UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO TECNOLÓGICO

COLEGIADO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO

Kaio Silva Rosa

Integração do Suporte aos *Frameworks* de Segurança às Ferramentas FrameWeb.

Vitória, ES

2021

Kaio Silva Rosa

Integração do Suporte aos *Frameworks* de Segurança às Ferramentas FrameWeb.

Monografia apresentada ao Curso de Engenharia de Computação do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Engenharia de Computação.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Colegiado do Curso de Engenharia de Computação

Orientador: Prof. Dr. Vítor E. Silva Souza

Vitória, ES

2021

Kaio Silva Rosa

Integração do Suporte aos *Frameworks* de Segurança às Ferramentas FrameWeb./ Kaio Silva Rosa. – Vitória, ES, 2021-

30 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Vítor E. Silva Souza

Monografia (PG) – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Colegiado do Curso de Engenharia de Computação, 2021.

1. Palavra-chave1. 2. Palavra-chave2. I. Souza, Vítor Estêvão Silva. II. Universidade Federal do Espírito Santo. IV. Integração do Suporte aos *Frameworks* de Segurança às Ferramentas FrameWeb.

CDU 02:141:005.7

Kaio Silva Rosa

Integração do Suporte aos *Frameworks* de Segurança às Ferramentas FrameWeb.

Monografia apresentada ao Curso de Engenharia de Computação do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Engenharia de Computação.

Trabalho aprovado. Vitória, ES, (dia) de (mês) de (ano):

Prof. Dr. Vítor E. Silva Souza
Orientador

Prof. Dr. Vinícius Fernandes Soares Mota
Universidade Federal do Espírito Santo

Cleisson Santos Guterres
Universidade Federal do Espírito Santo

Vitória, ES
2021

Agradecimentos

Agradeço a minha família, em especial meus pais e minhas irmãs, por todo o suporte e carinho dados nessa minha jornada. Agradeço também a minha namorada por toda a compreensão e apoio.

Ao meu orientador, professor Vítor Estêvão Silva Souza, pela orientação e dedicação e por ser um espelho de profissional, e que se mostrou muito aberto e aos meus professores da contribuiriam bastante para minha formação.

E por fim agradeço, meus amigos pela companhia e pela trocas de experiências que foram essenciais para que eu chegasse até aqui.

Resumo

A popularização da Internet nos proporciona maior acesso a informação, porém também abre uma demanda por softwares bem escritos e seguros, já que se tornou um dos principais meios de comunicação.

A Engenharia Web nasce trazendo propostas e abordagens sistemáticas para o desenvolvimento Web. Também houve necessidade de evolução de ferramentas com o propósito de facilitar o desenvolvimento Web como *frameworks* que visam resolver problemas recorrentes de forma mais genérica, ajudando o desenvolvedor a focar em resolver os aspectos mais específicos da sua solução.

Neste contexto, o método FrameWeb foi proposto com o objetivo de trazer os conceitos destes *frameworks* para a Engenharia Web. Sua arquitetura é baseada nos tipos de *frameworks* mais populares e visa uma melhor comunicação no desenvolvimento e a documentação do projeto. FrameWeb segue uma abordagem orientada a modelos e também provê um editor gráfico, para auxiliar desenvolvedores no uso da linguagem de modelagem do FrameWeb, e um gerador de código para facilitar no desenvolvimento. As categorias de *frameworks* suportadas atualmente são: Controlador Frontal, Injeção de Dependência e Mapeamento Objeto/Relacional e Segurança.

Esse trabalho se propõe a trazer para as ferramentas FrameWeb atuais o suporte aos *frameworks* de segurança (autenticação e autorização) para o editor gráfico e gerador de código. Existem versões do editor gráfico e do gerador de código nas quais o suporte foi implementado, no contexto de um trabalho de mestrado, porém estas não estão integradas com as versões mais atuais das ferramentas FrameWeb.

Palavras-chaves: Engenharia Web, Frameworks, FrameWeb, Autenticação, Autorização, Role-based Access Control, Editor Gráfico, Geração de Código.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UML	Unified Modeling Language
RBAC	Role Based Access Control
WIS	Web-based Information System
JSF	Java Server Faces
CDI	Context and Dependency Injection
DAO	Data Access Object
JAAS	Java Authentication and Authorization Services
AQL	Acceleon Query Language
XML	Extensible Markup Language

Sumário

1 Introdução

Nos últimos anos, tem crescido a popularidade do desenvolvimento de softwares Web e, com isso, também uma demanda para uma abordagem mais metódica para o desenvolvimento dos mesmos. A Engenharia Web (PRESSMAN, 2014) nasce trazendo propostas e abordagens sistemáticas para o desenvolvimento Web. Junto com a Engenharia Web, também houve necessidade de evolução de ferramentas com o proposito de facilitar o desenvolvimento Web, como *frameworks* que visam resolver problemas recorrentes de forma mais genérica, permitindo ao desenvolvedor focar em resolver os problemas mais específicos da sua solução.

Neste contexto, o método FrameWeb (SOUZA, 2020) foi proposto com o objetivo de trazer os conceitos dos *frameworks* para a Engenharia Web. Sua arquitetura é baseada nos tipos de *frameworks* mais populares — a saber, Controlador Frontal, Injeção de Dependência, Mapeamento Objeto/Relacional e Segurança — e visa uma melhor comunicação no desenvolvimento e a documentação do projeto.

FrameWeb segue uma abordagem orientada a modelos e também provê um editor gráfico — FrameWeb Editor (CAMPOS; SOUZA, 2017) — para auxiliar desenvolvedores no uso da linguagem de modelagem do FrameWeb, e um gerador de código para facilitar no desenvolvimento (ALMEIDA; CAMPOS; SOUZA, 2017). As ferramentas foram desenvolvidas como *plugins* da IDE Eclipse (SILVA, 2019).

