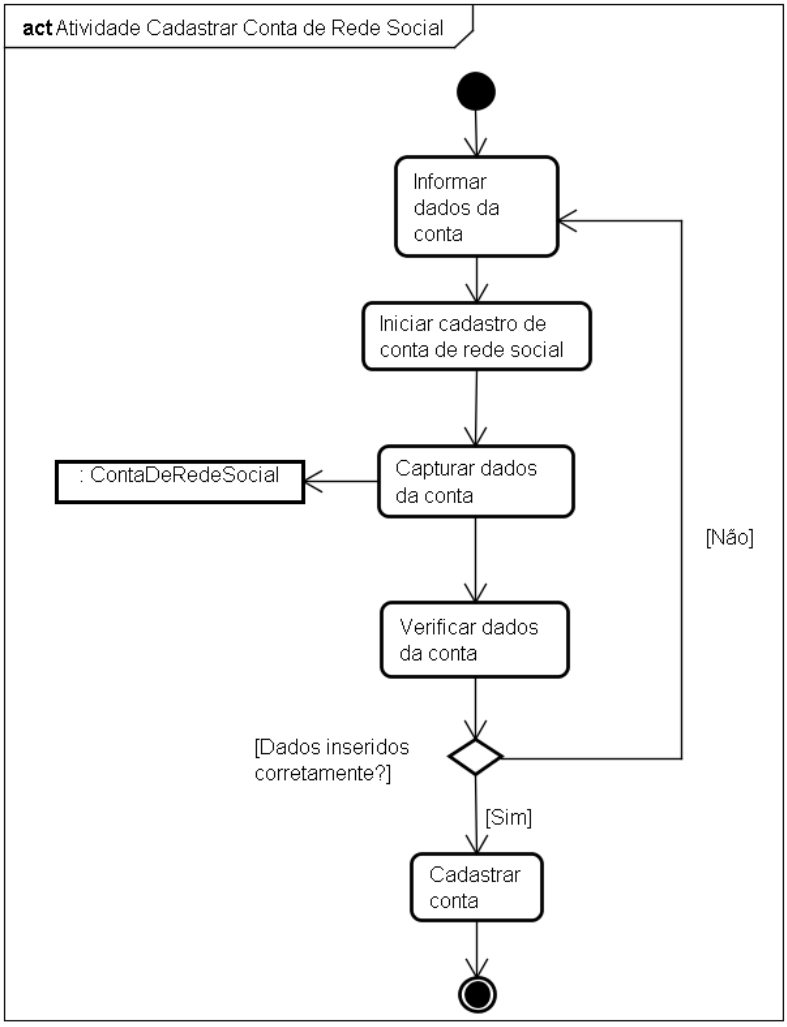


Figura 32 – Atividade Cadastrar Conta de Rede Social

4.6 DIAGRAMA DE SEQUÊNCIAS

Segundo Silva (2001) “Diagrama de sequência: é um diagrama de interação com ênfase na ordenação temporal das mensagens trocadas entre os objetos”.

Por meio deste diagrama é possível notar como cada ação realizada pelo usuário se comportará dentro da aplicação.

