Renan Vitor Henker Regis

Desenvolvimento do Núcleo do Sistema de Informação Gerencial do Movimento Espírita

Vitória, ES

2018

Renan Vitor Henker Regis

Desenvolvimento do Núcleo do Sistema de Informação Gerencial do Movimento Espírita

Monografia apresentada ao Curso de Ciência da Computação do Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Departamento de Informática

Orientador: Prof. Dr. Vítor E. Silva Souza

Vitória, ES

2018

Renan Vitor Henker Regis
Desenvolvimento do Núcleo do Sistema de Informação Gerencial do Movimento Espírita/ Renan Vitor Henker Regis. – Vitória, ES, 2018-
46 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Vítor E. Silva Souza

Monografia (PG) – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Departamento de Informática, 2018.

1. Palavra-chave1. 2. Palavra-chave2. I. Souza, Vítor Estêvão Silva. II. Universidade Federal do Espírito Santo. IV. Desenvolvimento do Núcleo do Sistema de Informação Gerencial do Movimento Espírita

CDU 02:141:005.7

Renan Vitor Henker Regis

Desenvolvimento do Núcleo do Sistema de Informação Gerencial do Movimento Espírita

Monografia apresentada ao Curso de Ciência da Computação do Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Trabalho aprovado. Vitória, ES, 17 de dezembro de 2018:

Prof. Dr. Vítor E. Silva Souza
Orientador

Prof. Lucia Catabriga, D.Sc
Universidade Federal do Espírito Santo

César Henrique Bernabé
Universidade Federal do Espírito Santo

Vitória, ES
2018

Agradecimentos

Em primeiro lugar agradeço a Deus que me sustentou e me guiou durante todo o tempo em que eu passei nesse curso. Por tudo o que passei dentro e fora da Universidade. Sem Ele, eu não estaria aqui.

Aos meus pais, Marlene e José, por terem me apoiado e incentivado a persistir no curso ao longo desses anos e por todo carinho e amor que sempre tiveram por mim.

A todos amigos feitos na Universidade. Aos professores do Departamento de Informática da UFES, por sempre estarem disponíveis em ajudar e a ensinar. Em especial ao meu orientador Vitor em estar disposto a me orientar e me ajudar a completar o curso, mesmo depois de tanto tempo inciado a orientação.

Resumo

Na Federação Espírita do Estado do Espírito Santo (FEEES) é feita toda a gerência de informações a respeito do movimento espírita do estado, além de realizar manualmente uma série de atividades e processos.

Este projeto visa modelar, projetar e desenvolver o módulo de cadastros básicos do Sigme (Sistema de Informação Gerencial do Movimento Espírita). Seu objetivo é informatizar e auxiliar a realização dos processos manuais de cadastro de pessoas, espíritas ou não, e instituições que são feitos por funcionários de cada instituição ligada à FEEES, a fim de reduzir o tempo gasto com essa atividade e obter um melhor controle das informações registradas. Além disso, qualquer pessoa poderá se cadastrar no sistema para que no futuro, por meio de outros módulos do Sigme, essa pessoa possa se cadastrar em algum evento, receber mala direta e utilizar de outras funcionalidades que possam ser adicionadas ao sistema.

O desenvolvimento do módulo do sistema seguiu um processo de Engenharia de Software, utilizando métodos e técnicas de modelagem e desenvolvimento orientado a objetos, em particular o método FrameWeb para projeto de aplicações Web baseadas em *frameworks*.

Palavras-chaves: Java, Aplicação Web, JSF, FrameWeb.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UML	Unified Modeling Language
API	Application Programming Interface
CDI	Contexts and Dependency Injection
DAO	Data Access Object
EL	Expression Language
FrameWeb	Framework-based Design Method for Web Engineering
HTML	HyperText Markup Language
Java EE	Java Enterprise Edition
JPA	Java Persistence API
JSF	Java Server Faces
JVM	Java Virtual Machine
MVC	Model-View-Controller
JDBC	Java Database Connectivity
SQL	Structured Query Language

Sumário

1 Introdução

Uma série de atividades da Federação Espírita do Estado do Espírito Santo (FEEES) são realizadas por meio de seus departamentos, acompanhados pelos vice-presidentes, além da tesouraria e secretaria, todos coordenados pela presidência. As principais atividades realizadas pela FEEES dependem de um cadastro de pessoas físicas e jurídicas, sejam elas espíritas ou não, em especial o cadastro de instituições espíritas adesas à Federação.

Os objetivos de se registrar toda esta gama de entidades são vários, dentre eles: gerar etiquetas para correspondência, gerar boletos para cobrança, saber quem são as pessoas que ocupam os diversos cargos, gerar um site das casas espíritas adesas, e integrar, de várias outras maneiras, as casas espíritas e o público espírita ao sistema.

O módulo do Sistema de Informação Gerencial do Movimento Espírita (Sigme) desenvolvido tem por objetivo informatizar e auxiliar a realização dos processos manuais de cadastro que são feitos por funcionários de cada instituição ligada a FEEES, a fim de reduzir o tempo gasto com essa atividade e obter um melhor controle das informações registradas.

1.1 Objetivos

O objetivo geral deste trabalho é desenvolver um sistema Web que será utilizado para controlar cadastros de instituições e pessoas, além da importação de cadastros de outras bases de dados no formato Excel, utilizando os conceitos aprendidos ao longo do curso de Ciência da Computação. São objetivos específicos deste projeto:

- Levantar as necessidades e requisitos necessários para o sistema e produzir documentos com as especificações dos requisitos;
- Produzir o documento de projeto do sistema, utilizando o método FrameWeb (SOUZA, 2007);
- Desenvolver o sistema de acordo com a estrutura definida no documento de projeto, utilizando *frameworks* já existentes para auxiliar no desenvolvimento do sistema;
- Sugerir melhorias futuras para as limitações do sistema.

1.2 Metodologia

A metodologia usada para desenvolver este trabalho foi composta pelas seguintes atividades:

- Revisão Bibliográfica: estudo dos temas que foram necessários para a construção do sistema, tais como técnicas de Engenharia de Software, a linguagem de programação Java e seus *frameworks* para desenvolvimento Web, arquitetura de um sistema Web, projeto de banco de dados e o método FrameWeb;
- Levantamento de requisitos: fase de levantamento de todos os requisitos das funcionalidades que foram implementadas no sistema;
- Especificação de requisitos: fase de análise de todos os requisitos levantados anteriormente, onde os mesmos foram definidos com mais detalhes e criados diagramas e modelos para melhor compreensão;
- Projeto de sistema: fase de definição das tecnologias e arquiteturas utilizadas para o desenvolvimento do sistema e documentação do projeto arquitetural do sistema utilizando o método FrameWeb;
- Implementação: após definida a arquitetura, o sistema foi implementado conforme os documentos definidos anteriormente. Em seguida, o protótipo do sistema implementado foi testado;
- Escrita da monografia: finalmente, uma monografia que descreve todo o trabalho realizado foi escrita. Vale ressaltar que a mesma foi escrita em *LaTeX* utilizando o editor *TeXstudio* e o template *abnTeX* que atende os requisitos das normas da ABNT (Associação Brasileira de Normas Técnicas) para elaboração de documentos técnicos e científicos brasileiros.

1.3 Organização do Texto

Esta monografia é estruturada em cinco partes e contém, além da presente introdução, os seguintes capítulos:

- **Capítulo** 2 – Referencial Teórico: apresenta uma revisão da literatura acerca dos temas relevantes ao contexto deste trabalho, a saber: Engenharia de Software, desenvolvimento Web e FrameWeb;
- **Capítulo** 3 – Especificação de Requisitos: apresenta a especificação de requisitos do sistema, descrevendo o minimundo e exibindo os seus diagramas de classes e casos de uso;

- **Capítulo** 4 – Projeto Arquitetural e Implementação: apresenta a arquitetura do sistema, assim como as partes principais de sua implementação, ilustradas por capturas de telas do sistema;
- **Capítulo** 5 – Considerações Finais: apresenta as conclusões retiradas deste trabalho.

2 Referencial Teórico

Este capítulo apresenta os principais conceitos teóricos para o desenvolvimento do módulo de cadastros básicos do Sigme e está organizado em 3 seções que são: Especificação de Requisitos (Seção 2.1), Projeto de sistemas e implementação (Seção 2.2) e o método *FrameWeb* (Seção 2.3).

2.1 Especificação de Requisitos

A especificação de requisitos compreende a primeira etapa do processo de desenvolvimento de um software. Nesta etapa são definidas as funcionalidades que o sistema deve conter e os critérios de qualidade sob os quais o sistema deverá operar. É caracterizada por um esforço conjunto entre clientes, usuários e especialistas de domínio a fim de traçar uma linha conjunta de entendimento entre todas as partes envolvidas a respeito de como um software se comportará e de como será utilizado pelo usuário. Trata-se de uma atividade complexa que não se resume somente a perguntar às pessoas o que elas desejam, mas sim analisar cuidadosamente a organização, o domínio da aplicação e os processos de negócio no qual o sistema será utilizado (SOMMERVILLE; KOTONYA, 1998).

Uma parte essencial dessa fase é a elaboração de modelos descrevendo o que o software tem de fazer (e não como fazê-lo), dita Modelagem Conceitual. Até este momento, a ênfase está sobre o domínio do problema e não se deve pensar na solução técnica, computacional a ser adotada (FALBO, 2017).

Com base nessas e em outras definições, pode-se dizer que os requisitos de um sistema incluem especificações dos serviços que o sistema deve prover, restrições sob as quais ele deve operar, propriedades gerais do sistema e restrições que devem ser satisfeitas no seu processo de desenvolvimento (FALBO, 2017). Os requisitos de software de um sistema podem ser divididos em funcionais e não funcionais.

Requisitos funcionais são declarações de serviços que o sistema deve prover, descrevendo o que o sistema deve fazer (SOMMERVILLE et al., 2003). Já os não funcionais, descrevem restrições sobre os serviços ou funções oferecidas pelo sistema (SOMMERVILLE et al., 2003), as quais limitam as opções para criar uma solução para o problema (PFLEEGER, 2004).

Um dos modelos criados durante a fase de requisitos, o modelo de casos de uso visa capturar e descrever as funcionalidades que um sistema deve prover para os atores que interagem com o mesmo, isto é, os usuários do sistema (SOUZA, 2007). Com isso, o usuário tem a descrição passo a passo de todas as funcionalidades do sistema e de como

pode prosseguir para executar determinada tarefa.

Os casos de uso cadastrais de baixa complexidade que envolvem as funcionalidades de criar, alterar, excluir e consulta, podem ser descritos na forma de tabela como proposto por FALBO (2017). Existem diferentes tipos de usuários no sistema, que são chamados de atores. Através dessa caracterização é possível identificar quais tarefas no sistema cada usuário pode ou não executar.

Ainda como produto dessa fase do desenvolvimento de um software, a modelagem conceitual estrutural visa modelar de forma computacional os requisitos e os casos de uso identificados anteriormente. Por meio deste modelo, é possível identificar de forma clara os tipos de entidade e os tipos de relacionamentos presentes no sistema. O propósito da modelagem conceitual estrutural, segundo o paradigma orientado a objetos, é definir as classes, atributos e associações que são relevantes para tratar o problema a ser resolvido (FALBO, 2017).

2.2 Projeto de sistemas e implementação

A fase de Projeto durante a especificação de um software tem por objetivo definir e especificar uma solução a ser implementada (FALBO, 2011). Diferente da fase de levantamento e especificação de requisitos, em que se está preocupado em buscar o que deve ser feito, a fase de projeto busca definir como serão implementadas as funcionalidades propostas e quais as tecnologias serão utilizadas no desenvolvimento do software. Esta fase é decomposta em duas etapas, sendo que a primeira consiste em definir uma arquitetura de software, seguido do detalhamento e projeto dos elementos da arquitetura e por fim o projeto detalhado.

A arquitetura de software de um programa ou sistema computacional é a estrutura ou estruturas do sistema, abrangendo os componentes de software, as propriedades externamente visíveis desses componentes e as relações entre eles (PRESSMAN, 2011), ou seja, a etapa de arquitetura de software tem como objetivo obter os relacionamentos dos componentes do sistema assim como identificar tais componentes. Ao final desta etapa, segue-se para o detalhamento que em (FALBO, 2011) é definido como tendo o objetivo de projetar em um maior nível de detalhes cada um dos elementos estruturais definidos na arquitetura, o que envolve a decomposição de módulos em outros módulos menores. Com o resultado dessa etapa, já é possível identificar os primeiros módulos de implementação, onde no caso do paradigma orientado a objetos, seriam as classes.

Por fim, o detalhamento do projeto tem por objetivo refinar e detalhar os elementos mais básicos da arquitetura do software: as interfaces, os procedimentos e as estruturas de dados. Deve-se escrever como se dará a comunicação entre os componentes da arquitetura, a comunicação do sistema em desenvolvimento com outros sistemas e com as pessoas que

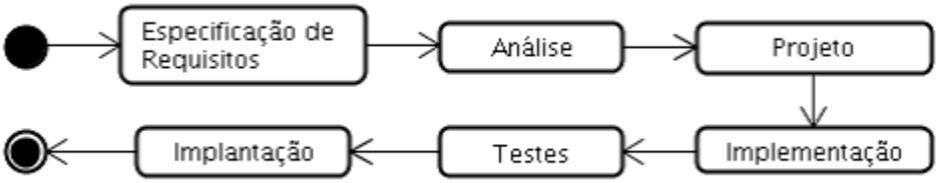

Figura 1 – Um processo de desenvolvimento de software simples sugerido pelo *FrameWeb*.

irão utilizá-lo (FALBO, 2011).

Ao final da fase de Projeto de software, tem-se a arquitetura na qual o sistema será montado juntamente com o projeto dos componentes do sistema. Esses componentes, como proposto em (FOWLER, 2002), podem ser classificados da seguinte forma: interface com o usuário, lógica de negócio e persistência de dados. A interface com o usuário é o componente responsável por gerenciar todo o contato do usuário com o sistema, seja para receber dados quanto para apresentar dados. O componente de lógica de negócio é encarregado de aplicar todas as regras de negócios e requisitos do sistema. É uma camada intermediária de um sistema, que recebe os dados vindo da interface com o usuário e após aplicada a lógica de negócio repassa os dados para a camada seguinte. A persistência de dados tem como tarefa guardar os dados de forma a ser possível recuperá-los quando necessário em um sistema de banco de dados integrando com o sistema.

2.3 O método FrameWeb

FrameWeb é um método de projeto para construção de sistemas de informação Web (*Web Information Systems* – WISs) baseado em *frameworks*. O *FrameWeb* é baseado em metodologias linguagens de modelagens bastante difundidas na área de Engenharia de Software sem, no entanto, impor nenhum processo de desenvolvimento especifico (SOUZA, 2007).

Sendo um método para a fase de Projeto, o *FrameWeb* espera que as fases apresentadas na Figura 1 sejam contempladas no processo de desenvolvimento e que sejam construídos diagramas de casos de uso e de classes de domínio (ou similares) durante as fases anteriores à fase de Projeto.

Este método assume que existem tipos determinados de *frameworks* utilizados durante todo o resto do desenvolvimento do software, que engloba a definição de uma arquitetura básica para o sistema e a implementação.

A arquitetura proposta para o método *FrameWeb* foi baseada no padrão Camada de Serviço (FOWLER, 2002). Tal arquitetura é composta de três camadas (MARTINS, 2016):

- Camada de Negócio (*Business Tier*): responsável pelas funcionalidades relacionadas às regras de negócio da aplicação. Esta camada é dividida em duas: Lógica de Domínio (*Domain*) e Lógica de Aplicação (*Application*);
- Camada de Apresentação (*Presentation Tier*): responsável por funcionalidades de interface com o usuário (incluindo as telas que serão apresentadas a ele). Esta camada, segundo o padrão proposto, é também dividida em outras duas: Visão (*View*) e Controle (*Control*);
- Camada de Acesso a Dados (*Data Access Tier*): responsável por funcionalidades relacionadas à persistência de dados.

De acordo com a especificação original do método, a fase de Projeto concentra as propostas principais: (i) definição de uma arquitetura padrão que divide o sistema em camadas, de modo a se integrar bem com os *frameworks* utilizados; (ii) proposta de um conjunto de modelos de projeto que trazem conceitos utilizados pelos *frameworks* para esta fase do processo por meio da criação de um perfil UML que faz com que os diagramas fiquem mais próximos da implementação (SOUZA, 2007).

A modelagem utilizando o método *FrameWeb* é baseada em quatro principais modelos, que, juntos, são capazes de descrever todo o fluxo de execução de um sistema Web, e além disso, estrutura de banco de dados, de páginas Web, etc. Os modelos são:

- Modelo de Entidades: é um diagrama de classes da UML que representa o domínio do problema e o mapeamento do mesmo para a estrutura do banco de dados;
- Modelo de Persistência: é um diagrama utilizado para representar as classes responsáveis pela persistência dos objetos de domínio, seguindo o padrão de projeto DAO (*Data Access Object*);
- Modelo de Navegação: é utilizado para representar os componentes existentes nos pacotes de Visão (*View*) e Controle (*Controllers*), que são responsáveis pela interação com o usuário. Além disso, este modelo é capaz de mostrar o fluxo existente entre diferentes páginas de um sistema Web (como certas funcionalidades existentes em uma página, levam o usuário para outra página, e assim sucessivamente);
- Modelo de Aplicação: é um diagrama utilizado para representar os serviços existentes no pacote de Aplicação (responsáveis por implementar os casos de uso de um sistema) e as dependências de componentes de outras camadas (Controle e Persistência).

Esses quatro modelos serão descritos na Seção 4.3 no contexto dos módulos desenvolvido neste trabalho.

3 Especificação de Requisitos

Este capítulo apresenta a especificação de requisitos para a construção do módulo de cadastros básicos do sistema Sigme. Os requisitos levantados se deram a partir de entrevistas com o orientador deste trabalho e da análise de documentos produzidos por ele a partir de entrevistas feitas junto à diretoria e funcionários da FEEES.

Na Seção 3.1 é apresentada a descrição do escopo do projeto, na Seção 3.2 são apresentados diagramas de casos de uso e na Seção 3.3 são apresentados os diagramas de classes. Os requisitos funcionais, requisitos não funcionais e regras de negócio podem ser encontrados no Documento de Especificação de Requisitos que está disponível no Apêndice ao final dessa monografia.

3.1 Descrição do Escopo

A Federação Espírita do Estado do Espírito Santo (FEEES) é uma organização religiosa, que objetiva unificar, orientar, coordenar e dinamizar o movimento espírita no Estado. O escopo do sistema proposto se refere à parte de cadastro de pessoas e instituições, que é base para muitas outras funcionalidades e é parte do núcleo do sistema. Ele é o responsável por registrar informações sobre pessoas físicas e jurídicas associadas à federativa, sejam elas espíritas ou não, e ainda registrar informações sobre instituições.

Os objetivos de se registrar toda esta gama de entidades são vários, dentre eles: gerar etiquetas para correspondência, gerar boletos para cobrança, saber quem são as pessoas que ocupam os diversos cargos, gerar um site das casas espíritas adesas, e integrar, de várias outras maneiras, as casas e o público espírita. Tais funcionalidades estão fora do escopo deste projeto e serão desenvolvidas em outros módulos do Sigme.

O módulo do sistema desenvolvido tem por objetivo informatizar e auxiliar a realização dos processos manuais de cadastro que são feitos por funcionários de cada instituição ligada à FEEES, a fim de reduzir o tempo gasto com essa atividade e obter um melhor controle das informações registradas.

O primeiro passo para utilização deste módulo do Sigme será o cadastro das informações da instituição principal, que via de regra é a federativa, e do administrador do sistema. A partir disso o administrador deve ocupar um cargo de presidente da instituição principal. Então o administrador pode cadastrar e manipular informações de cadastros de novas instituições e espíritas. Além disso nesse módulo o presidente pode delegar, por exemplo, a um assessor de TI a administração do sistema como um todo.

Por ser um sistema Web, um visitante não identificado também pode acessar o

sistema e se cadastrar como um espírita, e poderá ativar ou desativar seu cadastro. Assim em módulos futuros ele poderá se cadastrar para algum evento, receber mala direta, etc. Uma vez cadastrado no sistema, o espírita pode solicitar o acesso ao Sigme se porventura esquecer a senha. Um administrador também pode solicitar que o sistema envie por e-mail uma notificação a todos que possuem cadastros inativos (que ainda não receberam notificação deste tipo) informando-os que podem ativar seus cadastros se assim desejarem.

Além disso o módulo proposto também fornece a função de importar cadastros de espíritas de outras bases de dados que as instituições possuem. Esta funcionalidade é bastante importante, visto que já existe um cadastro de pessoas e instituições na FEEES e, desta maneira, evita o retrabalho dos funcionários da Federação.

3.2 Modelo de Casos de Uso

A Tabela 1 descreve os atores identificados para o módulo de cadastros básicos e suas respectivas funções.

Tabela 1 – Descrição dos atores identificados

Ator	**Descrição**
Administrador	Funcionário da Feees responsável pelo controle do núcleo de cadastros do sistema.
Espírita	Usuário comum cadastrado no Sigme.

O administrador do sistema é o usuário que foi cadastrado na primeira execução do Sigme ou um espírita que ganhou a função de administrador, concedida por outro administrador através da alteração de seu cadastro. O administrador tem permissão total no sistema, podendo assim alterar, inserir e excluir qualquer cadastro. O espírita é um usuário comum, que neste módulo somente tem acesso ao seu cadastro, e pode alterar seus dados, ativar e desativar seu cadastro.

As Figuras 2 e 3 mostram os diagramas de casos de uso do módulo de cadastros básicos, demonstrando quais casos de uso cada ator pode executar.

Os casos de uso **Cadastrar Espírita** e **Cadastrar Instituição** são do tipo cadastrais e portanto incluem inclusão, consulta, alteração e exclusão. Através do caso de uso Cadastrar Espírita, o administrador pode também dar permissão para um espírita se tornar um administrador do sistema, apenas informando que ele possui esta função.