1.1 Motivação e Justificativa

A versão atual das ferramentas de apoio ao FrameWeb dão suporte aos *frameworks* dos tipos Controlador Frontal, Injeção de Dependência e Mapeamento Objeto/Relacional, tanto na parte do editor gráfico quanto na geração de código.

O suporte aos *frameworks* de segurança, mais especifico aos *frameworks* de autenticação e autorização, foi adicionado ao FrameWeb mais recentemente no contexto de um trabalho de mestrado (PRADO; SOUZA, 2018; PRADO, 2020). Por este motivo, atualmente as ferramentas FrameWeb não dão suporte para os *frameworks* de segurança, apesar de já previsto no seu modelo. Existem versões do editor gráfico[1] e do gerador de código[2] nas quais o suporte foi implementado, porém estas não estão integradas com as versões mais atuais das ferramentas FrameWeb.[3] Logo, exite uma necessidade dessa integração do editor gráfico e da ferramenta de geração de código à base de código principal

[1] <https://github.com/Rodolfocostapr/FrameWeb>.
[2] <https://github.com/Rodolfocostapr/CodeGenRodolfo>.
[3] <https://github.com/nemo-ufes/FrameWeb>.

das ferramentas.

Esse trabalho de projeto de graduação traz o código existente para a versão mais recente da ferramenta, além de fazer testes para validação da integração.

1.2 Objetivos

O objetivo geral desse projeto de graduação é fazer a integração do suporte aos *frameworks* de segurança às ferramentas FrameWeb (editor gráfico e gerador de código).

Os objetivos específicos são: (i) fazer com que seja possível representar funcionalidades de autenticação e autorização nos modelos no editor gráfico; e (ii) fazer com que seja possível a geração de código referente a tais funcionalidades de segurança providas por estes *frameworks*.

1.3 Método de Desenvolvimento do Trabalho

O projeto de graduação foi iniciado com leituras de artigos sobre o método FrameWeb, *frameworks* de segurança e assuntos relacionados. Também foram objetos de estudo as técnicas e ferramentas de Desenvolvimento Orientado a Modelos (PASTOR et al., 2008), para que pudessem ser utilizadas para realizar a integração do código existente. Foram utilizados tutoriais das ferramentas utilizadas no desenvolvimento do editor e da ferramenta de geração de código. Por fim, foi feita a integração do código das ferramentas e testes para a validação do mesmo.

1.4 Organização da Monografia

Essa monografia está organizada em cinco capítulos, incluindo essa introdução.

- O Capítulo 2 apresenta o que foi estudado e ferramentas utilizadas para o desenvolvimento da integração;
- O Capítulo 3 apresenta a principal contribuição do trabalho, a integração feita e suas mudanças no *plugin* FrameWeb;
- O Capítulo 4 apresenta a avaliação da integração do suporte aos *frameworks* de segurança no *plugin*, comparando o que é gerado pelo editor;
- O Capítulo 5 apresenta as considerações finais do trabalho.

2 Referencial Teórico e Tecnologias Utilizadas

Neste capítulo serão apresentadas as tecnologias e conceitos que auxiliarão no processo de integração do suporte aos *frameworks* de segurança às ferramentas FrameWeb.

2.1 Engenharia Web

Estamos em um momento onde é difícil não ver os impactos da Internet. Com essa popularidade, cresce a demanda de softwares desenvolvidos para a plataforma web, de bancos online a redes sociais. A grande vantagem do desenvolvimento para essa plataforma é a disponibilidade, já que tendo um navegador se torna possível acessar esses softwares. Antes da formalização da Engenharia Web, temos softwares desenvolvidos para Web que funcionam mas com difícil manutenibilidade. A Engenharia Web traz conceitos essenciais para manter uma qualidade alta com foco em boas práticas e feita de maneira sistemática. Esses conceitos, de acordo com Olsina, Lafuente e Rossi (2001), são:

- Usabilidade: o software deve ser de fácil uso a todos os tipos seus usuários, independente do grau de conhecimento do mesmo;
- Funcionalidade: o software deve se comportar bem, atendendo todas as demandas para as quais foi construído, buscando todas as informações corretamente e as operações funcionando;
- Eficiência: o software deve ter um tempo de resposta para as suas funcionalidades bem curto de acordo com o que é esperado;
- Confiabilidade: o software deve garantir a validação dos dados de entrada dos usuários e dar um *feedback* de acordo com o esperado;
- Manutenibilidade: o software deve possibilitar a manutenção, a expansão e o aumento de funcionalidades, sem grandes problemas.

A metodologia da Engenharia Web tem o objetivo de fazer com que esses atributos levem a um software com alta qualidade, já que esses atributos são utilizados de forma análoga na Engenharia de Software.