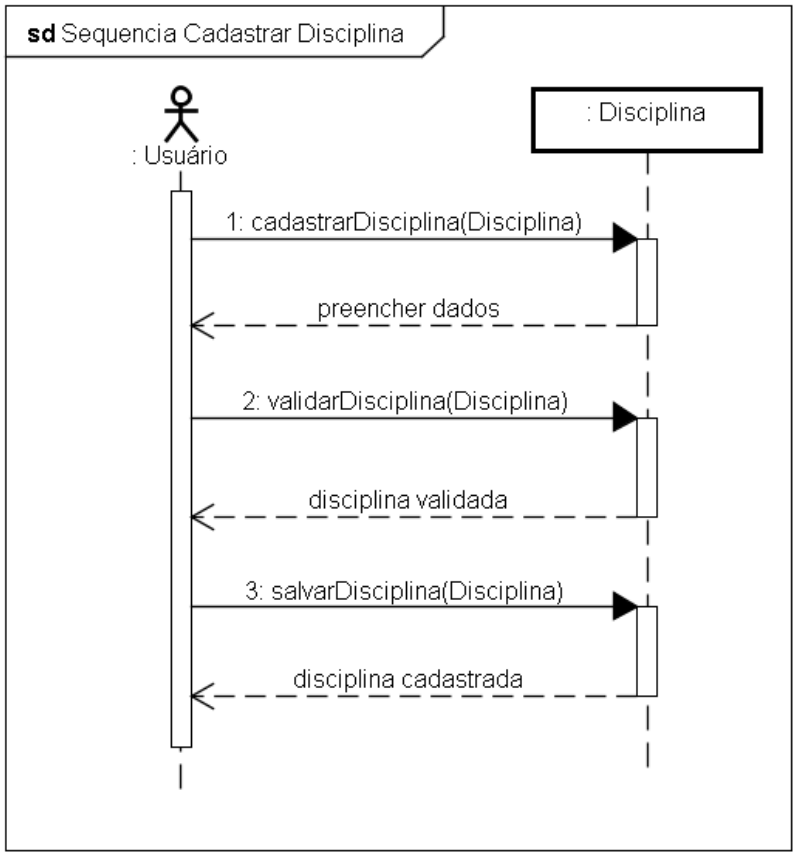


Figura 33 – Sequência Cadastrar Disciplina

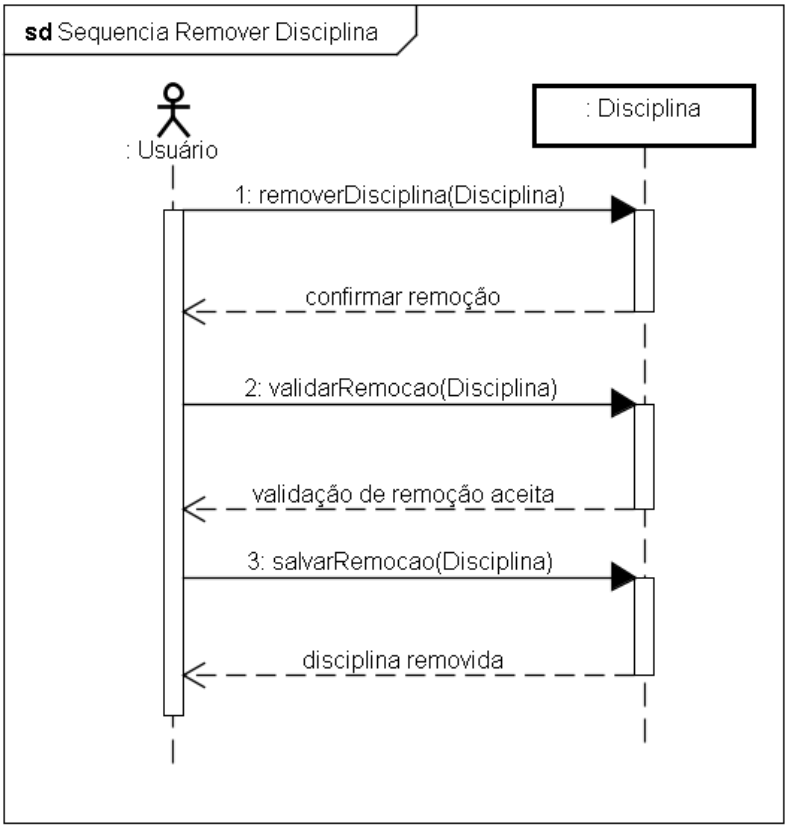


Figura 34 – Sequência Remover Disciplina

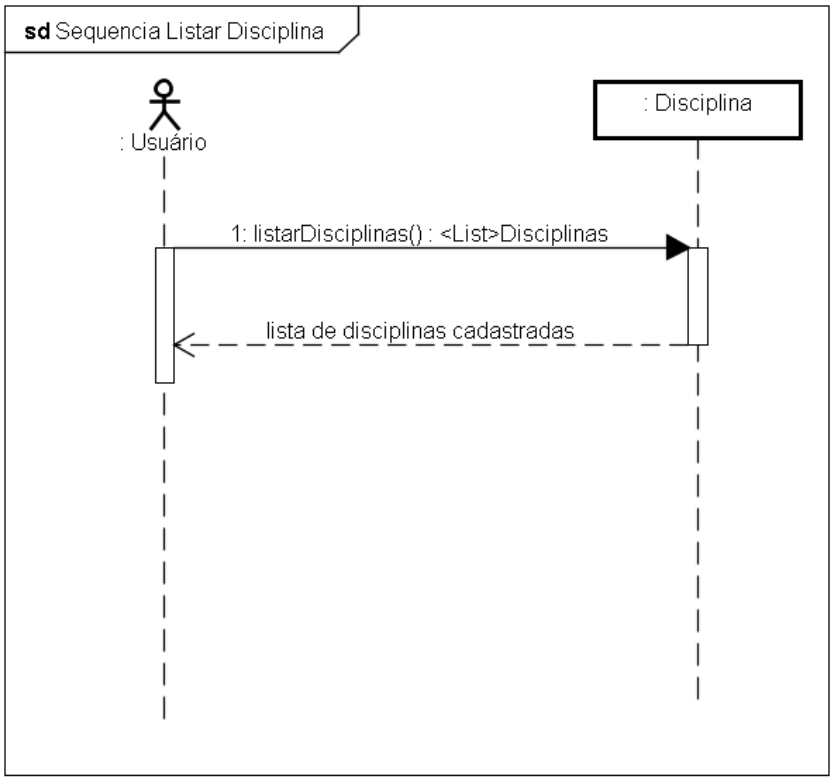


Figura 35 – Sequência Listar Disciplinas

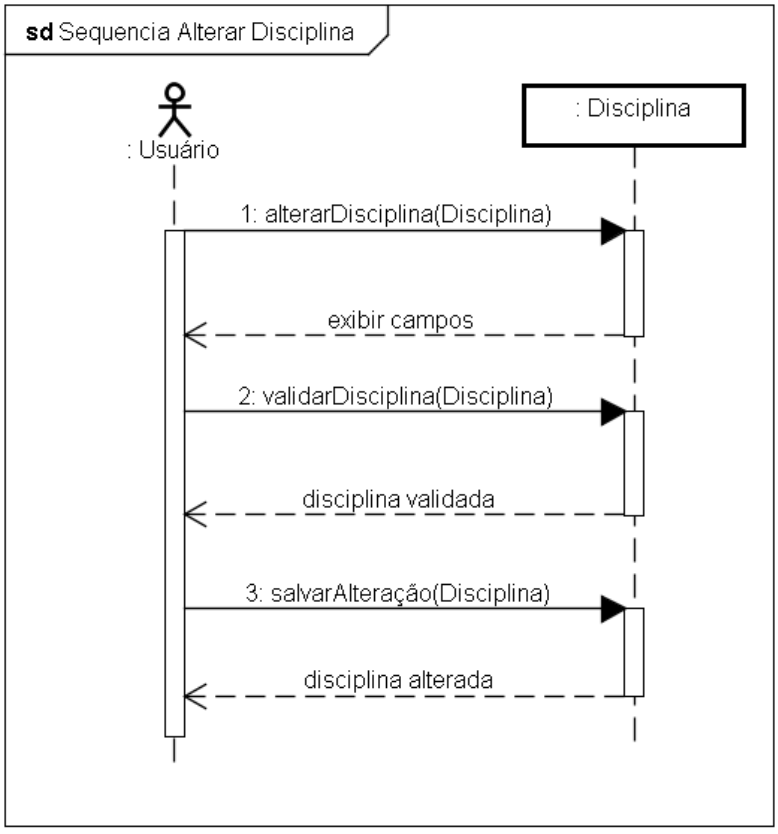


Figura 36 – Sequência Alterar Disciplina

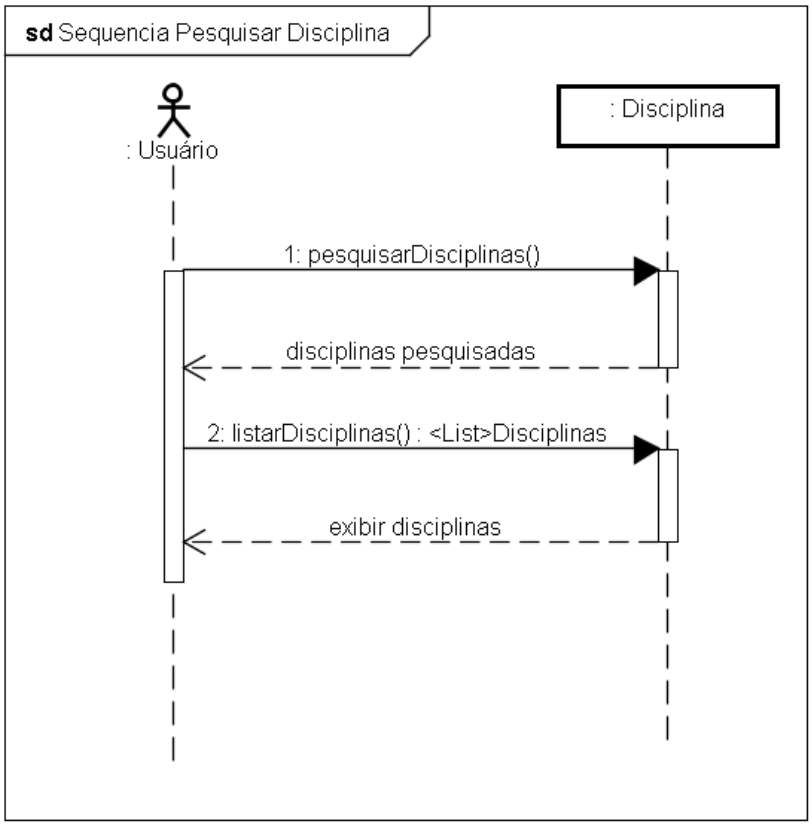


Figura 37 – Sequência Pesquisar Disciplina

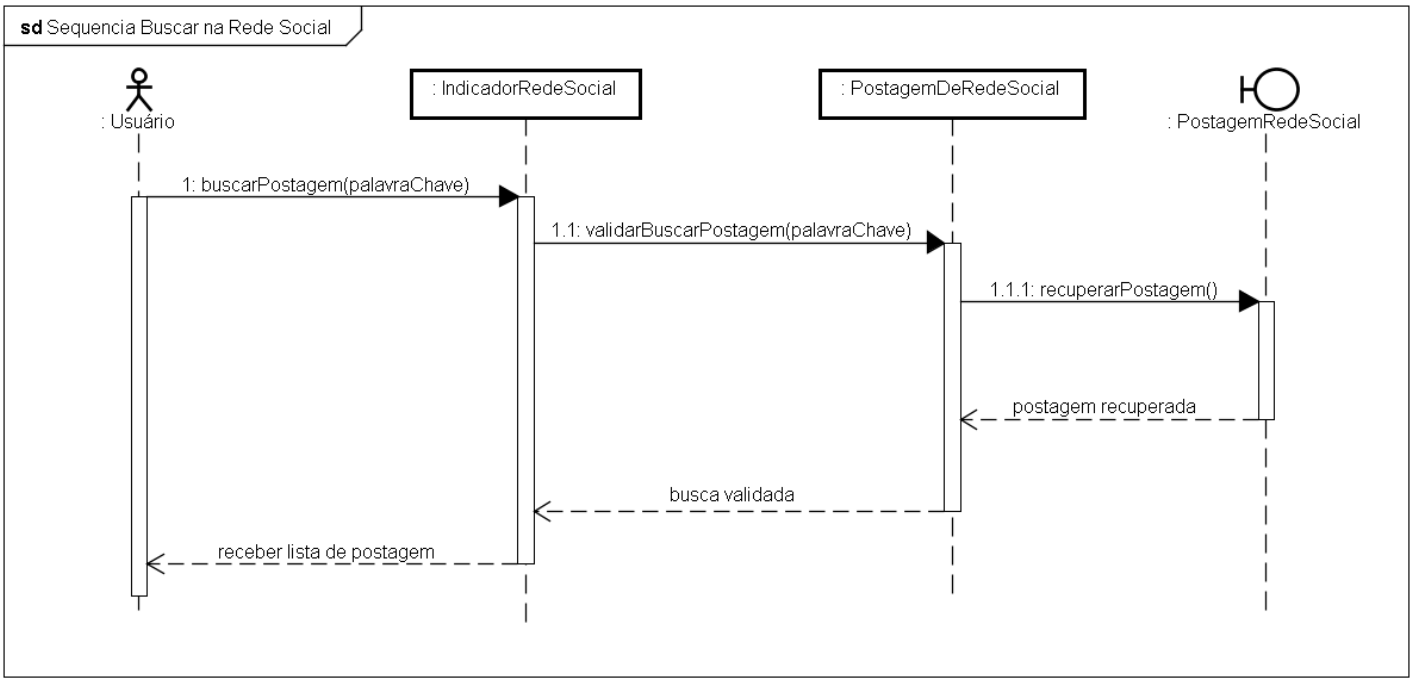
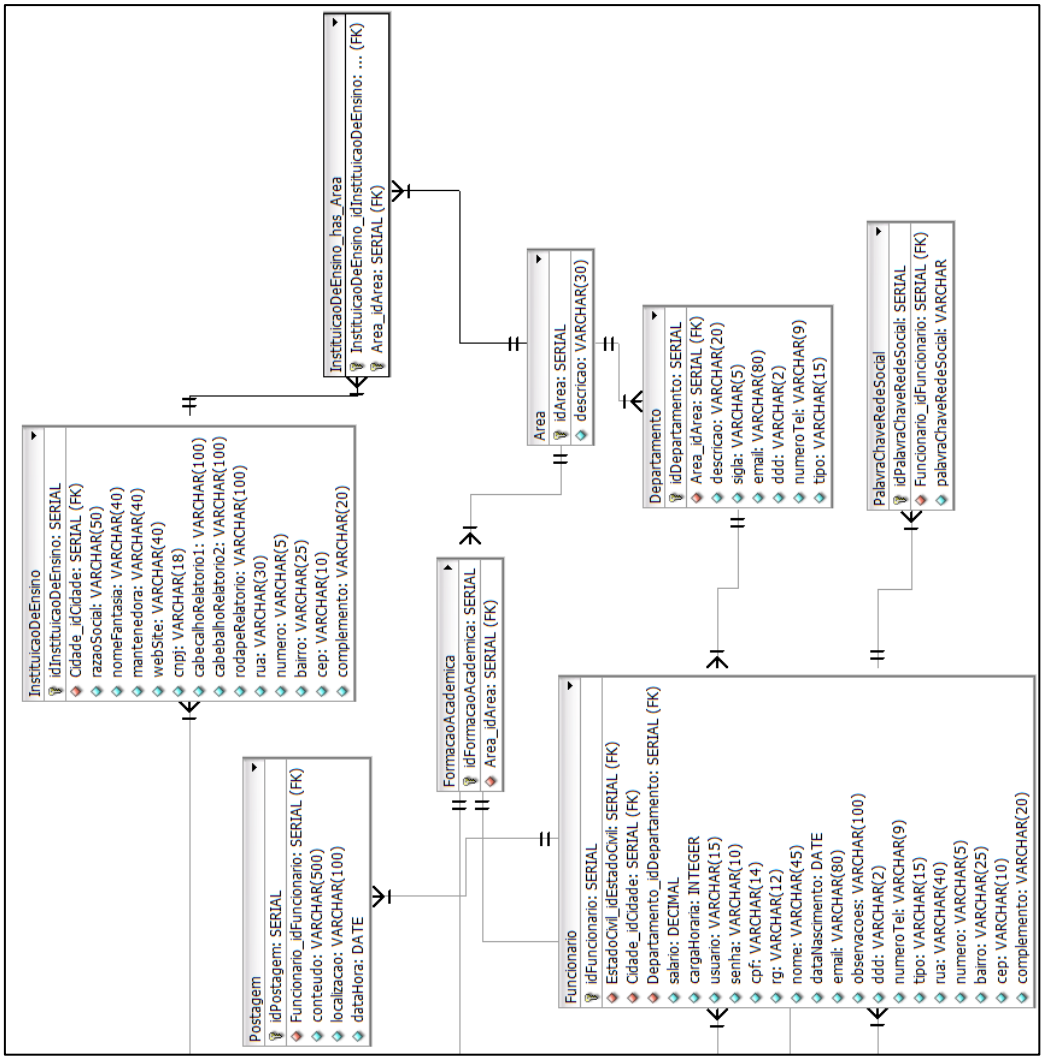


Figura 38 – Sequência Buscar Postagem na Rede Social

4.7 MODELO ENTIDADE-RELACIONAMENTO

O Modelo entidade-relacionamento (MER) descreve a estrutura lógica do banco de dados (BD). O MER consiste em uma coleção de entidades – objeto que possuí características distinguíveis de outros objetos, no BD é chamado de tabela –, atributos – características que identificam as entidades –, e associações entre estas entidades. Através do MER, é possível haver visualização de como as tabelas do BD deverão ser construídas.



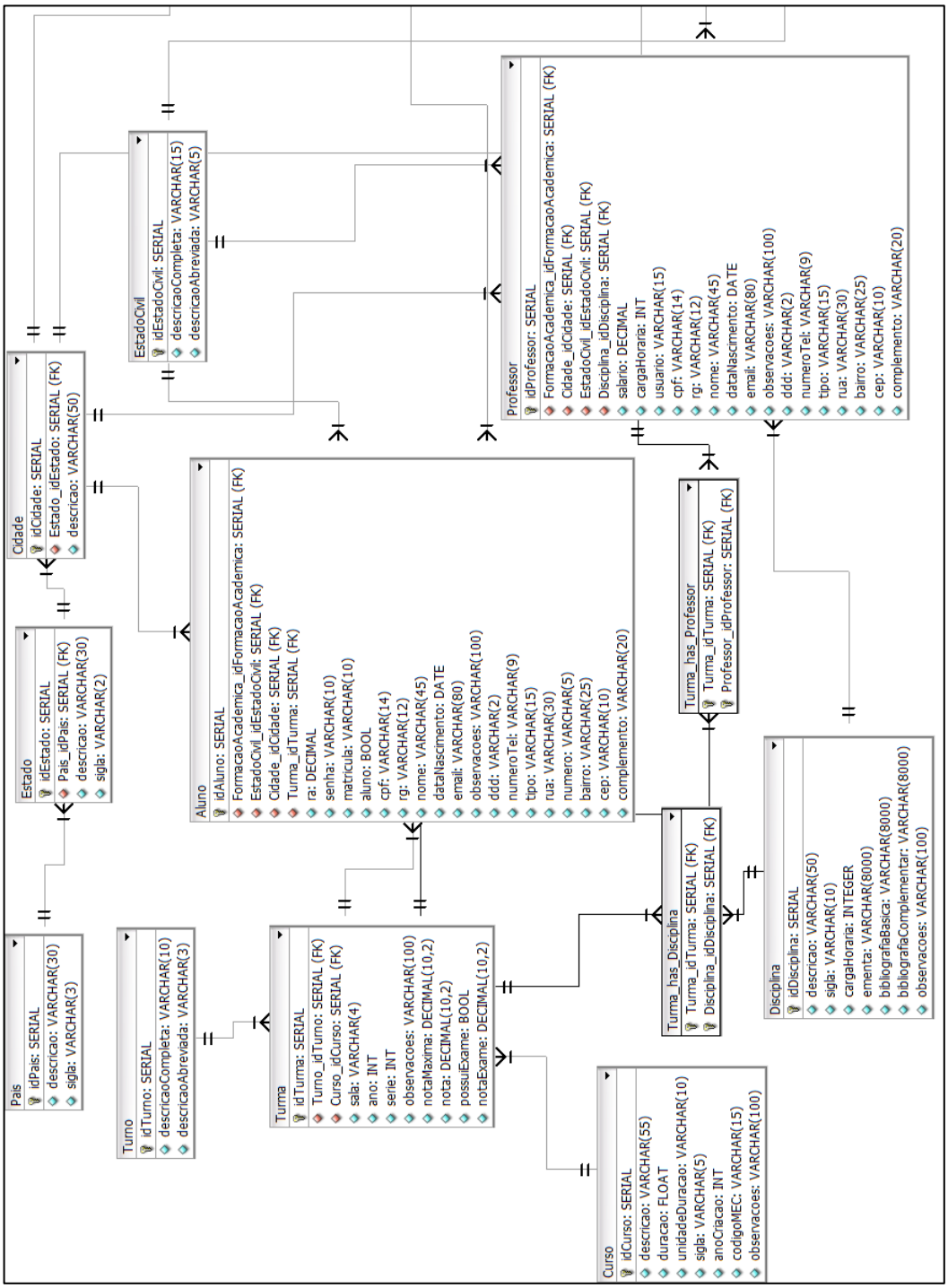


Figura 39 – Modelo Entidade-Relacionamento

4.8 DIAGRAMA DE CLASSES

Segundo Lee et al. (2001, p.510), as classes definem os objetos que serão utilizados dentro do sistema, no qual estão presentes atributos que na maior parte das vezes

são de tipos de dados primitivos e operações que podem ser aplicados sobre os objetos.

