Silas Louzada Campos

FrameWeb Editor: Uma Ferramenta CASE para suporte ao Método FrameWeb

Vitória, ES

2017

Silas Louzada Campos

FrameWeb Editor: Uma Ferramenta CASE para suporte ao Método FrameWeb

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

2017

Silas Louzada Campos
FrameWeb Editor: Uma Ferramenta CASE para suporte ao Método FrameWeb/ Silas Louzada Campos. – Vitória, ES, 2017-
45 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, 2017.

1. Web engineering. 2. Frameworks. 3. Model-driven development. 4. FrameWeb. I. Souza, Vítor Estêvão Silva. II. Universidade Federal do Espírito Santo. IV. FrameWeb Editor: Uma Ferramenta CASE para suporte ao Método FrameWeb

CDU 02:141:005.7

Silas Louzada Campos

FrameWeb Editor: Uma Ferramenta CASE para suporte ao Método FrameWeb

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, X de dezembro de 2017:

Prof. Dr. Vítor E. Silva Souza
Orientador

Prof. Dr. Davidson Cury
Universidade Federal do Espírito Santo

Profª. Camila Zacche de Aguiar, MSc
Universidade Federal do Espírito Santo

Vitória, ES
2017

Agradecimentos

Primeiramente agradeço a Deus por hoje estar aqui, apesar de todas as dificuldades encontradas durante o caminho, e por todas as pessoas maravilhosas que colocou em minha vida durante esta caminhada.

Agradeço a minha família, Helio, Santa, Lucas, por terem me apoiado desde o começo, por me darem força e todo o suporte necessário para chegar até aqui. Aos meus filhos, Beatriz e Matheus, por me esperarem ansiosos nos finais de semana para me receberem com muito amor e recarregarem minhas energias, sendo minha principal motivação para seguir em frente.

Agradeço aos meus amigos de curso, que sempre estiveram comigo nestes muitos anos de UFES, Lucas, César, Celso, Ana, Breno, João, Luiz, Luana, sempre ajudando nos momentos difíceis, e que irei levar para a vida toda.

Aos meus sócios e amigos de todas as horas, Victor e Cássio, e a todos os demais amigos do NEMO, Gabriel, Gabriela, Camila, Jordana.

Aos amigos de fora da UFES, que me acompanham na vida e acompanharam toda esta caminhada de perto, em especial à Carolina, minha irmã de coração, que sempre se fez muito presente e que com toda sua sinceridade sempre me aconselha e incentiva.

Agradeço à minha namorada, Tainá, por ser uma pessoa tão maravilhosa na minha vida, me dando todo o apoio no decorrer deste ano, e sempre me ajudando em tudo quanto possível.

E por fim, um obrigado em especial ao meu orientador, Vítor, por toda paciência, boa vontade e por todas as lições que aprendi com você durante este tempo trabalhando juntos, muito obrigado.

'São as nossas escolhas que revelam o que realmente somos,
muito mais do que as nossas qualidades.'
(Alvo Dumbledore)

Resumo

A Web tem sido amplamente utilizada como plataforma de implantação de sistemas de informação, e com o objetivo de facilitar a integração entre tecnologias e também o reuso de soluções já implementadas é comum o uso de *frameworks*, que oferecem um conjunto de classes e bibliotecas que resolvem problemas comuns em diversas aplicações. Porém, apesar de afetarem diretamente a arquitetura dos sistemas, os *frameworks* geralmente não são considerados no processo de desenvolvimento até a fase de implementação.

Dentro deste contexto, destacamos o método FrameWeb (SOUZA, 2007; SOUZA; FALBO; GUIZZARDI, 2009), um método baseado em *frameworks* para o desenvolvimento de sistemas de informação Web (Web Information Systems ou WISs). O método define uma arquitetura padrão para facilitar a integração com *frameworks* muito utilizados para o desenvolvimento nessa plataforma, além de propor um conjunto de modelos que traz para o projeto arquitetural do sistema conceitos inerentes a estes *frameworks*.

Este trabalho apresenta o *FrameWeb Editor*, uma ferramenta CASE(*Computer-Aided Software Engineering*) baseada nos metamodelos que definem a sintaxe da linguagem FrameWeb, provendo um editor gráfico para criação de modelos válidos nesta linguagem e servindo de base para outras funcionalidades, como por exemplo, geração de código.

Palavras-chaves: Engenharia Web. Frameworks. Desenvolvimento Dirigido por Modelos. FrameWeb.

Lista de ilustrações

Lista de abreviaturas e siglas

UML	Unified Modeling Language
MDD	Model Driven Development
WIS	Web-based Information Systems
MVC	Model-View-Controller
EMF	Eclipse Modeling Framework
DSL	Domain-Specific Language

Sumário

1 Introdução

Este capítulo apresenta as motivações e objetivos deste trabalho, assim como a metodologia utilizada para alcançar os resultados esperados, finalizando com a organização e estrutura da monografia.

1.1 Contextualização

Os avanços das tecnologias relacionadas a *Web* têm possibilitado a utilização desta plataforma para a implantação de aplicações mais complexas, das quais destacamos uma categoria de aplicações chamada de Sistemas de Informação Baseados na *Web* (*Web-based Information Systems - WISs*), que são considerados sistemas de informação tradicionais disponíveis na Internet.

Assim como nos sistemas de informação tradicionais, para o desenvolvimento destes sistemas se torna necessário o uso de conceitos e ferramentas relacionadas à Engenharia de Software, inaugurando, então, a Engenharia Web (PRESSMAN, 2011). Esta nova área busca estabelecer abordagens sistemáticas para o desenvolvimento de WIS's de qualidade, assim como sua implantação e manutenção (MURUGESAN et al., 2001).

Para o desenvolvimento de tais sistemas, é comum o reúso de software, que pode ser empregado de diferentes formas, como bibliotecas de componentes, geradores de aplicação, *templates* de código genéricos, todos envolvendo uma forma de abstração, seleção, e integração de artefatos de software (KRUEGER, 1992). Em particular, na Engenharia Web destacamos o uso de *frameworks*, que podem ser definidos como modelos reutilizáveis de todo ou parte de um sistema e que é representado por um conjunto de classes abstratas e a forma como suas instâncias interagem entre si (JOHNSON, 1997). Porém, apesar de afetarem diretamente a arquitetura dos sistemas, os *frameworks* geralmente não são considerados no processo de desenvolvimento até a fase de implementação.

Dentro deste contexto, destacamos o método FrameWeb *(Framework-based Design Method for Web Engineering)* (SOUZA, 2007; SOUZA; FALBO; GUIZZARDI, 2009), um método baseado em *frameworks* para o desenvolvimento de WIS's. O método define uma arquitetura padrão para facilitar a integração com *frameworks* muito utilizados para o desenvolvimento nessa plataforma, além de propor um conjunto de modelos que traz para o projeto arquitetural do sistema conceitos inerentes a estes *frameworks*.

Baseado em técnicas MDD (*Model-Driven Development*) (PASTOR et al., 2008), a sintaxe da linguagem de modelagem do método FrameWeb é definida formalmente por meio de metamodelos que permitem a inclusão de novas instâncias de *frameworks* e servem

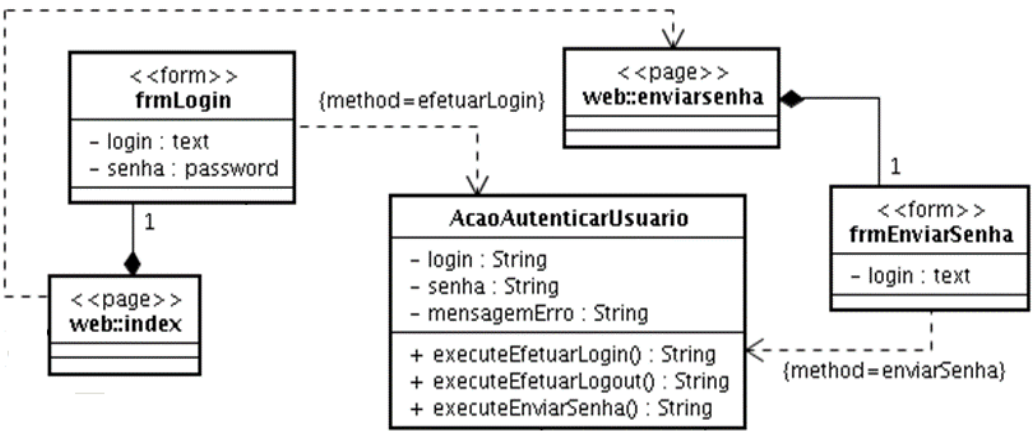

Figura 1 – Modelo FrameWeb construído utilizando-se um editor UML genérico (SOUZA, 2007)

.

de base para a construção de ferramentas que verifiquem a sintaxe dos modelos construídos, ofereçam geração de código, dentre outras funcionalidades possíveis (MARTINS; SOUZA, 2015; MARTINS, 2016).

1.2 Motivação

Em sua proposta original (SOUZA, 2007; SOUZA; FALBO; GUIZZARDI, 2009), o método FrameWeb definia extensões leves (*lightweight extensions*) ao meta-modelo da UML para representar componentes típicos da plataforma Web e dos *frameworks* utilizados, sendo criado um perfil UML para a construção dos diagramas, de forma que os modelos eram criados utilizando-se qualquer editor UML.[1] Estereótipos e restrições eram utilizados para relacionar as classes UML às classes específicas do método FrameWeb, como mostrado na Figura 1.

Esta abordagem possui algumas limitações, como:

- O perfil UML não é rigoroso, portanto as ferramentas de modelagem UML não possuem forma de prevenir que se incluam elementos que não pertencem aos modelos;
- Os componentes não estarem relacionados à sintaxe do método, não permitindo que o modelo seja processado posteriormente para validação e geração de código.

Observa-se, portanto, a necessidade de uma ferramenta CASE (*Computer-Aided Software Engineering*) mais apropriada para o método FrameWeb, ou seja, uma que

[1] Uma lista de diversas ferramentas UML e suas respectivas plataformas, preços, companhias, dentre outras informações, pode ser encontrada em: <http://www.objectsbydesign.com/tools/umltools_byCompany.html>.

permita que desenvolvedores criem modelos utilizando os elementos da UML incluídos na sintaxe de FrameWeb definida em (MARTINS; SOUZA, 2015; MARTINS, 2016), auxiliando desenvolvedores na criação de modelos FrameWeb válidos e servindo de base para outras funcionalidades como, por exemplo, geração de código.

1.3 Objetivos

Visando suprir uma necessidade do método FrameWeb, o objetivo geral deste trabalho é desenvolver uma ferramenta CASE que apoie o analista/desenvolvedor na utilização da abordagem FrameWeb para construção de sistemas de informação Web.

É proposta, portanto, a ferramenta *FrameWeb Editor*, que provê um ambiente gráfico para a criação de modelos FrameWeb, relacionando os componentes gráficos diretamente com os elementos definidos na sintaxe do método, além de oferecer suporte às características de extensibilidade do método e também à sua extensão FrameWeb-LD (CELINO et al., 2016), que propõe a integração de conceitos inerentes a *linked data*, auxiliando na criação de WISs integrados à Web Semântica.

1.4 Metodologia

O trabalho realizado compreendeu as seguintes atividades:

1. *Revisão Bibliográfica*: estudo sobre o método FrameWeb, *frameworks* e suas classificações, Desenvolvimento Orientado a Modelos, e a revisão do método e desenvolvimento dos metamodelos propostos em (MARTINS; SOUZA, 2015; MARTINS, 2016). Análise de projetos de graduação em que o método FrameWeb foi utilizado para o desenvolvimento de sistemas de informação.

2. *Estudo das Tecnologias*: estudo sobre os softwares disponíveis para a criação de ferramentas CASE seguindo a abordagem de desenvolvimento orientado a modelos como, por exemplo, a plataforma Eclipse RCP (Rich Client Platform) e seus plug-ins relevantes como, e.g., EMF,[2] Acelleo,[3] XText,[4] Sirius,[5] OBEO Designer,[6] etc.

3. *Implementação da Ferramenta de Modelagem*: desenvolvimento da ferramenta *FrameWeb Editor* utilizando o plugin Sirius, baseando-se no modelo da sintaxe abstrata para definir a sintaxe concreta, que contém características gráficas dos elementos

[2] <https://eclipse.org/modeling/emf/>
[3] <https://eclipse.org/acceleo/>
[4] <https://eclipse.org/Xtext/documentation/>
[5] <https://eclipse.org/sirius/overview.html>
[6] <http://www.obeodesigner.com/>

definidos no metamodelo. Ambos os modelos são especificados utilizando a sintaxe do EMF.

4. *Validação da Ferramenta*: engenharia reversa do sistema *Marvin*,[7] um sistema de apoio a atividades acadêmicas, e remodelagem de outras aplicações desenvolvidas em trabalhos de graduação aplicando o método FrameWeb e cujos modelos foram construídos utilizando-se um editor UML genérico.

5. *Integração com a Extensão FrameWeb-LD*: desenvolvimento do suporte para a extensão FrameWeb-LD de forma a trazer para o modelo de entidades os conceitos relacionados à *linked data* por meio de arquivos de vocabulários, caracterizando também a extensibilidade do método.

6. *Extensão da Ferramenta para dar Suporte a Metamodeladores*: desenvolvimento do suporte para a criação, visualização e edição dos arquivos relacionados aos *frameworks*, linguagem de programação e vocabulário na própria ferramenta. A extensibilidade e generalidade da ferramenta é atingida por meio destes arquivos.

7. *Redação da Monografia*: escrita da monografia, utilizando a linguagem *LaTeX*[8] e o modelo *abnTeX*[9] que atende os requisitos das normas da ABNT (Associação Brasileira de Normas Técnicas). Durante essa etapa, também foi publicado, no XVI Workshop de Ferramentas e Aplicações (WFA) do WebMedia 2017, um artigo sobre a ferramenta *FrameWeb Editor* (CAMPOS; SOUZA, 2017).