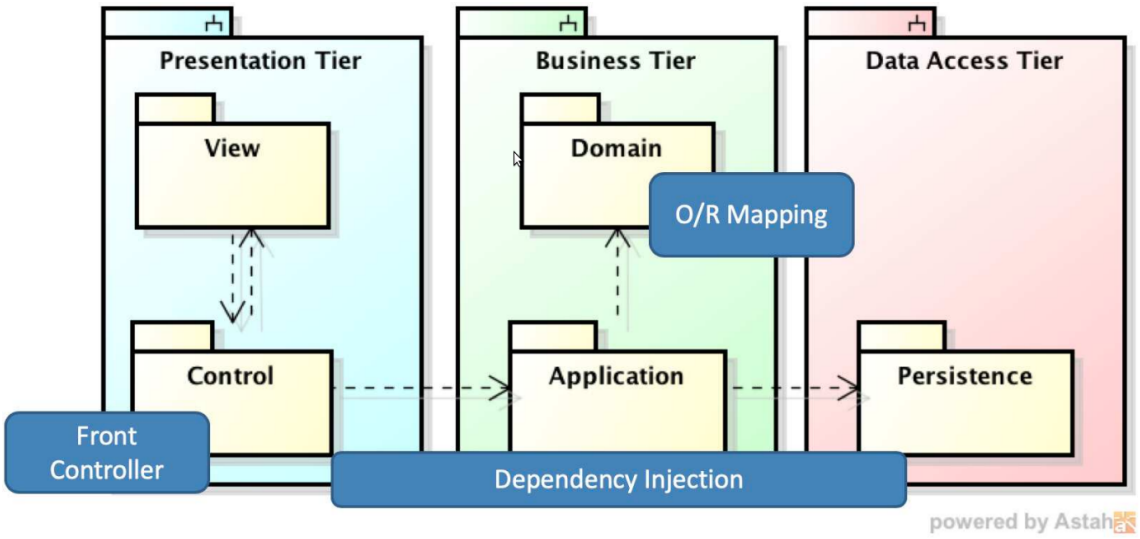

Figura 1 – Arquitetura padrão suportada pelo FrameWeb (SOUZA, 2020).

2.2 FrameWeb

O FrameWeb (*Framework-based Design Method for Web Engineering*) (SOUZA, 2020) é um método baseado em *frameworks* para o desenvolvimento de sistemas de informação Web (*Web Information System* – WIS). Nas propostas da Engenharia Web não havia um método que considerasse os aspectos do uso de *frameworks* na construção desses sistemas. O FrameWeb tem o intuito de dar uma melhor direção durante a fase de implementação, já que o uso de *frameworks* se tornou bastante popular.

Na arquitetura básica do FrameWeb, ele é dividido em três camadas, Apresentação, Negócio e Acesso a dados que são subdivididas como pode ser observado na a Figura 1.

Na camada de Apresentação, o pacote de Visão contém as páginas Web, arquivos de estilo (CSS), scripts na parte do cliente e artefatos ligados ao *front-end* da aplicação. No pacote Controle estão as classes controladoras, que lidam com as requisições feitas no pacote de visão e controlam os fluxos de dados da aplicação que usa a infraestrutura do *framework* Controlador Frontal e usa os serviços oferecidos pelo pacote de Aplicação.

Na camada de Negócio, o pacote de Aplicação contém a implementação dos casos de uso identificados na fase de análise de requisitos, e é uma camada de serviços independente da interface com o usuário, cujas dependências são conectadas pela *framework* de Injeção de Dependência.Já o pacote de Domínio contém as classes que representam o domínio do problema da aplicação alem das anotações que orientam a *framework* de Mapeamento Objeto/Relacional na persistência de dados.

Na camada de Acesso a Dados, o pacote Persistência contém as classes DAO (*Data Access Object*) que são responsáveis por persistir no banco de dados as instâncias

do pacote de Domínio por meio do *framework* de Mapeamento Objeto/Relacional para armazenar objetos do banco de dados relacional. Esse ultimo pacote é opcional e suas responsabilidades podem ser mescladas com o pacote de Aplicação, mas a centralização das operações de acesso a dados em classes DAO ajuda na manutenção do código.

O método FrameWeb apresenta um linguagem de modelagem para Web baseada em UML que tem a intenção de guiar a implementação das camadas descritas acima, através de componentes típicos dos *framework* utilizados. Os modelos propostos são:

- Modelo de Entidades: representa as classes do pacote de Domínio e seus metadados, estes serão usados para o mapeamento objeto/relacional;
- Modelo de Persistência: representa os DAOs que são responsáveis pela persistência dos dados gerados pelo sistema;
- Modelo de Navegação: representa os componentes que formam a camada de apresentação, são as páginas Web do sistema e seus atributos, que interagem com a camada de controle, pacotes de Visão e Controle;
- Modelo de Aplicação: representa as classes de serviço, que são responsáveis por implementar a lógica de negócio do sistema. Também demonstra quais classes de Controle possuem dependências para com classes de Aplicação e quais as dependências destas com as classes da Persistência.

O método FrameWeb possui ainda um editor gráfico associado, chamado *FrameWeb Editor* (CAMPOS; SOUZA, 2017). Tal editor foi implementado usando o Eclipse Sirius e foi proposto para criação de modelos baseados no método FrameWeb usando uma sintaxe definida formalmente por meio de metamodelos. Nele é possível criar os quatros modelos propostos pelo método, além de pacotes de definição de linguagem e de *frameworks*, que permitem ajustar os modelos genéricos do FrameWeb aos *frameworks* e plataforma de desenvolvimento específicos sendo utilizados.

Integrado ao editor gráfico, o *FrameWeb Code Generator* (ALMEIDA; CAMPOS; SOUZA, 2017) gera código a partir dos modelos criados no *FrameWeb Editor*. Os dados identificados pelo editor são transformados automaticamente em código fonte para o desenvolvedor. O *FrameWeb Code Generator* é extensível para diversas linguagens já que sua geração de código é feita através de *templates*. As ferramentas foram desenvolvidas como *plugins* da IDE Eclipse (SILVA, 2019), e encontram-se sob a licença MIT. O leitor interessado pode obter instruções com mais detalhes sobre a instalação e uso na página *Wiki* do repositório do projeto, disponível em: <https://github.com/nemo-ufes/FrameWeb/wiki/ToolsTutorial01>.