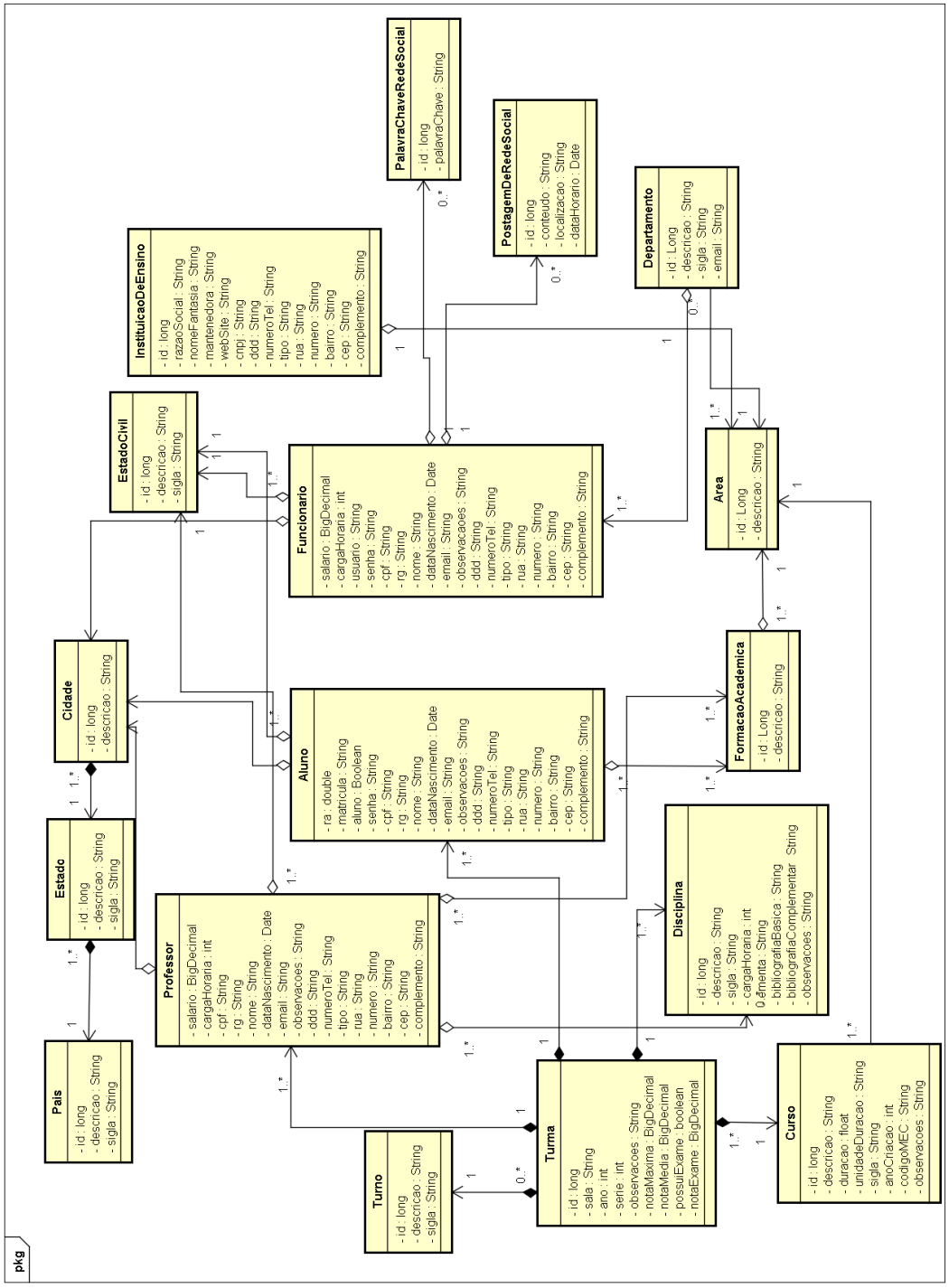


Figura 40 – Diagrama de classe

4.9 DESCRIÇÃO DE DESENVOLVIMENTO DO PROCESSO PRINCIPAL

O processo principal consiste em implementar um aplicativo utilizando a linguagem Java que seja capaz de obter informações de pessoas, na qual possuam relação com a instituição de ensino, por meio de redes sociais.

Para que tal objetivo seja alcançado, primeiramente é necessário que o desenvolvedor faça download dos artefatos necessários para desenvolvimento da linguagem Java, entre os artefatos estão o *JDK – Java Development Kit* – e uma *IDE*, nesta aplicação será utilizada o *Eclipse*.

Em seguida o desenvolvedor escolhe a rede social que deseja criar o aplicativo, verifique se a rede social possui um .jar – *Java Archive* – disponível para uso do desenvolvedor, da rede social de sua preferência, caso não o possua, o processo é encerrado; caso o possua o desenvolvedor deve primeiramente baixar o jar e então estudar os métodos disponibilizados pelo arquivo a fim de implementar o aplicativo para a movimentação que o aluno irá fazer.

Após verificar o .jar e estudar os métodos de desenvolvimento, o desenvolvedor cria o aplicativo seguindo a documentação previamente escrita em sua qualificação, caso a tenha feito; caso não a tenha feito, fazê-la.

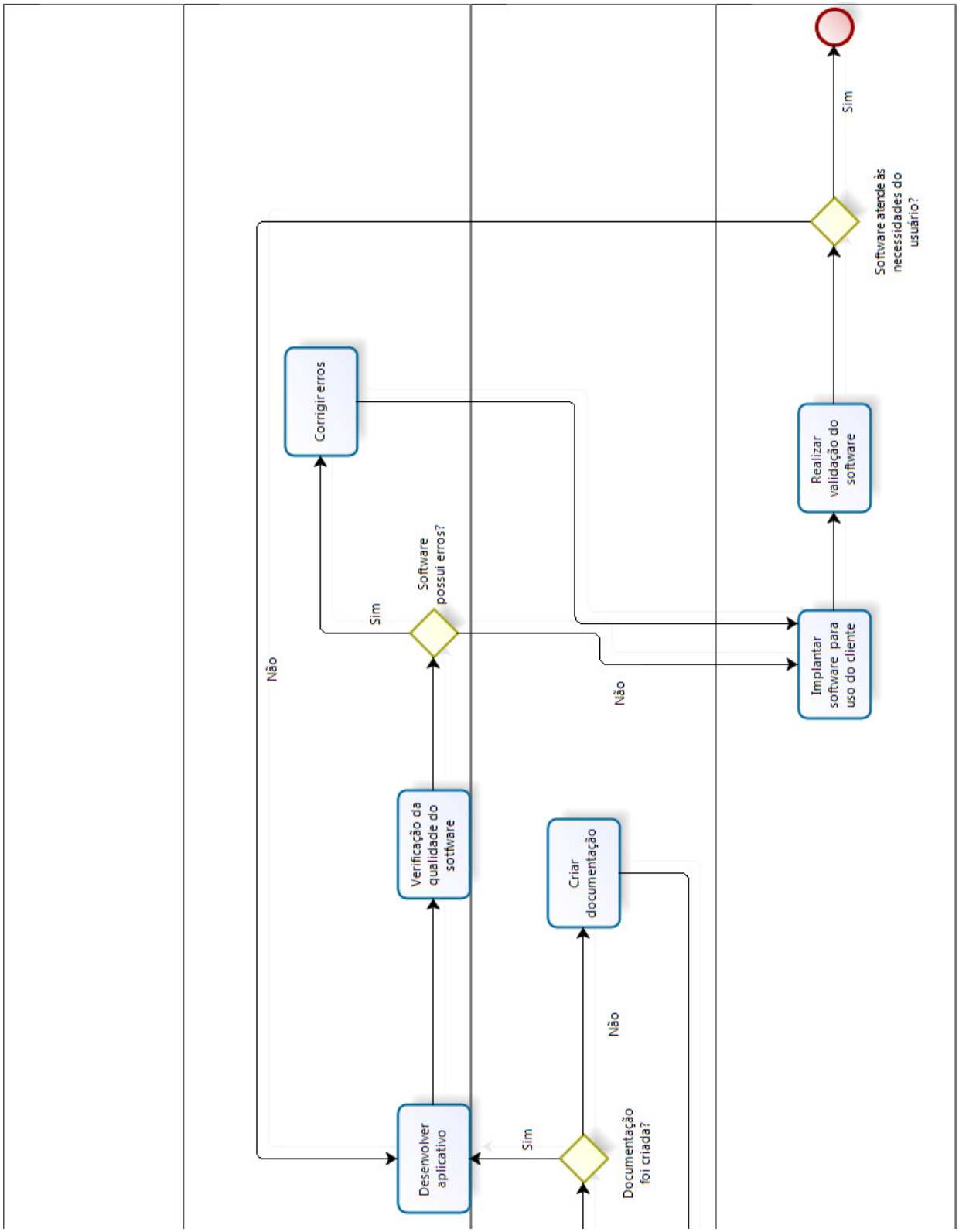
A próxima atividade do processo baseia-se em verificar a qualidade do aplicativo desenvolvido, para isto é preciso realizar os testes das funcionalidades do mesmo; caso passe o *software* passe com sucesso pelos testes, o aplicativo irá para a fase de implantação; caso contrário deverá então ser corrigido os erros de implementação para que então seja realizada a fase de implantação.

A fase de implantação e validação do *software* consiste em instalar o software na máquina do cliente, e então é feita a validação do software. Caso o software passe no teste de validação, o processo é encerrado com sucesso; caso contrário, o software deve voltar para a fase de desenvolvimento.

Para que se que o processo seja visto mais claramente, foi desenvolvido um diagrama de processo chamado de Swimlane que será exibido na Figura 41.

O diagrama *Swimlane* é uma implementação *Business Process Model and Notation* (BPMN) de ordem cronológica voltada para a área de negócio de um projeto, sua

representação é feita através de raias de piscina, que por sua vez representam cada departamento em que cada atividade deverá ser executada.