Para inserir um novo espírita, basta informar nome, CPF, e-mail, nome preferido, data de nascimento, sexo, endereço, telefones de contato e instituições caso já tenha frequentado ou frequenta alguma. Para inserir uma nova Instituição é necessário informar nome, sigla, endereço, site (se possuir), e-mail, telefones de contato (número e tipo – principal, celular, fax, etc.), o tipo de instituição (casa espírita não-adesa, casa espírita

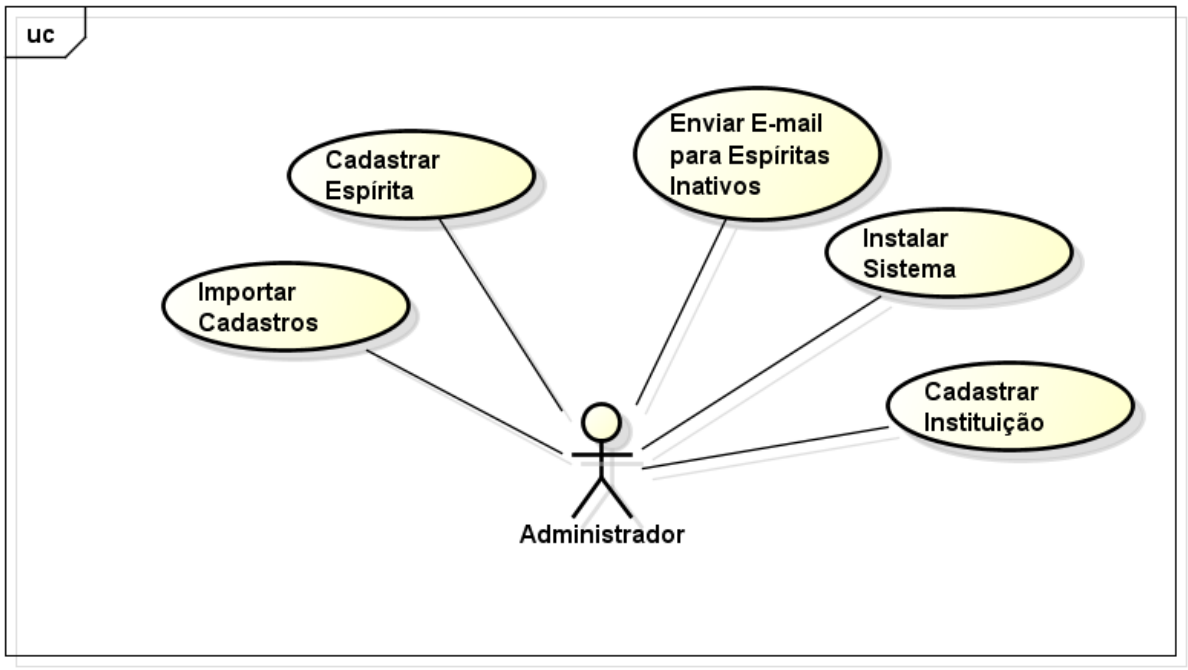

Figura 2 – Diagrama de casos de uso do ator **Administrador**.

adesa, federativa espírita, outra instituição espírita, não-espírita).

Na primeira utilização do sistema, o caso de uso **Instalar Sistema** é iniciado. Nesse momento é necessário informar os dados do administrador do sistema a ser cadastrado: nome completo, CPF, e-mail, nome preferido, senha e confirmação da senha. Em seguida deverá ser informado o nome e a sigla da instituição que será registrada com a principal. E por fim é necessário informar dados para a configuração de envio de e-mail: endereço do servidor de e-mail, porta, usuário e senha.

O administrador também pode **Importar Cadastros** de outras bases de dados em formato CSV. Após o administrador selecionar o arquivo de importação, o sistema irá apresentar uma tela em que será mostrado todos os cadastros a serem importados. Se existirem cadastros que possuam o mesmo CPF o administrador deve decidir qual deverá ser utilizado. Depois da confirmação dessa decisão, os cadastros serão registrados no Sigme.

Para aqueles espíritas que estão com o cadastro inativo, o administrador pode **Enviar e-mail para Espíritas Inativos**, informando-os sobre a ativação do cadastro. Somente poderá ser enviado para aqueles que nunca receberam um e-mail como esse.

Um usuário não identificado pode **Cadastrar-se** no Sigme, informando o e-mail. O sistema então enviará um código por e-mail que deve ser informado para o usuário se cadastrar. Tendo confirmado no sistema o código recebido, o usuário informa nome completo, CPF, nome preferido, a senha e confirmação da senha. Após isso o usuário é registrado como espírita e pode alterar, ativar e desativar seu cadastro.

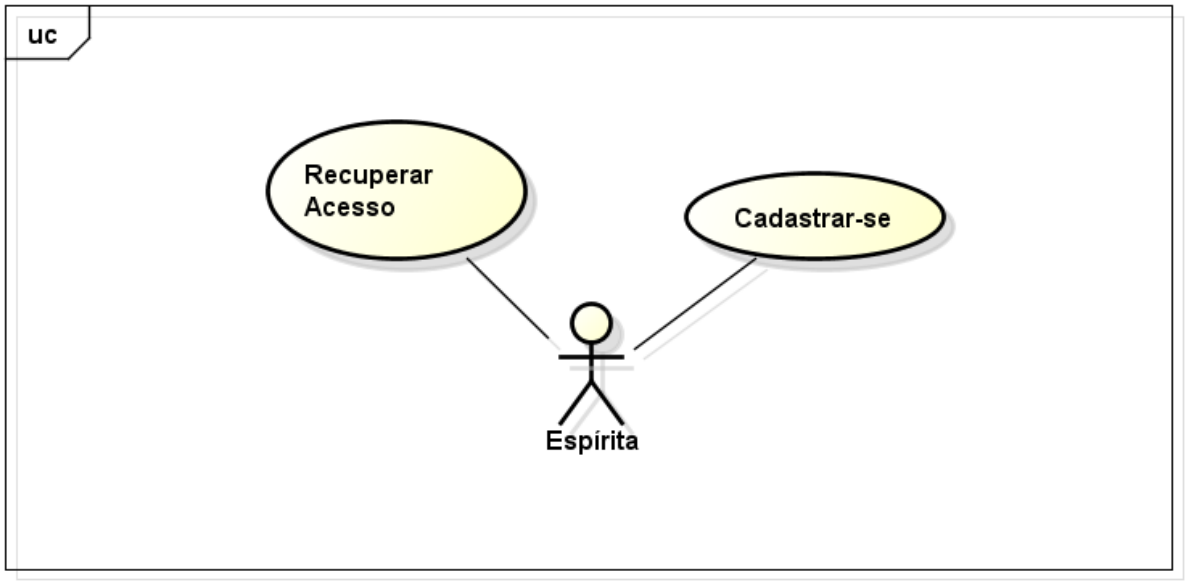

Figura 3 – Diagrama de casos de uso do ator **Espírita**.

Se o espírita esquecer sua senha, ele poderá **Recuperar o Acesso**, informando o e-mail. Ele receberá um link de acesso para o sistema, em que ele poderá escolher uma nova senha.

A documentação completa dos requisitos do sistema pode ser encontrada nos Apêndices desta monografia.

3.3 Modelo Estrutural

O modelo conceitual estrutural visa capturar e descrever as informações (classes, associações e atributos) que o sistema deve apresentar para prover as funcionalidades descritas na Seção 3.2. Na prática, o modelo estrutural tem o objetivo de obter as informações do mundo real e as inserir em um contexto de desenvolvimento de software. Neste modelo é possível identificar como que as ações executadas pelo usuário são tratadas no sistema.

A Figura 4 apresenta o diagrama de classes gerado para o módulo de cadastros básicos do Sigme. Nesse diagrama estão inclusos os atributos e classes de dois pacotes que compõem o módulo de cadastros básicos, que serão apresentados separados e mais bem detalhados na Seção 4: os módulos **UtilSigme** e **Núcleo**.

No cadastro de uma nova instituição, o administrador do sistema informa alguns dados necessários, tais como nome, sigla, site, e-mail, se é parte de uma regional e o **Tipo de Instituição** em uma tela de cadastro específica. Então o sistema cria um novo objeto da classe **Instituição** e o salva. No diagrama também é possível ver quais os possíveis tipos de instituições que podem ser cadastrados no sistema.

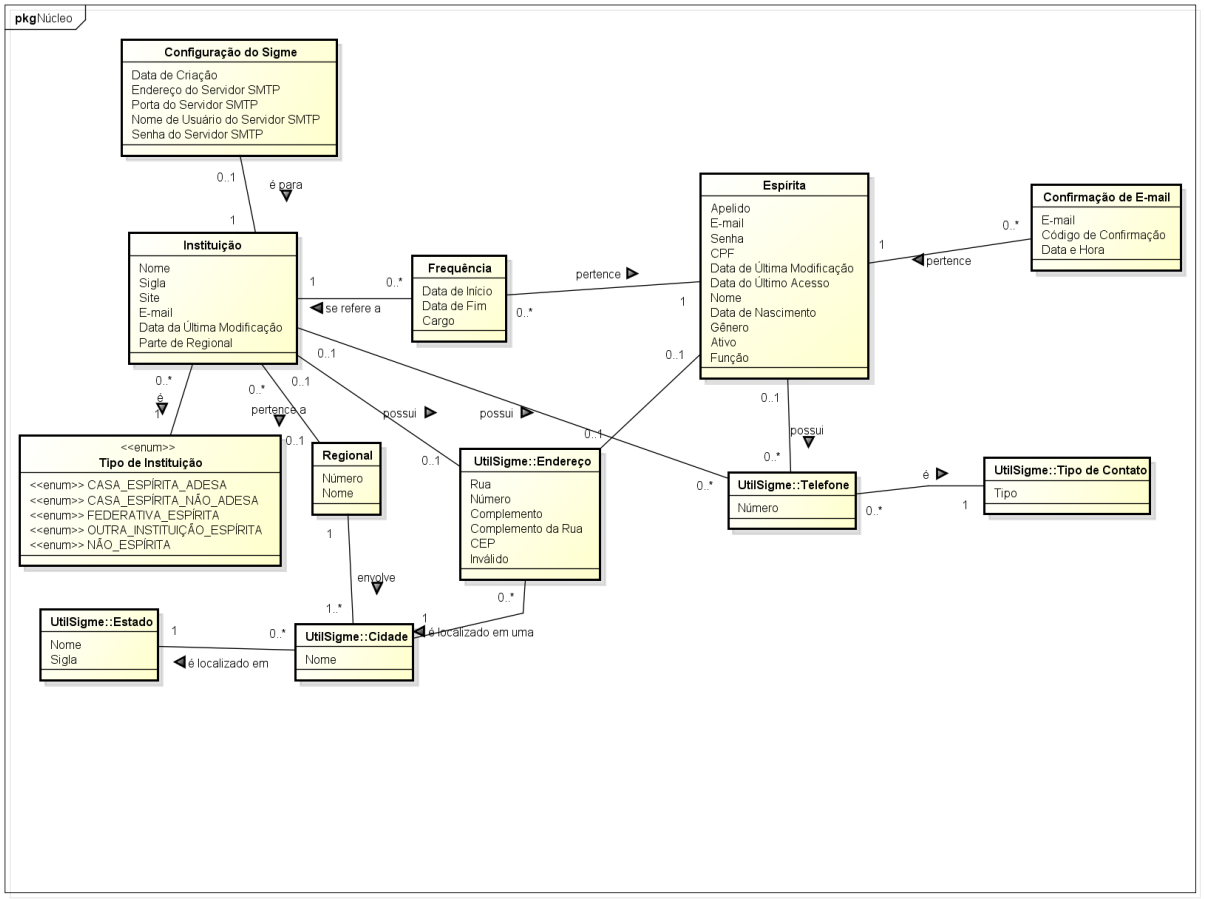

Figura 4 – Diagrama de classes do módulo de cadastros básicos.

Esse fluxo também se repete para a criação de um novo **Espírita**: o administrador ou um usuário que deseja se cadastrar no sistema, informa dados necessários para o cadastro em uma tela de cadastro específica. O sistema cria um novo objeto da classe Espírita com os dados informados e o registra. Mas um **Telefone** e **Endereço** só podem ser cadastrados a partir da tela de cadastro de espírita ou de uma instituição. Informações sobre confirmações de e-mails enviados são registrados no objeto **Confirmação de E-mail**. No diagrama também é visto a classe **Configuração do Sigme**, onde são registrados dados sobre o servidor de envio de e-mails.

É importante também identificar restrições de integridade do sistema, limitações que não são possíveis de serem representadas no diagrama mas são importante para manter a consistência e integridade de dados a todo o tempo. A partir disso as seguintes restrições foram identificadas:

- Um telefone não pode pertencer, ao mesmo tempo, a uma instituição e a um espírita;
- Um endereço não pode pertencer, ao mesmo tempo, a uma instituição e a um espírita;
- Uma instituição não pode estar numa cidade que não faça parte da sua regional.

3.4 Modelo Dinâmico

Os diagramas de classes gerados pela atividade de modelagem conceitual estrutural representam apenas os elementos estáticos de um modelo de análise orientada a objetos. É preciso, ainda, modelar o comportamento dinâmico da aplicação. Um modelo dinâmico indica como o sistema irá responder a eventos ou estímulos externos e auxilia o processo de descoberta das operações das classes do sistema (FALBO, 2017).

A Figura 5 mostra o diagrama de gráfico de estados para a classe Espírita, que consiste em uma máquina de estados pelos quais objetos de uma particular classe podem passar ao longo de seu ciclo de vida e as transições possíveis entre esses estados. Diagramas de gráfico de estados (ou diagramas de transição de estados) são usados principalmente para modelar o comportamento de uma classe, dando ênfase ao comportamento específico de seus objetos (FALBO, 2017).

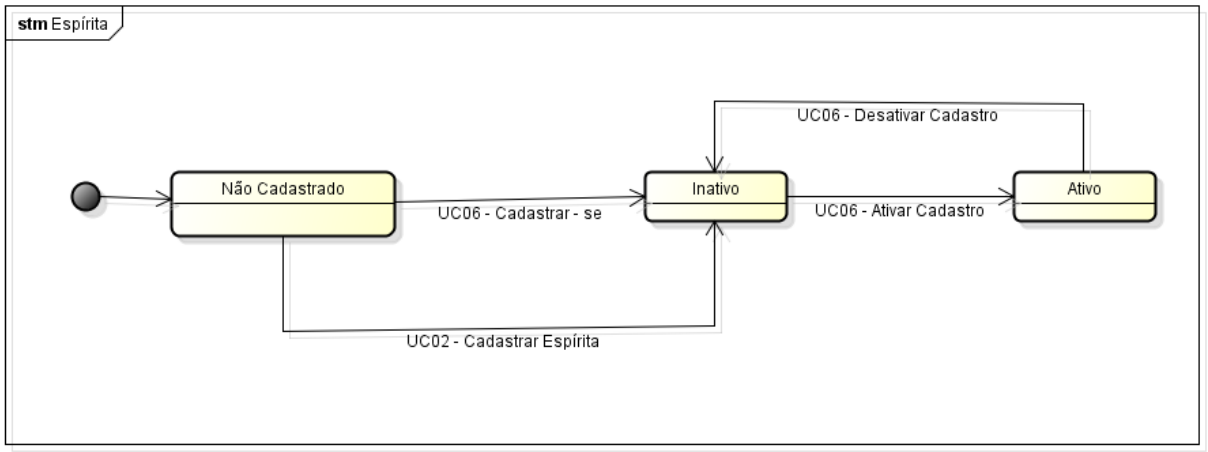

Figura 5 – Diagrama de estados da classe **Espírita**.

Os objetos da classe **Espírita** possuem três estados no sistema: **Não Cadastrado**, **Inativo** e **Ativo**. Quando uma pessoa que não possui um cadastro acessa o sistema para se cadastrar, completa o preenchimento das informações necessárias e submete seu cadastro, através do caso de uso **Cadastrar-se**, ele passa do estado **Não cadastrado** para o estado **Inativo**. Também é possível ir para o estado **Inativo** através do caso de uso **Cadastrar Espírita**, que o administrador executa. Um espírita inativo então, pode ativar seu cadastro através do caso de uso **Ativar Cadastro**, passando a estar assim no estado **Ativo**. Também é possível pelo caso de uso **Desativar Cadastro** um espírita retornar para o estado **Inativo**.

4 Projeto Arquitetural e Implementação

Como proposto na Seção 2.2, após as fases de especificação e análise de requisitos, ocorre a fase de projeto do sistema. Esta etapa é responsável pela modelagem da implementação do sistema, incorporando, aos requisitos, as tecnologias a serem utilizadas.

Neste capítulo são apresentados o projeto arquitetural e a implementação do sistema, por meio de suas principais telas. Na Seção 4.1, as tecnologias utilizadas para a implementação são descritas; na Seção 4.2, está descrita a arquitetura de software do sistema; na Seção 4.3 os modelos FrameWeb são apresentados. Por fim, na Seção 4.4, são apresentadas algumas telas do sistema.

4.1 Tecnologias Utilizadas

A linguagem de programação escolhida para ser realizada a implementação do projeto foi Java, devido a independência da plataforma e por oferecer, de forma nativa, suporte a diversos *frameworks* e APIs.

Para reduzir a complexidade do desenvolvimento, implantação e gerenciamento de aplicações, foi utilizado neste trabalho o Java EE (*Java Enterprise Edition*), que é um conjunto de de especificações de APIs e tecnologias, que são implementadas por programas servidores de aplicação.

O *JavaServer Faces*(JSF) (MANN, 2005) é um *framework* para construção de interfaces com usuários baseada em componentes para aplicações Web. Com o auxilio dessa API é possível efetuar a criação das páginas Web utilizando componentes visuais pré-prontos e a integração com o restante da aplicação.

JSF é um padrão de especificação Java para desenvolvimento Web e faz parte do Java EE utilizado nesse trabalho. Tal padrão, possui um modelo de programação que permite a abstração de componentes HTML e o pacote de controle, dentro da camada de apresentação.

O *Contexts and Dependency Injection* (CDI) para Java EE é um *framework* gratuito para a injeção de dependências disponível na plataforma de desenvolvimento Java EE 7. Com esta ferramenta é possível integrar as diferentes camadas de arquitetura e serviços de transação. O CDI é um conjunto de serviços que facilitam o trabalho de desenvolvedores por permitir, entre outras coisas, que páginas JSF façam referências a essas classes injetadas utilizando linguagem de expressões unificada (JENDROCK et al., 2014).

Para o acesso a banco de dados foi utilizada a API JPA (Java Persistence API) (DE-

MICHIEL; SHANNON, 2013), que é uma API para persistência de dados por meio de mapeamento objeto-relacional das classes de domínio. Após a definição de JPA como sendo um padrão a ser seguido para desenvolvimento em Java, diversas implementações surgiram ou foram atualizadas para se adequar a nova realidade, como foi o caso de um dos mais populares: o Hibernate (BAUER; KING, 2005).

Para auxiliar no desenvolvimento desse projeto, foi utilizado o JButler que é uma ferramenta desenvolvida pelo Núcleo de Estudos em Modelagem Conceitual e Ontologias (NEMO) da Universidade Federal do Espírito Santo (UFES) que tem como objetivo auxiliar a implementação de aplicações Web que utilizam os frameworks JSF, JPA e CDI. A ferramenta implementa, genericamente, as operações básicas existentes entre a aplicação e o banco de dados, fazendo com que o desenvolvedor se preocupe apenas em adaptar e indicar quais as entidades de domínio que serão utilizadas e/ou mapeadas, diminuindo, assim, drasticamente o tempo gasto para implementar tais funcionalidades.

4.2 Arquitetura de Software

A arquitetura de *software* da ferramenta Sigme, no caso desse projeto o módulo de cadastros básicos, nessa seção também chamado de **Núcleo** (**Core**), está organizada em três camadas seguindo o proposto pelo *FrameWeb*, que são: **Camada de Apresentação** (*Presentation Tier*), **Camada de Negócio** (*Business Tier*), e **Camada de Acesso a Dados** (*Data Access Tier*).

A primeira camada contém os pacotes de Visão (*View*) e Controle (*Control*), a segunda contém o de Domínio (*Domain*) e o de Aplicação (*Application*) e a terceira somente o pacote de Persistência (*Persistence*). Cada pacote será explicado melhor nas próximas seções.

A Figura 6 apresenta a visão geral das camadas e seus pacotes juntamente com o relacionamento que existe entre eles e as tecnologias Java EE utilizadas em cada pacote, já descritas na Seção 4.1.

A Figura 7 apresenta a subdivisão do módulo Núcleo, no pacote **br.org.feees-.sigme.core**, nas camadas descritas acima, a saber: **A Camada de Apresentação** (**control**), **Camada de Negócio** (**domain** e **application**) e **Camada de Acesso a Dados** (**persistence**).

O pacote de visão (**view**) que aparece na Figura 7 contém o arquivo de mensagens que são utilizadas no sistema e podem ser apresentadas ao usuário. Em outra pasta do projeto, chamada *WebContents*, encontra-se a maior parte dos componentes da *View*. Algumas classes e atributos da classe Espírita (*Spiritist*) que antes apareceram no documento de especificação de requisitos foram movidos para o pacote **org.feees.sigme.people**, por

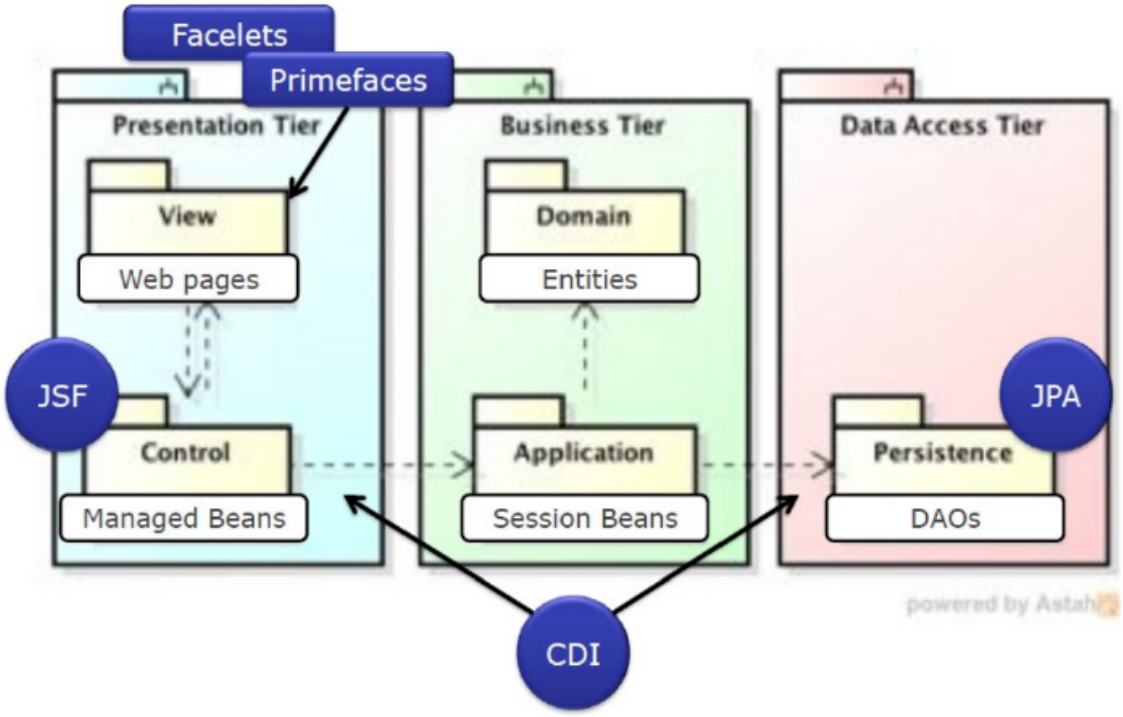

Figura 6 – Arquitetura de Software do Sistema proposto.(LIMA, 2015)

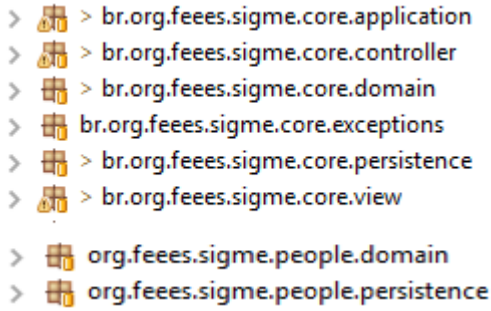

Figura 7 – Subdivisão em pacotes dos módulos **Núcleo**(*Core*) e **UtilSigme**.

motivo de reuso de código. Além disso, o pacote de exceções (**exceptions**), que reúne as classes Java que controlam exceções específicas do Sigme, também aparece na Figura 7.