1.5 Organização do Texto

Este texto está dividido em quatro partes principais além desta introdução, que seguem:

- **Capítulo 2** – Referencial Teórico: Neste capítulo é apresentada a base teórica do método FrameWeb, assim como conceitos de *Model-Driven Development* e os meta-modelos FrameWeb.

- **Capítulo 3** – FrameWeb Editor: Apresenta a ferramenta FrameWeb Editor proposta neste trabalho, apresentando suas interfaces e funcionalidades, assim como a relação da ferramenta com os meta-modelos da linguagem, demonstrando também a forma como atinge as características de extensibilidade propostas pelo método e por fim apresentando a arquitetura da ferramenta.

[7] <https://github.com/dwws-ufes/Marvin>
[8] <http://www.latex-project.org/>
[9] <http://www.abntex.net.br>

- **Capítulo 4** – Validação: Nesse capítulo são apresentados trabalhos previamente realizados no desenvolvimento de sistemas de informação Web utilizando o método FrameWeb e editores UML genéricos, sendo agora remodelados através do FrameWeb Editor.

- **Capítulo 5** – Considerações Finais: apresenta as conclusões obtidas ao final deste trabalho, bem como as dificuldades encontradas e as perspectivas de trabalhos futuros para a ferramenta e para o método FrameWeb em geral.

2 Referencial Teórico

Este capítulo apresenta a base teórica do desenvolvimento deste trabalho, apresentando principalmente uma descrição do método FrameWeb, seus fundamentos e sua aplicação prática, a Seção 2.1 apresenta uma introdução à modelagem dirigida por modelos, a Seção 2.2 apresenta o método FrameWeb e a Seção 2.3 apresenta os principais trabalhos relacionados.

2.1 *Model-Driven Development*

No desenvolvimento de sistemas é comum o uso de modelos que ajudem a compreender problemas complexos e suas possíveis soluções através de abstrações, porém normalmente esses modelos não são vistos como principal prioridade no processo de desenvolvimento de software. O Desenvolvimento Orientado a Modelos (*Model Driven Development* ou MDD) propõe uma nova abordagem em comparação com esta visão tradicional, tendo como objetivo a concentração dos esforços na construção de modelos que representem todas as características, funcionalidades e recursos do sistema, de forma que, posteriormente, por meio de transformações entre modelos dos diversos níveis de abstração obtém-se o software final (SELIC, 2003).

Portanto, em MDD temos modelos que são o próprio projeto do sistema, e que são evoluídos e atualizados no decorrer do processo de desenvolvimento, sendo a implementação derivada diretamente destes modelos. Por meio desta abordagem, são utilizados modelos para a especificação dos elementos de um sistema. Estes modelos, por sua vez, possuem uma especificação dos componentes que o integram e podem ser utilizados no mesmo. Esta especificação pode ser feita utilizando uma linguagem gráfica e a este modelo dá-se o nome de metamodelo.

Desta forma, ao empregar o paradigma MDD pode-se utilizar linguagens para a criação dos modelos, execução de transformações e geração de código. Estas linguagens formalizam um domínio e, para isso, necessitam atender a certos requisitos, devendo especificar uma linguagem abstrata, uma linguagem concreta e uma semântica. Nesse sentido o MDD pode ser usado na prática para o desenvolvimento de Linguagens Específicas de Domínio (Domain Specific Languages - DSL), que são utilizadas posteriormente para o desenvolvimento dos modelos que representam o sistema a ser desenvolvido e dos quais a implementação do sistema será diretamente derivada.

Em MDD, uma sintaxe abstrata é representada por um metamodelo, que descreve todos os conceitos e regras de modelagem relacionados à linguagem em questão, de forma

que o desenvolvimento dos modelos desta linguagem será totalmente baseado nestes metamodelos. Estes modelos criados com base na sintaxe abstrata, são instâncias do metamodelo que a define, e expressam seu significado por meio de uma notação ou representação gráfica, esta especificação das representações dos componentes é chamada sintaxe concreta da linguagem.

Além desta especificação dos conceitos e regras de modelagem, e de sua notação e representação gráfica, é necessário que se forneça significado para cada elemento da linguagem, uma semântica, e também a construção de uma ferramenta CASE que dê suporte à linguagem definida, por meio de ferramentas para a construção dos modelos especificados, dentre outras funcionalidades.

Existem diversas ferramentas de modelagem que surgiram para facilitar a utilização desta abordagem, auxiliando a criação dos modelos citados. Uma delas é o Eclipse Modeling Framework(EMF), um *framework* de modelagem e geração de código para construção de ferramentas e outras aplicações com base em modelos estruturados de dados (GRONBACK, 2009). O EMF, juntamente com alguns de seus "*plug-ins*", possibilita não só a criação de modelos, mas também a aplicação de diversas transformações nos mesmos e possui todos os recursos necessários para definir linguagens específicas de domínio (DSLs).

Dentre os "plug-ins" para o EMF, destacamos o OBEO Designer que, assim como sua versão aberta Sirius, produz ferramentas gráficas, e que foi utilizado neste trabalho para o desenvolvimento do FrameWeb Editor, conforme descrito na Seção 3.6.

2.2 O Método FrameWeb

O método FrameWeb (*Framework-based Design Method for Web Engineering*) (SOUZA, 2007; SOUZA; FALBO; GUIZZARDI, 2009) é um método baseado em técnicas de Engenharia de Software orientadas a objetos e direcionado à construção de Sistemas de Informação Web (*Web-based Information Systems*, ou WIS) que possuam uma infraestrutura arquitetônica baseada no uso de *frameworks*.

Assumindo que serão utilizados determinados tipos de *frameworks* durante o desenvolvimento, o FrameWeb define uma arquitetura de software padrão que divide o sistema em camadas, propondo uma extensão da UML para a construção de modelos de projeto que representem elementos comuns a estes *frameworks*.

A arquitetura lógica padrão para WISs definida pelo método FrameWeb é baseada no padrão arquitetônico *Service Layer* (Camada de Serviço), proposta por Randy Stafford em (FOWLER, 2002), que divide o sistema em três grandes camadas, como mostra a Figura 2. São elas:

- **Camada de Negócio**, referente às funcionalidades relacionadas às regras de negócio

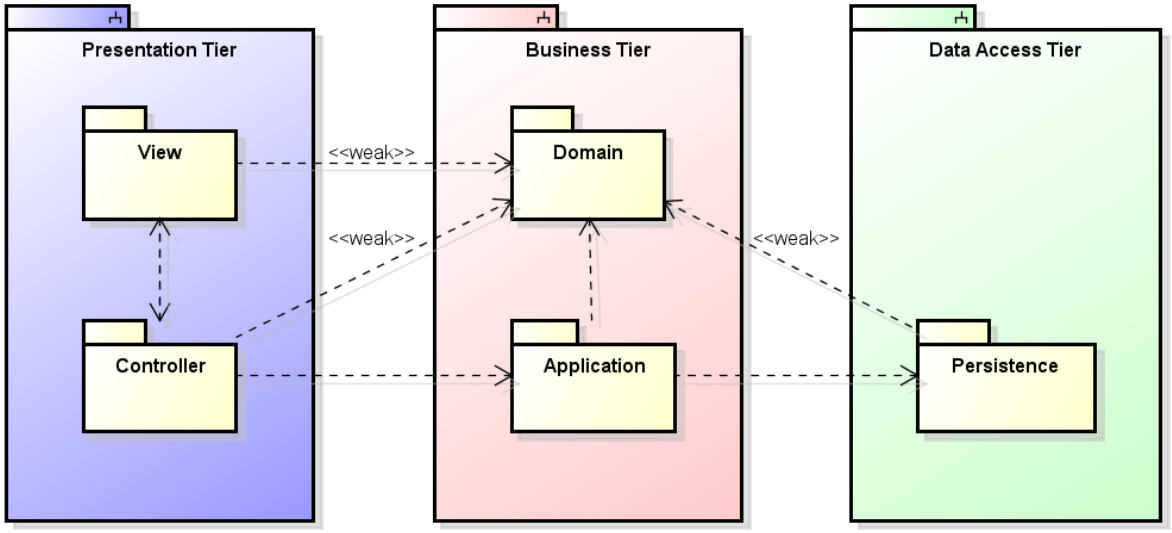

Figura 2 – Proposta de arquitetura FrameWeb (SOUZA, 2007).

da aplicação;

- **Camada de Apresentação**, referente às funcionalidades de interface com o usuário;
- **Camada de Acesso a Dados**, referente às funcionalidades para persistência dos dados.

Dentro desta arquitetura, temos ainda o chamado padrão arquitetural MVC (*Model-View-Controller*) (GAMMA et al., 1994), em que se baseia o acesso feito pela camada de apresentação às funcionalidades do sistema.

Com base neste padrão arquitetural, a linguagem FrameWeb define quatro tipos de diagramas, todos eles baseados no Diagrama de Classes da UML, que são utilizados para representar os componentes específicos de cada camada em questão, abrangendo componentes típicos da plataforma Web e relacionados aos *frameworks* utilizados. São eles:

- **Modelo de Entidades** (originalmente Modelo de Domínio), que representa os objetos de domínio do problema e sua persistência por meio do *framework* de mapeamento objeto/relacional (e.g., Java Persistence API[1]);
- **Modelo de Persistência**, referente à camada de acesso aos dados, seguindo o padrão de projeto *Data Access Object* (DAO) (ALUR; CRUPI; MALKS, 2003);
- **Modelo de Navegação**, para representar os diferentes componentes que, integrados ao *framework* Web (controlador frontal, e.g., JavaServer Faces[2]), formam a interface com o usuário e compõem a camada de Apresentação (Visão e Controle);

[1] <http://jcp.org/en/jsr/detail?id=338>
[2] <http://jcp.org/en/jsr/detail?id=344>

- **Modelo de Aplicação**, para representar as classes de serviço, responsáveis pela implementação dos casos de uso (funcionalidades do WIS), e suas dependências, gerenciadas pelo *framework* de injeção de dependências (e.g., Contexts and Dependency Injection for Java[3]).

2.2.1 Modelo de Entidades

No modelo de entidades, sugere-se o desenvolvimento a partir do modelo de classes construído na fase de análise de requisitos, adequando o mesmo à plataforma de implementação escolhida e adicionando os respectivos mapeamentos de persistência. Estes mapeamentos são meta-dados que indicam aos *frameworks* de Mapeamento Objeto/Relacional (*Object/Relational Mapping* – ORM) como transformar os objetos e seus atributos em tabelas e colunas.

A Figura 3 mostra um modelo de entidades referente ao Portal do LabES, um sistema de informação Web desenvolvido para o Laboratório de Engenharia de Software da Universidade Federal do Espírito Santo (LabES / UFES) apresentado em (SOUZA, 2007). Neste exemplo podemos observar algumas propriedades próprias deste tipo de modelo, como por exemplo a restrição de associação `fetch=lazy` definida na relação entre `Usuario` e `Area`, e que indica a estratégia de recuperação de uma associação, podendo ser normal (*eager*) ou preguiçosa (*lazy*).

Outros tipos de propriedades definidas nestes modelos são:

- Estratégia de mapeamento de herança;
- Coleção que implementará uma associação: bag (lista não indexada), lista, conjunto ou mapa;
- Ordenação de uma associação: natural (da classe) ou por colunas (c1 asc, c2 desc etc.);
- Cascateamento de uma associação: nada, criação, alteração, exclusão, atualização ou tudo.

2.2.2 Modelo de Persistência

O modelo de persistência é responsável por representar as classes DAO existentes, definindo a persistência das instâncias das classes de domínio, e para sua correta construção é necessária a criação de interfaces e implementações dos DAOs base, assim como a especificação de quais classes de domínio precisarão de um DAO.

[3] <http://jcp.org/en/jsr/detail?id=346>

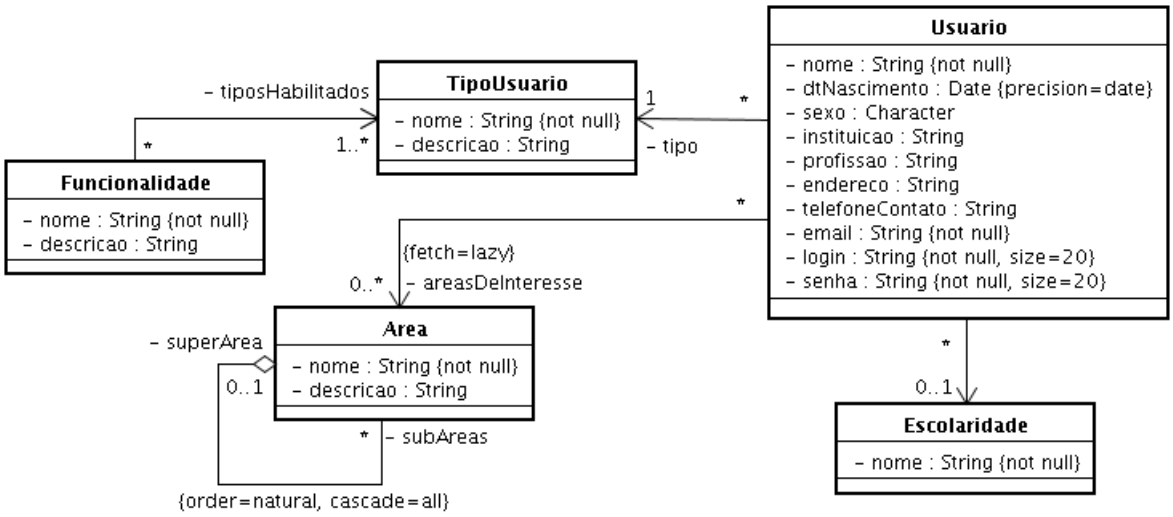

Figura 3 – Modelo de Entidades desenvolvido com o método FrameWeb (SOUZA, 2007).

Estes DAOs base, especificam operações muito comuns em todas as interfaces, como por exemplo operações de salvar e excluir, e desta forma evita a repetição deste tipo de operação em todas as interfaces definidas.

Portanto, para cada classe de domínio que necessite de um DAO, o modelo de persistência apresentará uma interface e uma classe concreta DAO implementando esta interface, que define os métodos de persistência relacionados à mesma.