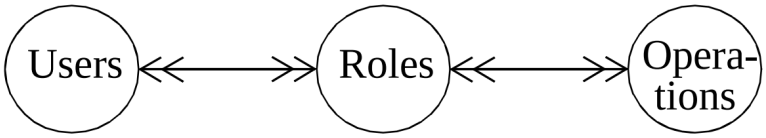

Figura 2 – Principais conceitos do RBAC e suas relações (FERRAIOLO; CUGINI; ‘KUHN, 1995)

2.3 Role Based Access Control

O processo de autenticação consiste em verificar as credenciais do usuário, ou seja, se ele é quem diz que é. Já o processo de autorização consiste em verificar se o usuário tem permissão de acesso a um determinado recurso. Existem vários métodos de verificação desses processos que, no geral, são feitos com auxílio de *frameworks* de segurança. Tais *frameworks* são abrangentes e tendem a seguir uma infraestrutura relacionada à segurança de uma aplicação. O método de controle de acesso que o *FrameWeb* abrange é o *RBAC*, ou *Role-Based Access Control*. O RBAC se baseia em papéis, o que proporciona uma grande flexibilidade nas políticas criadas, já que as permissões estão associadas a esses papéis.

Segundo Ferraiolo, Cugini e ‘Kuhn (1995), a noção central do *RBAC* é das operações que representam ações associadas a papéis (*Roles*) e usuários (*Users*) que são apropriadamente feitos membros dos papéis. Os papéis podem ser criados para vários cargos em uma organização. Os usuários tendem a receber papéis acordo com suas competências e habilidades e as principais ações administrativas são a concessão e a revogação desses papéis.

2.4 Model Driven Development

Desenvolvimento Orientado a Modelos (PASTOR et al., 2008) é uma metodologia que ajuda na compreensão de sistemas complexos, já que foca na criação de modelos que ajudam a visualizar o sistema como um todo. Segundo Martins (2016), o uso das técnicas MDD para projeto e desenvolvimento de sistemas deve levar em consideração aspectos não só relacionados à fase de projeto, desenvolvimento e manutenção, mas a todas as fases de um projeto de desenvolvimento de software.

Usando técnicas de MDD, foi formalizada uma linguagem Específica de Domínio (*Domain Specific Language* ou DSL) para o FrameWeb, por meio de metamodelos que estendem um parte do metamodelo do UML (*Partial UML Meta-model*). Esses metamodelos são subdivididos em cinco componentes que são os modelos propostos pelo FrameWeb e o componente *FrameWork Meta-model*, como pode ser observado na Figura 3, que permite a especificação de regras e modelar construtos específicos as *frameworks* que serão usadas pela aplicação.

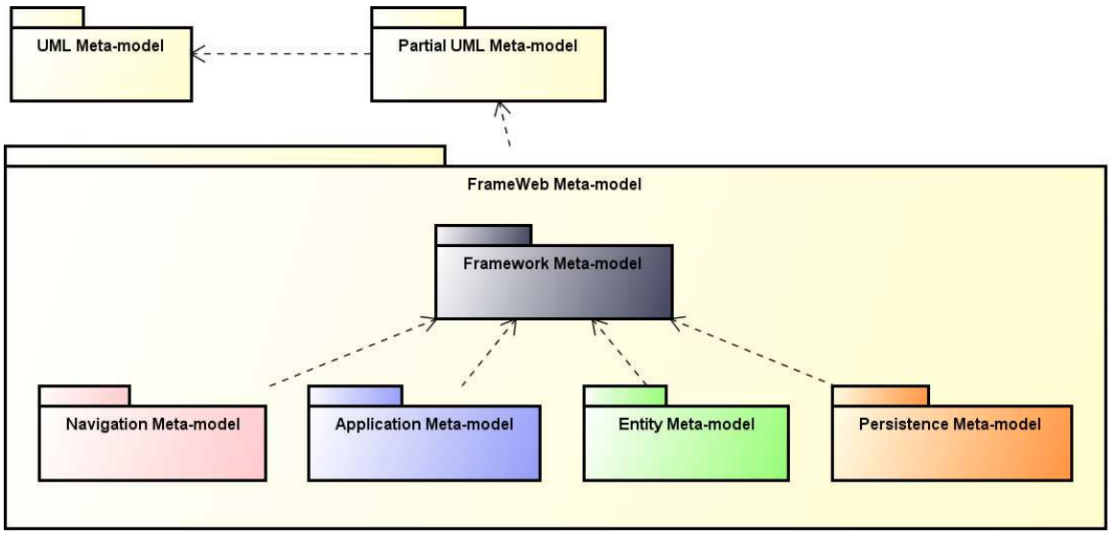

Figura 3 – Metamodelos que definem a linguagem FrameWeb (SOUZA, 2020).

Definida a linguagem do FrameWeb, é possível então validar os metamodelos e, com base neles, foram criadas as ferramentas de apoio ao FrameWeb. A linguagem do FrameWeb foi estendida para especificar, em seus modelos, a configuração das propriedades de autenticação e autorização dos *frameworks* usando os conceitos de RBAC (PRADO, 2020). Novos construtos de linguagem foram adicionados e as ferramentas de edição e geração de código também foram estendidas para dar suporte a esses construtos.