Na Seção 4.3 serão apresentados os modelos *FrameWeb* relativos a cada camada do sistema mencionadas anteriormente. Vale mencionar que os modelos tentam se aproximar do que está sendo implementado, por isso classes e atributos estão em inglês.

4.2.1 Camada de Apresentação

A camada de apresentação foi subdividida em visão (*View*) e controle (*Control*). A parte da visão é formada pelas páginas Web. A parte de controle contém os controladores que realizam a comunicação entre a interface e a aplicação.

As páginas Web do Sigme que fazem parte do pacote de visão da camada de

apresentação estão numa pasta chamada *WebContents* que contém uma subpasta para cada um dos casos de uso básicos (CRUD — *Create, Retrieve, Update, Delete*). Isso ajuda na organização dos arquivos.

Esse padrão de visão consiste em duas páginas html, sendo a primeira chamada `form.xhtml`, que é responsável por obter os dados das entidades para que possam ser modificados e armazenados no banco de dados. A segunda página é a `list.xhtml` é responsável por recuperar e exibir para o usuário as informações das entidades que estão armazenadas no banco de dados, geralmente em formato tabular.

4.2.2 Camada de Negócio

A camada de negócio foi subdividida em domínio (*Domain*) e aplicação (*Application*). A parte do domínio é formada pelas entidades de negócio (classes de domínio), enquanto a aplicação contém as validações dos dados e implementação dos casos de uso.

Os pacotes **core.domain** e **people.domain** contém a definição de cada uma das classes do módulo proposto. Cada uma dessas entidades está definida em um arquivo .java e, através de anotações, identificam como o mapeamento objeto/relacional irá ocorrer. Através desse mapeamento, o JPA irá gerenciar os objetos no banco de dados automaticamente, sem precisar de nenhuma intervenção do desenvolvedor. É nesse momento que é realizado também o relacionamento entre as classes do sistema utilizando as anotações `@OneToMany`, `@ManyToOne` ou `@ManyToMany` de JPA.

Por fim, nesse pacote existe um arquivo para cada classe com o mesmo nome e um *underscore* no final (chamada de *static meta-model* ou metamodelo estático), que declara os atributos que poderão ser utilizados para realizar as consultas no banco de dados utilizando os conceitos de Criteria API. Essas consultas serão implementadas na camada de persistência.

O pacote **core.application** contém os componentes que fazem a comunicação entre a apresentação (controladores) e a persistência (DAOs), implementando as funcionalidades do sistema descritas em seus casos de uso (vide Seção 3.2). Faz também as validações das informações antes de chamar os métodos de acesso a dados. Essas validações serão feitas ao tentar criar, modificar ou excluir uma entidade.

4.2.3 Camada de Acesso a Dados

A camada de acesso a dados possui os pacotes responsáveis pela persistência, **core.persistence** e **people.persistence**, que contêm os objetos responsáveis por fazer a comunicação com o banco de dados. Esses objetos são conhecidos como DAO (*Data Access Object*) e serão responsáveis por armazenar e recuperar os dados do banco de dados.

Sobre a arquitetura do banco de dados, conforme explicado anteriormente, o Sigme utiliza o JPA para fazer o mapeamento objeto/relacional e, através desse mapeamento, o próprio JPA irá gerenciar os objetos no banco de dados automaticamente. Com isso, foi utilizada a anotação `@Entity` nas classes do domínio para realizar a persistência dos dados.

4.3 Modelos *FrameWeb*

Nesta seção, os modelos *FrameWeb* que já foram citados na Seção 2.3 serão apresentados. Esses modelos também estão divididos nas camadas da arquitetura do sistema, conforme citado na Seção 4.2.

4.3.1 Modelo de Domínio

O **Modelo de Domínio** é um diagrama de classes da UML que representa os objetos de domínio do problema e seu mapeamento para a persistência em banco de dados relacional. A partir dele são implementadas as classes da camada de domínio na atividade de implementação (SOUZA, 2007).

O modelo de domínio é construído com base no diagrama de classe montado na fase de Análise de Requisitos, adicionando também informações acerca da plataforma de desenvolvimento escolhida, como tipos de atributos e navegabilidade de associações.

Diferente da abordagem original do *FrameWeb* proposta em 2007, todos os atributos que não podem ser nulos tiveram a *tag not null* omitida e os que são nulos tiveram a *tag null* acrescida de forma a diminuir a poluição visual com repetições desnecessárias no diagrama.

Todas as classes de domínio estendem de `PersistentObjectSupport` do JButler, sendo que essa herança não é mostrada nos diagramas com o intuito de não polui-lo com várias associações.

As figuras 8 e 9 apresentam os modelos de domínio para os módulos **Núcleo** e **UtilSigme**, respectivamente.

4.3.2 Modelo de Navegação

O **Modelo de Navegação** é um diagrama de classe da UML que representa os diferentes componentes que formam a camada de Apresentação, como páginas Web, formulários HTML e classes de ação (SOUZA, 2007). Esse modelo é utilizado pelos desenvolvedores para direcionar a codificação das classes e componentes dos pacotes Visão e Controle.

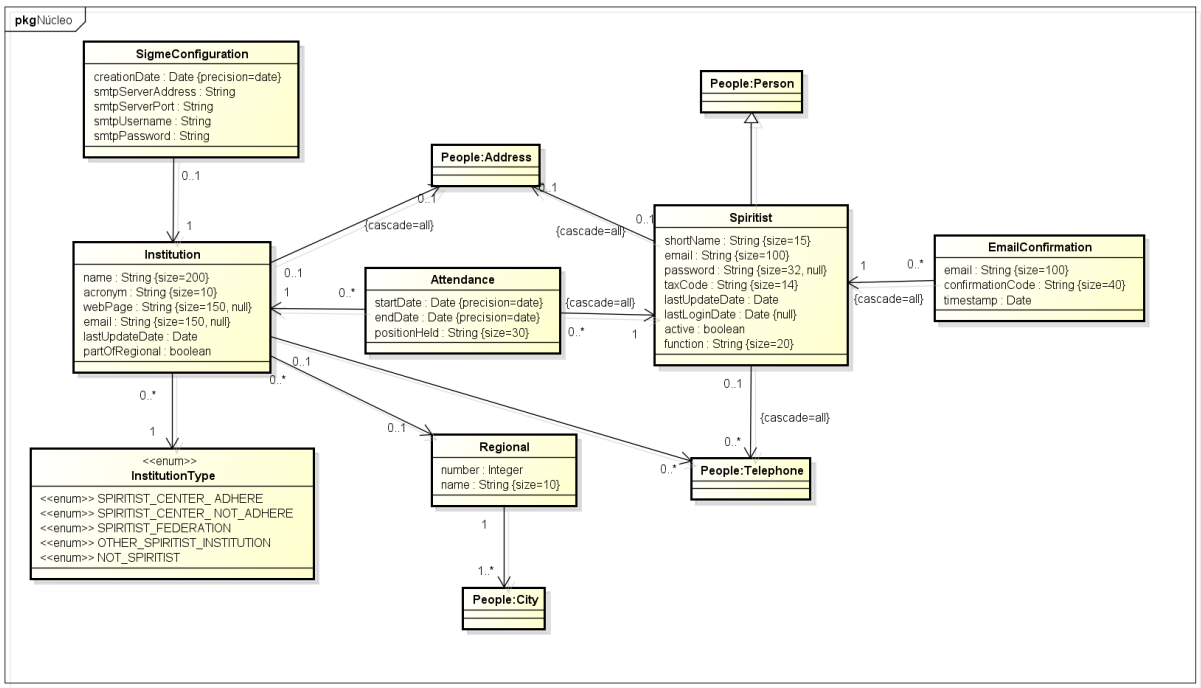

Figura 8 – Modelo de Domínio para o módulo Núcleo.

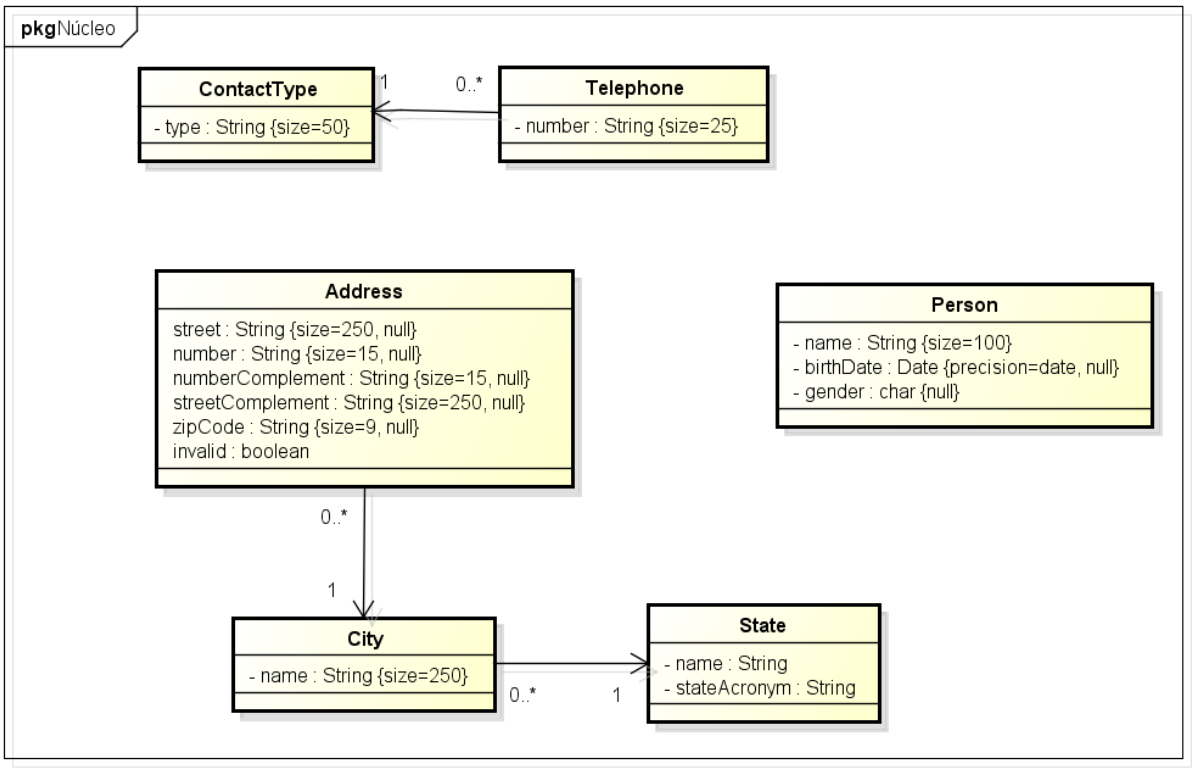

Figura 9 – Modelo de Domínio para o módulo UtilSigme.

Conforme a especificação original do *FrameWeb*, vários estereótipos podem ser utilizados para representar informações nos modelos. Logo abaixo é listado o nome do componente e o significado de cada um presente neste projeto.

- Sem esteriótipo : Uma classe de ação, para qual o *framework* controlador delega a execução da ação.
- *«page»* : Uma página web estática ou dinâmica.
- *«form»* : Um formulário HTML.

Em formulários HTML, atributos representam campos do formulário, que devem ter seus tipos definidos de acordo com a biblioteca de componentes utilizada. Como neste trabalho foi utilizado o *PrimeFaces*, os tipos ficarão com o prefixo "p" (ex.: p.inputText, p.dataTable, etc.). A classe de ação é o principal componente do modelo. Suas associações de dependência ditam o controle de fluxo quando uma ação é executada. Relacionamentos de dependência entre páginas e modelos indicam um link HTML entre as mesmas, enquanto associações de composição entre páginas ou modelos e formulários denotam a presença daquele formulário dentro da página ou modelo.

As funcionalidades criar, editar, excluir e visualizar (abreviadas de CRUD, do inglês *create*, *read*, *update* e *delete*), seguem um mesmo fluxo de execução e de interação com o usuário. Tais funcionalidades são similares para todos os casos de uso cadastrais devido à utilização da ferramenta JButler. Esse fluxo de execução similar é representado pela Figura 10 que é um modelo de navegação genérico, que também serve como o modelo de navegação do JButler.

Deve existir, para cada entidade, uma página chamada `list`, que irá conter a listagem de todos os objetos deste tipo existentes no banco. Essa página possui um formulário relacionado (`manageEntityList`), que armazena todas as entidades sendo mostradas na página citada anteriormente. Ao selecionar uma entidade da tabela, a mesma é atribuída ao atributo `entity` existente em `ManageEntityController`, sendo possível então executar certas ações, tais como: atualizar o objeto selecionado (`update`), ler o objeto selecionado (`retrieve`), e excluir o objeto selecionado (`delete`). Além disso, o usuário pode criar um novo objeto (`create`). Todas estas funcionalidades, ao serem executadas, irão chamar os respectivos métodos existentes em `ManageEntityController`.

Ao executar tais métodos, existem certos retornos, identificados nas relações de dependência saindo de `ManageEntityController`. Por exemplo, ao executar a funcionalidade `create`, o usuário será redirecionado para a página `form`, onde possui um formulário que contém todos os campos necessários para o cadastro de uma nova entidade, chamado `ManageEntityForm`. Nessa página o usuário poderá incluir os dados do novo objeto, e

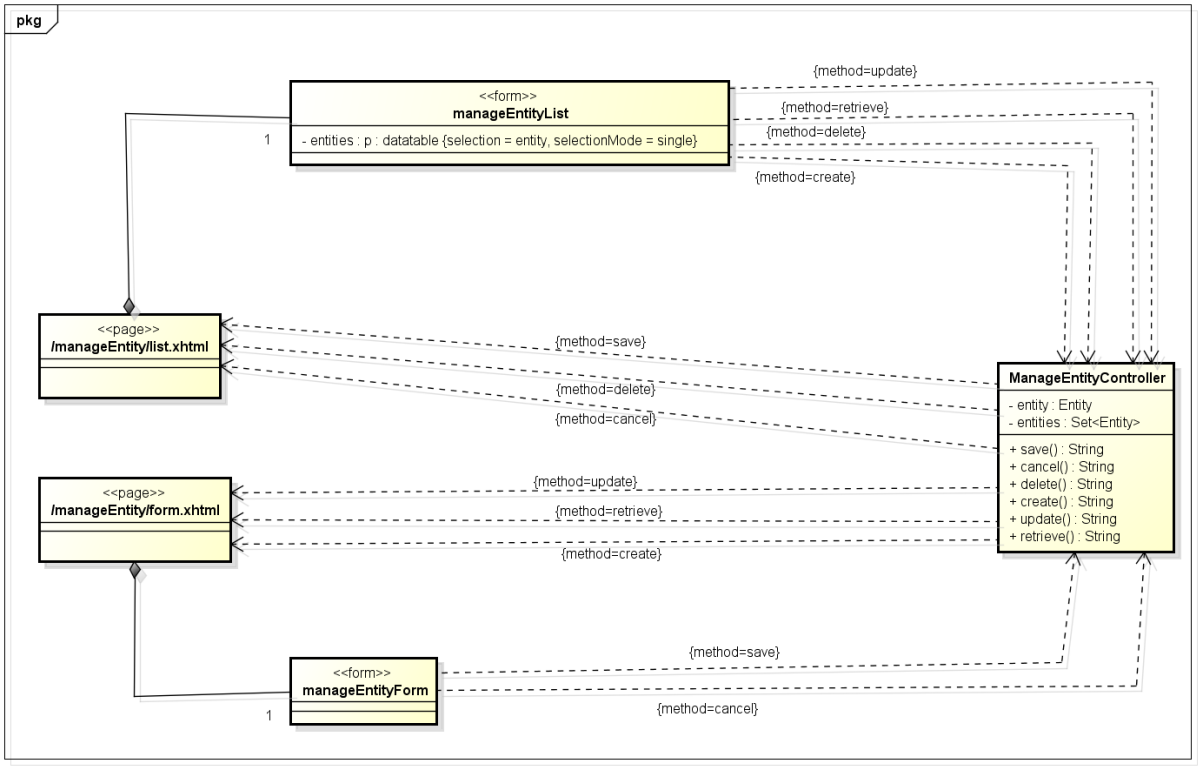

Figura 10 – Modelo de Navegação de um CRUD JButler (LIMA, 2015).

então executar o método `save`, que irá salvar o objeto, e como retorno, será redirecionado para a página inicial (contendo a listagem de todas as entidades existentes no banco).

Para os casos de uso que apresentam funções diferentes de apenas as básicas de cadastro, o modelo de navegação mostrado anteriormente não pode ser aplicado. As Figuras 11 e 12 apresentam os modelos de navegação para os casos de uso **Instalar Sistema** e **Enviar E-mail para Espíritas Inativos**, respectivamente.

Como descrito no caso de uso **Instalar Sistema**, no modelo apresentado na Figura 11, o fluxo inicia-se na página `index` onde o usuário preenche o formulário de informações sobre o administrador do sistema e registra essas informações através da chamada do método `registerAdministrator`. Após isso, a página com o formulário `InstallSystemOwner` é mostrada ao usuário. Ele então preenche o formulário com informações sobre a Instituição principal do sistema e submete essas informações através do método `registerOwnerInstitution`, tendo como resultado o redirecionamento para a página de configurações de envio de email chamada de `smtp`. O usuário então informa os dados sobre o servidor de e-mail e o submete esses dados por meio do método `saveSmtp` e a página final desse fluxo, `done`, é apresentada. Todos os métodos utilizados estão presentes na classe `InstallSystemController`.

No modelo de navegação para o caso de uso **Enviar Email Para Espíritas Inativos** (Figura 12) a primeira página a ser apresentada ao usuário é a listagem de espíritas inativos: `InactiveSpiritistsList`, onde o usuário pode selecionar vários registros.

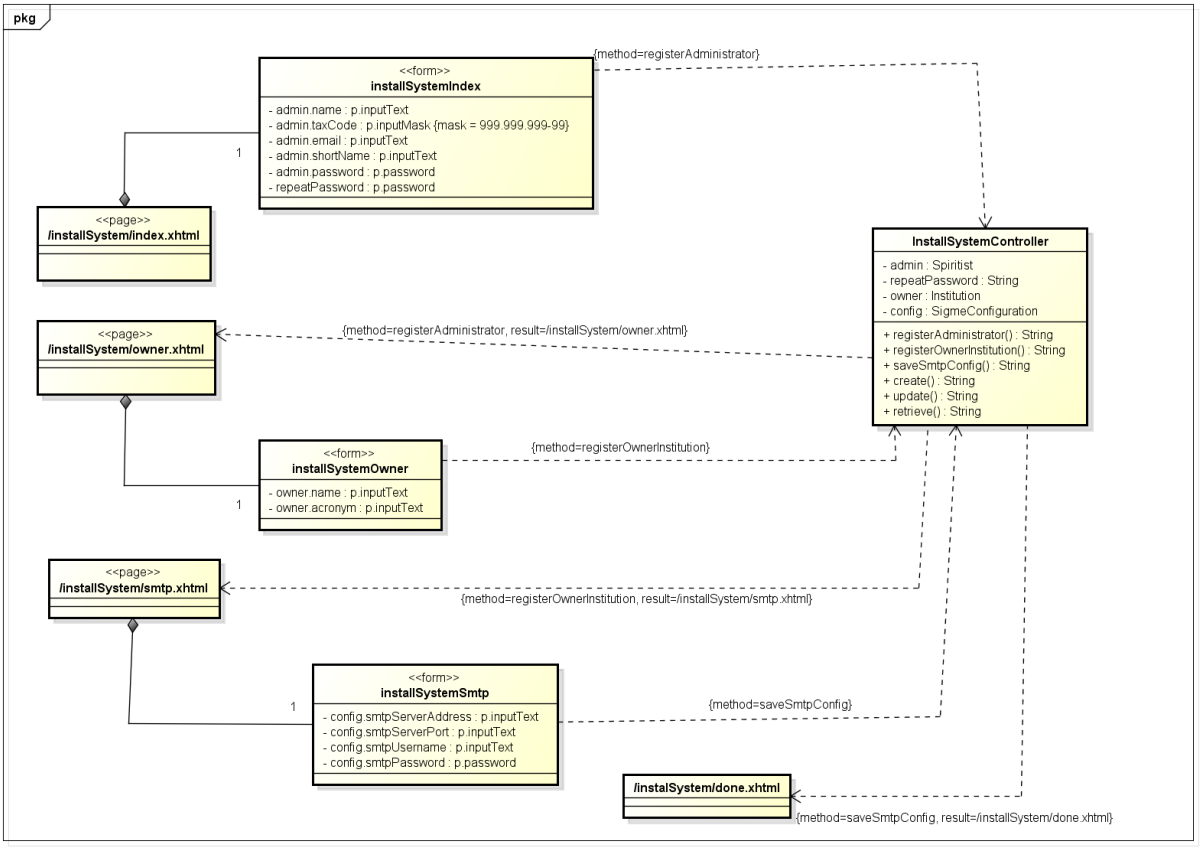

Figura 11 – Modelo de Navegação para o caso de uso **Instalar Sistema**.