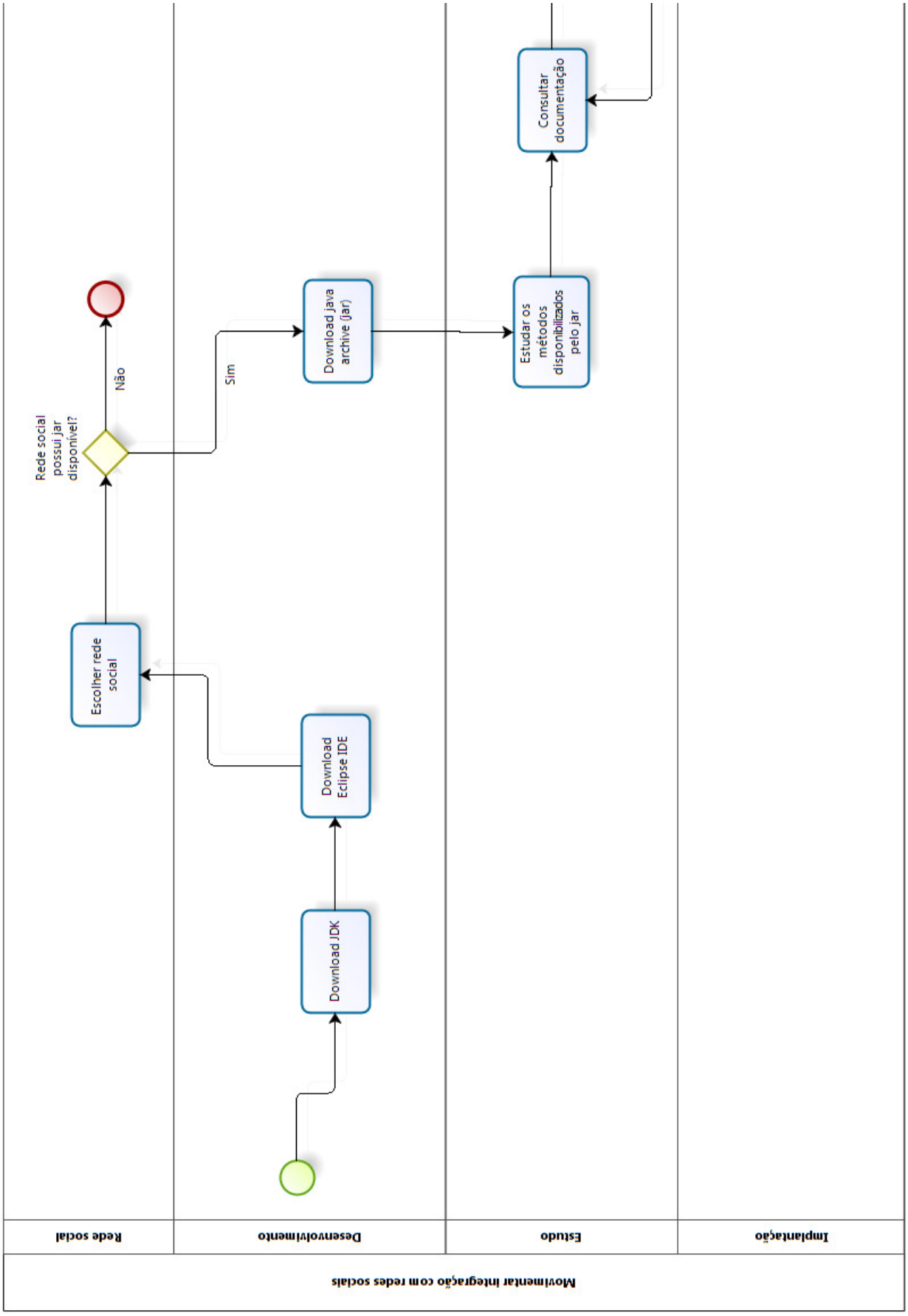


Figura 41 – Diagrama Swimlane

5. ESTRUTURA DO PROJETO

Neste capítulo será exibida a estrutura em etapas do desenvolvimento do sistema de *SRM* através da estrutura analítica de trabalho aqui ilustrada, bem como a duração de execução de cada uma das atividades realizadas, e, além disto também é possível que haja melhor entendimento com relação ao orçamento do projeto.

5.1 WORK BREAKDOWN STRUCTURE (WBS)

O *Work Breakdown Structure* (WBS) é definido como o processo de subdivisão hierárquica do trabalho em componentes para gerenciamento mais fácil, tendo como objetivo principal identificar elementos terminais, como produtos, serviços e resultados a serem realizados em um projeto, fornecendo uma base para a maior parte do planejamento de projetos (RUGGIERI, 2010).

Para ilustrar as fases e etapas da metodologia adotada, a Figura 42 apresenta o diagrama de *WBS*, visando à organização do trabalho e a orientação dos resultados desejados:

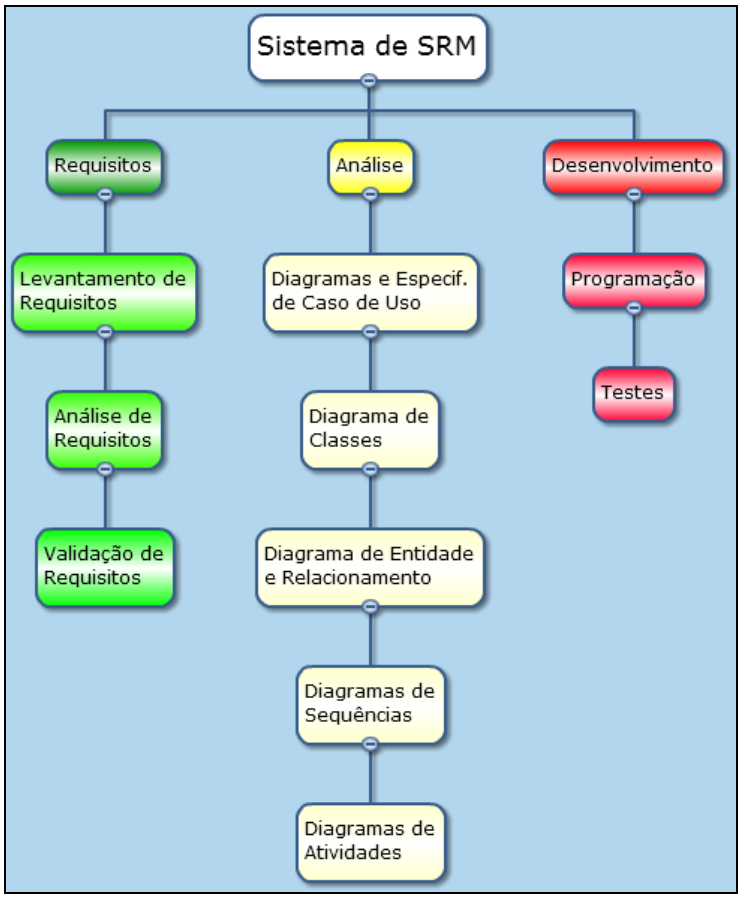


Figura 42 – Estrutura Work Breakdown

5.2 SEQUENCIAMENTO DE ATIVIDADES

Para um melhor gerenciamento de tempo e preço do projeto, foi desenvolvido um diagrama de sequenciamento, no qual demonstra as atividades que serão realizadas, bem como sua duração.

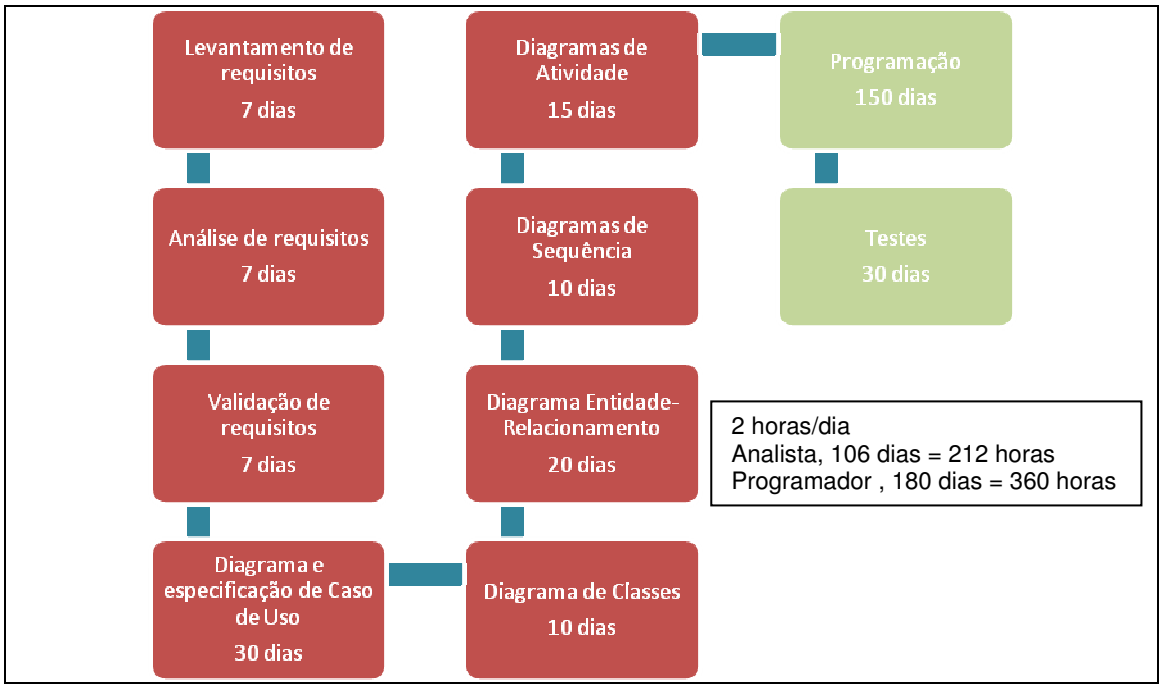


Figura 43 – Diagrama de sequenciamento de atividades

5.3 ORÇAMENTO

Para o desenvolvimento deste trabalho, será necessário a utilização os seguintes recursos e artefatos:

- 01 Analista de Sistemas
- 01 Programador Java
- 01 Notebook
- 01 Impressora

Analista de Sistemas			
Analista	Quantidade Horas	Valor Hora	Total
Priscila Rodrigues da Silva	212	R$ 25.00	R$ 5,300.00
TOTAL			R$ 5,300.00

Tabela 17 – Valor do Analista

Desenvolvedor			
Analista	Quantidade Horas	Valor Hora	Total
Priscila Rodrigues da Silva	360	R$ 25.00	R$ 9,000.00
TOTAL			R$ 9,000.00

Tabela 18 – Valor do Desenvolvedor

Equipamentos				
Equipamento	Valor (inicial)	Vida Útil	Total	Valor (final)
Notebook	R$ 2,500.00	R$ 3.70	R$ 1,332.00	R$ 1,168.00
Impressora	R$ 500.00	R$ 0.65	R$ 234.00	R$ 266.00
TOTAL			R$ 1,566.00	

Tabela 19 – Valor do equipamento

Softwares			
Software	Quantidade Licenças	Valor Licença	Total
NetBeans IDE 7.2.1	1	R$ -	R$ -
PostgreSQL v. 9.1	1	R$ -	R$ -
Astah	1	R$ -	R$ -
Windows Seven	1	R$ 750.00	R$ 750.00
TOTAL			R$ 750.00

Tabela 20 – Valor do Software

Valor Total do Projeto		
Mão de Obra	R$	14,300.00
Equipamentos	R$	1,566.00
Softwares	R$	750.00
TOTAL	R$	16,616.00

Tabela 21 – Valor Total do Projeto

6. IMPLEMENTAÇÃO DO SISTEMA

A implementação do projeto foi feita utilizando a plataforma *Java*, com os *frameworks JPA* 2.0, com a implementação *EclipseLink* e *JSF* 2.1 com *PrimeFaces* 3.5, juntamente com o ambiente de desenvolvimento *Netbeans* 7.2.1 e banco de dados *PostgreSQL* 9.1.

6.1 ORGANIZAÇÃO DO PROJETO

A organização do projeto foi feita através de pacotes, onde cada pacote tem uma função específica no sistema:

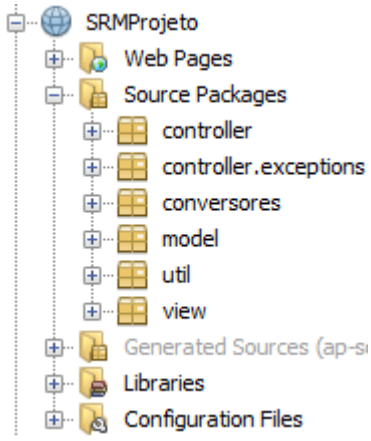


Figura 44 - Estrutura de pacotes do sistema

Pacote ***controller***: pacote que faz o controle entre as classes dos pacotes *model* e *view*. Nesta classe serão lançadas as exceções presentes no pacote *controller.exceptions*.

Pacote ***controller.exceptions***: pacote que contém as exceções para serem utilizadas posteriormente pelo pacote *controller*.

Pacote ***conversores***: pacote requerido pelo *framework JSF* para transformar objeto em texto, e texto em objeto.

Pacote ***model***: pacote em que estão as classes que representam as tabelas no banco de dados, ou entidades do sistema

Pacote ***view***: pacote onde se encontram os *managed bean*, classes *Java* que são utilizadas para interagir com as páginas *JSF* do sistema.

Além dos pacotes que possuem as classes *Java*, há uma pasta na qual contém todas as páginas do sistema, esta nomeada "*Web Pages*". Dentro desta pasta há além das páginas XHTML, outras pastas que possuem arquivos diversos, como css, imagens, e XML*s* responsáveis pela configuração de alguns comportamentos do sistema.

6.2 PRINCIPAIS CLASSES DO SISTEMA

Dentre todas as classes que compõe o sistema, algumas principais são importantes destacar:

SocialUtil.java – Classe responsável por prover a conexão com as redes sociais *Facebook* e *Twitter*.