2.5 Ferramentas utilizadas

A seguir, listamos as ferramentas utilizadas neste projeto:

- **Sirius**: é um *plug-in* do Eclipse dedicado à criação de ambientes gráficos de trabalho personalizados para modelagem. Como o editor atual usa esse *plug-in* para criação dos modelos propostos pelo FrameWeb, ele foi utilizado para atualizações necessárias;
- **EMF (*Eclipse Modeling Framework*)**: parte do Eclipse, é um *framework* de modelagem para aplicações baseadas em modelos de dados estruturados. Foi utilizado neste trabalho para colocar em prática os estudos de desenvolvimento orientado a modelos para a atualização/ajustes do metamodelo necessário;
- **JTwig**: é um sistema que permite criar *templates*. Esse sistema combina dados estáticos com dados dinâmicos para produzir conteúdo. O *template* é uma representação intermediária do conteúdo e específica como a saída será gerada. Neste trabalho foi utilizado na atualização do gerador de código, que na versão desenvolvida por Prado (2020) está escrita em CSharp.

3 Contribuição do Trabalho

Esse capitulo tem como objetivo descrever as ferramentas FrameWeb e descrever as mudanças feitas no FrameWeb Editor e FrameWeb Code Generator para que o suporte aos *frameworks* de segurança fosse contemplado, mais especificamente o *Role-Based Access Control* (RBAC).

O *FrameWeb Editor* (CAMPOS; SOUZA, 2017) oferece suporte à criação dos quatro tipos básicos de modelos FrameWeb: Entidade, Persistência, Aplicação e Navegação. É possível acessar representações e painéis de criação de componentes, diferentes para cada tipo de modelo citado. Sua base foi desenvolvida utilizando o Sirius por meio da definição de dois modelos relacionados ao FrameWeb. O modelo da sintaxe abstrata, ou meta-modelo, define os elementos que poderão ser criados no editor gráfico, assim como suas propriedades e é especificado por meio do Eclipse Modeling Framework (EMF), um *framework* de modelagem e geração de código para construção de ferramentas. E o modelo da sintaxe concreta, que é o *viewpoint specification* do Sirius, define a características gráficas dos elementos definidos no meta-modelo. Os arquivos criados por meio da ferramenta são arquivos XML salvos no formato .frameweb e processados pelo gerador de código.

O *FrameWeb Code Generator* (ALMEIDA; CAMPOS; SOUZA, 2017) foi criado para realizar a transformação do modelo em código. Este processo é responsável pela criação de arquivos (artefatos de código) para um *WIS* modelado no *FrameWeb Editor*. Nesse processo, o gerador de código carrega os modelos produzido pelo editor que são mapeados com os *templates JTwig* para a geração dos arquivos de código-fonte de saída para o sistema. As ferramentas utilizam arquivos .frameweb para carregar as definições de linguagem e dos *frameworks* utilizados.

Para a integração do suporte aos *frameworks* de segurança (SOUZA, 2020), usando RBAC, foram feitas adições aos modelos de Entidades, de Navegação e de Aplicação.

Os conceitos de RBAC, que precisam ser desenvolvidos no momento do design são: **User**, **Role** e **Permission** com associação de N para N entre eles e são representados no modelo de entidades como **AuthUser**, **AuthRole** e **AuthPermission**, e o seu comportamento é definido no modelo de navegação. A parte da autorização é feita pelas classes de serviço no modelo de aplicação.

A Figura 4 apresenta fragmentos dos três metamodelos, focado nas mudanças feitas para dar suporte a autenticação e autorização (SOUZA, 2020). As meta-classes já existentes no metamodelo do FrameWeb são as apresentadas em verde, rosa e azul para os modelos de Entidades, Navegação e Aplicação.

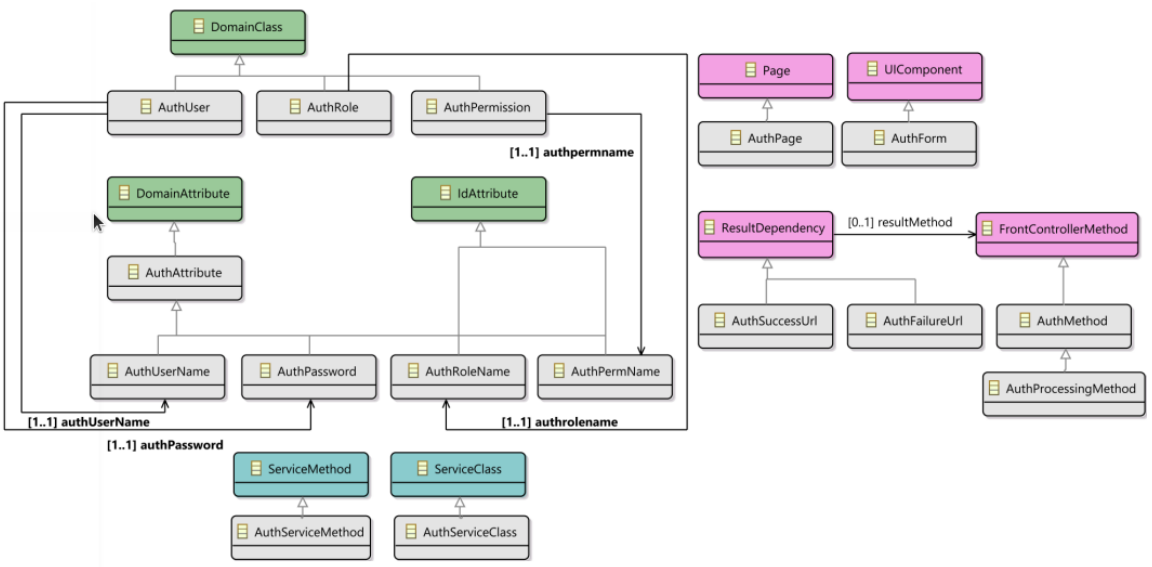

Figura 4 – Fragmentos dos meta-modelos de Entidades, Navegação e Aplicação, modificados para dar suporte aos *frameworks* de segurança (PRADO, 2020).