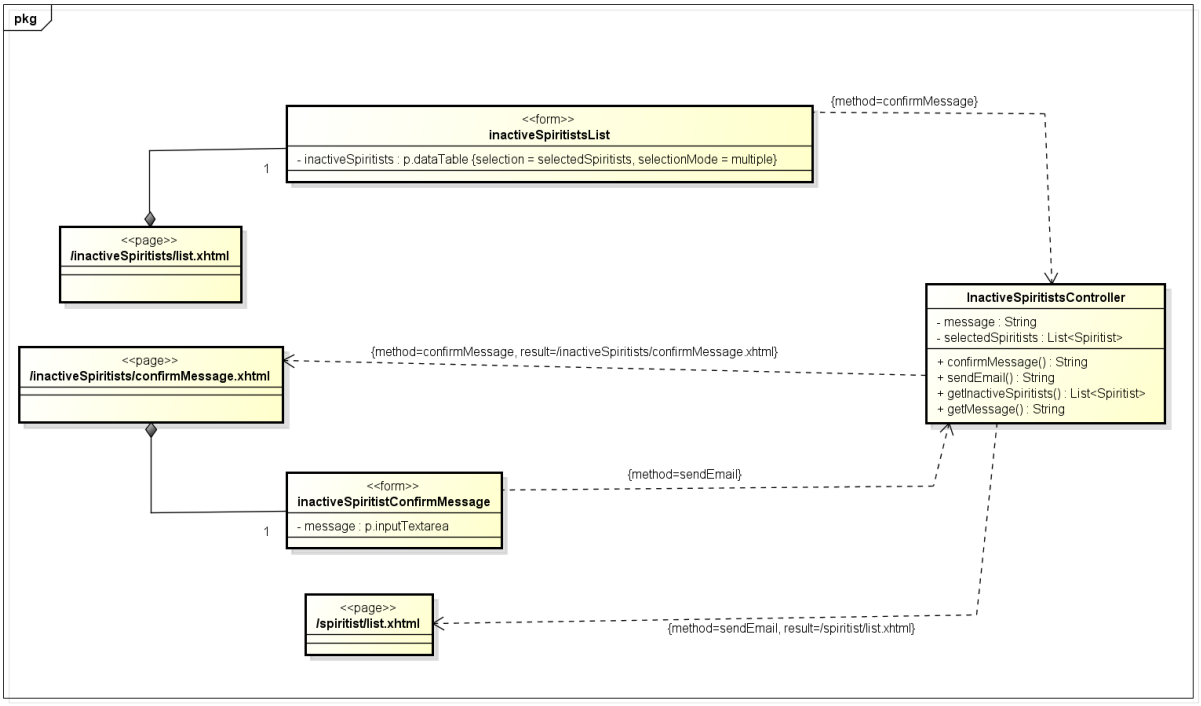

Figura 12 – Modelo de Navegação para o caso de uso **Enviar E-mail para Espíritas Inativos**.

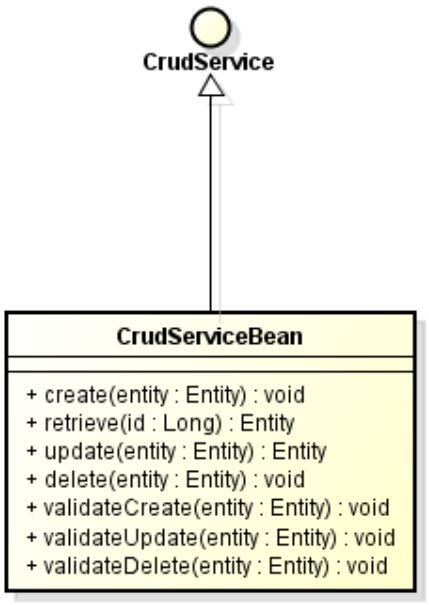

Figura 13 – Modelo de Aplicação genérica da ferramenta JButler (LIMA, 2015).

Esse modo de seleção pode ser visto no atributo `selectionMode = multiple` no componente `p:dataTable` no formulário. Após a seleção, através do método `confirmMessage`, a página `confirmMessage` é apresentada ao usuário onde, é mostrado um componente `p:inputTextarea`, que pertence a um fomulário(`InactiveSpiritistConfirmMessage`) para o usuário inserir a mensagem que quer enviar. Através do método `sendEmail`, e-mails são enviados aos espíritas selecionados e a página de listagem de espíritas inativos é mostrada novamente.

4.3.3 Modelo de Aplicação

O **Modelo de Aplicação**, consiste em um diagrama de classes da UML que representa as classes de serviço, responsáveis pela codificação dos casos de uso, e suas dependências. Esse diagrama é utilizado para guiar a implementação das classes do pacote Aplicação e a configuração das dependências entre os pacotes Controle, Aplicação e Persistência, ou seja, quais classes de ação dependem de quais classes de serviço e quais DAOs são necessários para que as classes de serviço alcancem seus objetivos (SOUZA, 2007).

As classes de aplicação `SpiritistServiceBean` e `InstitutionServiceBean` estendem de `CrudServiceBean` do pacote JButler. Tal classe está representada na Figura 13 de forma genérica. Da mesma forma dos diagramas anteriores essa herança não é mostrada no diagrama acima com o intuito de não poluir o diagrama com várias associações. A Figura 14 mostra o modelo de aplicação para o módulo **Núcleo**.

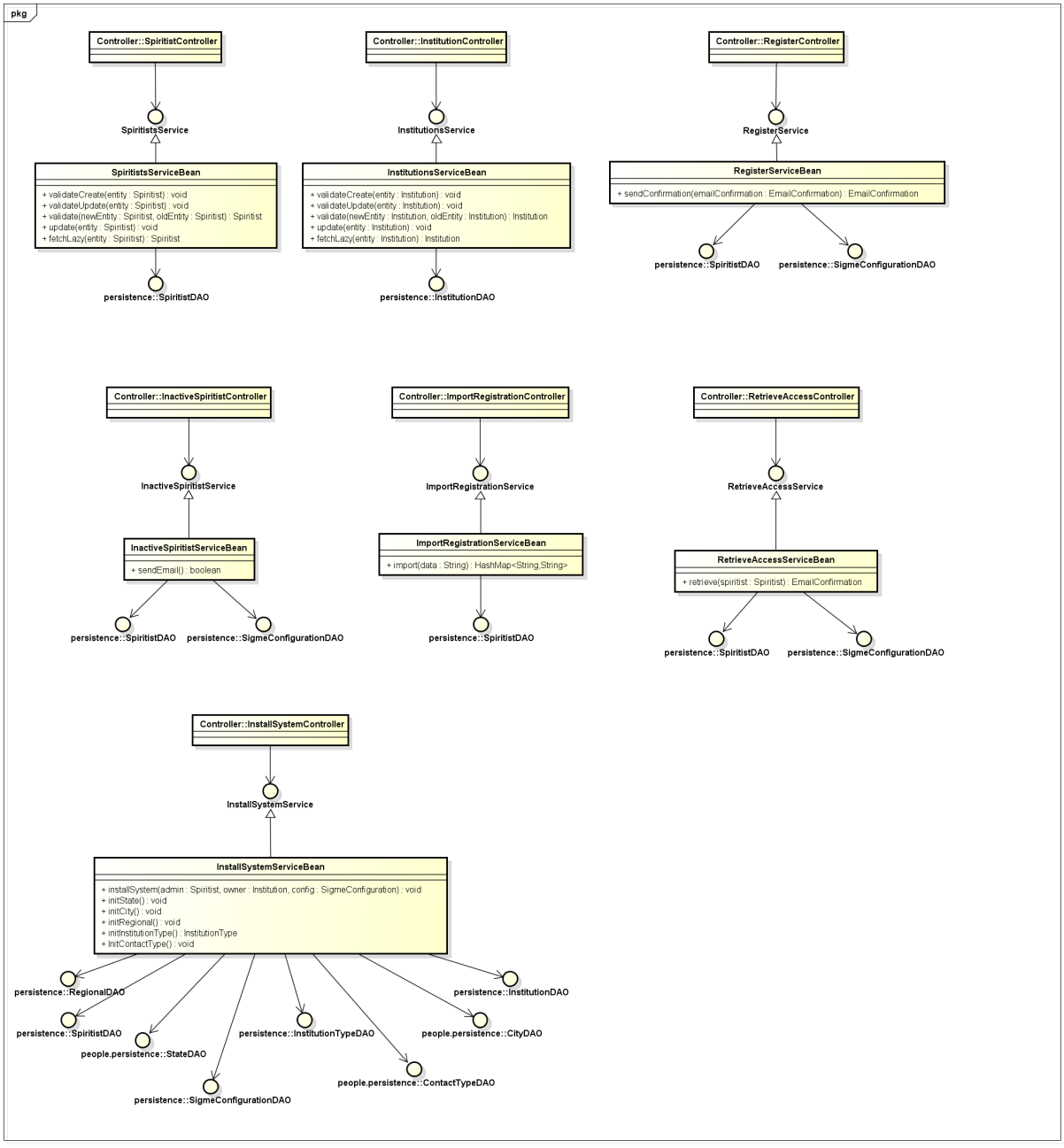

Figura 14 – Modelo de Aplicação para o módulo **Núcleo**.

4.3.4 Modelo de Persistência

O **Modelo de Persistência** é um diagrama de classes UML que representa os DAOs existentes, responsáveis pela persistência das instâncias das classes de domínio. Esse diagrama guia a construção das classes DAO, que pertencem à **Lógica de Acesso a Dados** (SOUZA, 2007).

O **Modelo de Persistência** é composto por todas as classes do domínio mapeadas para um banco de dados através do mapeamento objeto/relacional feito pelo JPA. Seguindo o padrão DAO, essas classes são oferecidas, por meio de interfaces, para o nível superior (de aplicação) cujas operações possíveis são ilustradas no modelo (COSTA, 2015).

Existem diversas funcionalidades que são comuns a qualquer classe (ou objeto) que possa ser salvo em um banco de dados: listar todos os objetos, excluir um objeto, salvar um objeto, atualizar um objeto, retornar um objeto dado um identificador, etc. Com o objetivo de evitar a repetição destas funcionalidades em cada DAO, o JButler nos provê um DAO base que implementa as principais operações utilizadas em um banco de dados. Tanto a interface `BaseDAO` quanto a classe `BaseJPADAO` são declaradas usando tipos genéricos, deixando a cargo de suas sub-interfaces e sub-classes a especificação a classe gerenciada por cada DAO para permitir que qualquer classe herde as funcionalidades existentes em seus DAOs base.

Segundo os padrões estabelecidos por *FrameWeb*, todas as interfaces DAO são subinterfaces de `BaseDAO`, enquanto todas as implementações JPA são sub-classes de `BaseJPADAO`, ambas do pacote JButler, herdando todos os métodos básicos, por exemplo: `retrieveAll()`, `save()`, `delete()`, `retrieveById()`, etc. Os demais métodos que foram declarados no diagrama se referem a consultas específicas que devem ser disponibilizadas para o funcionamento de determinados casos de uso (último passo do processo de construção do **Modelo de Persistência**). A Figura 15 exibe o modelo de persistência base do JButler.

Vale notar que o nome das classes já indica qual tecnologia de persistência foi utilizada, esse sistema de nomenclatura é mais uma sugestão do *FrameWeb* para simplificar o processo de software. As figuras 16 e 17 mostram os diagramas de persistências para os módulos **Núcleo** e **UtilSigme**.

Note que a relação de herança entre os DAOs específicos e o DAO base não é representada explicitamente nos diagramas para evitar poluição visual. Esta também é uma recomendação do *FrameWeb*. Além disso, como é possível perceber, o Modelo de Persistência não define nenhuma extensão da UML para representar os conceitos necessários da camada de acesso a dados, mas apenas regras que tornam essa modelagem mais simples e rápida, por meio da definição de padrões.

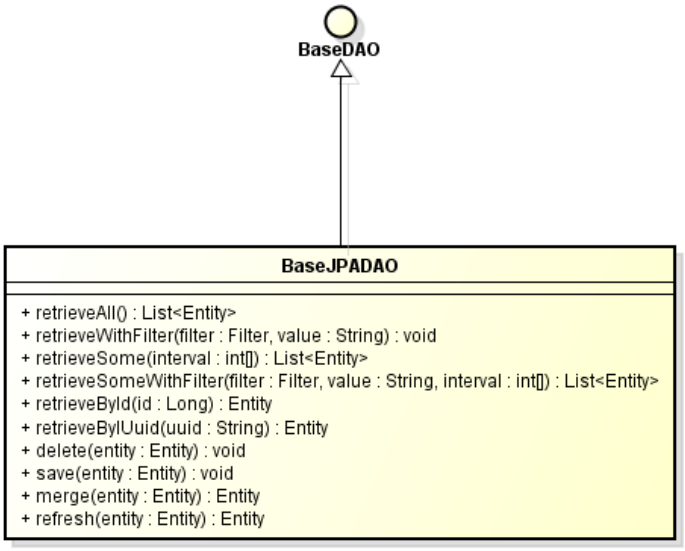

Figura 15 – Modelo de Persistência genérico da ferramenta JButler (LIMA, 2015).

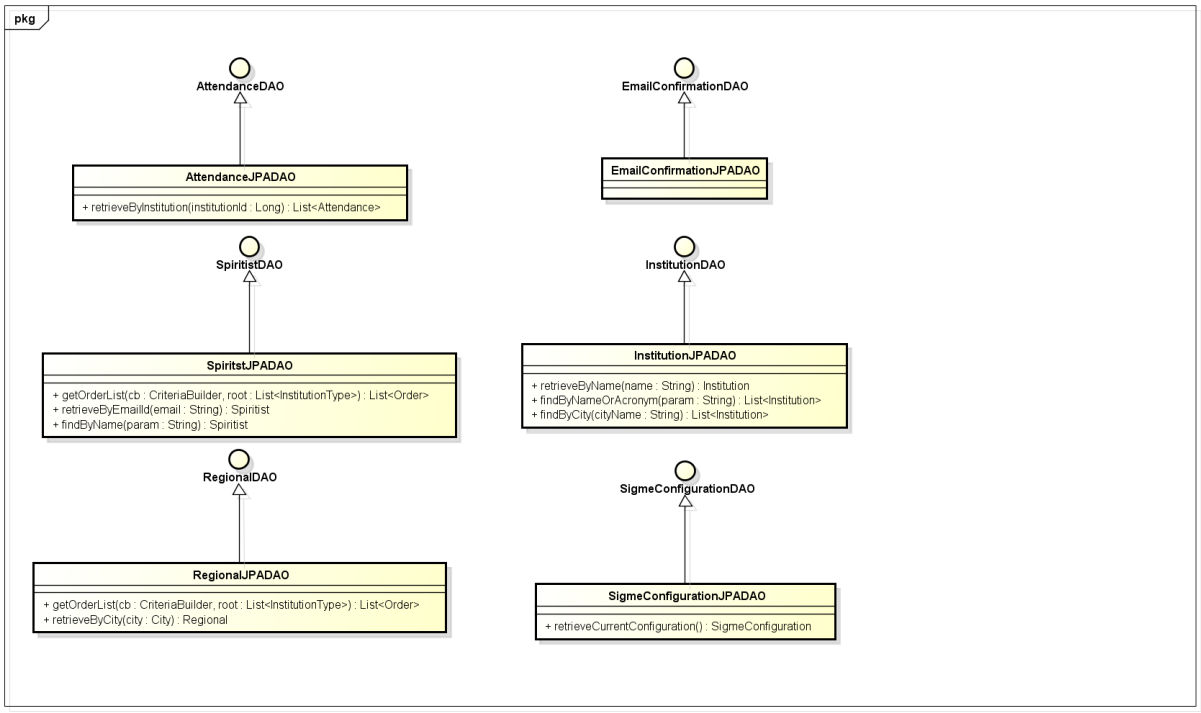

Figura 16 – Modelo de Persistência para o módulo **Núcleo**.

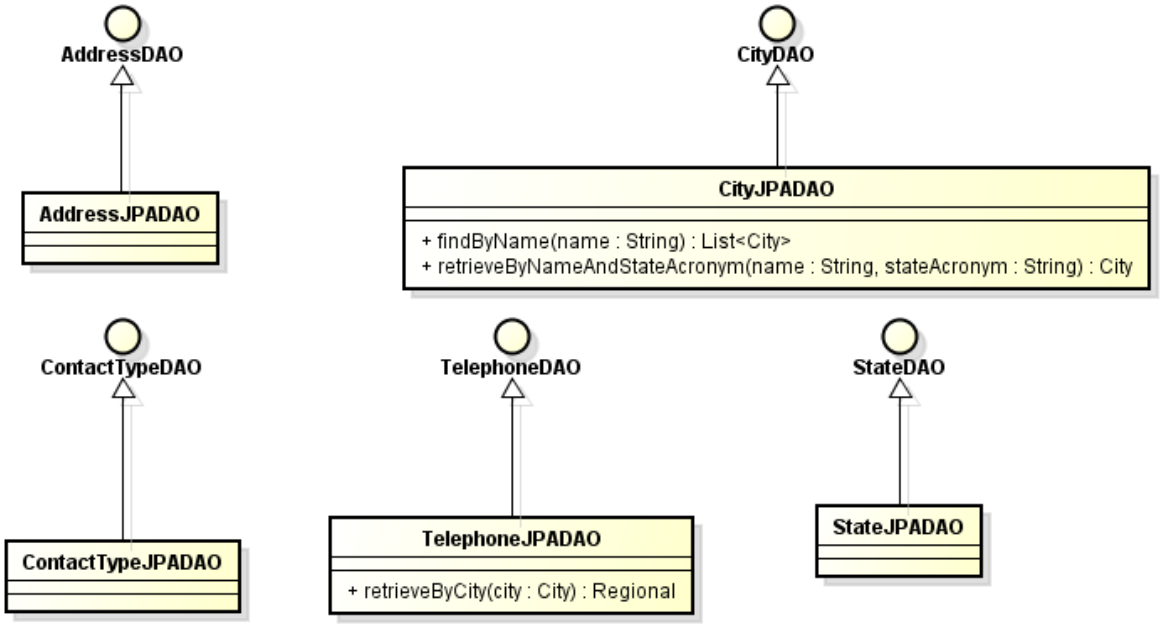

Figura 17 – Modelo de Persistência para o módulo **UtilSigme**.

4.4 Apresentação do Sistema

Depois da definição do escopo do projeto, dos requisitos funcionais e não funcionais na fase de Análise de Requisitos, como mostrado no Capítulo 3, das tecnologias utilizadas, da Arquitetura de Software e dos modelos construídos, como mostrado nas Seções 4.1, 4.2 e 4.3, a fase de implementação é responsável por agrupar todas essas informações de forma consistente a fim de construir o sistema. Esta seção tem por objetivo mostrar o resultado final do projeto, indicando suas principais funcionalidades a partir de captura de telas.

O primeiro contato com o sistema é feito na primeira execução do sistema, em que o usuário através de um link acessa telas para a configuração inicial do Sigme, que permitem informar dados de configuração do administrador do sistema, da instituição principal e do servidor de e-mails utilizado. As figuras 18 e 19 mostram o fluxo dessas telas.

A Figura 20 mostra a tela inicial do Sigme. Uma vez que as configurações iniciais foram salvas anteriormente, o usuário verá duas opções no menu lateral: A primeira delas é a opção Entrar em que o usuário será redirecionado para a tela de *login*, que permite a autenticação no sistema para a utilização da ferramenta. O usuário informa o e-mail e a senha (Figura 21) para ter acesso ao sistema.

A segunda é funcionalidade de o usuário se cadastrar no sistema: o usuário informa primeiro um e-mail (Figura 22) que será usado para acesso ao sistema. Depois informa o código de confirmação recebido por e-mail (Figura 23) e termina seu cadastro informando seu nome completo, CPF, nome preferido e a senha (Figura 24).

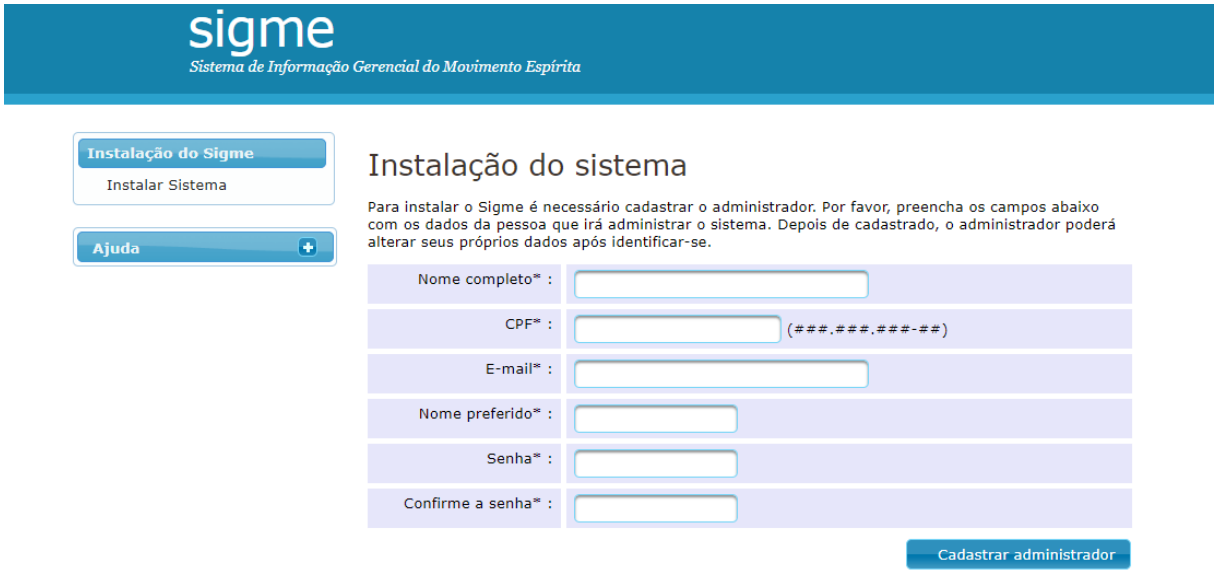

Figura 18 – Tela de cadastro do Administrador.