Uma das características da especificação deste modelo, é o fato de não ser necessária a repetição dos métodos do DAO na implementação e na interface, podendo ser modelados em apenas um dos dois e assim reduzindo a quantidade de elementos modelados. Além disso, sugere-se o uso de um padrão de nomes para as classes DAO, sendo `<nome da classe de domínio>DAO` para as interfaces, e `<nome da interface><tecnologia de persistência>` para as classes concretas.

A Figura 4 mostra um modelo de persistência que inclui interfaces DAO para as classes de domínio `Usuario`, `Area`, `TipoUsuario` e `Funcionalidade`, além de implementações das interfaces para o *framework* ORM Hibernate.

2.2.3 Modelo de Navegação

No modelo de navegação são analisados os casos de uso definidos durante a Especificação de Requisitos, criando classes controladoras e seus métodos, identificando como os dados serão submetidos pelos clientes para criar as páginas e formulários e assim adicionar os atributos ao controlador.

Após esta definição e análise dos resultados possíveis a partir dos dados de entrada, são criados os diferentes componentes para a interface com o usuário, utilizando estereótipos

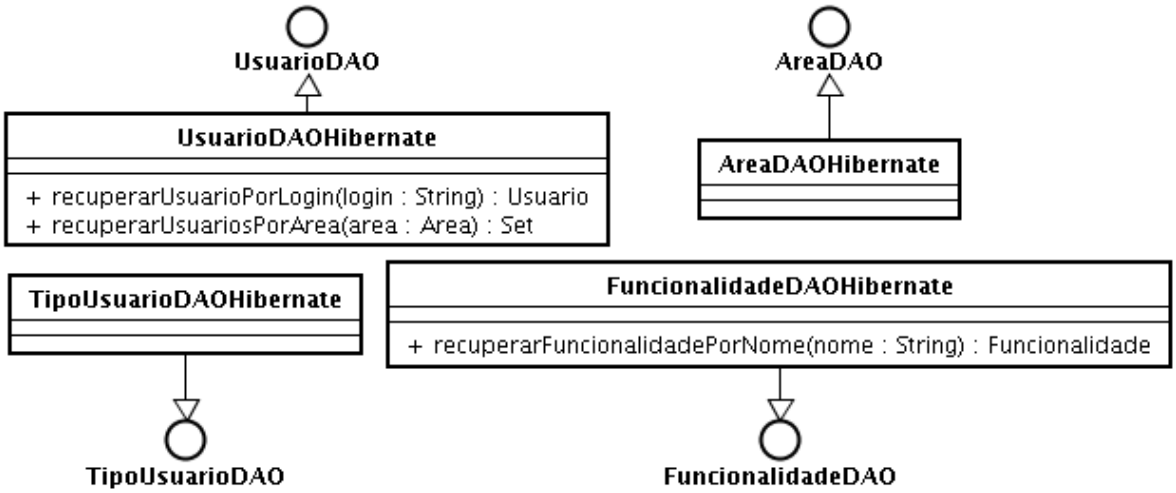

Figura 4 – Modelo de persistência desenvolvido com o método FrameWeb (SOUZA, 2007).

para representar os diferentes tipos de componentes, como `«page»` para páginas Web, `«form»` para formulários HTML e `«binary»` para arquivos que podem ser recuperados e exibidos pelo navegador, como imagens e documentos.

Estas páginas Web podem conter:

- Atributos, que representam informações a serem exibidas pela página;
- Relações entre páginas, que representam links HTML entre elas;
- Composição entre páginas e formulários, que representam a inclusão destes dentro da mesma.

Já nos formulários, os atributos representam os campos do formulário, definindo seus respectivos tipos, como `input`, `checkbox`, `button`, etc. ou tags definidas pelo framework utilizado, como por exemplo `multiSelectListbox` no PrimeFaces.[4]

2.2.4 Modelo de Aplicação

O modelo de aplicação representa as classes de serviço do sistema e é utilizado para a configuração das dependências entre os controladores, serviços e os DAOs necessários para que as classes de serviço alcancem seus objetivos.

Sendo assim, todas as classes controladoras que dependem de uma classe de serviço devem também aparecer no modelo, e da mesma forma, quando a classe de serviço depende de algum DAO, a interface deste deve aparecer no diagrama e estar associada à classe de serviço em questão. Neste modelo também vale a regra aplicada ao modelo de persistência,

[4] <https://www.primefaces.org/>

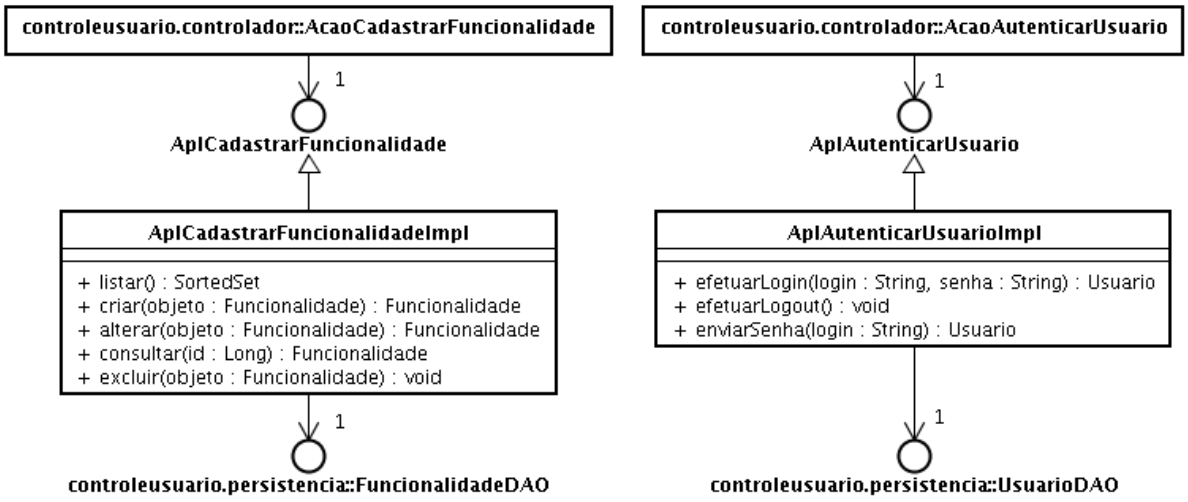

Figura 5 – Modelo de Aplicação desenvolvido com o método FrameWeb (SOUZA, 2007).

em que cada classe de serviço deve ter uma interface e uma implementação.

A Figura 5 apresenta um modelo de aplicação que representa a dependência entre à classe controladora `AcaoCadastrarFuncionalidade`, à classe de serviço `AplCadastrarFuncionalidadeImpl`, e seu respectivo DAO `FuncionalidadeDAO`, referente ao pacote `controleusuario` do Portal do LabES (SOUZA, 2007).

2.2.5 Metamodelos FrameWeb

Baseado em técnicas de Desenvolvimento Orientado a Modelos (*Model-Driven Development* ou MDD) (PASTOR et al., 2008), a sintaxe da linguagem FrameWeb é formalmente definida por meio de metamodelos, que especificam a sintaxe abstrata da linguagem, descrevendo os conceitos e as regras de modelagem específicas do método FrameWeb, e que consequentemente permitem a criação de ferramentas e mecanismos para que esta linguagem possa ser utilizada de forma correta e direcionada.

A Figura 6 apresenta um fragmento do metamodelo de entidades, demonstrando os possíveis mapeamentos e propriedades adicionais do método FrameWeb relacionadas aos atributos e relações citados na Seção 2.2.1. Por exemplo, em um modelo de entidade, uma associação será uma instância da classe `DomainAssociation` especificada no metamodelo, e desta forma podem ser definidas as propriedades `collection`, `cascade`, `fetch` e `order` nesta relação.

Além dos metamodelos para os quatro tipos básicos de modelos FrameWeb, também foi definido um metamodelo de *frameworks*, que define as classes, propriedades e relações para a especificação de um modelo de definição de *framework*, que contem os componentes do *framework* em questão e é usado para a inserção destes componentes nos modelos FrameWeb construídos, trazendo para os modelos FrameWeb os conceitos e propriedades

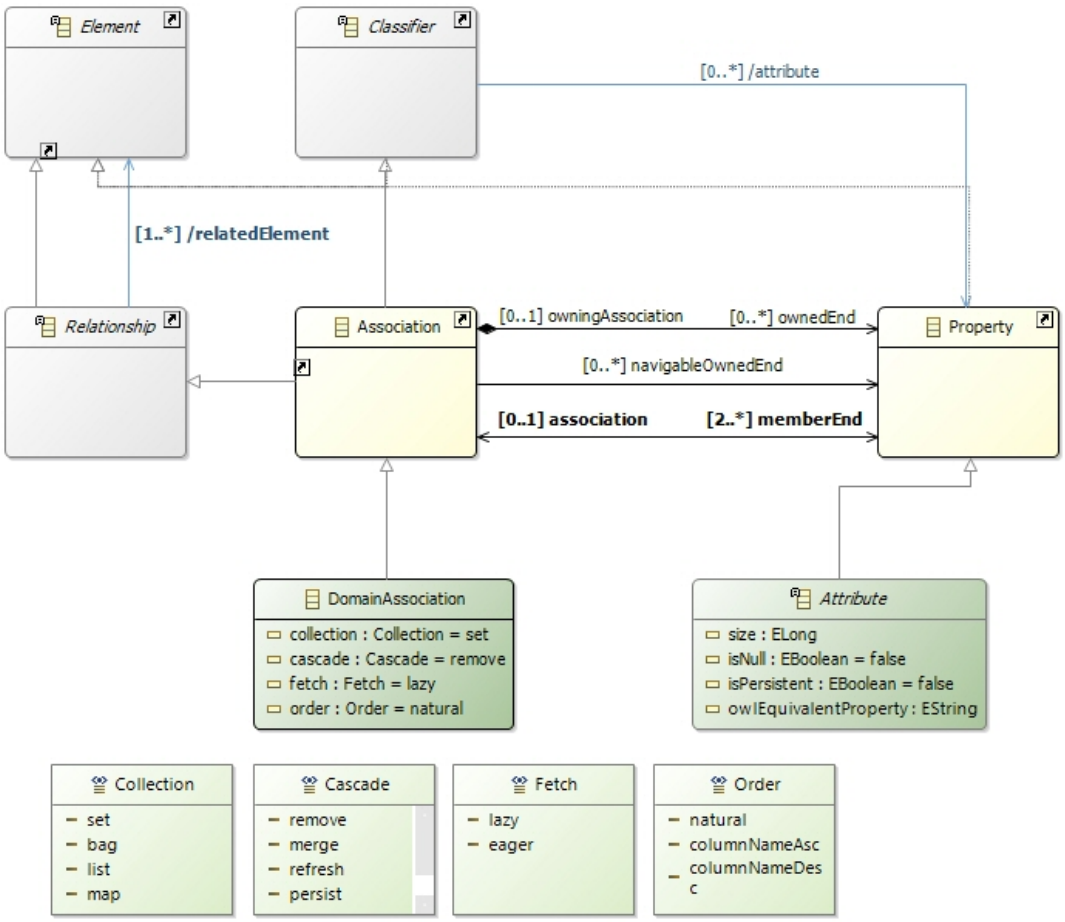

Figura 6 – Fragmento do metamodelo de Entidades do método FrameWeb.

específicas dos *frameworks* utilizados. A Figura 7 apresenta um fragmento do metamodelo de *frameworks*, onde em HTML por exemplo, componentes de uma página do tipo `inputText`, `body`, `h1`, etc.. seriam instâncias da classe `Tag` especificada no metamodelo.

Estes metamodelos são fundamentais para a construção de ferramentas que verifiquem a sintaxe dos modelos construídos, ofereçam geração de código, dentre outras funcionalidades possíveis (MARTINS; SOUZA, 2015; MARTINS, 2016), tendo sido utilizados como base (sendo adaptado à medida do necessário) para o desenvolvimento da ferramenta FrameWeb Editor apresentada neste trabalho.

2.2.6 FrameWeb-LD

FrameWeb-LD (CELINO et al., 2016), é uma extensão do método FrameWeb que propõe uma abordagem para o desenvolvimento de WIS's com integração à Web Semântica, que tem como principal ideia interligar dados (*Linked Data*) ao invés de documentos e adicionar semântica (significado) a essas ligações.

Esta proposta envolve:

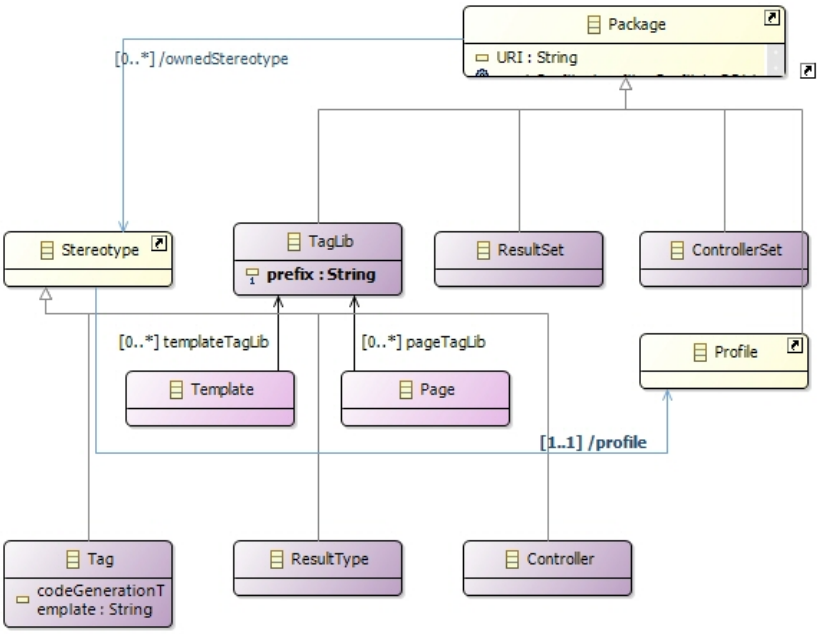

Figura 7 – Fragmento do metamodelo de *frameworks* do método FrameWeb.