As variáveis desta classe são variáveis de autenticação providas pelos aplicativos criados, em ambas as redes sociais. Esta classe foi feita seguindo o padrão que é disponível nos *sites* em que as *APIs* foram baixadas: twitter4j.org/en/code-examples.html e facebook4j.org/en/index.html#source_code.

É preciso também destacar que o sistema é construído para ser utilizado via Web, portanto é necessário que seja criada uma classe *Singleton* para a criação das *factory* das redes sociais – requeridas para que todo acesso à rede social seja realizado –. É preciso criar esta classe específica pois muitas pessoas podem acessar o sistema ao mesmo, e sempre gerar esta *factory* requer grande espaço em memória.

Ao invés de a criar sempre quando for preciso, simplesmente desenvolvemos uma classe que continuadamente ficará ativa na memória utilizando o tipo *static*, juntamente com seus atributos também do tipo *static*, garantindo assim apenas uma instância dela esteja em memória, e a chamando sempre quando necessário; sendo este o conceito do Padrão de projeto *Singleton*.

```
package util;

// imports omitidos
```

```
public class SocialUtil {

    // ######################    FACEBOOK     ######################
    private static final String appId = "6820512351571";
    private static final String appSecret = "9bde012d3062350ff156541e";
    private static final String commaSeparetedPermissions =
"manage_pages,publish_stream,read_stream";
    private static final String accessTokenFb = "CAAJsUyZCBUBAIMz9fGHclS80SUiZ8";

    public static Facebook getFacebook() {
        Facebook facebook = new FacebookFactory().getInstance();
        facebook.setOAuthAppId(appId, appSecret);
        facebook.setOAuthPermissions(commaSeparetedPermissions);
        facebook.setOAuthAccessToken(new AccessToken(accessTokenFb, null));
        return facebook;
    }
    // ######################    TWITTER     ######################

    private static final String consumerKey = "Wwx4xEPruS0QDzLw";
    private static final String consumerSecret = "Bvma4hCCZheB2E7nXuFnkkooeNX0nE";
    private static final String accessToken = "18737118435-tfHUKKQIfZHtgXmiQ8Ff";
    private static final String tokenSecret = "9VebLQRmppLiAAAevUu90EbvZ19rpw";

    public static Twitter getTwitter() {
        ConfigurationBuilder builder = new ConfigurationBuilder();
        builder.setDebugEnabled(true);

        builder.setOAuthConsumerKey(consumerKey);
        builder.setOAuthConsumerSecret(consumerSecret);
        builder.setOAuthAccessToken(accessToken);
        builder.setOAuthAccessTokenSecret(tokenSecret);
        Configuration configuration = builder.build();
        TwitterFactory factory = new TwitterFactory(configuration);
        Twitter twitter = factory.getInstance();
        return twitter;
    }
}
```

JPAUtil.java – Classe *Singleton* responsável pela criação da *entity manager factory.*

A *entity manager factory* é a interface utilizada necessária para a criação de uma conexão com o banco de dados. Para criá-la, primeiramente é preciso configurar o arquivo *persistence.xml*, que é criado assim que o *JPA* é marcado para ser utilizado no projeto.

O arquivo *pesistence.xml* contém todos os dados necessários para que a conexão seja feita, tais como, classes a serem persistidas, *driver* de conexão, url de conexão, usuário, senha e outros. Este é um modo de realizar a conexão com o banco de

dados, há também modo, por injeção de dependências, a qual não foi feita no projeto.

```
package util;

// imports omitidos

public class JPAUtil {
    private static final EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("SRMProjectPU");

    public static EntityManager getEntityManager() {
        return emf.createEntityManager();
    }

    public static EntityManagerFactory getEmf() {
        return emf;
    }
}
```

SocialMB.java – *Managed Bean* utilizado para o controle da página XHTML, na qual é responsável pela integração com as redes sociais *Facebook* e *Twitter*.

As classes *Java* do tipo *Managed Bean* são padrão para o *framework JSF*, requeridas para que os dados possam ser capturados e manipulados pelo usuário na tela, a partir disto é possível que operações como criar, excluir, exibir e alterar sejam feitas.

Para declarar que uma classe *Java* é também um *Managed Bean*, é preciso anotá-la com @ManagedBean e uma outra anotação na qual diz que tipo de escopo aquela página terá, o que interfere no tempo de vida de todos os objetos que estão neste escopo. Neste caso, o escopo utilizado é o de sessão, explicitado a partir da anotação @SessionScoped.

Na versão 2.0 do *JSF*, além do @SessionScoped, há outros escopos, tais como, @RequestScoped, @ApplicationScoped, @ViewScoped.

```
package view;

// imports omitidos

@ManagedBean
@SessionScoped
public class SocialMB {

    //retorna lista de posts buscados
    private List<Postagem> posts;
    //faz conexão com o controller para salvar a postagem no banco
```

```
    private PostagemDAO postagemJPA;
    //seta cada objeto da lista
    private Postagem postagem;
    //objeto que dá a palavra chave para busca
    private Palavrachaveredesocial palavra;
    //variável de referência no xhtml
    private String post;
    //detalhes de cada post
    private Postagem postDet;
    //verifica se a checkbox foi selecionada para a busca
    private boolean twitter;
    private boolean facebook;
    //verifica se a checkbox foi selecionada para a postagem
    private boolean twitterPost;
    private boolean facebookPost;

    public SocialMB() {
        this.postagem = new Postagem();
        this.post = new String();
        this.postagem = new Postagem();
        this.posts = new ArrayList<Postagem>();
        this.postDet = null;
        this.palavra = new Palavrachaveredesocial();
        this.postagemJPA = new PostagemJpaController(JPAUtil.getEmf());
    }

    // ################## Método de postagem nas redes sociais ##################
    public void postar(String post) {
        System.out.println("Salvando post " + post);
        FacesMessage.Severity severity = FacesMessage.SEVERITY_INFO;
        String message = null;
        //teste verifica se a checkbox do twitter está selecionada
        //e se a quantidade de caractéres está dentro de 140
        if (this.twitterPost && this.post.length() <= 140) {
        //caso dê algum erro no bloco “try”, o bloco “catch” é executado
            try {
                // linha abaixo é responsável pela postagem do post no twitter
                Status status = SocialUtil.getTwitter().updateStatus(post);
                message = post + " - Postado com sucesso no Twitter";
                System.out.println("Mensagem postada com sucesso!");
                severity = FacesMessage.SEVERITY_INFO;
                this.post = new String();
            } catch (TwitterException e) {
                message = "Ops, algum erro ocorreu.";
                severity = FacesMessage.SEVERITY_ERROR;
                e.printStackTrace();
            }
        } else {
            message = "A quantidade máxima de caracteres para o Twitter é de 140";
            severity = FacesMessage.SEVERITY_WARN;
        }
        FacesMessage msg = new FacesMessage(severity, message, null);
        FacesContext.getCurrentInstance().addMessage(null, msg);
        //linha abaixo verifica se a checkbox do
        //facebook para postagem está selecionada
        if (this.facebookPost) {
            try {
                //linha abaixo posta a mensagem digitada no social.xhmtl no
                //facebook
```

```
                SocialUtil.getFacebook().postStatusMessage(post);
                message = post + " - Postado com sucesso no Facebook";
                System.out.println("Mensagem postada com sucesso!");
                severity = FacesMessage.SEVERITY_INFO;
                this.post = new String();
            } catch (Exception e) {
                message = "Ops, algum erro ocorreu.";
                severity = FacesMessage.SEVERITY_ERROR;
                e.printStackTrace();
            }
        //linha abaixo verifica se ao menos uma checkbox de postagem foi
        //selecionada
        } else if (this.facebookPost == false && this.twitterPost == false) {
            message = "Selecione ao menos uma rede social";
            severity = FacesMessage.SEVERITY_WARN;
        }
        FacesMessage msg2 = new FacesMessage(severity, message, null);
        FacesContext.getCurrentInstance().addMessage(null, msg2);
    }

    // ################## Método de pesquisa em redes sociais #################
    public void search(Palavrachaveredesocial palavra) {
        FacesMessage.Severity severity = FacesMessage.SEVERITY_INFO;
        String message = null;
        try {
            if (this.facebook) {
                // new Reading().limit(100) obtém 1st-100th resultados
                ResponseList<Post> posts =
SocialUtil.getFacebook().searchPosts(palavra.getPalavrachave(), new
Reading().limit(100));
                // loop que pega cada postagem que foi obtida pela linha anterior
                for (Post post : posts) {
                    this.postagem.setDatahora(post.getCreatedTime());
                    this.postagem.setConteudo(post.getMessage());
                    this.postagem.setLocalizacao("Facebook: " +
post.getFrom().getName());
                    this.posts.add(this.postagem);
                // linha abaixo grava cada postagem recebida na busca no banco
                    this.postagemJPA.create(this.postagem);
                    message = "Resultados com: " + palavra.getPalavrachave();
                }
            }
            if (this.twitter) {
                //query recebe a palavra chave que foi passada no xhtml
                Query query = new Query(palavra.getPalavrachave());
                //definição de quantos tweets serão retornados
                query.setCount(100);
                //result recebe as autorizações do twitter junto com a palavra a
                //ser buscada
                QueryResult result = SocialUtil.getTwitter().search(query);
                for (Status status : result.getTweets()) {
                    //se a lingua for "pt" então é gravada
                    if (status.getUser().getLang().equals("pt")) {
                        this.postagem.setConteudo(status.getText());
                        this.postagem.setDatahora(status.getCreatedAt());
                        this.postagem.setLocalizacao("Twitter: " +
status.getUser().getLocation());
                        this.posts.add(this.postagem);
                        this.postagemJPA.create(this.postagem);
```

```
                        message = "Resultados com: " + palavra.getPalavrachave();
                    }
                }
            } else if (this.facebook == false && this.twitter == false) {
                message = "Selecione ao menos uma rede social";
                severity = FacesMessage.SEVERITY_ERROR;
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        FacesMessage msg = new FacesMessage(severity, message, null);
        FacesContext.getCurrentInstance().addMessage(null, msg);
    }

    // ###########  Método que mostra os detalhes de cada postagem  ###########
    public void detalhe(Postagem post) {
        System.out.println("Vendo detalhes do post " + post.getLocalizacao());
        this.postDet = new Postagem();
        this.postDet.setConteudo(post.getConteudo());
        this.postDet.setIdpostagem(post.getIdpostagem());
        this.postDet.setDatahora(post.getDatahora());
        this.postDet.setLocalizacao(post.getLocalizacao());
    }

    // ############## Método que configura o relatório ##############
    public void preProcessPDF(Object document) throws IOException,
BadElementException, DocumentException {
        Document pdf = (Document) document;
        pdf.setPageSize(PageSize.A4);
        pdf.open();
        //aqui pega o contexto para formar a url da imagem
        ServletContext servletContext = (ServletContext)
FacesContext.getCurrentInstance().getExternalContext().getContext();
        String logo = servletContext.getRealPath("") + File.separator + "img" +
File.separator + "sociallogo.png";

        //adiciona a img ao pdf
        pdf.add(Image.getInstance(logo));
        //adiciona um paragrafo ao pdf, alinha também ao centro
        Paragraph p = new Paragraph("Relatório de postagem");
        p.setAlignment("center");
        p.spacingAfter();
        pdf.add(p);
        p.spacingAfter();
    }

    // ################ GETTERS AND SETTERS ################
    public List<Postagem> getPosts() {
        return this.posts = this.postagemJPA.findPostagemEntities();
    }
    // ################ GETTERS AND SETTERS PADRÕES OMITIDOS ################
}
```

Social.xhtml – Página *XHTML* responsável pela integração com as redes sociais *Facebook* e *Twitter*.

Esta página, assim como todas as outras, está dentro da pasta nomeada "*Web Pages*", está página faz a relação com o Managed Bean chamado SocialMB. Aqui é onde o *PrimeFaces* é utilizado através da tag prefixa 'p', tornando assim os componentes ricos.