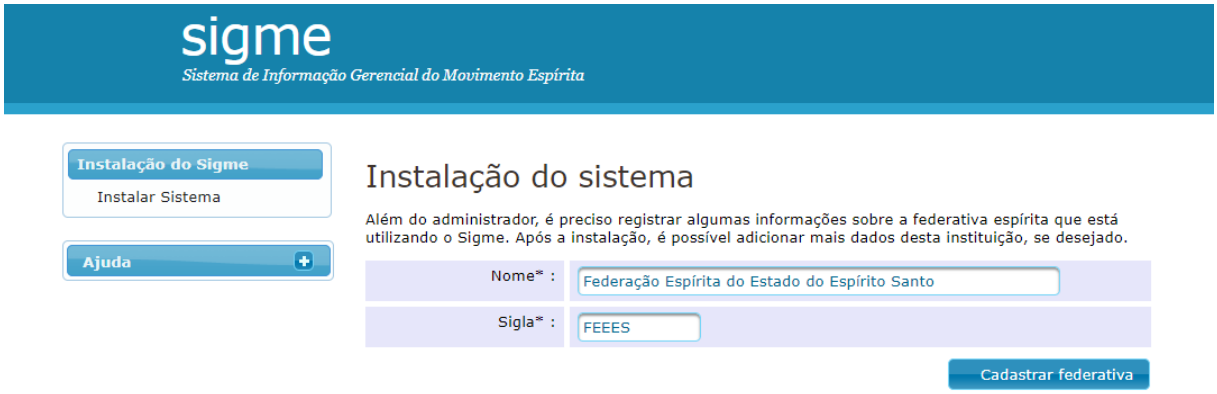

Figura 19 – Tela de cadastro da Instituição Principal.

As funcionalidades do tipo CRUD possuem telas que seguem exatamente o mesmo padrão e fluxo. Para evitar repetição, apenas uma das telas será mostrada, exibindo as funcionalidades de listagem, cadastro, leitura, atualização e exclusão de um espírita. A partir da tela de listagem (Figura 25), ao selecionar um registro, alguns botões são habilitados na parte inferior. Clicando em **Consultar**, **Alterar** ou **Adicionar**, o usuário será redirecionado para a tela de cadastro de um item conforme pode ser visto na Figura 26. Importante notar que ao clicar em **Consultar**, os campos do formulário serão somente leitura. Ao clicar em **Alterar** os campos do formulário já virão preenchidos com as informações do espírita selecionado, sendo possível editar as mesmas. E por último, ao clicar em **Adicionar**, o formulário será exibido em branco, possibilitando a criação de um novo Espírita.

A funcionalidade **Excluir** pode ser acionada a partir da tela de listagem e possibilita

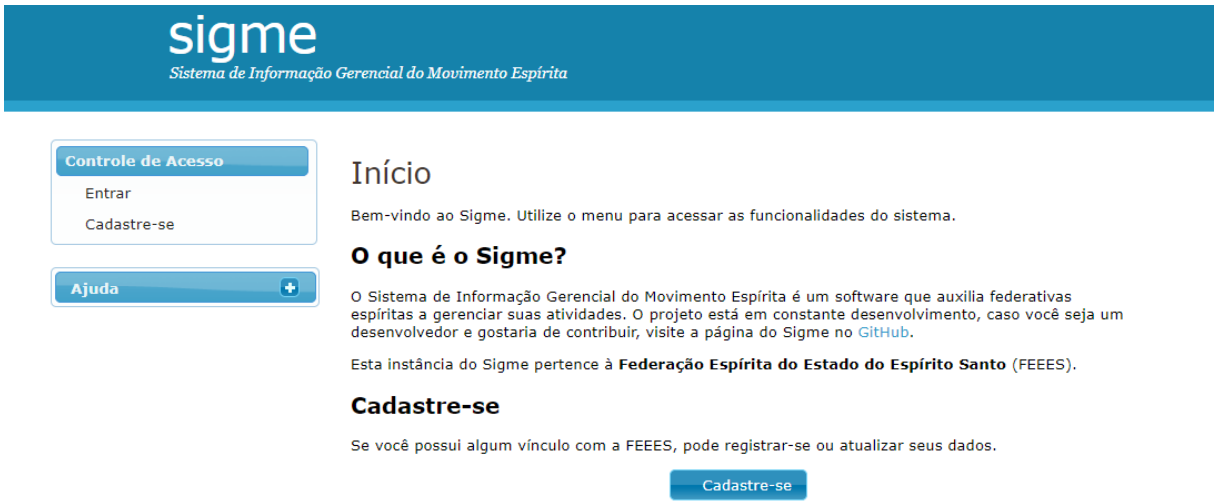

Figura 20 – Tela inicial após configuração inicial do sistema.

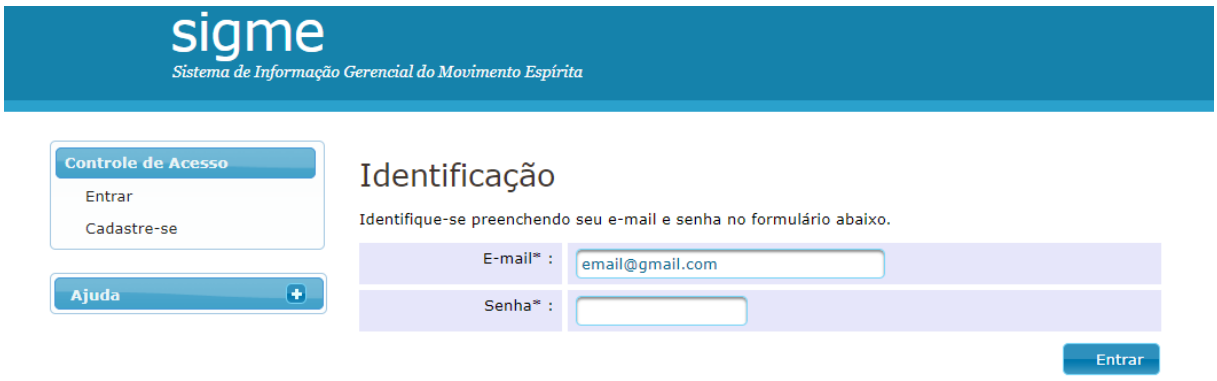

Figura 21 – Tela de *login*.

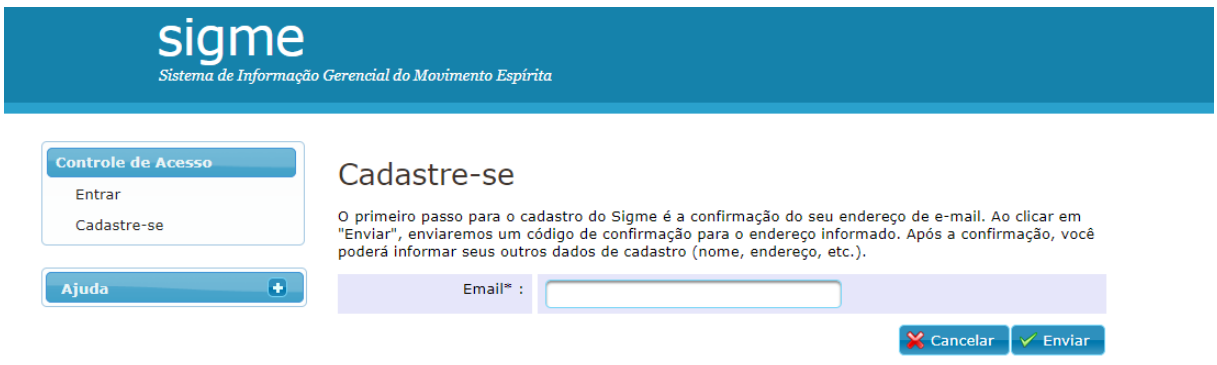

Figura 22 – Tela em que o usuário informa o e-mail para se cadastrar.

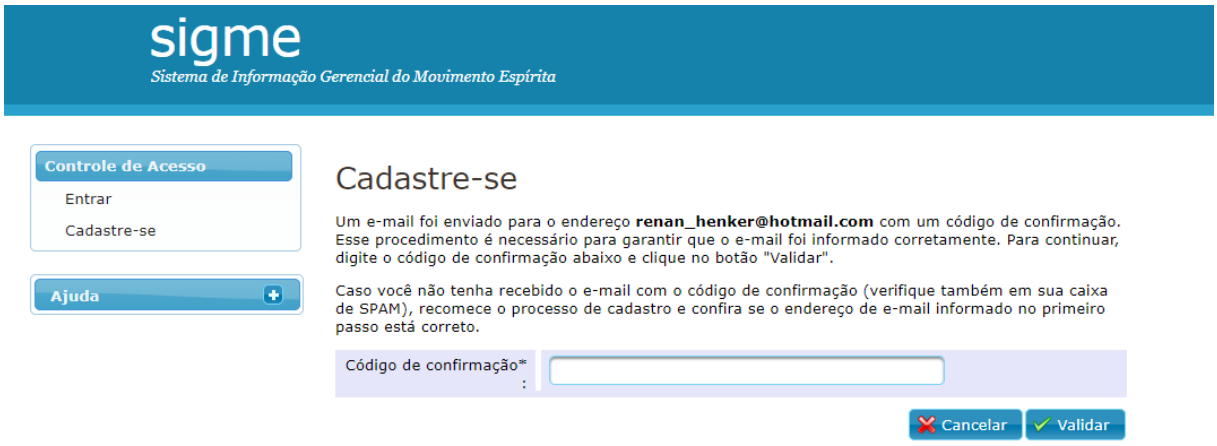

Figura 23 – Tela inicial após configuração inicial do sistema.

Figura 24 – Tela de cadastro das informações do novo usuário.

Figura 25 – Tela de listagem de espíritas.

Figura 26 – Tela de cadastro das informações do novo usuário.

a exclusão de um item (neste caso, de um espírita). Ao clicar no botão, um novo painel será exibido, para confirmação da exclusão. O usuário poderá, então, confirmar a mesma, excluindo o item, ou cancelar. A Figura 27 exibe tal comportamento para o CRUD **Espírita**.

Para os casos de uso que fogem do escopo de cadastros básicos a estrutura é um pouco diferente. Para a tela de *Envio de E-mail para espíritas Inativos*, por exemplo, não é possível nem criar um novo registro, alterar nem excluir os existentes. Para acessar essa funcionalidade, é necessário clicar no botão **Enviar e-mail** na tela de listagem de espíritas. Assim, o usuário será redirecionado para a tela em questão onde é possível ver em formato de tabela os espíritas inativos, conforme mostra a Figura 28. Para enviar o e-mail, o usário pode alterar uma mensagem padrão ou enviar assim mesmo, no campo de mensagem (Figura 29). Clicando no botão **Enviar** os e-mails são enviados.

Figura 27 – Tela de cadastro das informações do novo usuário.

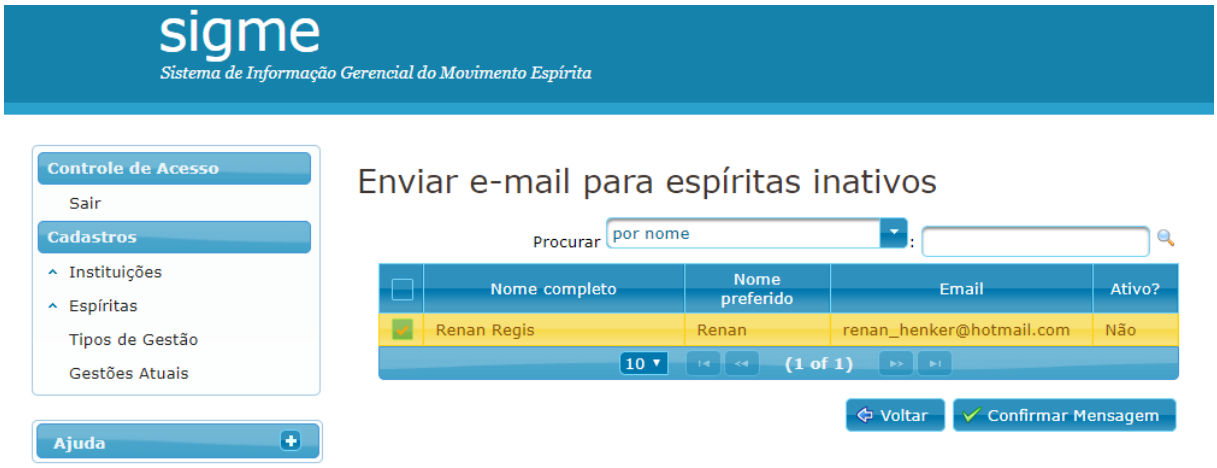

Figura 28 – Tela de listagem de espíritas inativos.

Figura 29 – Tela de listagem de espíritas inativos.

5 Considerações Finais

Este capítulo apresenta as conclusões do trabalho realizado, mostrando suas contribuições. Por fim, são apresentadas suas limitações e perspectivas de trabalhos futuros.

5.1 Conclusões

Devido à grande quantidade processos de cadastros de espiritas e instituições feitos manualmente pelos funcionários de cada instituição ligada a FEEES, surgiu a necessidade da criação de um sistema Web capaz de informatizar e auxiliar a realização desses processos a fim de reduzir o tempo gasto com essa atividade e obter um melhor controle das informações registradas.

Todos os objetivos levantados no Capítulo 1 foram atingidos, exceto algumas partes da implementação devido ao período de tempo e condições do autor. Toda a documentação proposta foi produzida de acordo com os padrões de Engenharia de Software. Foram levantados os requisitos, em seguida foi feita a análise de tais requisitos, resultando no Documento de Especificação de Requisitos. Este documento contém informações sobre requisitos funcionais, não funcionais, regras de negócio, descrição do sistema, definição de atores, casos de uso e diagramas de classe. Após a finalização do documento de requisitos finalizado, foi criado o Documento de Projeto, contendo todas as informações relacionadas à arquitetura do sistema e foram criados os modelos propostos na Seção 2.3, seguindo a abordagem *FrameWeb*.

Ao final de todas as etapas, ficou evidente a dificuldade e a complexidade de ter um software integralmente documentado. Além de que qualquer alteração seja no escopo, é necessário atualizar vários documentos a fim de manter a consistência. Porém, ficou clara a importância de ter feito tal controle para a manutenção e expansão do sistema.

Uma das maiores dificuldades encontradas foi o aprendizado do *FrameWeb*, um modelo que não é tão simples de ser inteiramente compreendido. Apesar da dificuldade, o *FrameWeb* foi um método bastante útil, pois sua modelagem é bem próxima do comportamento do sistema na Web.

Por fim, é importante citar como as disciplinas cursadas durante o curso ajudaram no desenvolvimento deste trabalho, Engenharia de Software, Projeto de Sistemas, Programações, Lógica, Banco de Dados, dentre outras. Uma das coisas mais difíceis no final de um curso é conseguir utilizar todos os conhecimentos adquiridos durante o curso e colocá-los em prática. Este trabalho proporcionou isso, sendo possível colocar em prática boa parte das disciplinas estudadas durante o curso.

5.2 Limitações e Perspectivas Futuras

A partir dos resultados alcançados, algumas limitações e melhorias podem ser observadas, o que dá margem para a realização de trabalhos futuros. Sendo assim, alguns trabalhos surgirão a partir deste. Dentre as possíveis melhorias e limitações, são eles:

- Atualizar os *frameworks* utilizados no Sigme para melhoria, seja de segurança, facilidade de desenvolvimento, etc. Devido a evolução dos *frameworks* para aplicações Web ser cada vez mais constante, o Sigme se encontra defasado nesse sentido;
- Adicionar a funcionalidade para gerenciar e controlar permissões e perfis de usuário para que os usuários possam interagir com cada funcionalidade dentro de sua alçada dentro do sistema;
- Adicionar funcionalidade para geração de relatórios informativos;
- Adicionar funcionalidade para gerenciamento de eventos relacionados ao movimento espírita: local, data tipo de reunião, etc..;
- Adicionar a gerência do site da instituição relacionada, dando aos responsáveis pelas diferentes áreas da federativa (presidência, vice-presidências, secretaria, tesouraria, departamentos, regionais, etc.) a capacidade de adicionar, editar e excluir conteúdo sua responsabilidade no site da instituição.

Referências

BAUER, C.; KING, G. Hibernate in action. Manning Greenwich, 2005. Citado na página 25.

COSTA, T. M. *Resgate - Despacho de ambulância automatizado*. Monografia (Projeto de Graduação) — Universidade Federal do Espírito Santo, 2015. Citado na página 35.

DEMICHIEL, L.; SHANNON, B. Java platform, enterprise edition 7 (java ee 7) specification. *JSR, JCP*, 2013. Citado na página 25.

FALBO, R. A. *Projeto de Sistemas*. [s.n.], 2011. 68 p. Disponível em: <http://www.inf.ufes.br/~falbo/files/Notas_Aula_Projeto_Sistemas_2.pdf>. Citado 2 vezes nas páginas 15 e 16.

FALBO, R. A. *Engenharia de Requisitos: Notas de Aula*. 2017. Disponível em: <https://inf.ufes.br/~falbo/files/ER/Notas_Aula_Engenharia_Requisitos.pdf>. Acesso em: 15.11.2018. Citado 3 vezes nas páginas 14, 15 e 23.

FOWLER, M. *Patterns of Enterprise Application Architecture*. 2. ed.. ed. [S.l.]: Addison-Wesley, 2002. Citado na página 16.

JENDROCK, E. et al. *The Java EE 7 Tutorial*. [S.l.]: Addison-Wesley Professional, 2014. v. 1. Citado na página 24.

LIMA, L. V. F. *SAP - Sistema de Apoio ao Professor*. Monografia (Projeto de Graduação) — Universidade Federal do Espírito Santo, 2015. Citado 5 vezes nas páginas 6, 26, 31, 33 e 36.

MANN, K. D. *Java Server Faces in Action*. [S.l.]: Dreamtech Press, 2005. Citado na página 24.

MARTINS, B. F. S. *Uma abordagem dirigida a modelos para o projeto de Sistemas de Informação Web com base no método FrameWeb*. [S.l.], 2016. Citado na página 16.

PFLEEGER, S. L. Engenharia de software: teoria e prática. *2ª Edição, Prentice Hall*, 2004. Citado na página 14.

PRESSMAN, R. S. *Engenharia de Software - Uma abordagem profissional*. 7ª edição. ed. [S.l.]: McGraw-Hill, 2011. ISBN 9788563308337. Citado na página 15.

SOMMERVILLE, I.; KOTONYA, G. *Requirements engineering: processes and techniques*. [S.l.]: John Wiley & Sons, Inc., 1998. Citado na página 14.

SOMMERVILLE, I. et al. *Engenharia de software*. [S.l.]: Addison Wesley São Paulo, 2003. v. 6. Citado na página 14.

SOUZA, V. E. S. *FrameWeb: um Método baseado em Frameworks para o Projeto de Sistemas de Informação Web*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, 2007. Citado 7 vezes nas páginas 11, 14, 16, 17, 28, 33 e 35.

Apêndices

Documento de Requisitos

Projeto: Módulo de Cadastros Básicos do **Sigme –** Sistema de Informação Gerencial do Movimento Espírita

Registro de Alterações:

Versão	Responsáveis	Data	Alterações
0.1	Renan Vitor Henker Regis	22/04/2015	Versão parcial inicial
1.0	Renan Vitor Henker Regis	19/08/2018	Versão Final

1. Introdução

Este documento apresenta os requisitos de usuário do módulo de cadastros básicos da ferramenta Sigme – Sistema de Informação Gerencial do Movimento Espírita – e está organizado da seguinte forma: a seção 2 contém uma descrição do propósito do sistema; a seção 3 apresenta uma descrição do minimundo apresentando o problema; e a seção 4 apresenta a lista de requisitos de usuário levantados junto ao cliente.

2. Descrição do Propósito do Sistema

O Sigme tem como propósito informatizar e auxiliar a realização das principais atividades da Federação Espírita do Estado do Espírito Santo (Feees) que são realizadas por meio de seus departamentos, acompanhados pelos vice-presidentes, além da tesouraria e secretaria, todos coordenados pela presidência. As principais atividades são o cadastro de pessoas, espíritas ou não, e o cadastro de instituições.

3. Descrição do Minimundo

A Federação Espírita do Estado do Espírito Santo (Feees) é uma organização religiosa, que objetiva unificar, orientar, coordenar e dinamizar o movimento espírita no Estado. O escopo do sistema proposto se refere à parte de cadastro de pessoas e instituições, que é base para muitas outras funcionalidades e é parte do núcleo do sistema. Ele é o responsável por registrar informações sobre pessoas físicas e jurídicas associadas à federativa, sejam elas espíritas ou não, e ainda registrar informações sobre instituições.

Os objetivos de se registrar toda esta gama de entidades são vários, dentre eles: gerar etiquetas para correspondência, gerar boletos para cobrança, saber quem são as pessoas que ocupam os diversos cargos, gerar um site das casas espíritas adesas, e integrar, de várias outras maneiras, as casas espíritas e o público espírita ao sistema. Tais funcionalidades estão fora do escopo deste projeto, sendo desenvolvidas em outros módulos do Sigme.

De todas as instituições espera-se registrar: nome, sigla, endereço, endereço na Internet (site), e-mail, telefones de contato (número e tipo – principal, celular, fax, etc.),

tipo de instituição (casa espírita não-adesa, casa espírita adesa, federativa espírita, outra instituição espírita, não-espírita) e a data da última atualização dos dados cadastrais.

Das pessoas físicas espera-se registrar: nome completo, nome curto, e-mail (também utilizado como login), senha, endereço, telefones (números e tipos), data de nascimento, sexo e a data da última atualização dos dados cadastrais. Dos espíritas, espera-se registrar também as instituições que frequenta (ou frequentou) e se exerce (ou exerceu) algum cargo nestas instituições.

Um endereço, seja de instituição ou pessoa física, é composto de: rua, número, complemento, bairro, cidade, estado, CEP e uma indicação de se o endereço está retornando correspondências.