- Uma extensão do metamodelo FrameWeb, permitindo que os mapeamentos relacionados a *linked data* sejam incorporados aos modelos;
- Uma ferramenta para geração de código, auxiliando os desenvolvedores na publicação dos dados.

Os conceitos referentes a *linked data* são incorporados ao modelo de entidades do método FrameWeb adicionando anotações RDF a este modelo, desta forma, o metamodelo foi extendido para permitir a inclusão destas anotações, que especificam como os dados do WIS se relacionam aos vocabulários existentes na Web Semântica. Um fragmento deste metamodelo é apresentado na Figura 8.

2.3 Trabalhos Relacionados

A seguir são apresentados alguns métodos que fazem uso de MDD no desenvolvimento de Sistemas de Informação Web e suas respectivas ferramentas, sendo apresentados com foco comparativo ao contexto de aplicação do método FrameWeb e da ferramenta CASE apresentada neste trabalho.

O método UWE (UML-based Web Engineering) (KOCH et al., 2008b) é um método MDD para desenvolvimento de aplicações Web e que possui a ferramenta CASE ArgoUWE,[5] um plugin para a ferramenta ArgoUML[6] em que os diagramas são construídos

[5] <http://uwe.pst.ifi.lmu.de/toolargoUWE.html>
[6] <http://argouml.tigris.org/>

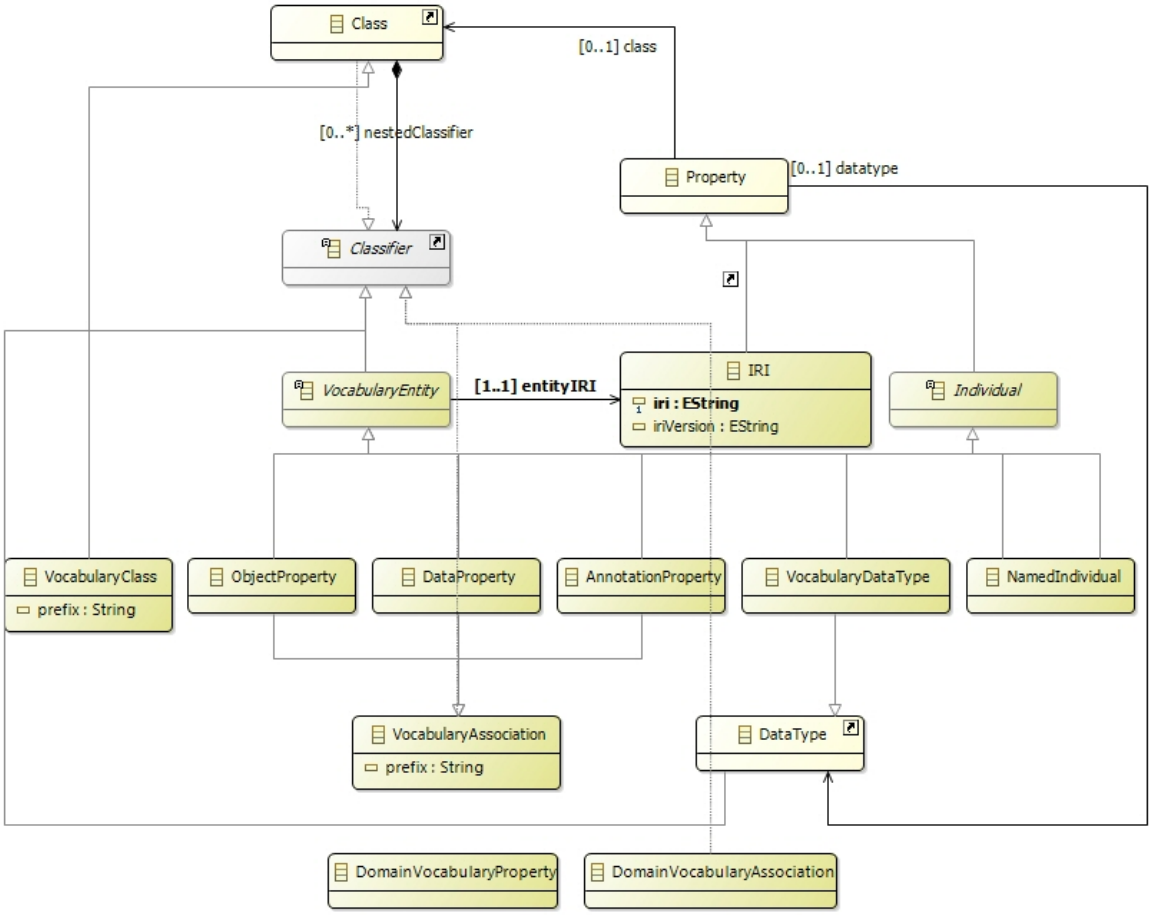

Figura 8 – Fragmento do metamodelo de Vocabulário do método FrameWeb.

utilizando elementos de modelagem específicos para representar os conceitos introduzidos na metodologia UWE. Esse método assemelha-se ao FrameWeb quanto à separação da modelagem de conteúdo bem como a separação entre as partes independentes de plataforma e as dependentes. O grande diferencial se faz no fato do método não fazer menção aos conceitos e funcionalidades disponíveis com o uso de *frameworks*, que são incorporados aos modelos FrameWeb na ferramenta FrameWeb Editor por meio da importação dos arquivos de definições de *frameworks*, que podem ser criados e atualizados na própria ferramenta, permitindo a extensibilidade com relação à atualização e surgimento de novos *frameworks* disponíveis para utilização.

O RUX-Method (CALVARY et al., 2003) é um método para desenvolvimento de aplicações Web com características próprias para a criação de interfaces sensíveis a contexto. O método estrutura todo o ciclo de vida do desenvolvimento e oferece suporte à geração de código para diferentes dispositivos e plataformas. Este método também possui uma ferramenta implementada chamada RUX-Tool (LINAJE et al., 2009), que oferece suporte à modelagem e geração de código automática de interfaces com o usuário multiplataforma e para diferentes dispositivos. Por outro lado, assim como UWE, distancia-se do método FrameWeb e de seu editor principalmente com relação à extensibilidade, sendo muito

dependente de sua biblioteca de componentes.

O padrão IFML (BARESI; GARZOTTO; PAOLINI, 2001) é uma linguagem visual independente de plataforma baseada em uma abordagem tradicional do padrão MVC, que traz aos modelos conceitos inerentes à navegabilidade das aplicações e operações que afetam seus conteúdos e estados de navegação considerando recursos integrados da *hypermedia*. O IFML possui uma ferramenta chamada IFML Editor,[7] que possui a mesma arquitetura do *FrameWeb Editor*, sendo desenvolvida com base na tecnologia Sirius (VIYOVIC; MAKSIMOVIC; PERISIC, 2014). Porém como os demais métodos, não faz menção ou oferece funcionalidades referentes aos *frameworks* utilizados, abrangendo somente a parte independente de plataforma no desenvolvimento dos modelos.

O OOH4RIA (MELIÁ et al., 2008) é um método baseado no *framework* da Google™ (Google Web Toolkit - GWT)[8] para plataforma Java utilizando uma abordagem dirigida por modelos. O método é focado no desenvolvimento de RIAs (*Rich Internet Applications*) e possui a ferramenta OOH4RIA Tool (MELIÁ et al., 2009), que permite representar os modelos e transformações para acelerar o processo de desenvolvimento de uma RIA implementada com o *framework* GWT. A principal limitação desta abordagem é que sob certo ponto de vista o fato de ser baseado no GWT enrijece o método, pois sua visão independente de plataforma não é de fato totalmente independente, já que está intimamente relacionada a aspectos da plataforma escolhida.

[7] <http://ifml.github.io/>
[8] <http://www.gwtproject.org/>

3 FrameWeb Editor

Esta seção apresenta o *FrameWeb Editor*, ferramenta CASE desenvolvida inteiramente neste trabalho e baseada nos metamodelos desenvolvidos em (MARTINS; SOUZA, 2015; MARTINS, 2016). A Seção 3.1 apresenta uma visão geral da ferramenta, a Seção 3.2 descreve seu uso e suas principais funcionalidades, a Seção 3.3 descreve a relação da ferramenta construída com os metamodelos que definem formalmente a linguagem, a Seção 3.4 descreve como a ferramenta atende às características de extensibilidade propostas pelo método, a Seção 3.5 apresenta as funcionalidades de verificação sintática dos modelos, a Seção 3.6 apresenta sua arquitetura e a Seção 3.7 conclui.

Além deste capítulo, um vídeo também descreve o *FrameWeb Editor* e algumas de suas características. O vídeo encontra-se disponível em <https://youtu.be/3cpkBcGIfUw>.

3.1 Visão Geral

A ferramenta *FrameWeb Editor* tem como objetivo principal guiar os desenvolvedores na utilização do método FrameWeb e, para isso, possui algumas características como:

- Suporte à construção dos quatro tipos de diagramas FrameWeb (Entidades, Persistência, Navegação e Aplicação), com interfaces e painéis personalizados para cada tipo de diagrama;
- Funcionalidades comuns em ferramentas de modelagem UML, como recursos de "arrastar-soltar" para fácil criação de relações, de forma a facilitar a adesão e utilização de desenvolvedores já habituados a este tipo de modelagem;
- Restrição das possibilidades de criação de componentes apenas aos específicos do método, baseado no modelo que está sendo desenvolvido;
- Validação dos modelos construídos, de forma a impedir ações inválidas e verificar restrições de integridade, auxiliando no desenvolvimento de modelos válidos;
- Fácil inserção de componentes e propriedades relacionadas aos frameworks e da linguagem de programação utilizada.

Desta forma, para o desenvolvimento de um WIS utilizando efetivamente o método FrameWeb é necessário apenas um conhecimento prévio de suas definições e propriedades apresentadas por Souza (2007), utilizando o *FrameWeb Editor* para o desenvolvimento dos

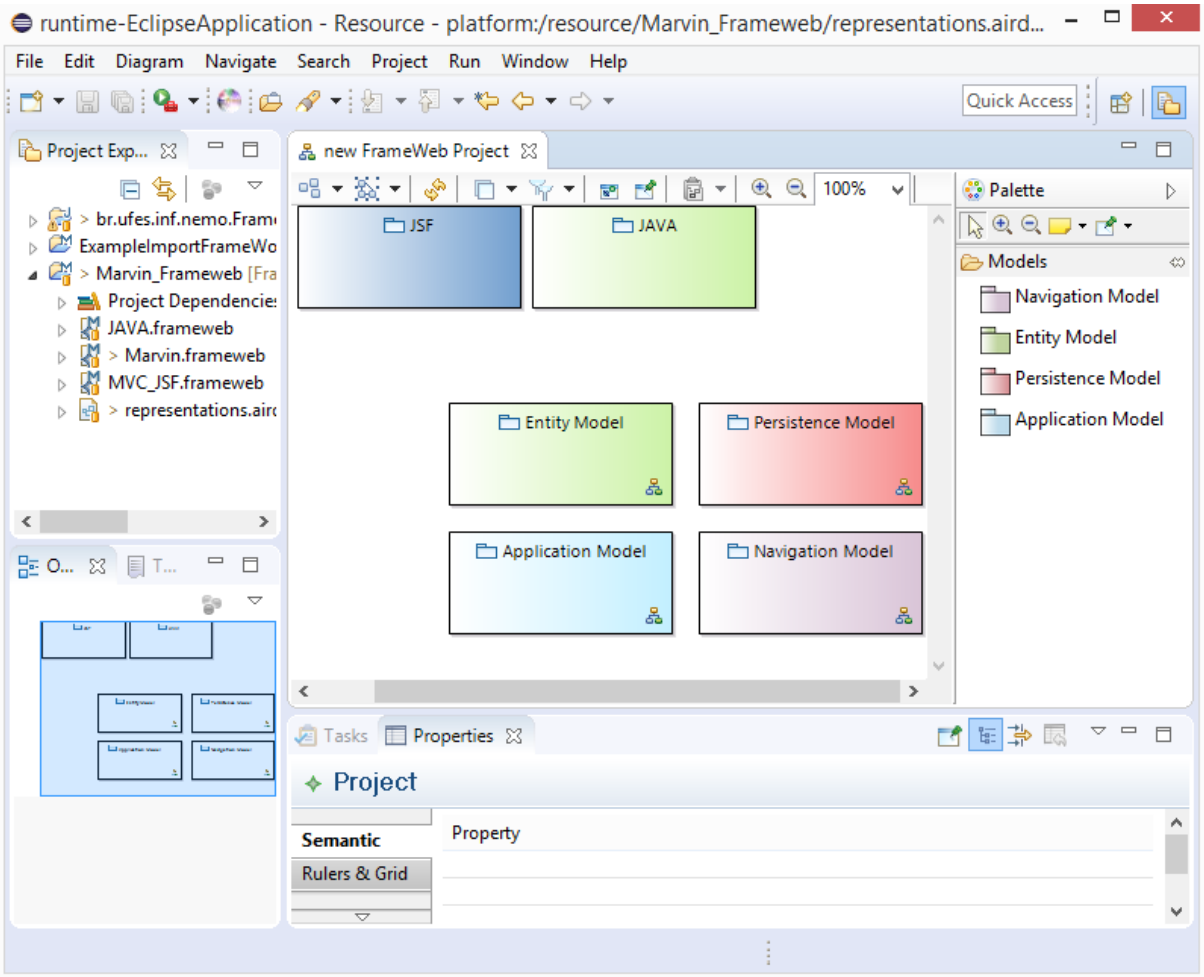

Figura 9 – Diagrama de projeto de um WIS modelado com FrameWeb usando o editor.

modelos e posteriormente sendo realizada a geração de código utilizando uma ferramenta externa, apresentada por Almeida, Campos e Souza (2017), e que futuramente será integrada ao próprio editor para automatização do processo.