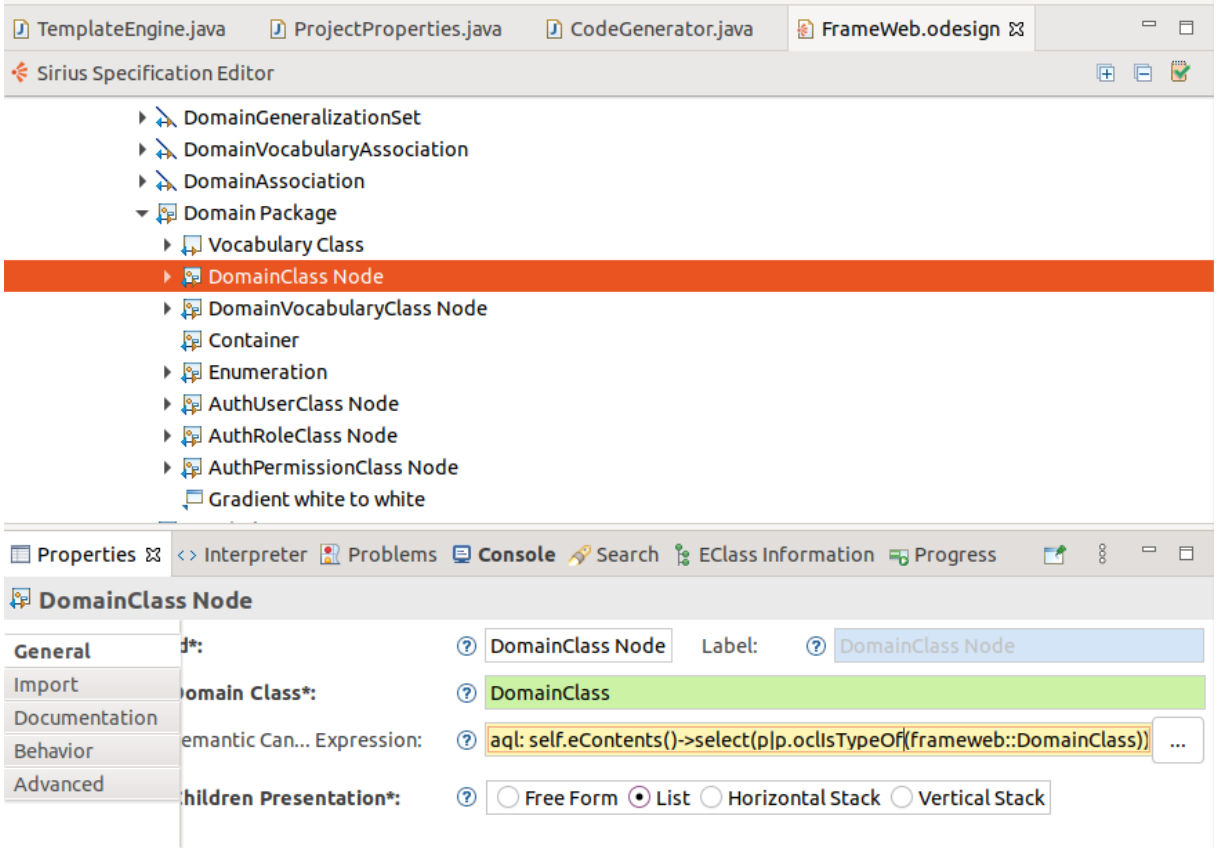

Figura 5 – Definição dos nós adicionados no editor gráfico pela visão do *viewpoint specification* do Sirius.

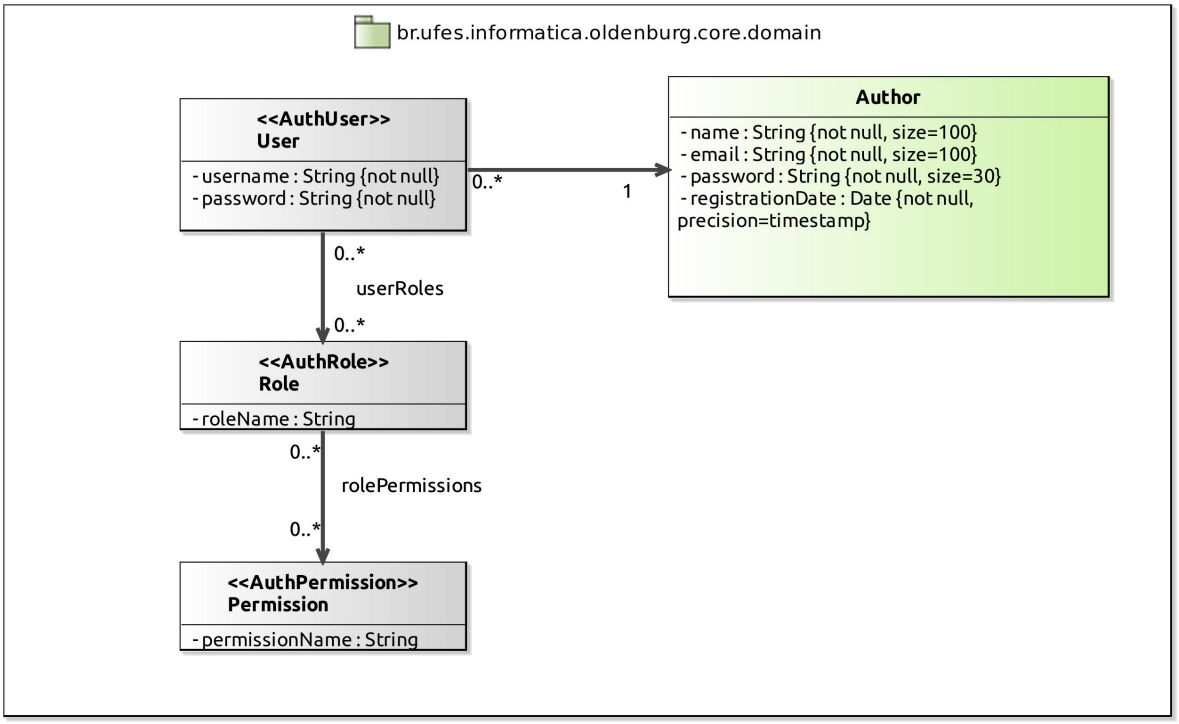

Figura 6 – Modelo de entidades com os construtos de autenticação e autorização.