No módulo de cadastro básicos são incluídas as seguintes funcionalidades:

a) Um administrador pode gerenciar cadastros de instituições e espíritas;
b) Um visitante não identificado pode se cadastrar e tornar-se um usuário do Sigme com cadastro ativo se o e-mail informado ainda não foi cadastrado;
c) Um espírita ativo pode a qualquer momento alterar, desativar e reativar seu cadastro;
d) Um administrador pode importar cadastros de outras bases de dados que a instituição já possua (assumindo que possam ser exportados para um formato padronizado);
e) Um espírita cadastrado no sistema pode a qualquer momento recuperar seu acesso ao Sigme.
f) Um administrador pode solicitar que o sistema envie por e-mail notificação a todos os cadastros inativos (que ainda não receberam notificação deste tipo) informando-os que podem ativar seus cadastros se assim desejarem;

Além das funcionalidades acima, é esperado que o Sigme proveja um mecanismo de controle de papéis que permita indicar as pessoas que atualmente ocupam cargos específicos nas instituições. Esse mecanismo deve permitir identificar, por exemplo, quem são os administradores do sistema.

4. Requisitos de Usuário

Tomando por base o contexto do sistema, foram identificados os seguintes requisitos de usuário:

Requisitos Funcionais

Identificador	Descrição	Prioridade	Depende de	Issue Github[1]
RF01	O sistema deve permitir o controle dos cadastros de instituições pelo administrador.	Alta		#6
RF02	O sistema deve permitir o controle dos cadastros de espíritas pelo administrador.	Alta	RN01, RN03, RN05	#4
RF03	O sistema deve permitir uma busca por cadastros de usuários duplicados.	Média	RF02, RF07	#27
RF04	O sistema deve permitir enviar e-mail para usuários que estão com cadastro inativo (e que nunca foram notificados a respeito).	Média	RF02, RN04	#24
RF05	O sistema deve permitir a importação de cadastros de outras bases de dados (Access, Excel, etc.).	Média	RF02, RNF08	#26
RF06	O sistema deve permitir a recuperação do acesso ao sistema de um espírita cadastrado.	Média	RF03	
RF07	O sistema deve permitir que visitantes se cadastrem como espíritas, incluindo a confirmação de posse do endereço de e-mail informado.	Alta	RN02	#19
RF08	O sistema deve permitir a ativação, desativação e alteração dos cadastros de espíritas, sendo que a alteração do endereço de e-mail exige confirmação.	Alta	RF07, RF02, RN05	#21, #22, #23 e #25
RF09	O sistema deve permitir que no primeiro acesso possa ser informados dados relativos a instituição principal, ao usuário administrador do sistema e as informações de envio de e-mails.	Alta	RF01, RF02, RF06	

Regras de Negócio

Identificador	Descrição	Prioridade	Depende de
RN01	Um espírita cadastrado pelo administrador encontra-se inativo e deve ativar o seu cadastro através do código recebido por e-mail para utilizar o sistema.	Alta	
RN02	Um visitante não identificado pode se cadastrar e tornar-se um usuário do Sigme com cadastro ativo se o e-mail informado ainda não foi cadastrado.	Alta	
RN03	Quando um espírita não possui senha, seu cadastro encontra-se automaticamente inativo.	Média	

[1] Refere-se ao número da *issue* no *issue tracker* do Sigme criado no Github: https://github.com/feees/Sigme/issues

RN04	Para efetivar o cadastro ou a alteração de e-mail é necessário que o visitante/usuário informe um código de confirmação recebido por e-mail.	Alta	

Requisitos Não Funcionais

Identificador	Descrição	Categoria	Escopo	Prioridade	Depende de
RNF01	O sistema deve prover controle de acesso às informações, evitando que elas sejam alteradas por pessoas não identificadas.	Segurança de Acesso	Sistema	Alta	
RNF02	A ferramenta deve ser de aprendizado fácil, não sendo necessário nenhum treinamento especial para seu uso.	Facilidade de Aprendizado	Sistema	Média	
RNF03	A ferramenta deve ser de fácil operação, não sendo necessário uso contínuo para uma boa operação do sistema.	Facilidade de Operação	Sistema	Alta	
RNF04	A ferramenta deve ter o cadastro de instituições e espíritas divididos e apresentados em partes de modo que não assuste o usuário com formulários muito grandes e complexos.	Atratividade	Sistema	Média	
RNF05	O sistema deve estar disponível como uma aplicação Web, acessível a partir dos principais navegadores disponíveis no mercado.	Portabilidade	Sistema	Alta	
RNF06	O sistema deve ser fácil de manter, de modo a acomodar novas funcionalidades.	Manutenibilidade	Sistema	Alta	
RNF07	O desenvolvimento do sistema deve explorar o potencial de reutilização de componentes, tanto no que se refere ao desenvolvimento com reúso quanto ao desenvolvimento para reúso.	Reusabilidade	Sistema	Média	
RNF08	O sistema deve ser projetado de forma a permitir integração com outra base de dados como, por exemplo o Access ou Excel, por meio de formatos não proprietários (como CSV).	Interoperabilidade	Sistema	Média	

Documento de Especificação de Requisitos

Projeto: **Sigme –** Sistema de Informação Gerencial do Movimento Espírita (Cadastros Básicos)

Registro de Alterações:

Versão	Responsáveis	Data	Alterações
1.0	Renan Vitor Henker Regis	14/07/2015	Versão parcial inicial, contendo as seções 1, 2, 3.1 e 4.1
1.1	Renan Vitor Henker Regis	02/05/2018	Atualização das seções 1,2,3,4 e inclusão da seção 5
1.2	Renan Vitor Henker Regis	05/05/2018	Atualização das seções 4 e 5
1.3	Renan Vitor Henker Regis	10/09/2018	Versão final.

1. Introdução

Este documento apresenta a especificação dos requisitos do módulo de cadastros básicos da ferramenta Sigme – Sistema de Informação Gerencial do Movimento Espírita. A atividade de análise de requisitos foi conduzida aplicando-se técnicas de modelagem de casos de uso, modelagem de classes e modelagem de comportamento dinâmico do sistema. Os modelos apresentados foram elaborados usando a UML. Este documento está organizado da seguinte forma: a seção 2 apresenta os subsistemas identificados, mostrando suas dependências na forma de um diagrama de pacotes; a seção 3 apresenta o modelo de casos de uso, incluindo descrições de atores, os diagramas de casos de uso e descrições de casos de uso; a seção 4 apresenta o modelo conceitual estrutural do sistema, na forma de diagramas de classes; a seção 5 apresenta o modelo comportamental dinâmico do sistema, na forma de diagramas de estado; finalmente, a seção 6 apresenta o glossário do projeto, contendo as definições das classes identificadas.

2. Identificação de Subsistemas

Devido a complexidade do módulo proposto para o presente projeto, não foi necessário dividir em subsistemas.

3. Modelo de Casos de Uso

O modelo de casos de uso visa capturar e descrever as funcionalidades que um sistema deve prover para os atores que interagem com o mesmo. Os atores identificados no contexto deste projeto estão descritos na tabela abaixo.

Tabela 1 – Atores.

Ator	Descrição
Administrador	Funcionário da Feees responsável pelo controle do núcleo de cadastros do sistema.
Espírita	Usuário cadastrado no Sigme.

A seguir, são apresentados os diagramas de casos de uso e descrições associadas.

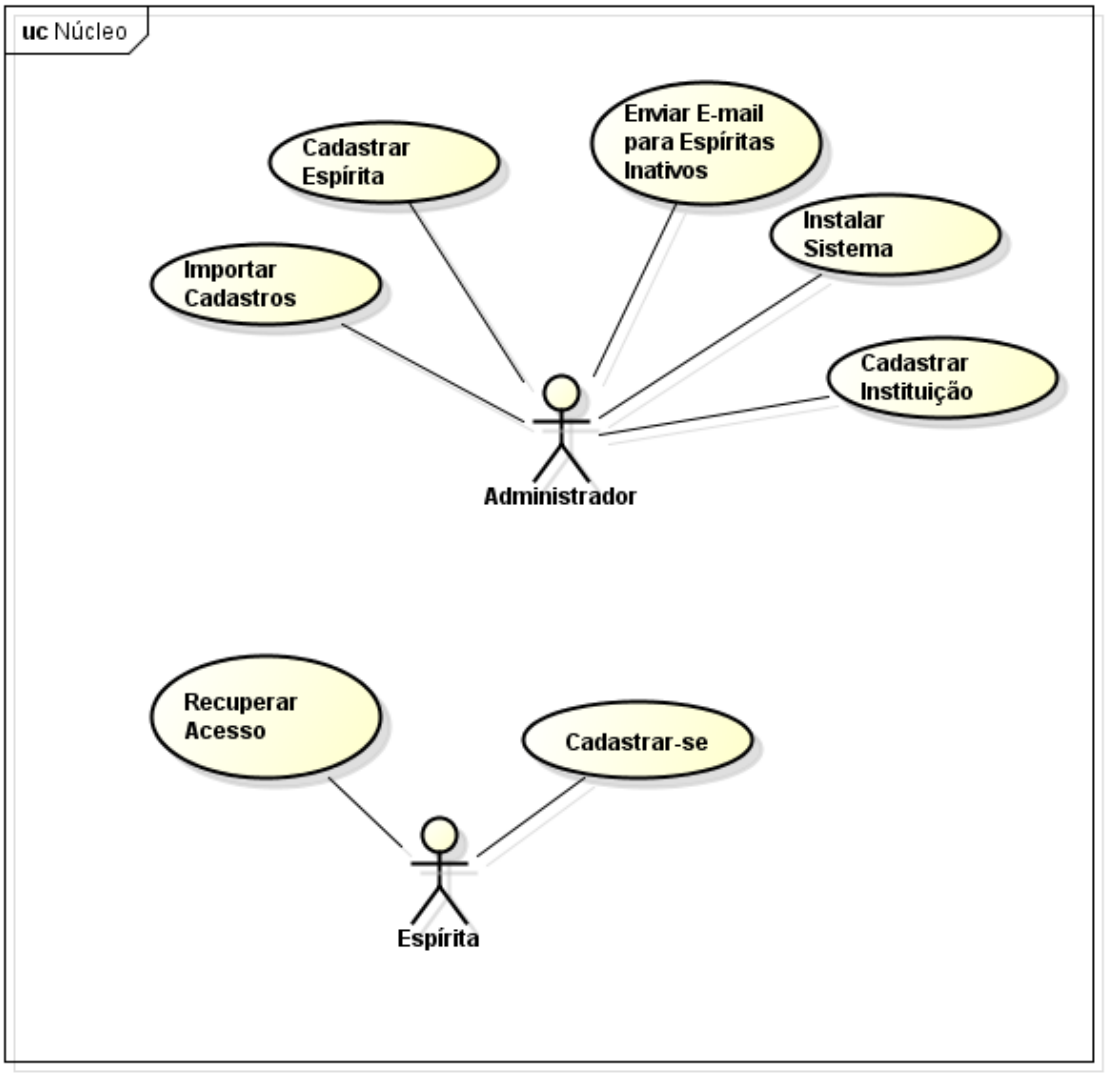

Figura 1 – Diagrama de Casos de Uso.

A seguir, são apresentadas as descrições de cada um dos casos de uso identificados. Os casos de uso cadastrais de baixa complexidade, envolvendo inclusão, alteração, consulta e exclusão, são descritos na tabela abaixo, segundo o padrão da organização.

Tabela 3 – **Casos de Uso Cadastrais.**

Subsistema	***Núcleo***				
Identificador	**Caso de Uso**	**Ações Possíveis**	**Observações**	**Requisitos**	**Classes**
UC01	**Cadastrar Instituição**	I,A,C,E	[I]: Informar nome, sigla, endereço, endereço na internet (site), telefones de contato, tipo de instituição e a regional (se fizer parte de uma).	RF01	Instituição, Tipo de Instituição, Endereço, Telefone
UC02	**Cadastrar Espírita**	I,A,C,E	[I]:Informar nome, CPF, e-mail, nome preferido, data de nascimento, sexo, endereço, telefones e instituições que frequentou ou frequenta	RF02, RN01	Espírita, Instituição, Frequência, Endereço, Telefone

A seguir, são apresentados os casos de uso de maior complexidade que não puderam ser descritos segundo os formatos tabulares simplificados. Esses casos de uso são descritos segundo o padrão de descrição completa de casos de uso definido.

Descrição de Caso de Uso

Projeto: Sigme
Identificador do Caso de Uso: UC03
Caso de Uso: Instalar Sistema
Descrição Sucinta: Este caso de uso permite que seja cadastrado o administrador, a instituição principal do sistema e as informações de envio de e-mails.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Instalar Sistema		1. O ator informa os dados do administrador do sistema a ser cadastrado: nome completo, CPF, e-mail, nome preferido, senha e confirmação da senha. 2. O sistema registra um novo espírita como administrador. 3. O ator informa nome e sigla de uma instituição. 4. O sistema registra a instituição como a principal. 5. O ator informa o endereço do servidor de e-mail, porta, usuário e senha. 6. O sistema registra os dados de envio de e-mail.

Fluxos de Eventos de Exceção

Nome do Fluxo de Eventos Normal Relacionado	Condição de Exceção	Descrição
Instalar Sistema	1 - Dados inválidos	1 – Uma mensagem de erro é retornando ao passo 1 para a correção dos dados.
Instalar Sistema	3 - Dados inválidos	3 – Uma mensagem de erro é retornando ao passo 1 para a correção dos dados.
Instalar Sistema	5 - Dados inválidos	5 – Uma mensagem de erro é retornando ao passo 1 para a correção dos dados.

Requisitos Relacionados: RF01, RF02, RF09

Classes Relacionadas: Instituição, Configuração do Sigme, Espírita

Descrição de Caso de Uso

Projeto: Sigme
Identificador do Caso de Uso: UC04
Caso de Uso: Enviar E-mail para Espíritas Inativos
Descrição Sucinta: Este caso de uso permite que seja enviado e-mail para espíritas que estão com o seu cadastro inativo.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Enviar E-mail Para Cadastros Inativos		1. O sistema apresenta a lista de espíritas cadastrados no sistema cujos cadastros estejam inativos. 2. O administrador marca ou desmarca os cadastros que receberão o e-mail e clica no botão de enviar. 3. O sistema apresenta uma mensagem padrão para envio, sendo possível que administrador altere a mensagem a ser enviada se quiser. 4. O sistema pede a confirmação do envio do e-mail. 5. O administrador confirma o envio do e-mail. 6. O sistema envia um e-mail para todos espíritas com cadastro inativo, que foram previamente selecionados.

Fluxos de Eventos Variantes

Nome do Fluxo de Eventos Normal Relacionado	Variante	Descrição
Enviar E-mail Para Cadastros Inativos	2 – Nenhum Espírita selecionado	3 – O sistema informa que o administrador deve selecionar ao menos um Espírita e retorna ao passo 1

Requisitos Relacionados: RF04, RN01, RN03

Classes Relacionadas: Espírita, Endereço, Telefone

Descrição de Caso de Uso

Projeto: Sigme
Identificador do Caso de Uso: UC05
Caso de Uso: Importar Cadastros
Descrição Sucinta: Este caso de uso permite que seja importado cadastrados de espíritas de outras bases de dados (em formato CSV).

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Importar Cadastros		1. O administrador seleciona e informa ao sistema de qual arquivo em formato CSV serão importados os cadastros. 2. O sistema apresenta as informações dos espíritas do arquivo CSV em uma tabela. 3. O sistema pede ao administrador a confirmação do cadastro de todos espíritas da tabela. 4. O administrador confirma o cadastro. 5. O sistema registra os cadastros de espíritas.

Fluxos de Eventos Variantes

Nome do Fluxo de Eventos Normal Relacionado	Variante	Descrição
Importar Cadastros	2 – Existem cadastros diferentes com o mesmo e-mail e/ou CPF.	2 – O sistema apresenta os cadastros de espíritas que possuem alguma informação repetida. 3 – Para cada cadastro o administrador seleciona que ação deseja tomar: 3.1 – Informa que deseja cancelar a importação daquele cadastro. 3.2 – Informa que deseja alterar o e-mail e/ou CPF do cadastro a ser importado. 3.3 – Informa que deseja sobrescrever os dados do cadastro existente com os do cadastro a ser importado. 4 – O sistema apresenta os cadastros a serem realizados com as alterações feitas pelo administrador. 5 - O sistema pede ao administrador a confirmação do cadastro de todos espíritas da tabela. 6 - O administrador confirma o cadastro. 7 - O sistema registra os cadastros de espíritas.

Fluxos de Eventos de Exceção

Nome do Fluxo de Eventos Normal Relacionado	Condição de Exceção	Descrição
Importar Cadastros	1 – Formato de arquivo inválido.	1a – O sistema retorna ao passo 1 e informa ao administrador que deve ser selecionado outro arquivo.

Requisitos Relacionados: RF02, RF03, RF05

Classes Relacionadas: Espírita, Endereço, Telefone

Descrição de Caso de Uso

Projeto: Sigme
Identificador do Caso de Uso: UC06
Caso de Uso: Cadastrar-se
Descrição Sucinta: Este caso de uso permite que um usuário não identificado se cadastre no Sigme e espíritas possam consultar, alterar, ativar e desativar seu cadastro.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Incluir Novo Espírita	Não estar cadastrado no sistema.	1. Um visitante não identificado informa o e-mail. 2. Se o e-mail informado não foi cadastrado ainda no sistema, um código de confirmação para o cadastro é enviado para este e-mail pelo sistema. 3. O visitante informa o código recebido por e-mail. 4. O visitante informa nome completo, CPF, nome preferido, a senha e a confirmação da senha. 5. O sistema registra o visitante como espírita.
Ativar Cadastro	Já é um espírita com cadastrado inativo no sistema.	1. O espírita informa o CPF, a senha e a confirmação da senha. 2. O sistema marca o cadastro do espírita como ativo.
Desativar Cadastro	Já é um espírita com cadastrado ativo no sistema.	1. O espírita informa que deseja desativar seu cadastro. 2. O sistema altera o cadastro do espirita para inativo.
Alterar Cadastro	Já é um espírita com cadastrado ativo no sistema.	1. O espírita informa os novos dados, sendo que o e-mail não pode ser alterado ainda pois necessita de confirmação por e-mail. 2. O sistema registra os novos dados.
Alterar E-mail	Já é um espírita com cadastrado ativo no sistema.	1. O espírita informa que deseja alterar o e-mail. 2. O espírita informa o novo e-mail. 3. O sistema envia um código de confirmação para o e-mail informado. 4. O espírita informa o código recebido por e-mail. 5. O sistema registra a alteração de e-mail.

Fluxos de Eventos Variantes

Nome do Fluxo de Eventos Normal Relacionado	Variante	Descrição
Incluir Novo Espírita	2 – O e-mail informado já está cadastrado no sistema.	2a.1 – O sistema informa que existe ao menos um cadastro com o mesmo e-mail. 2a.2 – O visitante informa que deseja recuperar o acesso. 2a.3 – Incluir *Recuperar Acesso*.

Fluxos de Eventos de Exceção

Nome do Fluxo de Eventos Normal Relacionado	Condição de Exceção	Descrição
Incluir Novo Espírita	3 – Código de confirmação incorreto.	3 – O sistema retorna ao passo 3 e informa visitante que é necessário informar um código válido.

Incluir Novo Espírita	4 – Dados inválidos.	4 – O sistema retorna ao passo 4 e informa ao visitante para entrar com os dados corretos.
Ativar Cadastro	1– Dados inválidos.	1 – O sistema retorna ao passo 1 e informa ao visitante para entrar com os dados corretos.
Alterar Cadastro	1– Dados inválidos.	1 – O sistema retorna ao passo 1 e informa ao visitante para entrar com os dados corretos.
Alterar E-mail	4 – Código de confirmação incorreto.	4 – O sistema retorna ao passo 4 e informa visitante que é necessário informar um código válido.

Requisitos Relacionados: RF07, RF08, RN01, RN02, RN03, RN04

Classes Relacionadas: Espírita, Endereço, Telefone, Frequência

Descrição de Caso de Uso

Projeto: Sigme
Identificador do Caso de Uso: UC07
Caso de Uso: Recuperar Acesso
Descrição Sucinta: Este caso de uso permite que um espírita recupere o acesso ao Sigme.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Recuperar Acesso	Estar cadastrado no sistema.	1. Um visitante não identificado informa o e-mail utilizado no cadastro. 2. O sistema envia um link de acesso ao sistema para alterar a senha do espírita ao e-mail informado. 3. O espírita informa duas vezes a nova senha para confirmar. 4. A senha do espírita é alterada pelo sistema. 5. O sistema redireciona o espírita para tela de login.

Fluxos de Eventos Variantes

Nome do Fluxo de Eventos Normal Relacionado	Variante	Descrição
Recuperar Acesso	1 – Não existe espíritas cadastrados com o e-mail informado.	1 – O sistema informa que não existe espíritas cadastrados com o e-mail informado. 2 – O sistema redireciona o visitante ao passo 1.

Fluxos de Eventos de Exceção

Nome do Fluxo de Eventos Normal Relacionado	Condição de Exceção	Descrição
Recuperar Acesso	1 – Dados inválidos.	1 – O sistema retorna ao passo 1 e informa o visitante que é necessário informar um e-mail válido.
Recuperar Acesso	3 – Senhas não conferem.	3 – O sistema retorna ao passo 3 e informa ao espírita para entrar com senhas iguais.

Requisitos Relacionados: RF06, RF07, RF08, RN01, RN02, RN03, RN04

Classes Relacionadas: Espírita, Endereço, Telefone, Frequência

4. Modelo Estrutural

O modelo conceitual estrutural visa capturar e descrever as informações (classes, associações e atributos) que o sistema deve representar para prover as funcionalidades descritas na seção anterior. A seguir, são apresentados os diagramas de classes identificados no contexto deste projeto. Na seção 5 – Glossário de Projeto – são apresentadas as descrições das classes presentes nos diagramas apresentados nesta seção.

A Figura 2 apresenta o diagrama de classes.