3.2 Diagramas e Interface com o Usuário

Como citado no Capítulo 2, os modelos FrameWeb são baseados em quatro tipos básicos de diagramas: Entidades, Navegação, Persistência e Aplicação. O *FrameWeb Editor* oferece suporte para a criação destes quatro tipos de diagramas e, além destes, um novo tipo de diagrama, que oferece uma visão geral dos modelos construídos, e os *frameworks*, linguagens de programação e vocabulários que estão sendo utilizados no projeto, como mostra a Figura 9, que no projeto em questão, mostra que foram desenvolvidos os quatro tipos básicos de diagramas e mostra também que está sendo utilizada a tecnologia JavaServer Faces (JSF) como *framework* Web e a plataforma Java, representados pelos componentes em azul e verde no canto superior do diagrama.

Ainda na Figura 9, podemos analisar a interface principal da ferramenta, que

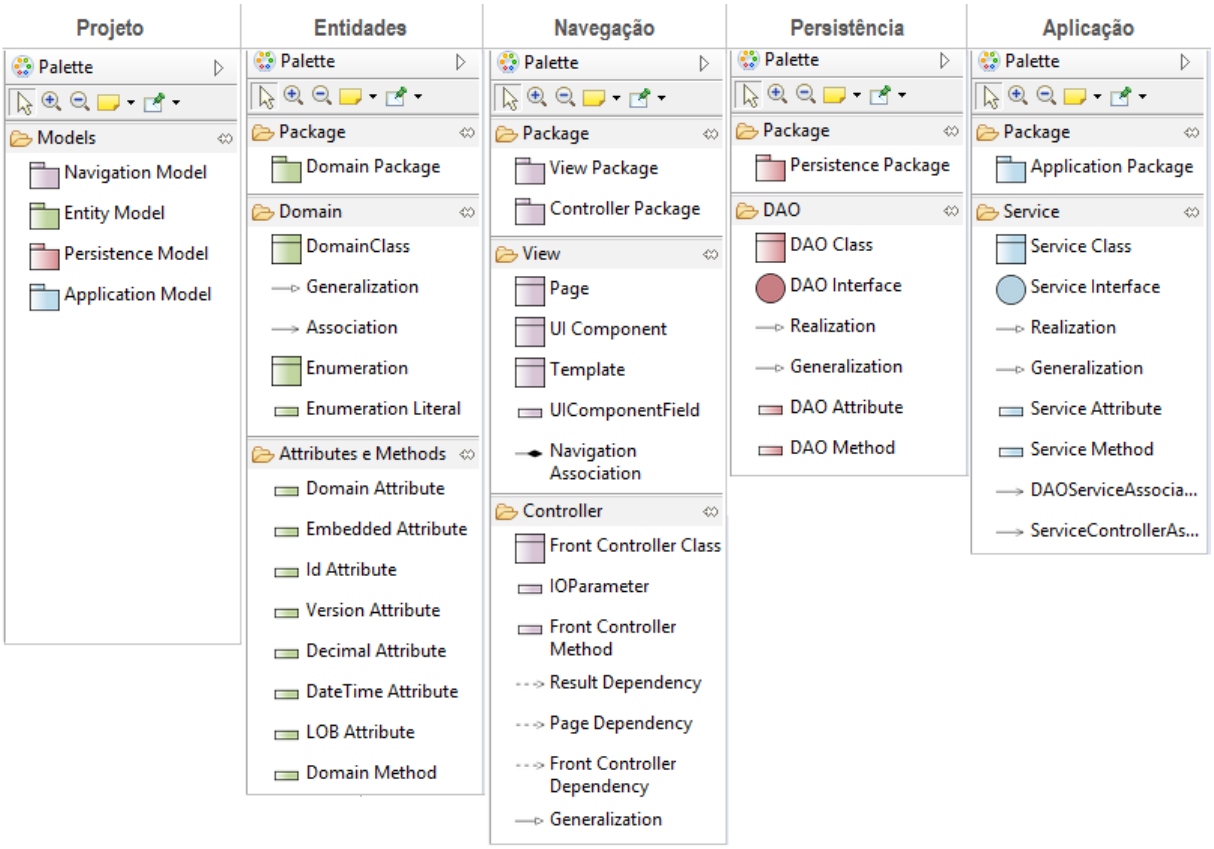

Figura 10 – Paletas de componentes para cada um dos tipos de modelos FrameWeb.

possui:

- A janela principal, onde os modelos são desenvolvidos;
- Um *Project Explorer*, que provê uma visão hierárquica do projeto e de seu *workbench*;
- Uma janela *Outline*, que apresenta um esboço geral do modelo em questão;
- Uma aba *Properties*, onde são vistas e alteradas todas as propriedades referentes ao componente selecionado;
- Um paleta de componentes, onde são apresentadas todas as possibilidades de componentes que podem ser criados no modelo em questão.

Esta paleta de componentes é diferente para cada tipo de modelo FrameWeb, sempre apresentando componentes específicos do modelo que está sendo construído, sendo cada componente relacionado diretamente ao respectivo elemento definido no metamodelo apresentado na Seção 2.2.5, facilitando o processo de desenvolvimento e impedindo que sejam adicionados componentes que não pertençam ao modelo em questão. A Figura 10 apresenta as paletas de componentes para cada um dos tipos de modelos FrameWeb.

A Figura 11 mostra um fragmento de um Modelo de Navegação do *Marvin*,[1] um sistema de apoio a atividades acadêmicas. Seguindo a estrutura MVC, temos neste fragmento do modelo a página `index.xhtml`, composta de um formulário que envia os dados para o controlador `GenerateBibliographyController`, que por sua vez é associado à página `bibliography.xhtml` por meio do método `generate` e novamente à pagina `index.xhtml` através do método `back`.

Os modelos construídos na ferramenta são salvos em arquivos `.frameweb` em um formato XML com uma sintaxe específica do EMF. Posteriormente, tais arquivos podem ser processados para o desenvolvimento de outras funcionalidades como, por exemplo, geração de código, conforme proposto em (ALMEIDA; CAMPOS; SOUZA, 2017).

3.3 Relação com os Metamodelos

A ferramenta foi desenvolvida com base nos metamodelos citados na Seção 2.2.5, e que definem os elementos que poderão ser criados no modelador gráfico, bem como suas propriedades e relações. Os metamodelos são especificados por meio do *Eclipse Modeling Framework* (*EMF*), sendo inicialmente definidos em (MARTINS; SOUZA, 2015; MARTINS, 2016) e vem sendo refinados à medida do necessário.

Neste trabalho não foram desenvolvidos novos metamodelos, o FrameWeb Editor apenas relaciona os componentes gráficos da ferramenta diretamente com os componentes especificados nos metamodelos desenvolvidos em (MARTINS; SOUZA, 2015; MARTINS, 2016). Isto significa que, diferentemente de modelos construídos utilizando um editor UML genérico, os componentes criados por meio do *FrameWeb Editor* possuem uma semântica relacionada ao método, o que possibilita o processamento destes modelos para novas funcionalidades, como por exemplo:

- A verificação e validação dos modelos construídos, conforme apresentado na Seção 3.5;
- O processamento dos modelos construídos para geração de código. Funcionalidade apresentada em (ALMEIDA; CAMPOS; SOUZA, 2017);
- Outras possíveis transformações de Modelo para Texto *(model to text - M2T)*, Modelo para Modelo *(model to model - M2M) e Texto para Modelo(text to model - T2M)*.

Atualmente está sendo desenvolvida a integração da ferramenta com o gerador de código. Tem-se ainda como trabalho futuro o desenvolvimento de uma transformação Texto para Modelo, aplicando engenharia reversa em sistemas já desenvolvidos para a manutenção e documentação dos mesmos.

[1] <https://github.com/dwws-ufes/Marvin>

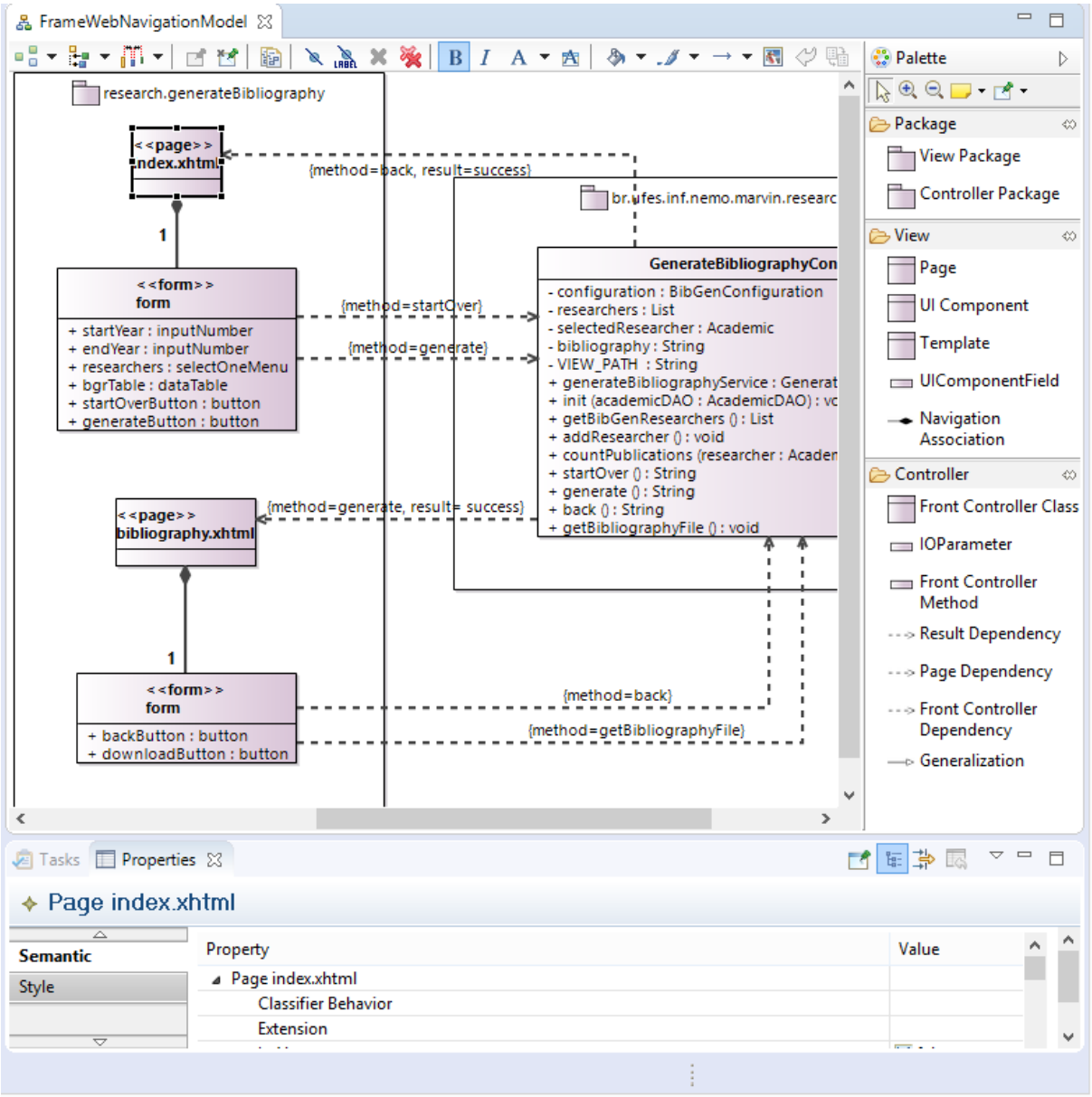

Figura 11 – Diagrama de Navegação do Marvin modelado com o *FrameWeb Editor*.

3.4 Extensibilidade

Uma das principais características e diferenciais do método FrameWeb com relação a outros métodos é a proposta de um método que possa se adequar aos diferentes *frameworks* utilizados, trazendo para os modelos os conceitos inerentes aos mesmos.

Com a grande variedade de *frameworks* existentes e a velocidade com que estes são atualizados e de que novos *frameworks* são criados, é necessária uma forma modular da ferramenta se adequar a estas constantes alterações. Isto é feito por meio dos chamados arquivos de definição de frameworks, que são arquivos `.frameweb` contendo todas as classes e propriedades relacionadas ao *framework* em questão, e que são especificados pelo método FrameWeb através do metamodelo de *framework*, conforme demonstrado na Seção 2.2.5.

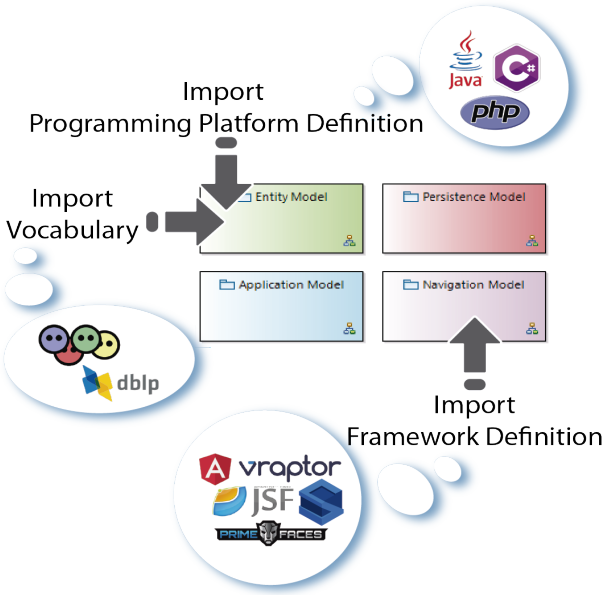

Figura 12 – Visão geral da extensibilidade da ferramenta com relação aos *frameworks*, plataformas de programação e vocabulários *linked data*.

Quando inseridos no projeto, estes arquivos possibilitam a utilização destes conceitos no desenvolvimento dos modelos.

Esta característica é também desenvolvida para dar suporte a diferentes linguagens de programação e para trazer aos modelos de entidades conceitos inerentes à *linked data* apresentados na extensão do método FrameWeb proposta por Celino et al. (2016) e relacionados aos vocabulários utilizados. A Figura 12 apresenta uma visão geral destes arquivos e de sua influência nos respectivos modelos FrameWeb.