Em uma versão antiga do FrameWeb Editor, foram feitas as extensões para incluir os conceitos de RBAC, que precisaram de ser atualizados para a nova versão já que o metamodelo sofreu modificações desde daquela versão, e uma versão do gerador de código existente era em CSharp (PRADO, 2020), e a nova versão do gerador de código usa o Eclipse E4 e *templates jTwig*, logo se fez necessário uma adequação para essas novas tecnologias.

No editor gráfico, modificações nos metamodelos relativos a estes três diagramas foram feitas através de criação de nós no *Sirius* como mostrado na Figura 5, que servirão para instanciar visualmente os componentes adicionados. No modelo de entidade temos os **AuthUser**, que representa o usuário, **AuthRole**, que representa um conjunto de atividades funcionais, e **AuthPermission**, que representa um conjunto de permissões associadas a *Role*. A Figura 6 apresenta um exemplo de modelo de entidades usando as novas meta-classes.

No modelo de navegação temos **AuthPage**, que representa a pagina de login, **AuthForm**, que representa o formulário usado para receber as credencias do usuário e o **AuthMethod**, que representa o método que é chamado no controlador para verificação dessas credenciais. Esses elementos são responsáveis pela parte visão e pelo login, assim como as URLs de processamento **AuthSuccessUrl**, que representa o redirecionamento em caso de sucesso e o **AuthFailureUrl**, que representa o redirecionamento no caso da tentativa falha de login. A Figura 7 apresenta um exemplo de modelo de navegação usando