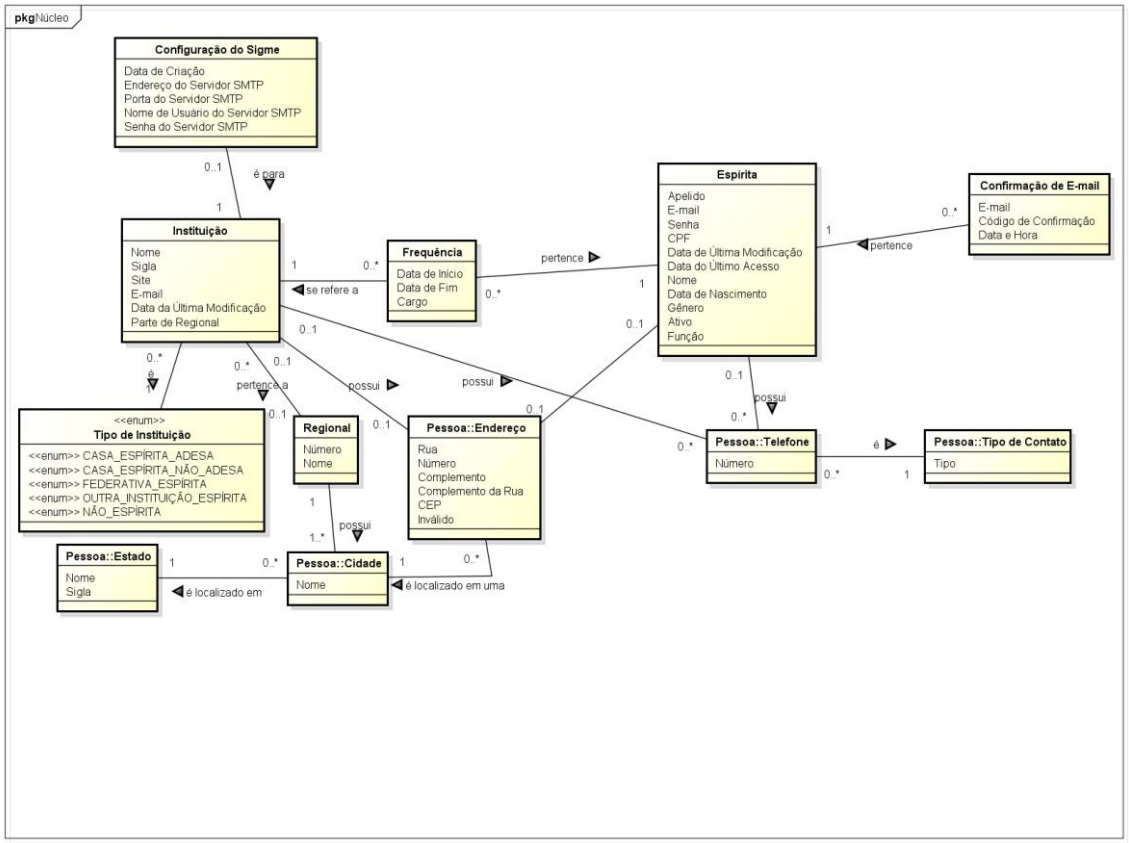

Figura 2 – Diagrama de Classes.

As seguintes restrições de integridade devem ser observadas:

- RI1: Um Telefone não pode pertencer, ao mesmo tempo, a uma instituição e a um espírita.
- RI2: Um Endereço não pode pertencer, ao mesmo tempo, a uma instituição e a um espírita.
- RI3: Uma Instituição não pode estar numa Cidade que não faça parte da sua regional.

5. Modelo Dinâmico

O modelo dinâmico visa capturar o comportamento dinâmico do sistema.

A seguir, são apresentado o diagrama de estados elaborado no contexto desse projeto.

A figura 3 apresenta o diagrama de estados da classe Espírita.

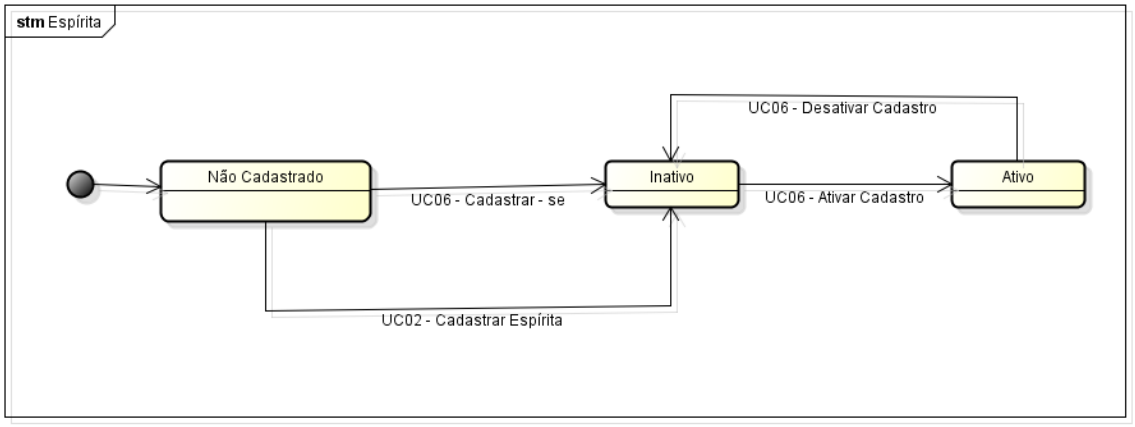

Figura 3 – Diagrama de Estados da Classe Espírita.

6. Glossário de Projeto

Esta seção apresenta as definições dos principais conceitos envolvidos no projeto.

- **Espírita:** classe que armazena as informações sobre espíritas cadastrados.
  - Apelido: Nome preferido do espírita.
  - E-mail: E-mail do espírita.
  - Senha: senha de acesso do espírita ao sistema.
  - CPF: CPF do espírita.
  - Data da última modificação: Data da última modificação do cadastro.
  - Data do último acesso: Data da última vez que o espírita acessou o sistema.
  - Nome: Nome do espírita.
  - Data de nascimento: Data de nascimento no formato DD/MM/AAAA
  - Gênero: Gênero do Espírita
  - Ativo: Indicador se o espírita está ativo ou inativo no sistema
  - Função: Tipo de função do usuário no sistema (Administrador, usuário comum, etc...).
- **Confirmação de E-mail**: classe que armazena as informações sobre os e-mails enviados como confirmação de um cadastro, ativação ou inativação de um cadastro.
  - E-mail: descrição do e-mail enviado.
  - Código de Confirmação: Código enviado para a confirmação.

    - Data e Hora: data e hora do envio.
- **Telefone**: classe que representa o telefone de um espírita ou instituição.
    - Número: número do telefone.
- **Tipo do Contato**: classe que representa o tipo de contato do telefone (Residencial, comercial, etc...).
    - Tipo: descrição do tipo.
- **Instituição:** classe que armazena as informações sobre as instituições espíritas cadastradas.
    - Nome: Nome da instituição.
    - Sigla: Sigla para a instituição.
    - Site: Endereço eletrônico principal da instituição.
    - E-mail: E-mail principal utilizado da instituição.
    - Data de última modificação: Data da última modificação do cadastro.
    - Parte de Regional: indica se esse tipo é parte de alguma regional.
- **Regional:** uma regional congrega as instituições espiritas de uma região.
    - Número: número da regional.
    - Nome: nome da regional.
- **Frequência:** classe que indica quanto tempo um espirita esteve associado a uma instituição.
    - Data de Início: data de entrada do espirita na instituição.
    - Data de Fim: data de saída do espirita da instituição.
    - Cargo: Cargo que o espírita ocupou no período indicado.
- **Configuração do Sigme:** classe que armazena informações sobre configurações relacionadas a envio de e-mails.
    - Data de Criação: Data em que foi feita a configuração do sigme.
    - Endereço do Servidor SMTP: endereço eletrônico do servidor de e-mail.
    - Porta do Servidor SMTP: Porta utilizada para envio de e-mails do sistema.
    - Nome do Usuário do Servidor SMTP: nome do usuário do servidor de e-mail.
    - Senha do Servidor SMTP: senha utilizada no servidor de e-mail.
- **Endereço:** classe que representa endereços.
    - Rua: nome da rua.
    - Número: número do endereço.
    - Complemento: informação adicional do endereço.
    - Complemento da rua: informação adicional da rua.
    - CEP: código postal do endereço.
    - Inválido: indicador do estado em que se encontra o endereço: sim ou não.

- **Cidade**: cidade em que se localiza a instituição ou espirita.
  - Nome: nome da cidade.
- **Estado:** estado em que se localiza a instituição ou espirita.
  - Nome: nome do estado.
  - Sigla: sigla do estado.

6.1 Tipos de Dados Específicos de Domínio

- **Tipo de Instituição**: tipos que uma instituição pode ser. Tipo enumerado que pode assumir os seguintes valores: CASA_ESPÍRITA_ADESA, CASA_ESPÍRITA_NÃO_ADESA, FEDERATIVA_ESPÍRITA, OUTRA_INSTITUIÇÃO_ESPÍRITA E NÃO_ESPÍRITA.

Documento de Projeto de Sistema

Projeto: **Sigme** - Sistema de Informação Gerencial do Movimento Espírita (Cadastros Básicos)

Registro de Alterações:

Versão	Responsáveis	Data	Alterações
1.0	Renan Vitor Henker Regis	12/05/2018	Versão Inicial, contendo as seções 1,2,3,4 e 5.
1.1	Renan Vitor Henker Regis	28/09/2018	Versão final.

1. Introdução

Este documento apresenta o documento de projeto (*design*) do módulo de cadastros básicos, também chamado de "Núcleo" neste documento, da ferramenta Sigme. Este documento está organizado da seguinte forma: a seção 2 apresenta a plataforma de software utilizada na implementação da ferramenta; a seção 3 trata de táticas utilizadas para tratar requisitos não funcionais (atributos de qualidade); a seção 4 apresenta o projeto da arquitetura de software; por fim, a seção 5 apresenta o projeto dos componentes da arquitetura.

2. Plataforma de Desenvolvimento

Na Tabela 1 são listadas as tecnologias utilizadas no desenvolvimento da ferramenta, bem como o propósito de sua utilização.

Tabela 1 – Plataforma de Desenvolvimento e Tecnologias Utilizadas.

Tecnologia	Versão	Descrição	Propósito
Java	8	Linguagem de programação orientada a objetos e independente de plataforma.	Desenvolvimento de aplicativos em linguagem de programação orientada a objetos e independente de plataforma.
Java EE	7	Conjunto de especificações de APIs e tecnologias, que são implementadas por programas servidores de aplicação.	Reduzir a complexidade do desenvolvimento, implantação e gerenciamento de aplicações, de modo que o desenvolvedor não se preocupe demasiadamente com segurança, escalabilidade e desempenho.
MySQL Server	5.6	Sistema Gerenciador de Banco de Dados Relacional gratuito.	Persistência dos dados manipulados pela ferramenta.
Java Persistence API (JPA)	2.0	API para persistência de dados por meio de mapeamento objeto-relacional.	Fazer mapeamento objeto-relacional.
JavaServer Faces (JSF)	2.2	*Framework* para a construção de interfaces de usuários baseada em componentes para aplicações web que incorpora o padrão arquitetural MVC (*Model-View-Controller*).	Criação das páginas web utilizando componentes visuais pré-prontos e sua integração com o restante da aplicação.
Contexts and Dependency Injection for Java EE (CDI)	1.1	*Framework* gratuito para injeção de dependências.	Integração das diferentes camadas da arquitetura e serviços de transação.
Wildfly	10.0	Servidor de aplicação baseado na plataforma	Prover um servidor para as páginas *web* e ser compatível

		Java EE e implementando na linguagem Java.	com a especificação Java EE 7.
Eclipse Java EE IDE for Web Developers	4.5.1	Ambiente de desenvolvimento (IDE) para a linguagem Java.	Facilitar a atividade de implementação de software.
Git	1.6	Sistema de Controle de Versão.	Controlar as várias versões do código-fonte da ferramenta.
Astah Community	6.1	Ferramenta para modelagem em UML.	Ferramenta de modelagem UML (*Unified Modeling Language*).

3. Atributos de Qualidade e Táticas

Na Tabela 2 são listados os atributos de qualidade considerados neste projeto, com uma indicação se os mesmos são condutores da arquitetura ou não e as táticas a serem utilizadas para tratá-los.

Tabela 2 – Atributos de Qualidade e Táticas Utilizadas.

Categoria	Requisitos Não Funcionais Considerados	Condutor da Arquitetura	Tática
Facilidade de Aprendizado, Facilidade de Operação	RNF02, RNF03	Sim	• Prover ao usuário a capacidade de entrar com comandos que permitam operar o sistema de modo mais eficiente. Para tal, as interfaces do sistema devem permitir, sempre que possível, a entrada por meio de seleção ao invés da digitação de campos.
Segurança de Acesso	RNF01	Sim	• Identificar usuários usando *login* e autenticá-los por meio de senha.
Atratividade	RNF04	Sim	• Os cadastros devem ser apresentados em partes de modo que não fiquem muito grandes e complexos para o usuário.
Manutenibilidade, Portabilidade	RNF05, RNF06	Sim	• Organizar a arquitetura da ferramenta segundo uma combinação de camadas e partições. • A camada de lógica de negócio deve ser organizada segundo o padrão Camada de Serviço. • A camada de gerência de dados deve ser organizada segundo o padrão DAO. • Separar a interface com o usuário do restante da aplicação, segundo o padrão MVC.
Reusabilidade	RNF07	Sim	• Reutilizar componentes e *frameworks* existentes. • Quando não houver componentes disponíveis e houver potencial para reuso, devem-se desenvolver novos componentes para reuso.

Interoperabilidade	RN08	Sim	• Utilizar componentes e *frameworks* que possam permitir a integração com outra base de dados além da utilizada no projeto (*MySQL*), como por exemplo o Access ou Excel, por meio de formatos não proprietários (como CSV).

4. Arquitetura de Software

A arquitetura de software da ferramenta Sigme, no caso desse projeto o módulo de cadastros básicos (*Core*), está organizada em três camadas seguindo o proposto pelo *FrameWeb*, que são: Camada de Apresentação (*Presentation Tier*), Camada de Negócio (*Business Tier*), e Camada de Acesso a Dados (*Data Access Tier*). De forma a dar suporte para a construção da aplicação a ferramenta de apoio *JButler* será utilizada. Essa ferramenta provê classes que auxiliam na implementação dos casos de uso cadastrais que seguem o modelo de arquitetura proposto.

A primeira camada contém os pacotes de Visão (*View*) e Controle (*Control*), a segunda contém o de Domínio (*Domain*) e o de Aplicação (*Application*) e a terceira somente o pacote de Persistência (*Persistence*). Cada pacote será explicado melhor nas próximas seções onde será descrito o módulo de cadastros básicos do Sigme. A Figura 1 apresenta a visão geral das camadas e seus pacotes juntamente com o relacionamento que existe entre eles e as tecnologias Java EE utilizadas em cada pacote.

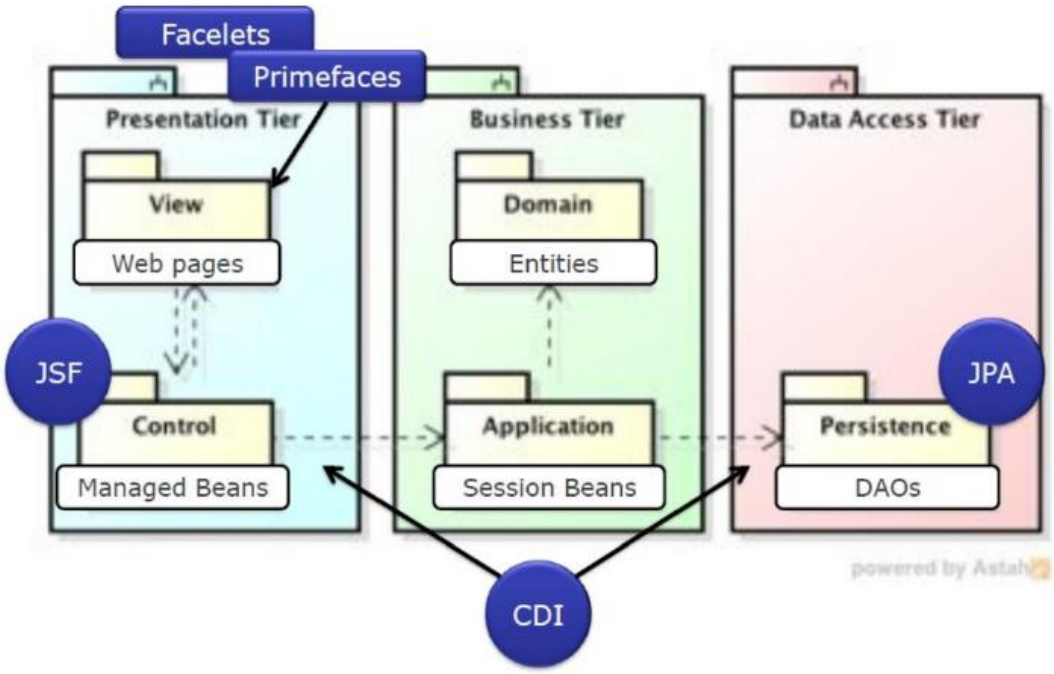

Figura 1 – Arquitetura de Software do Sistema proposto (LIMA,2015).

A figura 2 apresenta a subdivisão do módulo proposto nas camadas descritas acima, a saber: A Camada de Apresentação (*Control*), Camada de Negócio (*domain e application*) e Camada de Acesso a Dados (*persistence*). O pacote de Visão (*View*) que aparece na figura 2 contém o arquivo de mensagens que são utilizadas no sistema e podem ser apresentadas ao usuário. Em outra pasta do projeto, chamada WebContents, encontra-

se a maior parte dos componentes da View. Algumas classes e atributos da classe Espírita (*Spiritist*) que antes apareceram no documento de especificação de requisitos foram movidos para o pacote pessoa, por motivo de reuso de código. Além disso, o pacote de Exceções (*Exceptions*) que corresponde a classes Java que controlam exceções do específicas do Sigme, aparece na Figura 2.

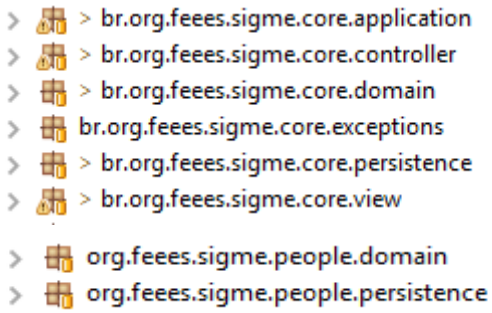

Figura 2 – Subdivisão dos módulos "Núcleo (*Core*)" do Sigme e "Pessoa (*People*)".

Nas próximas subseções serão apresentados os modelos FrameWeb relativos a cada camada do sistema mencionadas anteriormente.

4.1 - Camada de Apresentação

As funcionalidades criar, editar, excluir e visualizar (abreviadas de CRUD, do inglês *create, read, update e delete*), seguem um mesmo fluxo de execução e de interação com o usuário. Tais funcionalidades são similares para todos casos de uso cadastrais devido a utilização da ferramenta *JButler*. Esse fluxo de execução similar é representado abaixo através de um modelo de apresentação genérico.

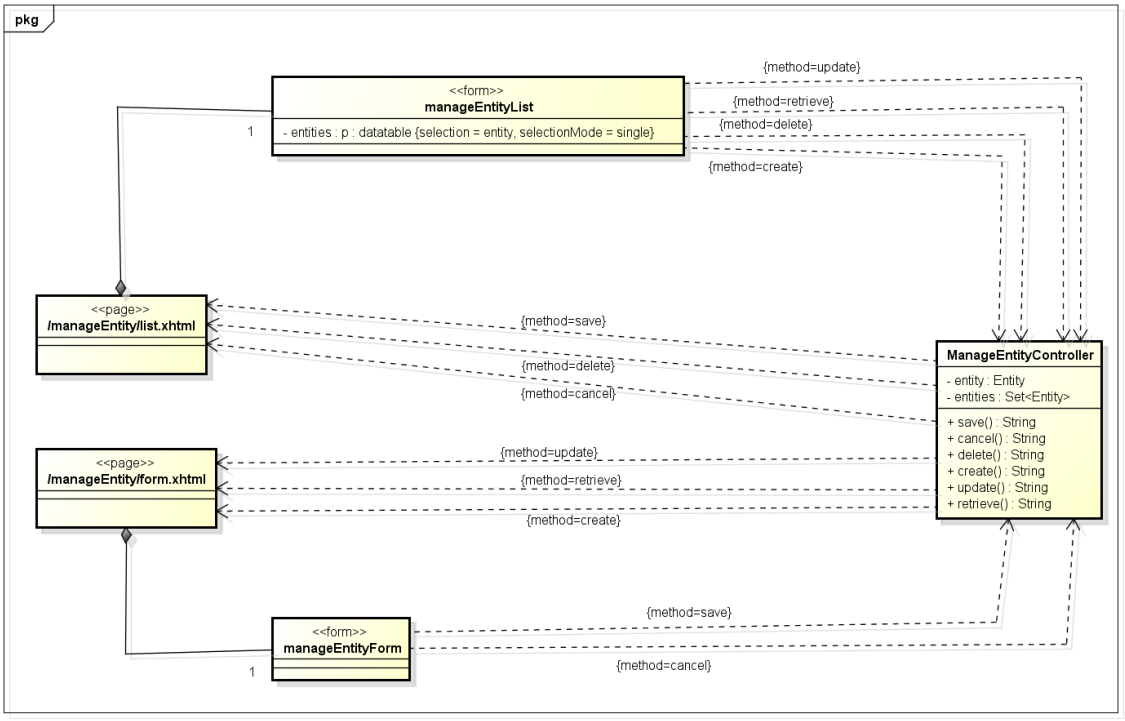

Figura 3 – Modelo de Navegação de um CRUD *JButler* (LIMA,2015), usado como base para as funcionalidades dos cadastros do módulo do Sigme.

Para os casos de uso que apresentam funções diferentes de apenas as básicas de cadastro, o modelo de navegação mostrado anteriormente não pode ser aplicado. Segue abaixo os modelos de navegação para os casos de uso "*Instalar Sistema*" e "*Enviar E-mail para Espíritas Inativos*", respectivamente.