Um arquivo de definição de linguagem de programação é um arquivo contendo os tipos de dados referentes àquela linguagem. Por exemplo, o arquivo `JAVA.frameweb` possui classes como `Date, String, etc...` Já os arquivos de definição de vocabulários possuem as entidades e propriedades do vocabulário em questão, possibilitando a ligação das mesmas com classes definidas pelo desenvolvedor no modelo de entidades, facilitando assim o desenvolvimento de sistemas preparados para a Web Semântica.

Por exemplo, para um determinado projeto desenvolvido em Java, utilizando JSF e a biblioteca de componentes PrimeFaces e mapeando os dados ao vocabulário DBLP, tais definições seriam importadas no editor, permitindo que:

- Os componentes PrimeFaces sejam usados como tipos dos atributos dos formulários representados em Modelos de Navegação;
- Regras específicas do JSF se apliquem na relação entre páginas, formulários e controladores nestes mesmos modelos;

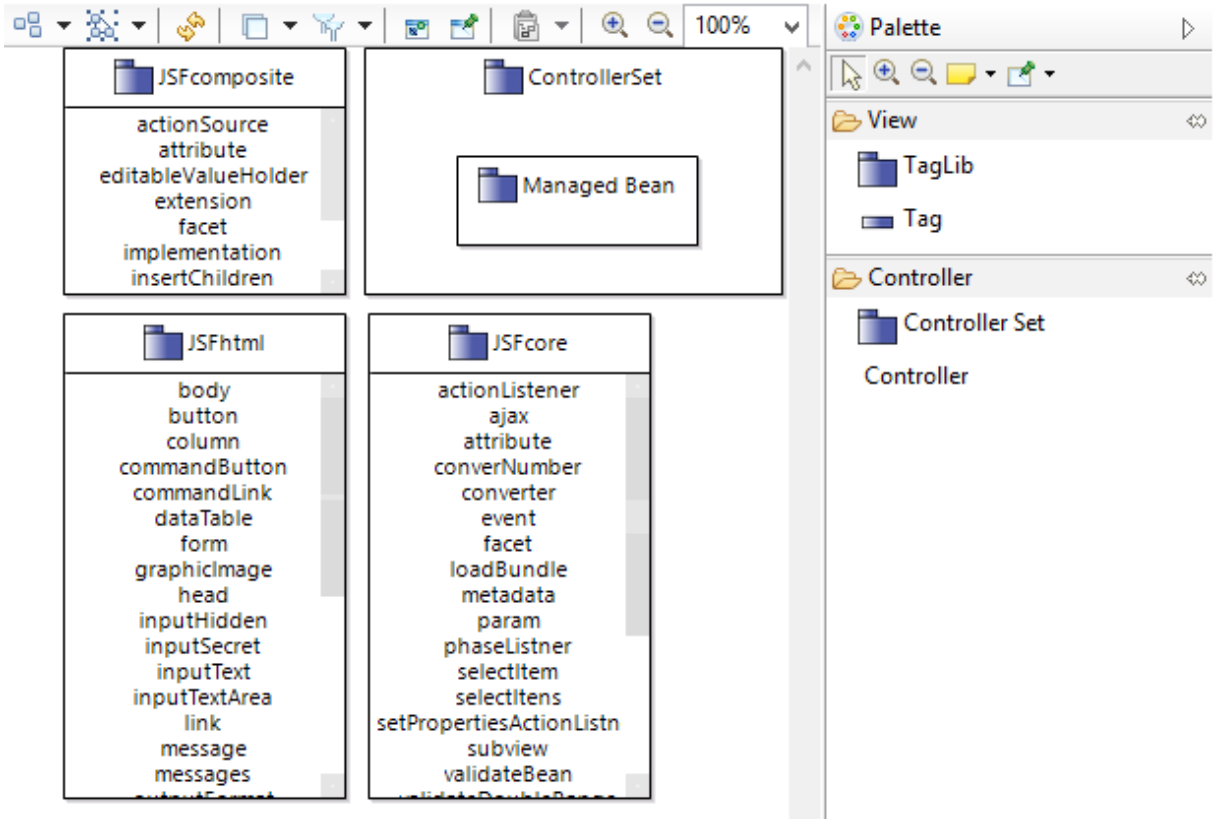

Figura 13 – Arquivo de definição do framework JSF, utilizando o *FrameWeb Editor*.

- Tipos básicos da linguagem Java sejam usados nos atributos das classes do Modelo de Entidades;
- Ligações entre as classes/atributos deste mesmo modelo sejam estabelecidas com classes e propriedades do vocabulário DBLP.

Além de implementar a funcionalidade de importação destes respectivos arquivos, a ferramenta *FrameWeb Editor* possui interfaces e painéis próprios para a visualização, criação e edição destes arquivos, como mostra a Figura 13, de forma que o desenvolvedor possa facilmente atualizar os mesmos à medida que for necessário e até mesmo criar novos arquivos caso queira utilizar determinado *framework*/linguagem/vocabulário que ainda não esteja implementado no repositório da ferramenta.

Esta abordagem permite que o desenvolvedor tenha a possibilidade de utilização do método FrameWeb no processo de desenvolvimento de suas aplicações independentemente da linguagem ou *framework* que utilizar, não aplicando restrições ou forçando a utilização de determinado *framework* pré-estabelecido, visto que ainda possam estar sendo utilizadas linguagens e *frameworks* legados que não possuam suporte em ferramentas mais atuais.

3.5 Verificação sintática dos Modelos

A ferramenta *FrameWeb Editor* tem como objetivo guiar o desenvolvedor na construção de modelos FrameWeb válidos, e portanto necessita de funcionalidades que garantam a integridade dos modelos desenvolvidos. Em Engenharia de Software, o termo *validação* é usado para identificar se um sistema atende aos seus requisitos originais impostos por parte do cliente, enquanto o termo *verificação* diz respeito à correção dos modelos desenvolvidos. Portanto, apesar da terminologia adotada pelo EMF e herdada pelo *FrameWeb Editor* se chamar *validate* em sua interface com o usuário, a funcionalidade oferecida pelo editor está relacionada com a verificação dos modelos.

Essa funcionalidade é implementanda de duas formas:

1. A ferramenta impede que sejam realizadas ações inválidas durante a modelagem, como, por exemplo, a criação de classes fora do pacote ao qual devem pertencer, a criação de relações entres classes que não podem ser relacionadas, impedindo que sejam criadas inconsistências;

2. A ferramenta possui a função *validate*, que conforme citado anteriormente, diz respeito à uma verificação, e que pode ser chamada por meio do menu de contexto, exibido ao clicar com o botão direto do mouse na área de modelagem. Tal função irá processar o modelo e indicar possíveis inconsistências, cobrindo casos que não podem ser restritos diretamente e também restrições próprias dos frameworks utilizados.

Este tipo de verificação e restrições de construção citadas no item 1 está relacionado diretamente com a sintaxe abstrata da linguagem, ou seja, os metamodelos, e aplicada no editor através da sintaxe concreta, conforme apresentado na Seção 3.6.

Com relação à função *validate* citada no item 2, é possível definir regras personalizadas para cada tipo de modelo, e que só serão aplicadas ao iniciar uma verificação em uma instância do diagrama em questão. Quando uma regra é violada, um marcador aparecerá nos elementos problemáticos, informando a origem da inconsistência.

Por exemplo, a Figura 14 demonstra a especificação de uma regra relacionada ao modelo de entidades e que diz que todo modelo de entidades deverá possuir pelo menos um `DomainPackage`. A figura mostra, ainda, como essa regra é apresentada no diagrama pelo editor ao ser chamada a função *validate*.

3.6 Arquitetura da Ferramenta

A arquitetura do *FrameWeb Editor* é ilustrada na Figura 15. A ferramenta foi desenvolvida com base na tecnologia Sirius (VIYOVIC; MAKSIMOVIC; PERISIC, 2014),

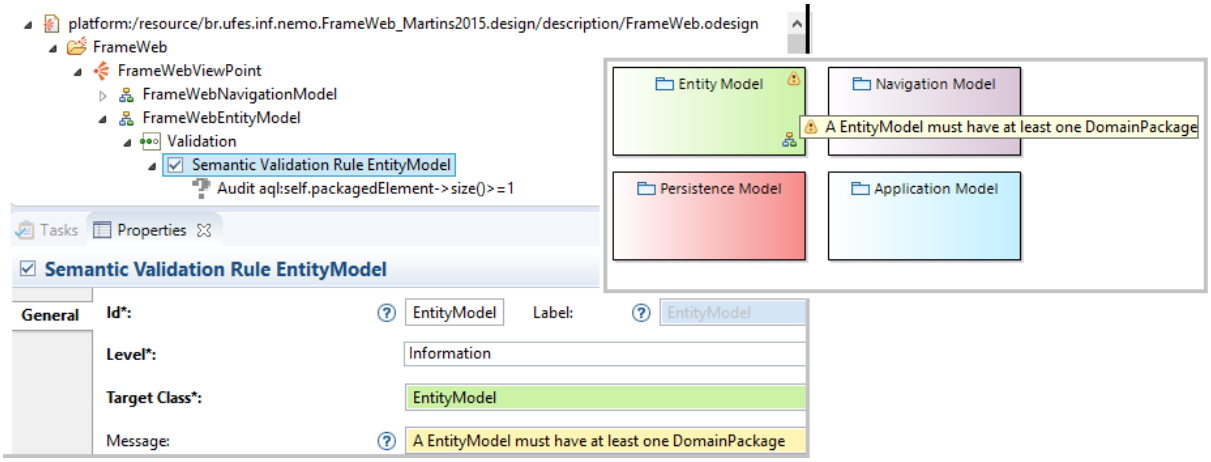

Figura 14 – Regra de validação especificada para o Modelo de Entidades.

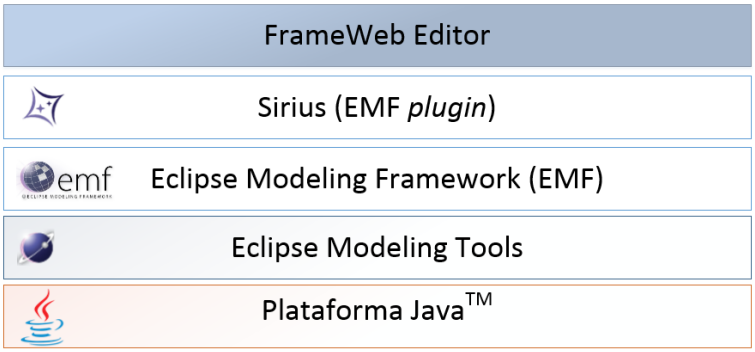

Figura 15 – Arquitetura do *FrameWeb Editor*.

que oferece suporte à definição de representações gráficas de componentes e provê diversas funcionalidades comumente utilizadas em editores gráficos.

A construção de uma ferramenta de modelagem gráfica utilizando o *Sirius* passa pela definição de dois modelos relacionados ao domínio em questão (em nosso caso, FrameWeb). O modelo da sintaxe abstrata, ou meta-modelo, define os elementos que poderão ser criados no modelador gráfico, bem como suas propriedades e relações, e é especificada por meio do *Eclipse Modeling Framework* (*EMF*), um *framework* de modelagem e geração de código para construção de ferramentas e outras aplicações com base em modelos estruturados de dados (GRONBACK, 2009). O modelo da sintaxe concreta, chamado pelo *Sirius* de *viewpoint specification*, define as características gráficas dos elementos definidos no meta-modelo. Ambos os modelos são especificados utilizando a sintaxe do *EMF*.

A Figura 16 mostra um fragmento da *viewpoint specification* definida no *Sirius*, em que o elemento gráfico *Page Node* e suas respectivas configurações de apresentação é associado à classe *Page* definida no meta-modelo e então utilizado no *FrameWeb Editor* como mostrado na Figura 11.

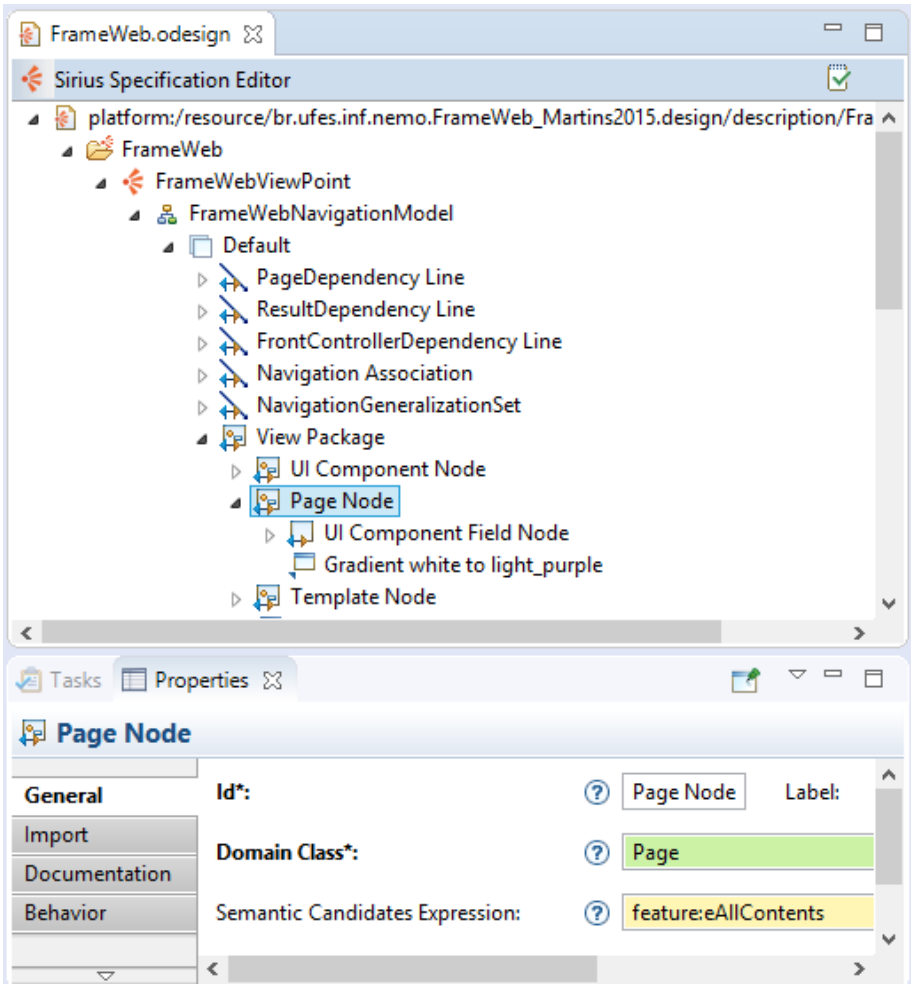

Figura 16 – Fragmento do FrameWeb Viewpoint Specification, referente à classe *Page*.