Trecho em que o usuário posta conteúdos nas redes sociais:

```
<p:panel id="panelPost" header="Postar nas Redes Sociais" toggleable="true"
toggleOrientation="horizontal" >
                <h:panelGrid columns="6">
                    <p:inputTextarea autoResize="false" value="#{socialMB.post}"
cols="120" rows="4" maxlength="63000" counter="counter" counterTemplate="{0}
caracteres restando."/>
                    Facebook
                    <p:selectBooleanCheckbox value="#{socialMB.facebookPost}" />
                    Twitter
                    <p:selectBooleanCheckbox value="#{socialMB.twitterPost}" />
                    <p:commandButton type="submit" value="Postar"
action="#{socialMB.postar(socialMB.post)}" icon="ui-icon-comment" />
                    <h:outputText id="counter" />
                </h:panelGrid>

</p:panel>
```

Tags importantes deste trecho:

- p:inputTextarea = componente *PrimeFaces* que cria um campo, permite ao usuário a inserção de algum conteúdo a ser posteriormente manipulado pela classe Managed Bean.

- p:selectBooleanCheckbox = componente *PrimeFaces* que exibe uma caixa de seleção, na qual é verificada posteriormente se a mesma foi ou não selecionada, para que assim alguma ação seja ocorrida.

- p:commandButton = componente *PrimeFaces* que cria um botão, na qual seguidamente ações serão realizadas. Neste caso a ação que acontece será executar o método nomeado como "postar", localizado na classe SocialMB, passando como parâmetro o valor que foi digitado anteriormente para a variável nomeada como "post", no p:inputTextarea.

Trecho em que as postagens são retornadas e exibidas para o usuário:

```
<p:panel id="panelResultTw" header="Lista de posts" toggleable="true"
toggleOrientation="horizontal">
      <p:dataTable value="#{socialMB.posts}" var="post" paginator="true" rows="20"
paginatorTemplate="{CurrentPageReport}  {FirstPageLink} {PreviousPageLink}
{PageLinks} {NextPageLink} {LastPageLink} {RowsPerPageDropdown}"  id="posts"
paginatorPosition="bottom" rowsPerPageTemplate="20,25,30" resizableColumns="true"
emptyMessage="Não há posts com esta palavra-chave" >
```

```
          <p:column filterBy="#{post.conteudo}" sortBy="#{post.conteudo}"
filterMatchMode="contains">
              <f:facet name="header">
                 <h:outputText value="Conteudo" />
              </f:facet>
              <h:outputText value="#{post.conteudo}" />
           </p:column>
           <p:column filterBy="#{post.localizacao}" sortBy="#{post.localizacao}">
               <f:facet name="header">
                  <h:outputText value="Localizacao" />
               </f:facet>
               <h:outputText value="#{post.localizacao}" />
            </p:column>
            <p:column filterBy="#{post.datahora}" sortBy="#{post.datahora}">
                <f:facet name="header">
                  <h:outputText value="Data" />
                </f:facet>
                <h:outputText value="#{post.datahora}" >
                   <f:convertDateTime pattern="dd/MM/yyyy - HH:mm:ss" />
                </h:outputText>
             </p:column>
             <p:column style="width:10%;text-align:center" headerText="Detalhes"
exportable="false">
                <p:commandLink update=":frm3:txtPost,:frm3:txtLoc, :frm3:txtData"
action="#{socialMB.detalhe(post)}">
                  <h:graphicImage value="/img/info.png" title="Detalhes"
onclick="dlgDet.show();"/>
                </p:commandLink>
              </p:column>
       </p:dataTable>
</p:panel>
```

Tags importantes deste trecho:

- p:panel = componente *PrimeFaces* de agrupamento de conteúdo que utiliza AJAX, permite a integração com menus, e é utilizado na página social.xhmtl para diferenciar o agrupamento entre a postagem e a pesquisa nas redes sociais.
- p:dataTable = componente *PrimeFaces* cria uma tabla na qual fornece soluções embutidas para muitos casos comuns como paginação, classificação, seleção, e filtragem de dados.
- p:column = *tag* utilizada para criar uma coluna dentro de uma tabela.

Trecho para inserção da palavra-chave para que o conteúdo seja posteriormente recuperado nas redes sociais:

```
<h:panelGrid columns="2">
       <h:panelGroup>
          <h:panelGroup>
```

```
          Palavra-chave:
             <p:inputText value="#{socialMB.palavra.palavrachave}" size="50"
maxlength="50"/>
          </h:panelGroup>
          <h:panelGroup>
             Facebook
             <p:selectBooleanCheckbox value="#{socialMB.facebook}" />
             Twitter
             <p:selectBooleanCheckbox value="#{socialMB.twitter}" />
             <p:spacer width="10px"/>
             <p:commandButton type="submit" icon="ui-icon-search" value="Pesquisar"
action="#{socialMB.search(socialMB.palavra)}" update=":form1:posts"/>
          </h:panelGroup>
       </h:panelGroup>
</h:panelGrid>
```

Tags importantes deste trecho:

- p:panelGrid = componente *PrimeFaces* utilizado para colocar os componentes que estão dentro desta *tag* em linhas e colunas.
- h:panelGroup = componente *JSF* utilizado para criar separação entre os componentes que estão dentro desta *tag*.

Para exemplificar a utilização do JPA 2.0 no projeto, foi escolhida a classe Turma, que está presente no pacote nomeado *model*.

```
package model;

// imports omitidos

@Entity
@Table(name = "turma")
@XmlRootElement
public class Turma implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "idturma")
    private Integer idturma;
    @Column(name = "sala")
    private String sala;
    @Column(name = "ano")
    private Integer ano;
    @Column(name = "serie")
    private Integer serie;
    @Column(name = "observacoes")
    private String observacoes;
    @Column(name = "notamaxima")
    private BigDecimal notamaxima;
    @Column(name = "nota")
    private BigDecimal nota;
```

```
    @Column(name = "possuiexame")
    private Boolean possuiexame;
    @Column(name = "notaexame")
    private BigDecimal notaexame;
    @Column(name = "qtdaluno")
    private Integer qtdaluno;
    @ManyToMany(mappedBy = "turmaList")
    private List<Disciplina> disciplinaList;
    @ManyToMany(mappedBy = "turmaList")
    private List<Professor> professorList;
    @JoinColumn(name = "turno_idturno", referencedColumnName = "idturno")
    @ManyToOne(optional = false)
    private Turno turnoIdturno;
    @JoinColumn(name = "curso_idcurso", referencedColumnName = "idcurso")
    @ManyToOne(optional = false)
    private Curso cursoIdcurso;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "turmaIdturma")
    private List<Aluno> alunoList;

    public Turma() {
    }
    // getters, setters, equals, hashCode e toString omitidos
}
```

Anotações *JPA* importantes:

@Entity: Informa que a classe é também um entidade, ou seja, uma tabela no banco de dados.

@Table: Utilizada para dar um nome à tabela no banco de dados através do parâmetro "*name*".

@Column: Cria ou identifica uma coluna no banco de dados – dependendo da configuração do arquivo *pesistence.xml*, alguns parâmetros são disponíveis para esta anotação, tais como:

- name: identifica o nome da coluna
- unique: identifica se o valor do campo poderá se repetir em outros registros da mesma tabela.
- updatable: identifica se o campo pode ser atualizado.
- insertable: identifica se o campo pode ser inserido.

@Id: Identifica a chave primária da entidade.

@GeneratedValue: Gerador automático de chave primária pelo banco de dados.

@Basic: Permite a declaração de um tipo de busca, eager ou lazy. Por padrão, o valor assumido é eager.

@JoinColumn: Parâmetros requeridos:

- name: nome do campo de chave estrangeira criado para armazenar o id da tabela relacionada
- referencedColumnName: nome da chave primária da tabela referenciada na chave estrangeira.

@ManyToOne: Identifica que haverá uma única instância referenciada do objeto estrangeiro anotado no banco de dados. Do outro lado desta anotação haverá uma lista de objetos da classe mapeada por @ManyToOne. Nesta outra classe, a lista estará mapeada por @OneToMany.

@OneToMany: É a anotação contrária à @ManyToOne, aqui indica que haverá uma lista de objetos da classe que foi mapeada por @ManyToOne.

6.3 INTERFACES DO SISTEMA

É necessário frisar que, todas as interfaces do sistema foram construídas seguindo um padrão, para isto, anteriormente foi preciso que as classes que serão manipuladas nas telas passassem por várias camadas.

Inicialmente foi necessário criar uma entidade, isto acontece no pacote nomeado *model*. Posteriormente é preciso desenvolver uma classe na qual faz a intermediação entre a classe criadas no pacote *model* e no pacote *view,* para isto é criada a classe *Java* presente no pacote *controller*.

Classes presentes neste pacote realizam regras de negócios, como verificações de integridade do banco de dados, como por exemplo, não é permitido excluir um país caso algum estado o esteja utilizando.

Tela de integração com as redes sociais: Esta tela é o resultado da página social.xhml. Aqui é possível realizar as integrações com as redes sociais *Facebook* e *Twitter*, através da realização de buscas de posts a partir de uma palavra chave, bem como a postagem nas redes sociais.

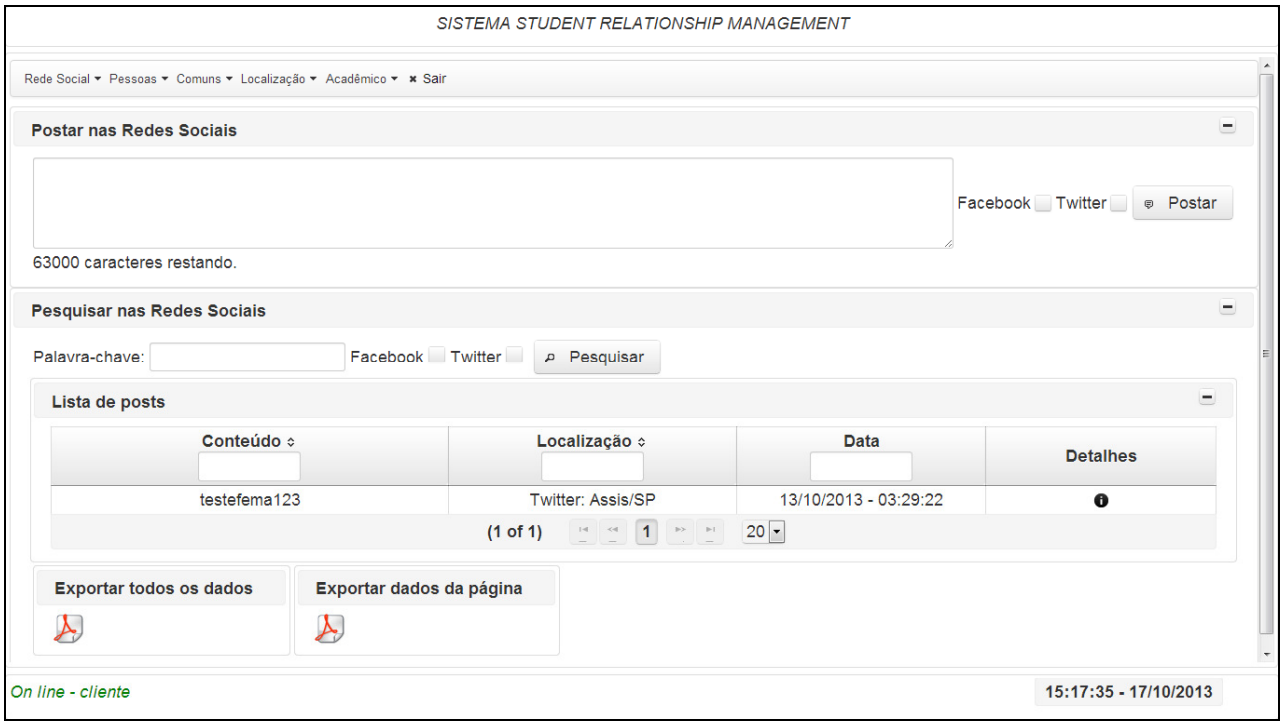


Figura 45 – Tela de integração com as redes sociais

Todos as postagens retornadas a partir da pesquisa nas redes sociais são persistidas no banco de dados, para que posteriormente estes dados sejam estudados e alguma tomada de decisão seja assumida.

Para postar alguma mensagem na rede social, o InputTextarea irá receber os dados para posterior postagem, e então é preciso selecionar em qual rede social a mensagem será postada, através das *checkboxes* localizadas no lado direito da tela, e então clicar no botão “Postar”. Caso não haja nenhuma *checkboxes* selecionada, o sistema envia uma mensagem de alerta; caso a mensagem ultrapasse 140 caracteres, e a *checkboxes* do Twitter esteja selecionada, o sistema envia mensagem de alerta, dizendo que a quantidade máxima de caracteres para o *Twitter* é de 140.