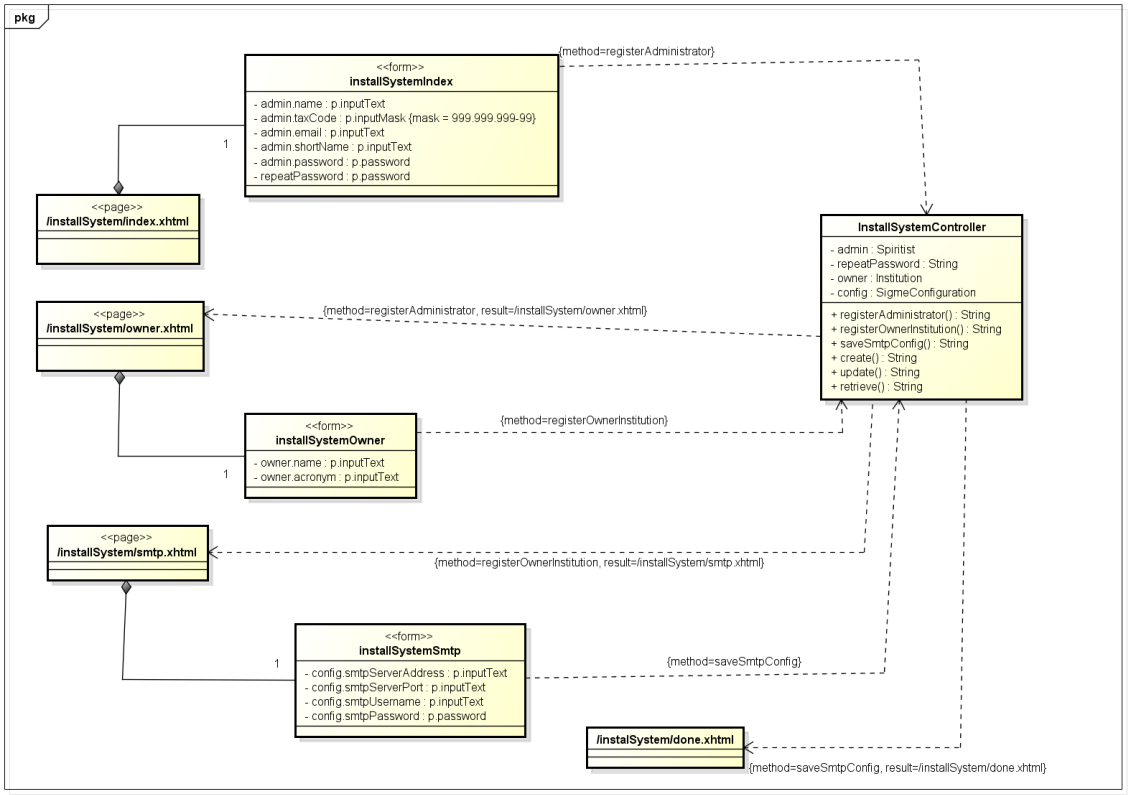

Figura 4 – Modelo de Navegação para o caso de uso "*Instalar Sistema*".

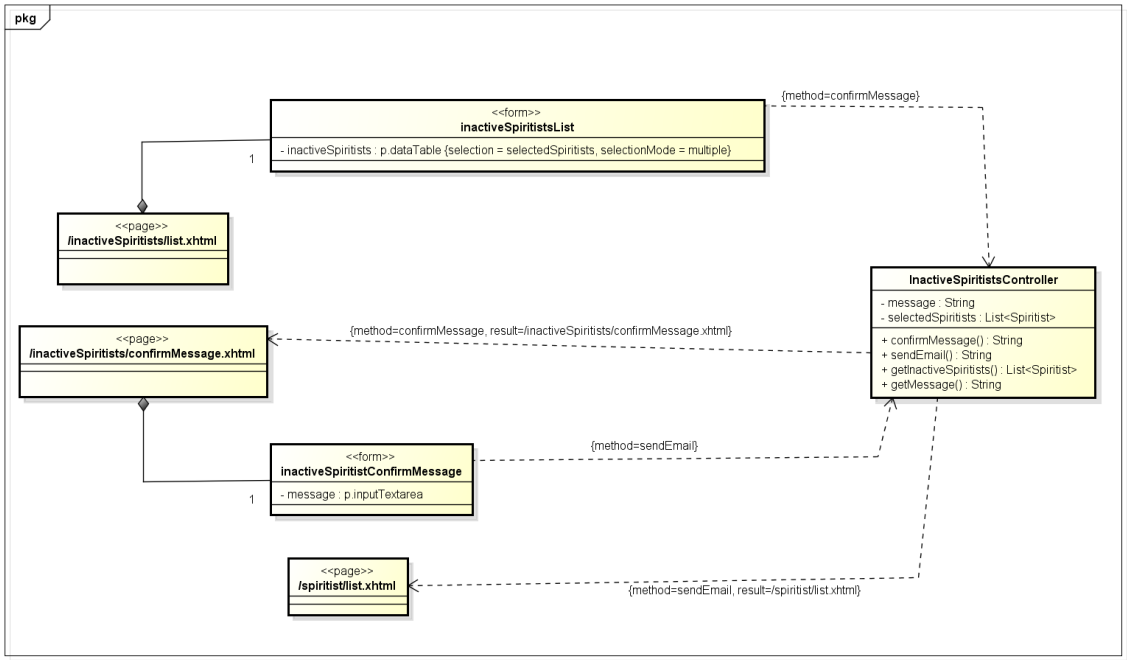

Figura 5 – Modelo de Navegação para o caso de uso "*Enviar E-mail para Espíritas Inativos*".

4.2 – Camada de Negócios

Diferente da abordagem original do *FrameWeb* proposta em 2007, todos os atributos que não podem ser nulos tiveram a *tag* {*not null*} omitida e os que são nulos tiveram a tag {*null*} acrescida de forma a diminuir a poluição visual com repetições desnecessárias no diagrama.

Todas as classes de domínio estendem de PersistentObjectSupport do pacote *JButler*, onde essa herança não é mostrada no diagrama acima com o intuito de não poluir o diagrama com várias associações.

Abaixo, seguem os modelos de domínio para os módulos "Núcleo" e "Pessoa".

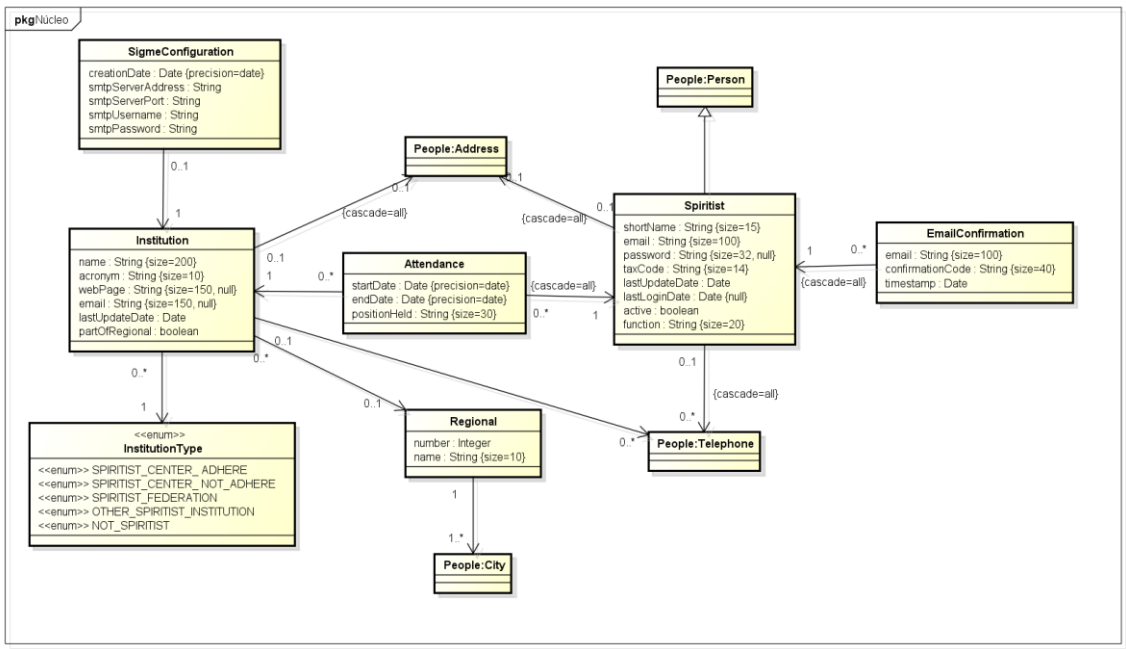

Figura 6 – Modelo de Domínio para o módulo "*Núcleo*".

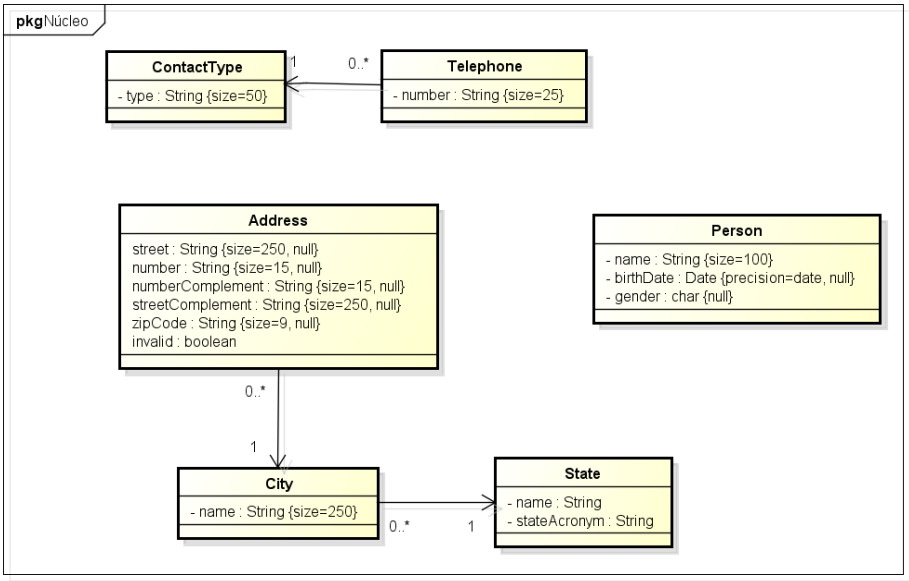

Figura 7 – Modelo de Domínio para o módulo "*Pessoa*".

As classes de aplicação *SpiritistServiceBean* e *InstitutionServiceBean* estendem de *CrudServiceBean* do pacote *JButler*, tal classe está representada abaixo de forma genérica. Da mesma forma dos diagramas anteriores essa herança não é mostrada no diagrama acima com o intuito de não poluir o diagrama com várias associações.

Abaixo, segue o modelo de aplicação para o módulo "Núcleo".

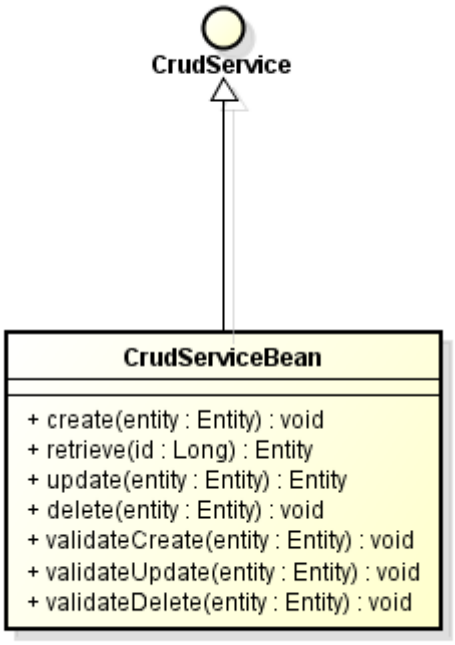

Figura 8 – Modelo de Aplicação genérica da ferramenta *JButler* (LIMA,2015).

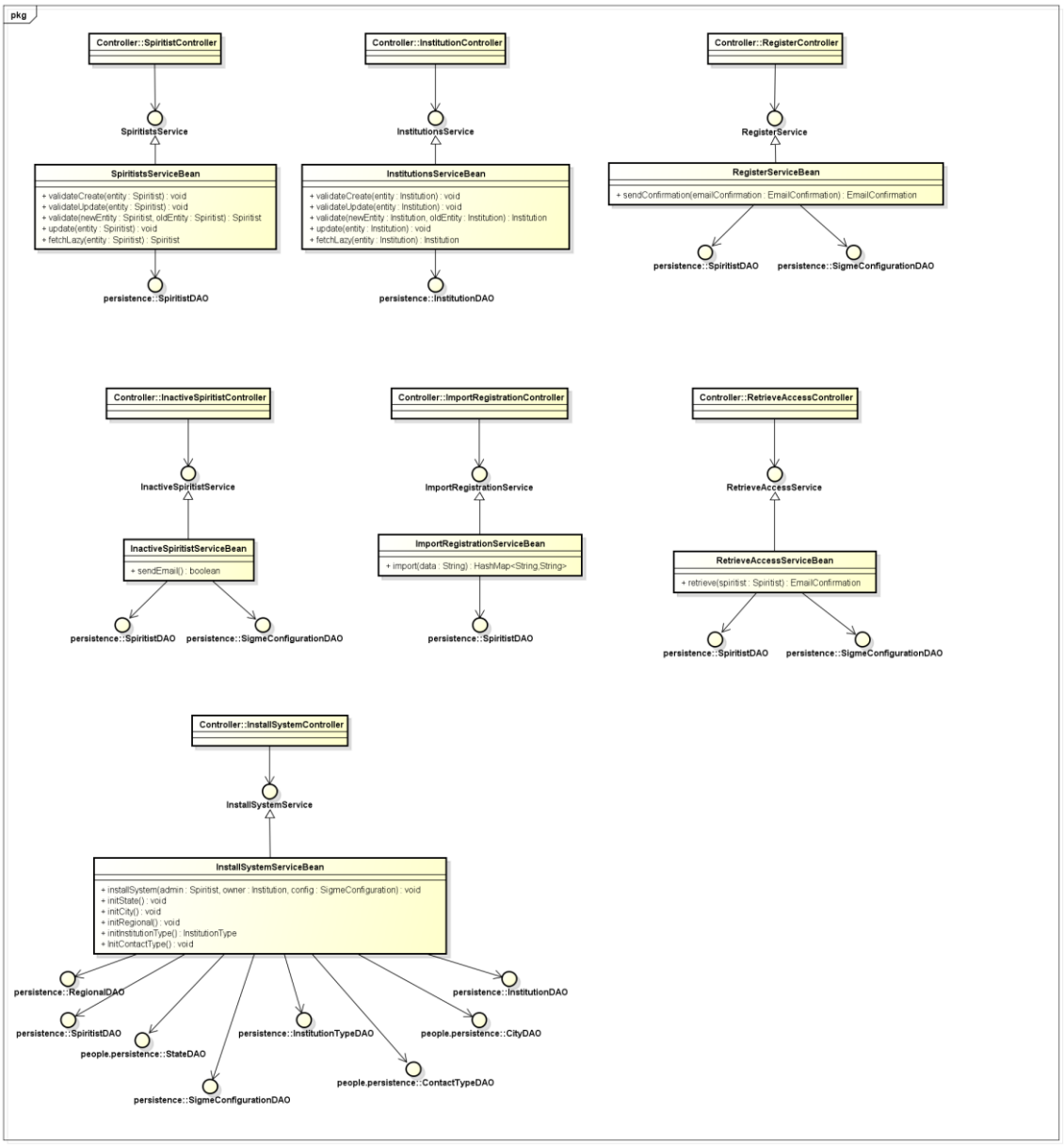

Figura 9 – Modelo de Aplicação para o módulo "*Núcleo*".

4.3 – Camada de Acesso a Dados

Nas figuras abaixo são apresentados os Modelos de Persistências desenvolvidos para o projeto Sigme, os mesmos foram usados na implementação do pacote de persistência.

Vale notar que o nome das classes já indica qual tecnologia de persistência foi utilizada, esse sistema de nomenclatura é mais uma sugestão do *FrameWeb* para simplificar o processo de software. Vale notar também que abaixo está representado o diagrama de persistência genérico provido pela ferramenta *JButler* e o modelo para o módulo "Núcleo".

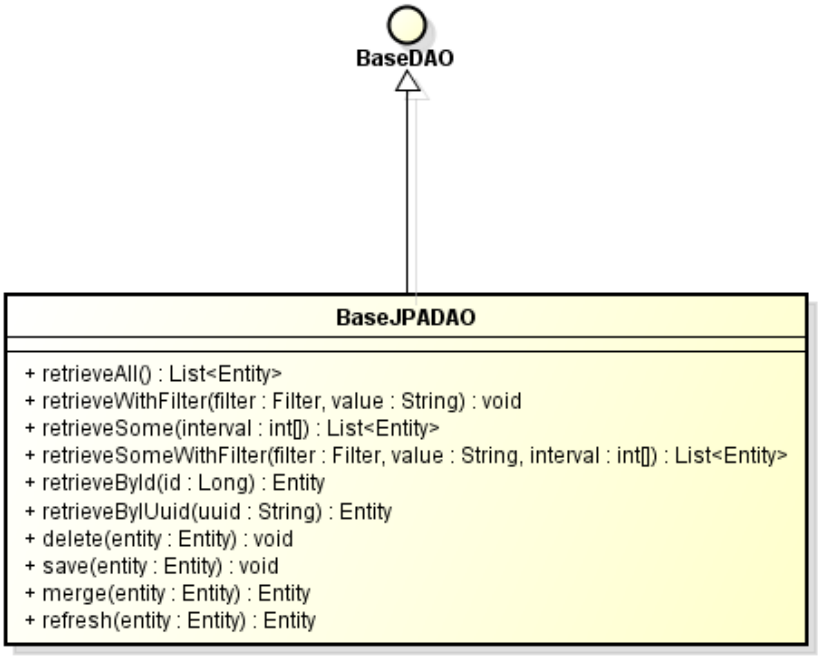

Figura 10 – Modelo de Persistência genérico da ferramenta *JButler* (LIMA,2015).

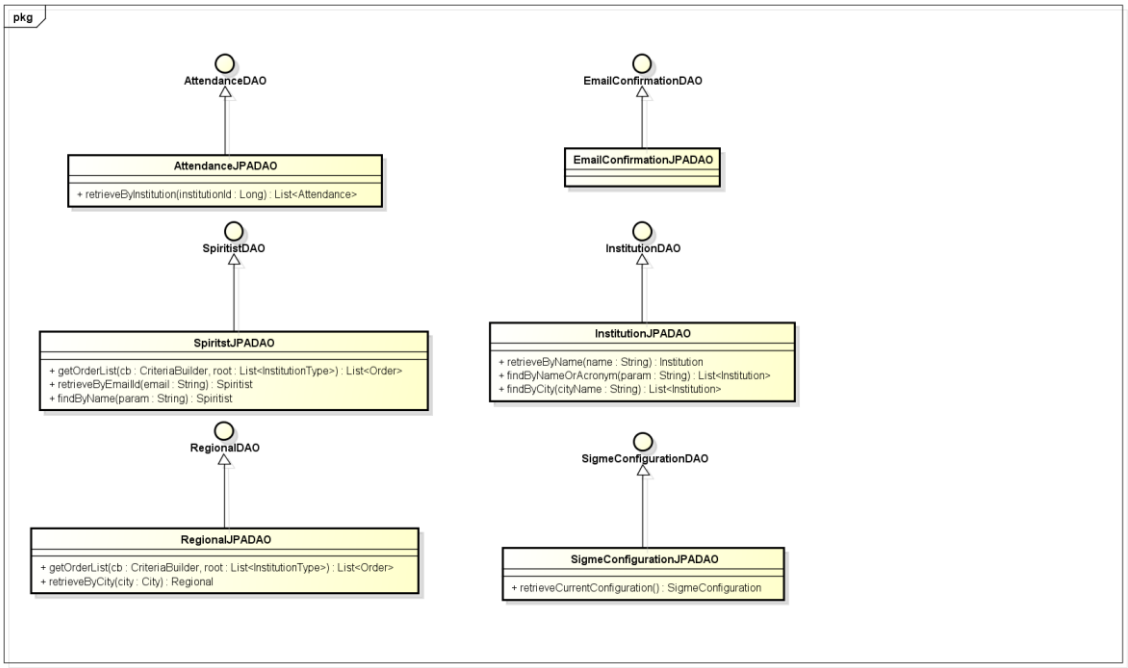

Figura 11 – Modelo de Persistência para o módulo "*Núcleo*".

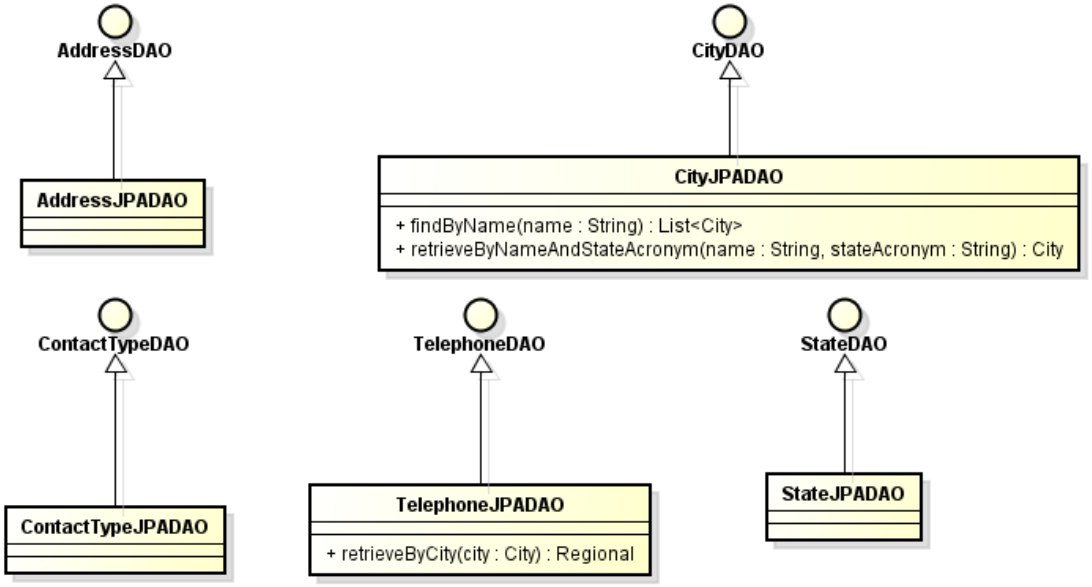

Figura 12 – Modelo de Persistência para o módulo "*Pessoa*".

Note que a relação de herança entre os DAOs específicos e o DAO base não é representada explicitamente nos diagramas para evitar poluição visual. Esta também é uma recomendação do *FrameWeb*, ficando, portanto, o desenvolvedor incumbido de derivar essa relação implicitamente ao analisar o modelo.

5 – Referências

LIMA, L. V. F. *SAP* - ***Sistema de Apoio ao Professor***. [S.l.], 2015.

SOUZA, VÍTOR ESTÊVÃO SILVA. ***FrameWeb: um Método baseado em Frameworks para o Projeto de Sistemas de Informação Web.*** Dissertação de Mestrado (Programa de Pós-Graduação em Informática) - Universidade Federal do Espírito Santo, 2007.