3.7 Conclusões do Capítulo

Neste capítulo apresentamos o *FrameWeb Editor*, ferramenta CASE voltada ao método FrameWeb. Por meio das funcionalidades e características apresentadas, a ferramenta visa facilitar a utilização do método FrameWeb guiando os desenvolvedores na construção de seus modelos de forma mais ágil e direcionada, diminuindo a curva de aprendizado para desenvolvedores que desejam começar a utilizar o método e apoiando os que já o utilizam, obtendo modelos válidos e consistentes ao final do processo de modelagem.

4 Validação

Para validar a ferramenta e sua fidelidade aos conceitos e entidades propostas pelo método FrameWeb, foram remodelados diversos trabalhos previamente desenvolvidos utilizando o método FrameWeb porém modelados através de editores UML genéricos. Neste capítulo, apresentamos os modelos do SCAP, um sistema para controle de afastamento de professores proposto, em (PRADO, 2015). Mais exemplos de projetos desenvolvidos utilizando o FrameWeb Editor podem ser encontrados no repositório do projeto em <https://github.com/nemo-ufes/FrameWeb/>

O SCAP é um sistema que auxilia professores e secretários do Departamento de Informática (DI) da UFES na tramitação de solicitações de afastamento, que necessitam ser avaliadas por professores do DI e pela diretoria do Centro Tecnológico (CT) juntamente com a Pró-Reitoria de Pesquisa e Pós-Graduação (PRPPG). O sistema facilita todo o processo, desde a solicitação até a análise e armazenamento da mesma (PRADO, 2015).

O SCAP foi implementado utilizando a linguagem de programação Java, na plataforma Java EE 7 (Java Enterprise Edition 7), sendo utilizado como controlador frontal o *framework* VRaptor, um *framework* MVC Web para desenvolvimento ágil com Java, e para a persistência o framework JPA (Java Persistence API), simplificando a lógica de acesso aos dados.

Utilizando-se o método FrameWeb, foram desenvolvidos, portanto, os quatro tipos básicos de modelos FrameWeb, especificando as classes, funcionalidades e demais propriedades do SCAP, como mostrado a seguir. A Figura 17 apresenta um fragmento do modelo de entidades, contendo as classes de domínio que foram utilizadas na implementação, e apresentando um comparativo do modelo original feito em um editor UML genérico, e o mesmo modelo desenvolvido com o FrameWeb Editor.

Conforme proposto pelo método FrameWeb, foram desenvolvidos modelos de navegação para todos os casos de uso do SCAP, a Figura 18 apresenta o diagrama para o caso de uso Cadastrar Afastamento, comparando novamente o modelo original com o modelo desenvolvido com o FrameWeb Editor, assim como nas figuras 19 e 20, que apresentam os modelos de Aplicação e Persistência, respectivamente.

Nos modelos apresentados, podemos observar que, graficamente, os modelos são muito similares, portanto a principal diferença entre ambos é que os modelos desenvolvidos utilizando o FrameWeb Editor possuem uma semântica atribuída a cada entidade do modelo, possibilitando a validação dos modelos e que os modelos possam ser processados posteriormente para geração de código, enquanto na abordagem adotada previamente os modelos serviriam apenas de referência, sendo a implementação feita de forma manual.

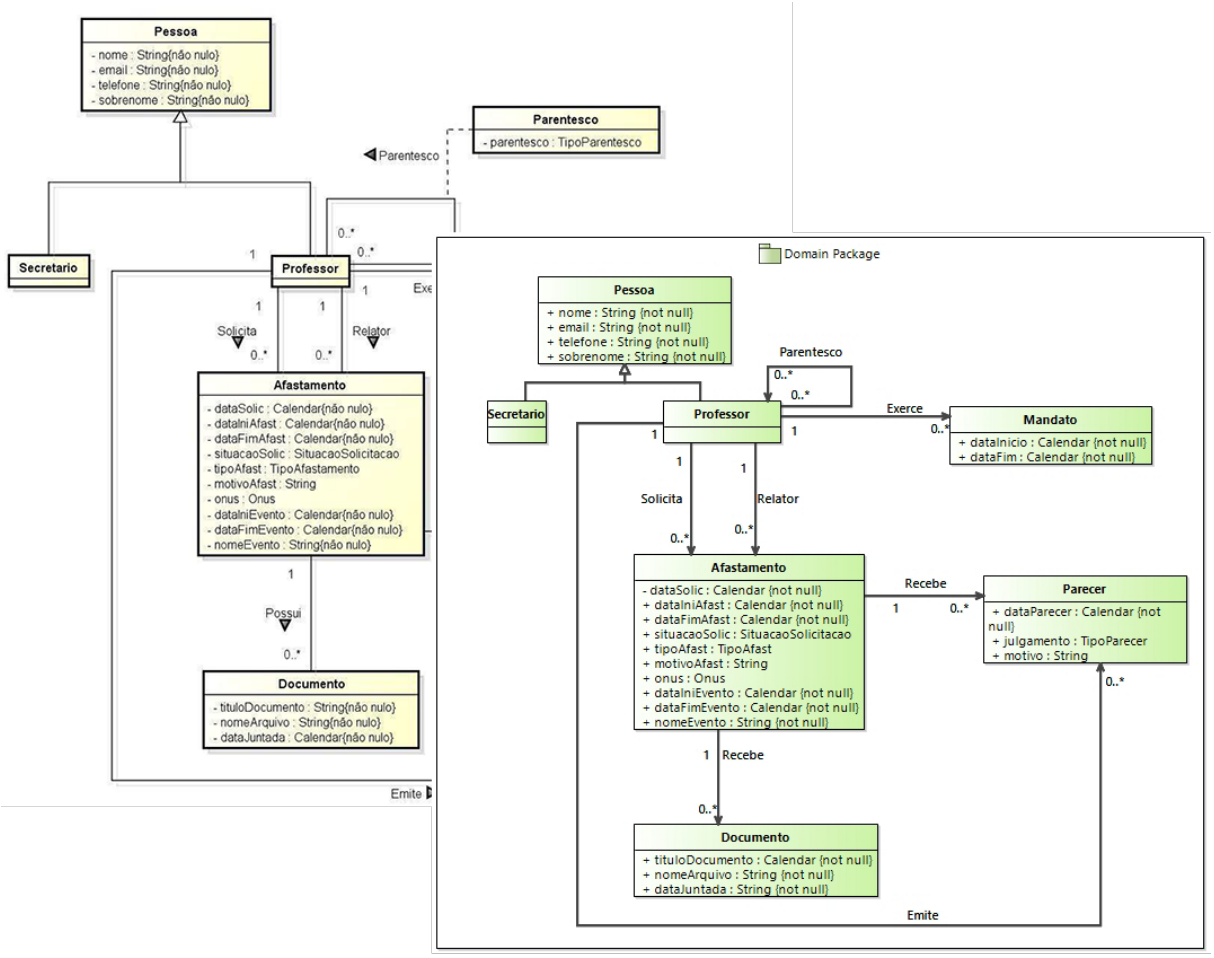

Figura 17 – Diagrama de entidades do SCAP, na esquerda, o modelo original proposto em (PRADO, 2015), na direita o modelo desenvolvido através do FrameWeb Editor.

Apesar das vantagens encontradas no uso do FrameWeb Editor para a utilização do método FrameWeb, a ferramenta ainda não se encontra em versão comercial, ou mesmo uma versão executável de fácil instalação, sendo necessária a configuração do ambiente de desenvolvimento do *Sirius* para sua execução, o que de certa forma impede que seja utilizada na prática por desenvolvedores. Tem-se como trabalhos futuros facilitar este processo, e assim possibilitando uma avaliação formal e sua utilização por um número maior de usuários.

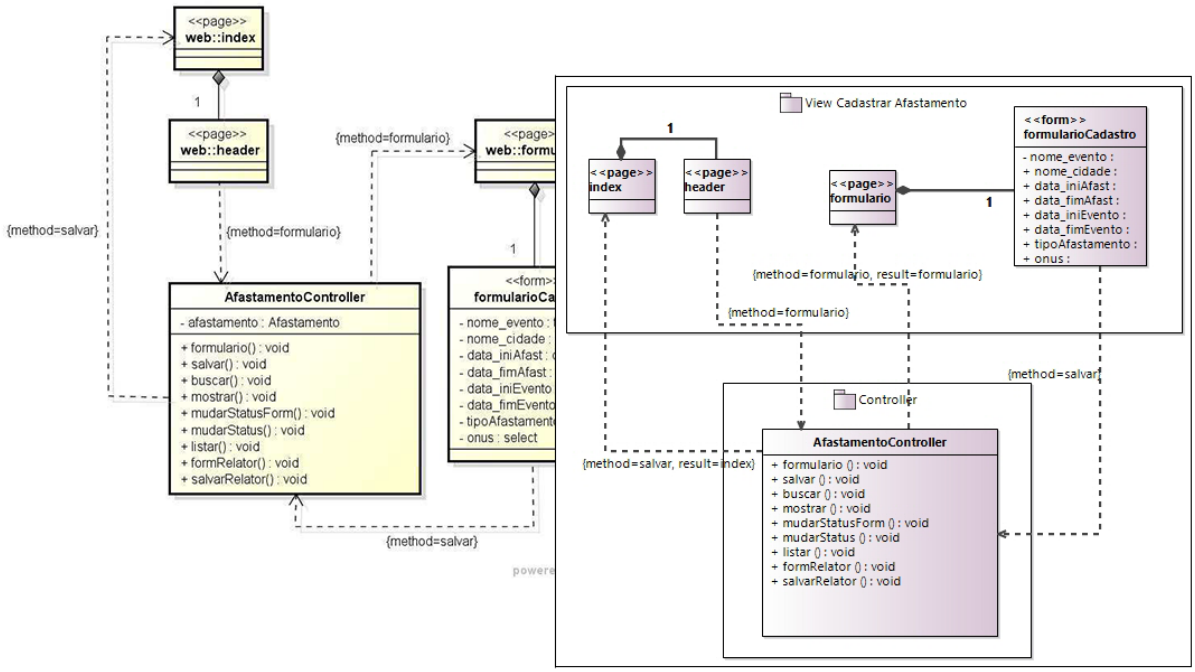

Figura 18 – Diagrama de navegação para o caso de uso Cadastrar Afastamento do SCAP, na esquerda, o modelo original proposto em (PRADO, 2015), na direita o modelo desenvolvido através do FrameWeb Editor.

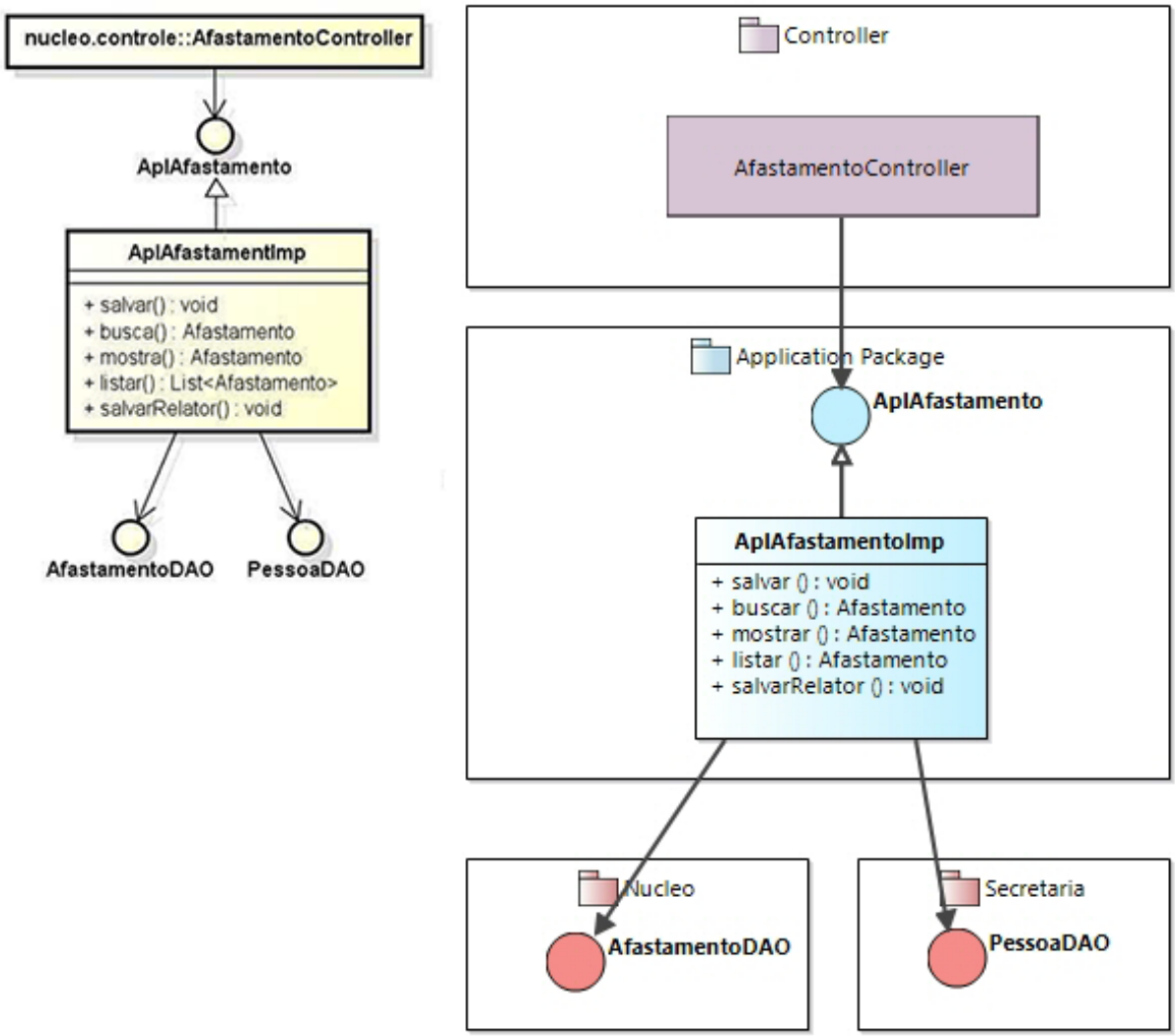

Figura 19 – Fragmento do diagrama de aplicação do SCAP, na esquerda, o modelo original proposto em (PRADO, 2015), na direita o modelo desenvolvido através do FrameWeb Editor.