O inputText indicado após a texto “palavra-chave” é utilizado para capturar a palavra e/ou frase para então ser realizada a pesquisa nas redes sociais. Após o campo ser preenchido é necessário selecionar alguma das *checkboxes* a seguir para escolher em qual/is redes sociais será aplicada a palavra-chave para busca. Seguidamente as postagens relacionadas à esta palavra-chave são exibidas na *data table* abaixo,

e, como é possível haver várias pesquisas, foi criado um campo de pesquisa no próprio componente, a fim de facilitar a localização de postagens.

Tela de cadastro, alteração, exclusão e consulta de alunos: Tela onde é possível manter as informações dos alunos da instituição de ensino.

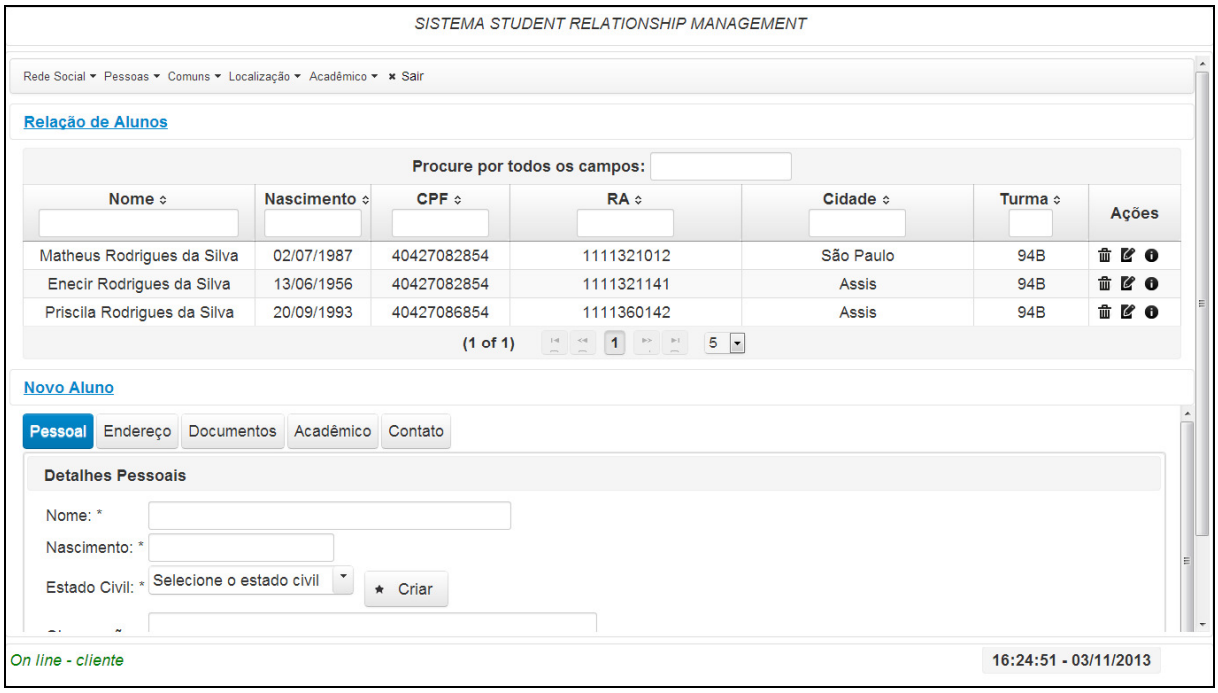


Figura 46 – Tela manter alunos

Atributos obrigatórios para cadastro de alunos:

- Nome: Nome completo do aluno.
- Nascimento: Data de nascimento do aluno, contendo dia, mês e ano.
- Estado civil: Estado civil do aluno.
- Cidade: Cidade do aluno.
- CPF: CPF do aluno.
- Turma: Turma em que o aluno está inserido.
- Formação Acadêmica: Formação Acadêmica do aluno.
- E-mail: Correio eletrônico do aluno.

Tela de cadastro, alteração, exclusão e consulta de professores: Tela onde é possível manter as informações dos professores da instituição de ensino.

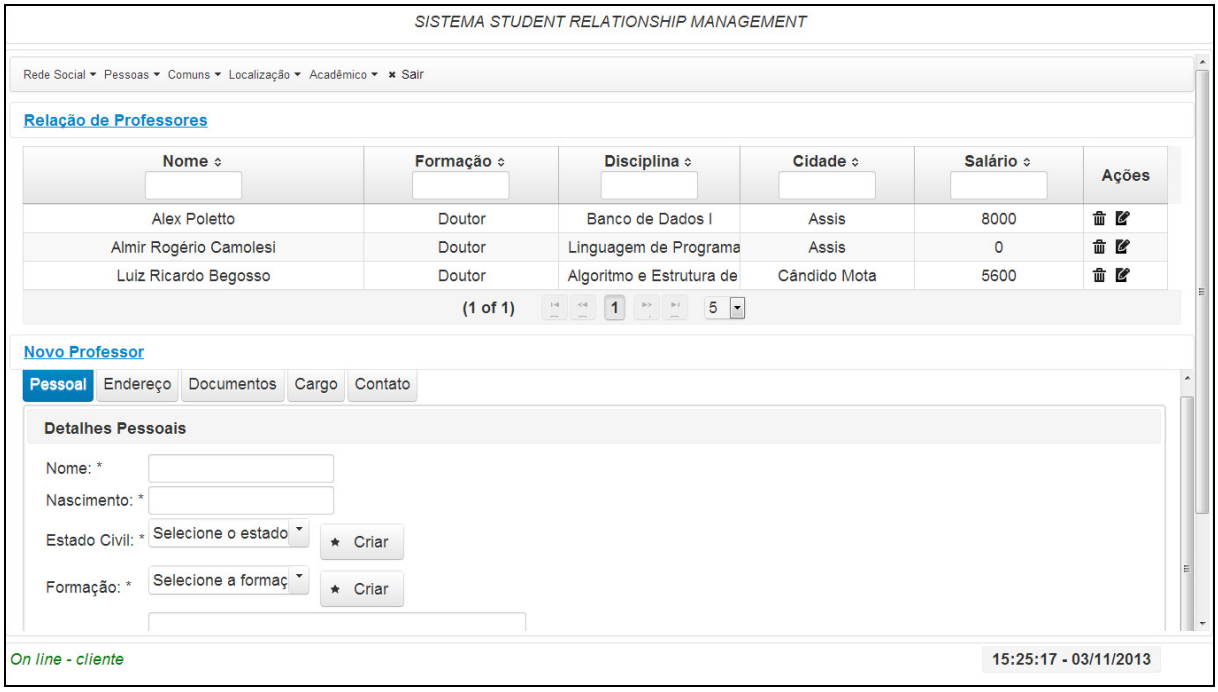


Figura 47 - Tela manter professores

Atributos obrigatórios para cadastro de professores:

- Nome: Nome completo do professor.
- Nascimento: Data de nascimento do professor, contendo dia, mês e ano.
- Estado civil: Estado civil do professor.
- Formação Acadêmica: Formação Acadêmica do professor.
- Cidade: Cidade do professor.
- CPF: CPF do professor.
- Disciplina: Disciplina que o professor leciona.
- E-mail: Correio eletrônico do professor.

Tela de cadastro, alteração, exclusão e consulta de turmas: Tela onde é possível manter as informações das turmas da instituição de ensino.

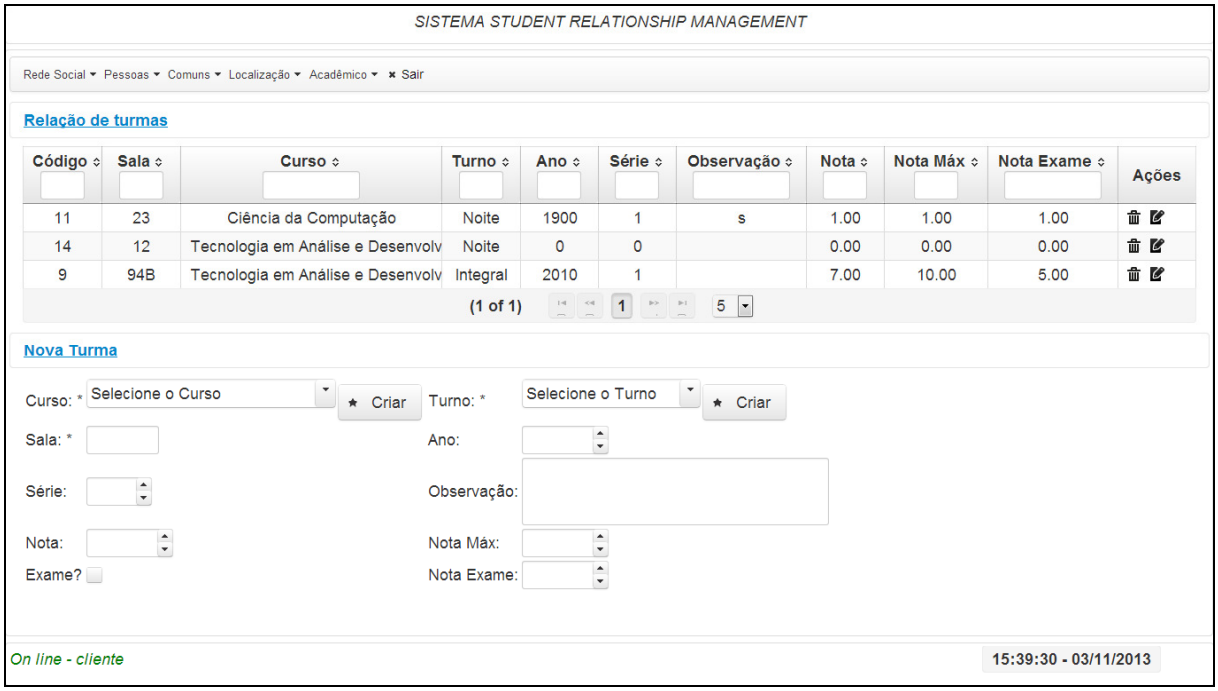


Figura 48 - Tela manter turmas

Atributos obrigatórios para cadastro de turmas:

- Curso: Nome do curso da turma.
- Sala: Identificação da sala de aula da turma na instituição de ensino.
- Turno: Identificação do turno em que a turma participa das aulas.

Tela de cadastro, alteração, exclusão e consulta de departamentos: Tela onde é possível manter as informações das turmas da instituição de ensino.

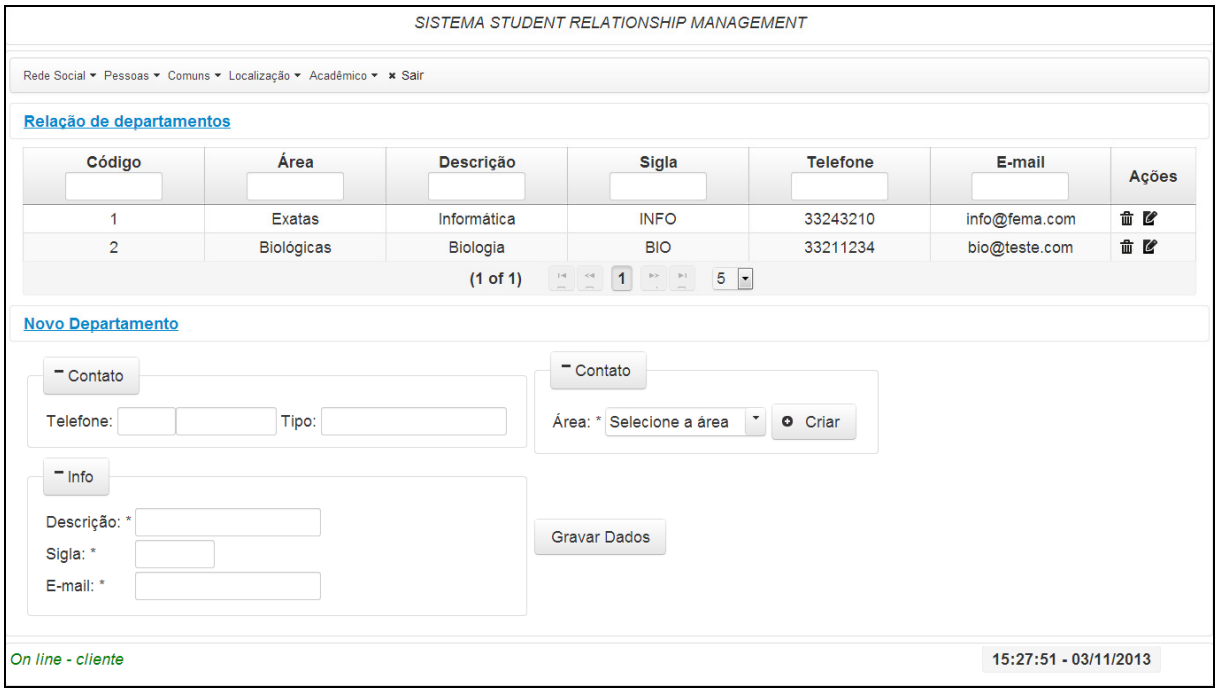


Figura 49 - Tela manter departamentos

Atributos obrigatórios para cadastro de departamentos:

- Descrição: Nome do departamento da instituição de ensino.
- Sigla: Sigla do departamento da instituição de ensino.
- E-mail: Correio eletrônico do departamento.
- Área: Área em que o departamento faz parte.