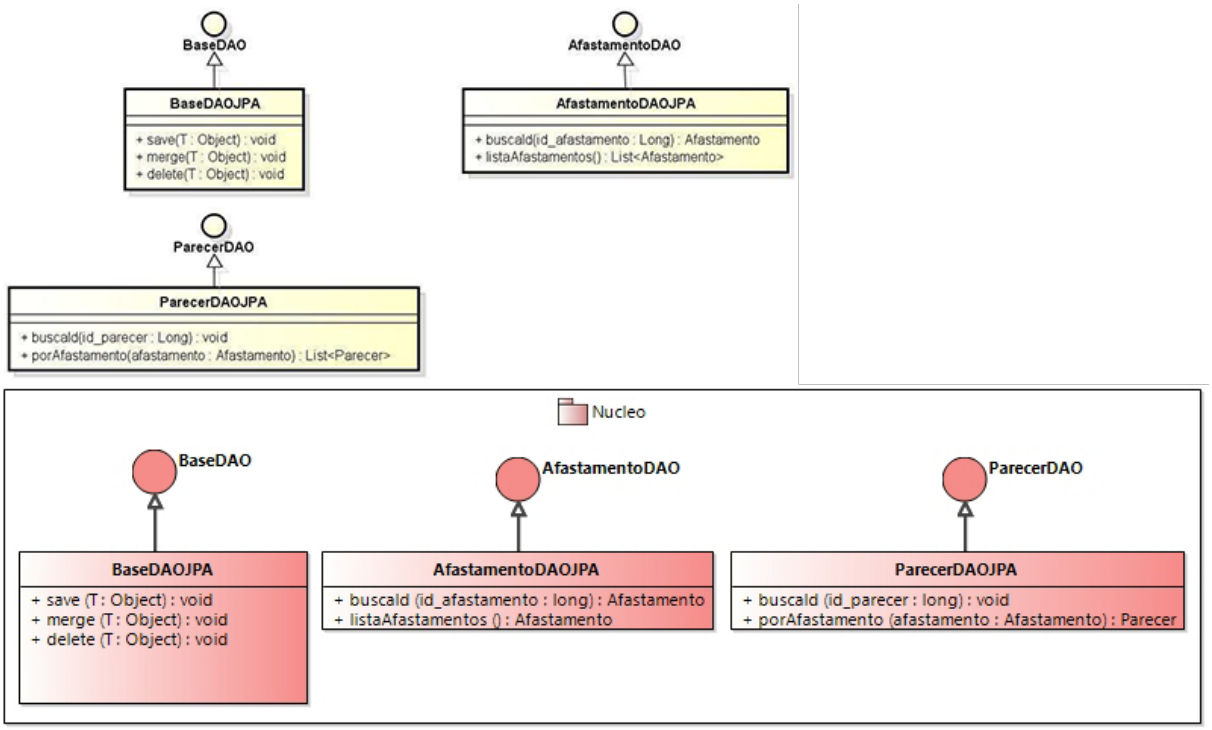

Figura 20 – Fragmento do diagrama de entidades do SCAP, acima, o modelo original proposto em (PRADO, 2015), abaixo o modelo desenvolvido através do FrameWeb Editor.

5 Considerações Finais

Conforme citado no Capítulo 1, em sua proposta original, o método FrameWeb define extensões leves do metamodelo UML, sendo os modelos FrameWeb desenvolvidos utilizando-se um editor UML genérico. Esta abordagem resultava em algumas limitações, como o fato do perfil UML não ser rigoroso, ou seja, não possuir forma de prevenir que se incluam elementos que não pertençam ao modelo, e o fato dos componentes não estarem relacionados à sintaxe do método, o que não permite que estes modelos sejam processados para validação e posteriormente para geração de código.

A ferramenta FrameWeb Editor proposta neste trabalho supre esta necessidade do método por uma ferramenta CASE mais apropriada para o método, oferecendo todo o suporte para que analistas/desenvolvedores utilizem o método FrameWeb para o desenvolvimento de sistemas de informação Web por meio de interfaces e painéis para a criação de componentes que contém apenas elementos próprios da linguagem, possibilitando uma modelagem mais direcionada, mantendo os recursos comumente utilizados em ferramentas de modelagem, e permitindo a verificação dos modelos para a garantia de modelos sintaticamente corretos.

Além deste suporte para o desenvolvimento dos modelos básicos do método FrameWeb, uma grande vantagem da utilização da ferramenta é a possibilidade de utilização dos arquivos de definição de *framework*/linguagem de programação/vocabulário, conforme demonstrado na Seção 3.4, o que permite a fácil integração dos modelos desenvolvidos com os conceitos relacionados à tecnologia utilizada. O suporte à utilização, criação e edição destes arquivos na própria ferramenta é muito vantajosa por possibilitar que se use qualquer *framework* de sua preferência, até mesmo *frameworks* antigos, que não possuam mais suporte em ferramentas mais atuais, pois se o arquivo ainda não existir no repositório do FrameWeb, o mesmo pode ser criado pelo próprio desenvolvedor para a utilização em seus projetos.

Além de oferecer as funcionalidades apresentadas, a ferramenta também serve de base para que sejam desenvolvidas outras funcionalidades. O *framework* EMF oferece plugins capazes não só de criar modelos, mas também de executar transformações de Modelo para Texto (*model to text - M2T*), transformações de Modelo para Modelo (*model to model - M2M*) e transformações de Texto para Texto (*text to text - T2T*), que podem ser utilizadas de forma integrada à ferramenta para a geração de código a partir dos modelos FrameWeb criados.

Portanto, Tem-se como trabalhos futuros a integração do gerador de código FrameWeb (ALMEIDA; CAMPOS; SOUZA, 2017) ao editor gráfico, assim como o desenvolvi-

mento de novas funcionalidades que facilitem a usabilidade, como por exemplo a utilização de padrões para uma modelagem mais automatizada e o aperfeiçoamento das verificações apresentada na Seção 3.5, em que seria possível definir autocorreções para determinadas inconsistências.

Também são trabalhos futuros a geração de um executável de fácil instalação, conforme citado na Seção 4, possibilitando a realização de avaliações e análises criteriosas de usabilidade da ferramenta, assim como experimentos comparativos com usuários usando a ferramenta proposta e usuários usando ferramentas CASE tradicionais no processo de projeto e desenvolvimento de sistemas de informação Web.

De modo geral, nossa perspectiva é que a ferramenta possa ser trabalhada para que evolua de protótipo para ferramenta comercial e ser usada por desenvolvedores Web na indústria, efetivamente realizando a transferência de tecnologia da academia.

Referências

ALMEIDA, N. V. d.; CAMPOS, S. L.; SOUZA, V. E. S. A model-driven approach for code generation for web-based information systems built with frameworks. In: ACM. *Proceedings of the 23rd Brazilian Symposium on Multimedia and the Web*. [S.l.], 2017. p. (to appear). Citado 3 vezes nas páginas 27, 29 e 41.

ALUR, D.; CRUPI, J.; MALKS, D. *Core J2EE Patterns: Best Practices and Design Strategies*. 2nd. ed. [S.l.]: Prentice Hall / Sun Microsystems Press, 2003. Citado na página 17.

BARESI, L.; GARZOTTO, F.; PAOLINI, P. Extending uml for modeling web applications. In: IEEE. *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*. [S.l.], 2001. p. 10–pp. Citado na página 25.

CALVARY, G. et al. A unifying reference framework for multi-target user interfaces. *Interacting with computers*, Oxford University Press, v. 15, n. 3, p. 289–308, 2003. Citado na página 24.

CAMPOS, S. L.; SOUZA, V. E. S. FrameWeb Editor: Uma Ferramenta CASE para suporte ao Método FrameWeb. In: *Anais do 16º Workshop de Ferramentas e Aplicações, 23º Simpósio Brasileiro de Sistemas Multimedia e Web*. Gramado, RS, Brazil: SBC, 2017. p. 199–203. Citado na página 13.

CELINO, D. R. et al. A Framework-based Approach for the Integration of Web-based Information Systems on the Semantic Web. In: *Proc. of the 22nd Brazilian Symposium on Multimedia and the Web*. [S.l.]: ACM, 2016. p. 231–238. Citado 3 vezes nas páginas 12, 22 e 31.

FALBO, R. A. SABiO: Systematic Approach for Building Ontologies. In: GUIZZARDI, G. et al. (Ed.). *Proc. of the Proceedings of the 1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering*. [S.l.]: CEUR, 2014. Nenhuma citação no texto.

FOWLER, M. *Patterns of Enterprise Application Architecture*. 1. ed. [S.l.]: Addison-Wesley, 2002. ISBN 9780321127426. Citado na página 16.

GAMMA, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. [S.l.]: Pearson Education, 1994. Citado na página 17.

GRONBACK, R. C. *Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit*. 1. ed. [S.l.]: Addison-Wesley Professional, 2009. ISBN 9780321635198. Citado 2 vezes nas páginas 16 e 34.

JOHNSON, R. E. Frameworks=(components+ patterns). *Communications of the ACM*, ACM, v. 40, n. 10, p. 39–42, 1997. Citado na página 10.

KOCH, N. et al. Web Engineering: Modelling and Implementing Web Applications, chap. UML-Based Web Engineering. In: *UML-Based Web Engineering*. London, UK: Springer London, 2008b. Citado na página 23.

KRUEGER, C. W. Software reuse. *ACM Computing Surveys (CSUR)*, ACM, v. 24, n. 2, p. 131–183, 1992. Citado na página 10.

LINAJE, M. et al. Automatic Generation of RIAs Using RUX-Tool and Webratio. In: *Web Engineering: 9th International Conference, ICWE 2009, Proceedings*. San Sebastián, Spain: Springer Berlin Heidelberg, 2009. p. 501–504. ISBN 978-3-642-02818-2. Citado na página 24.

MARTINS, B. F. *Evolução do Método FrameWeb para o Projeto de Sistemas de Informação Web Utilizando uma Abordagem Dirigida a Modelos*. Dissertação (Mestrado) — Programa de Pós-Graduação em Informática, Universidade Federal do Espírito Santo, 2016. Citado 5 vezes nas páginas 11, 12, 22, 26 e 29.

MARTINS, B. F.; SOUZA, V. E. S. A model-driven approach for the design of web information systems based on frameworks. In: ACM. *Proceedings of the 21st Brazilian Symposium on Multimedia and the Web*. [S.l.], 2015. p. 41–48. Citado 5 vezes nas páginas 11, 12, 22, 26 e 29.

MELIÁ, S. et al. Ooh4ria tool: Una herramienta basada en el desarrollo dirigido por modelos para las rias. In: *JISBD*. [S.l.: s.n.], 2009. p. 219–222. Citado na página 25.

MELIÁ, S. et al. A model-driven development for gwt-based rich internet applications with ooh4ria. In: IEEE. *Web Engineering, 2008. ICWE'08. Eighth International Conference on*. [S.l.], 2008. p. 13–23. Citado na página 25.

MURUGESAN, S. et al. Web Engineering: a New Discipline for Development of Web-Based Systems. In: MURUGESAN, S.; DESHPANDE, Y. (Ed.). *Web Engineering - Managing Diversity and Complexity of Web Application Development*. [S.l.]: Springer, 2001. cap. 1, p. 3–13. Citado na página 10.

PASTOR, O. et al. Model-driven development. *Informatik-Spektrum*, v. 31, p. 394–407, 2008. Citado 2 vezes nas páginas 10 e 21.

PRADO, R. C. do. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework VRaptor 4*. [S.l.], 2015. Citado 6 vezes nas páginas 7, 36, 37, 38, 39 e 40.

PRESSMAN, R. S. *Engenharia de Software - Uma Abordagem Profissional*. 7ª edição. ed. [S.l.]: McGraw-Hill, 2011. 780 p. Citado na página 10.

SELIC, B. The pragmatics of model-driven development. *IEEE software*, IEEE, v. 20, n. 5, p. 19–25, 2003. Citado na página 15.

SOUZA, V. E. S. *FrameWeb: um Método baseado em Frameworks para o Projeto de Sistemas de Informação Web*. Dissertação (Mestrado) — Programa de Pós-Graduação em Informática, Universidade Federal do Espírito Santo, 2007. Citado 11 vezes nas páginas 6, 7, 10, 11, 16, 17, 18, 19, 20, 21 e 26.

SOUZA, V. E. S.; FALBO, R. A.; GUIZZARDI, G. Designing Web Information Systems for a Framework-based Construction. In: HALPIN, T.; PROPER, E.; KROGSTIE, J. (Ed.). *Innovations in Information Systems Modeling: Methods and Best Practices*. 1. ed. [S.l.]: IGI Global, 2009. cap. 11, p. 203–237. Citado 4 vezes nas páginas 6, 10, 11 e 16.

VIYOVIC, V.; MAKSIMOVIC, M.; PERISIC, B. Sirius: A rapid development of DSM graphical editor. In: IEEE. *Intelligent Engineering Systems (INES), 2014 18th International Conference on*. [S.l.], 2014. p. 233–238. Citado 2 vezes nas páginas 25 e 33.