Tela de cadastro, alteração, exclusão e consulta de cursos: Tela onde é possível manter as informações das turmas da instituição de ensino.

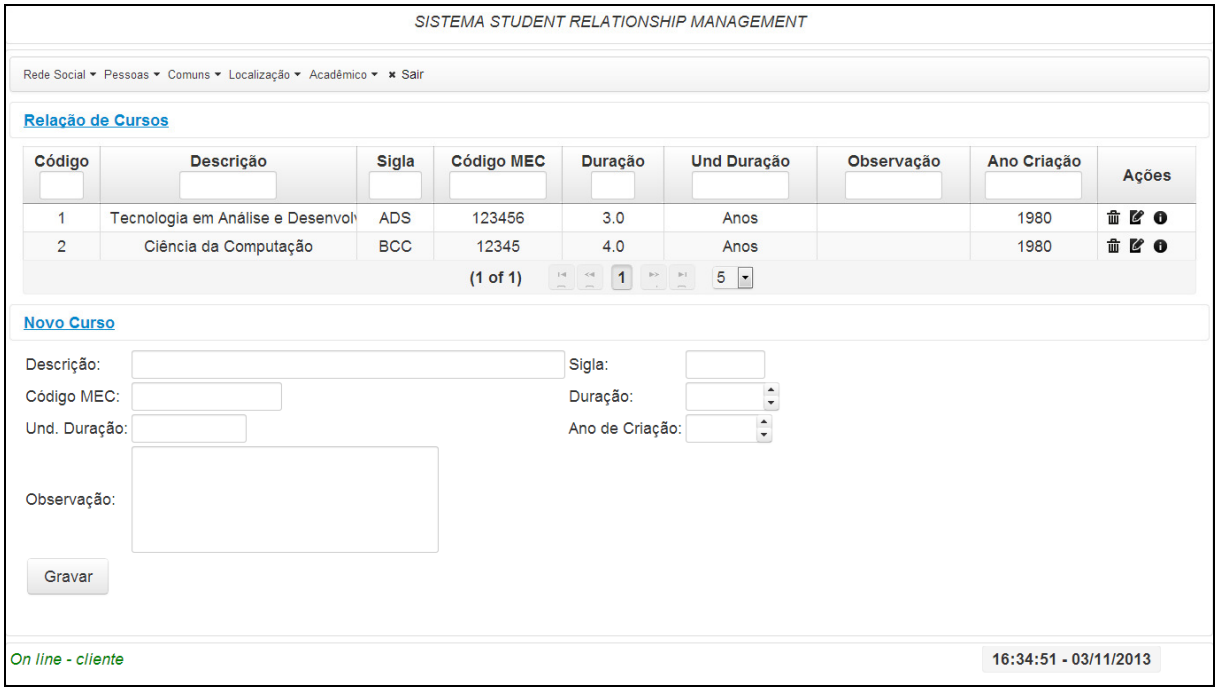


Figura 50 - Tela manter cursos

Atributos obrigatórios para cadastro de cursos:

- Descrição: Nome do curso disponibilizado pela instituição de ensino.
- Sigla: Sigla do nome do curso disponibilizado pela instituição de ensino.
- Código MEC: Código do curso no Ministério da Educação e Cultura.
- Duração: Quantidade semestres ou anos para a conclusão do curso.
- Unidade de duração: Unidade de duração do curso, como semestres ou anos.
- Ano de criação: Ano em que o curso foi criado.

Durante a implementação do projeto houveram alguns problemas devido à falta de experiência com os *frameworks* então conhecidos durante este ano, e já utilizados durante este trabalho, principalmente em relação as anotações *JPA* para a realização da persistência dos dados. Em contra partida, a facilidade em relação ao *JSF* e *PrimeFaces* foi relativamente grande, e, alguns fatores contribuíram para isto,

um dos fatores facilitadores foi por conta do mostruário exibido na página oficial do *PrimeFaces* na qual apresenta o funcionamento de cada componente.

Além da parte visual, o *PrimeFaces* possui também outros vários recursos embutidos, principalmente por parte de pesquisa de dados, suprindo assim por muitas vezes a necessidade de desenvolver códigos complexos envolvendo *JPA*, o qual em alguns momentos carece de métodos simplificados, especialmente em relação à parte de pesquisa de dados, tornando o código um tanto quanto trabalhoso, e, portanto mais passível de erro.

7. CONCLUSÃO

Considerado que a concorrência entre as universidades cresce a cada dia, e que ainda existem falhas e abandono de curso por parte dos alunos, principalmente nos primeiros anos dos cursos de graduação, é necessário criar estratégias para que este relacionamento seja melhor mapeado e de forma constante.

Neste sentido, o trabalho que foi desenvolvido busca atender as expectativas do aluno para com a instituição de ensino, mapeando informações por meio de redes sociais, deste modo utilizando-se de conceitos de *SRM*, e então, direcionar a instituição em relação às premências do aluno para um melhor e duradouro relacionamento.

É necessário ressaltar que o *SRM* é principalmente, uma estratégia de marketing, na qual visa conhecer o aluno em suas necessidades a modo de mantê-lo na instituição de ensino, sendo a tecnologia o meio para a captação de informações para posterior tomada de decisão.

7.1 TRABALHOS FUTUROS

Futuramente, é possível incrementar distintas redes sociais para busca de informações, consequentemente resultando em diferentes tipos de conteúdo, melhorando assim a análise feita a partir dos dados captados. Também como trabalho futuro pode-se realizar uma pesquisa na qual medirá de forma tangível a eficiência do conceito e das estratégias de *SRM*, podendo-se utilizar o software que foi desenvolvido neste trabalho.

REFERÊNCIAS

ALCÂNTARA, Daniel de Oliveira. **Da improvisação ao SAC – A origem do CRM.** 2003. 126p. Dissertação (Mestrado) – Marketing – Uninove, São Paulo, 2003.

BREAKENRIDGE, Deirdre. **Social Media Definition in a Cloud**. Disponível em: <http://www.deirdrebreakenridge.com/2010/01/social-media-definition-in-a-cloud/>. Acesso em: 27/02/2013.

CAELUM. **Apostila Java para Desenvolvimento Web.** Disponível em: <http://www.caelum.com.br/apostila-java-web/o-que-e-java-ee/> Acesso em: Julho de 2013.

ÇIVICI, **Çağatay. PrimeFaces User's Guide.** 4.5. ed. 2012.

DEVJR. **Tecnologia Java.** Disponível em: <http://devjr.wordpress.com/tecnologia-java/> Acesso em: Julho de 2013.

DEVMEDIA. **Introdução ao Java Micro Edition.** Disponível em: <http://www.devmedia.com.br/introducao-ao-java-micro-edition/3857> Acesso em: Julho de 2013.

GONÇAVES, Edson. **Dominando JavaServer Faces e Facelets utilizando Spring 2.5, Hibernate e JPA**, 1. ed. Rio de Janeiro: Editora Moderna Ltda, 2008.

GREENBERG, Paul. **CRM, Customer Relationship Management na velocidade da luz**: conquista e lealdade de clientes em tempo real na Internet. 1. ed. Rio de Janeiro: Campus, 2001.

GUEDES, Gilleanes T. A. **UML 2 UMA ABORDAGEM PRÁTICA,** 1. ed. São Paulo: Editora Novatec, 2009.

JAVAFREE. Fórum. **Tutorial Java: O que é Java?** Disponível em: <http://javafree.uol.com.br/artigo/871498/Tutorial-Java-O-que-e-Java.html> Acesso em: Julho de 2013.

KOLIFRATH, Jessica. **Using a Mind Map to Brainstorm Article Ideas.** Disponível em: <http://suite101.com/article/using-a-mind-map-to-brainstorm-article-ideas-a146717>. Acesso em: 11/03/2013.

KROENKE, David M. **Banco de Dados: Fundamentos, Projeto e Implementação**. 6. ed. Rio de Janeiro: Editora LTC – Livros Técnicos e Científicos S.A,1999.

KOSTOJOHN, Scoot; JOHNSON, Mathew; PAULEN, Brian. **CRM Fundamentals,** 1. ed. New York: Apress, 2011.

L3 CRM. **O conceito de CRM aplicado às instituições de ensino.** Disponível em: <http://www.l3crm.com.br/universo-crm/ebooks/12/ebook-sobre-srm/>. Outubro de 2012.

LEE, Richard C; TEPFENHART, William M. **UML e C++ – Guia Prático de Desenvolvimento Orientado a Objeto**, 1. Ed. Tradução: Celso Roberto Paschoa, São Paulo: MAKRON, 2001.

LIMA, Adilson da Silva. **UML 2.3 DO REQUISITO À SOLUÇÃO**, 1. ed. São Paulo: Editora Érica, 2012.

MOMJIAN, Bruce; **PostgreSQL: Introduction and Concepts,** 1. ed. Upper Saddle River: Pearson Education, 2000.

ORACLE. Java SE Technologies at a Glance. Disponível em: <http://www.oracle.com/technetwork/java/javase/tech/technologies-135120.html> Acesso em: Julho de 2013.

PEPPERS, Don; ROGERS, Martha; **CRM Series – Marketing 1 to 1. Aumentando o valor de seus clientes com CRM**. São Paulo: 2004. 108 p.

PIEDADE, Maria Beatriz. **Business Intelligence in Higher Education: Enhancing the teaching-learning process with a SRM system.** In: Information Systems and Technologies (CISTI), 2010, Leiria, Portugal.

PIEDADE, Maria Beatriz Guerra. **Business Intelligence no suporte ao conceito e à prática de Student Relationship Management em Instituições de Ensino Superior.** 2011. 267p. Tese (Doutorado) – Escola de Engenharia, Universidade do Minho, Braga, 2011.

ROUSE, Margaret. **One-to-one-marketing (1:1 marketing).** Disponível em: < http://searchcrm.techtarget.com/definition/one-to-one-marketing>. Julho de 2013.

RUGGIERI, Ruggero. **WBS – Uma ferramenta importante para o gerente de projetos.** Disponível em: <http://www.tiespecialistas.com.br/2010/11/wbs-%E2%80%93-uma-ferramenta-importante-para-o-gerente-de-projetos/#.UTA0B6J_CSo>. Acesso em: 01/03/2013.

SANTANA, Otávio Gonçalves de. **Por que Java?** Disponível em: <http://www.devmedia.com.br/por-que-java/20384>. Acesso em: 25/06/2013

SCM & CRM. Arquitectura Informacional. Disponível em: <http://pt.grupo2sii.wikia.com/wiki/CRM> Acesso em: Julho de 2013.

SILVA, Alberto Manuel Rodrigues da; VIDEIRA, Carlos Alberto Escaleira; **UML, Metodologias e Ferramentas CASE,** 1. ed. Porto: Centro Atlântico, 2001.

SILVA, Tamires Alves da. **Sistema web para gestão de eventos utilizando tecnologias Java e Google Maps API.** 2011. 106p. Trabalho de Conclusão de Curso – Fundação Educacional do Município de Assis – FEMA/Instituto Municipal de Ensino Superior de Assis – IMESA.