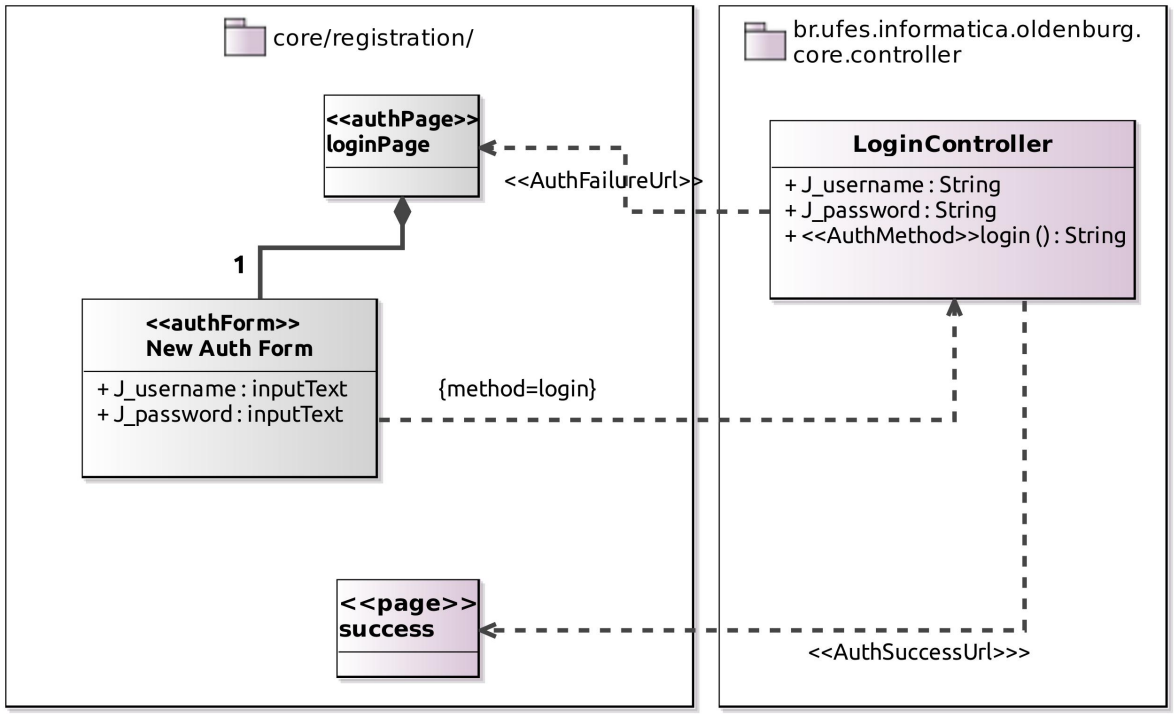

Figura 7 – Modelo de navegação com os construtos de autenticação e autorização

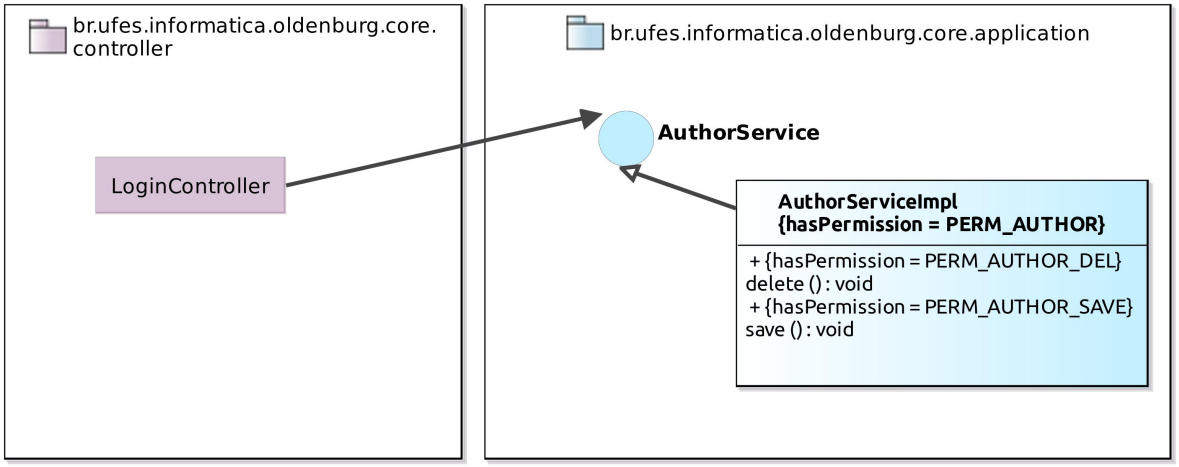

Figura 8 – Modelo de aplicação com os construtos de autenticação e autorização

as novas meta-classes.

Já no modelo de Aplicação temos **AuthService**, que representa a classe de serviço que contêm a parte de autorização que é feita através de *Permissions* usando o atributo **PermissionName** para nome da permissão requerida para o seu acesso. A Figura 8 apresenta um exemplo de modelo de aplicação usando as novas meta-classes. O AuthorServiceImpl é um **AuthService** que requer a uma permissão **PERM_AUTHOR** para ser acessada, e os métodos **delete()** e **save()** requerem **PERM_AUTHOR_DEL** e **PERM_AUTHOR_SAVE** respectivamente.

No gerador de código, é feita a busca das representações dos digramas de cada modelo FrameWeb é a extraída seus dados para que seja feito o mapeamento dessa representação com os templates *JTwig*, logo foram feitas modificações no código para adequar a busca pelos novos construtos e foram criadas *templates JTwig* que os mapeiam e geram o seu código de acordo com o *framework* de segurança utilizado. A Listagem 3.1 mostra um trecho do arquivo ClassCodeGenerator.java com as modificações. Os *templates* criados foram para o JAAS, um *framework* de segurança muito utilizado tanto para autenticação quanto para autorização, o intuito das modificações foi facilitar para os desenvolvedores a implementar RBAC. A Listagem 3.2 mostra um trecho do arquivo Auth com as modificações. Os mesmos princípios utilizados podem ser usados para criar *templates* para outros *frameworks*, como Apache Shiro, Spring Security entre outros. A Listagem 3.3 mostra um exemplo de código gerado pelo gerador de código.

Listagem 3.1 – Exemplo de do modificação feita no gerador de código

```
1  public static String render(AuthUser class_, String template) {
2    TemplateEngine templateEngineContext = new JtwigTemplateEngineImpl();
3    templateEngineContext.setTemplate(template);
4
5    templateEngineContext
6      .addParameter(PACKAGE, class_.getPackage())
7      .addParameter(CLASS, class_)
8      .addParameter(ATTRIBUTES, class_.getAttributes())
9      .addParameter(ASSOCIATIONS, class_.getAssociations()
10         .stream()
11         .filter(DomainAssociation.class::isInstance)
12         .map(DomainAssociation.class::cast)
13         .collect(Collectors.toList()))
14     .addParameter(METHODS, class_.getOperations()
15         .stream()
16         .filter(DomainMethod.class::isInstance)
17         .map(DomainMethod.class::cast)
18         .collect(Collectors.toList()))
19     .addParameter(GENERALIZATIONS, class_.getGeneralizations());
20
21   return templateEngineContext.getCode();
22 }
```

A implementação descrita neste trabalho está disponível no repositório público <https://github.com/nemo-ufes/FrameWeb>.

Listagem 3.2 – Exemplo de template criado para AuthServiceClass

```
1  /** TODO: generated by FrameWeb. Should be documented. */
2  @Stateless
3  public class {{ class.Name }} {% if generalizations is not empty %}extends {{ ((
       generalizations[0]).GeneralizationSets[0]).Name }} {% endif %}implements {{
       class.Name | replace({'Bean': '', 'Impl': ''}) }} {
4    /** Serialization id (using default value, change if necessary). */
5    private static final long serialVersionUID = 1L;
6    @Inject
7    private Event<LoginEvent> loginEvent;
8    @Resource
9    private SessionContext sessionContext;
10
11   {% for association in associations %}
12   /** TODO: generated by FrameWeb. Should be documented. */
13   @EJB
14   private {{ association.TargetMember.Type.Name }} {{ association.TargetMember.
         Type.Name | lower_first }};
15   {% endfor %}
16
17   {% for attribute in attributes %}
18   /** TODO: generated by FrameWeb. Should be documented. */
19   private {{ attribute.Type.Name }} {{ attribute.Name }};
20   {% endfor %}
21
22   {% for attribute in attributes %}
23   /** Getter for {{ attribute.Name }}. */
24   public {{ attribute.Type.Name }} get{{ attribute.Name | title }}() {
25     return {{ attribute.Name }};
26   }
27
28   /** Setter for {{ attribute.Name }}. */
29   public void set{{ attribute.Name | title }}({{ attribute.Type.Name }} {{
         attribute.Name }}) {
30     this.{{ attribute.Name }} = {{ attribute.Name }};
31   }
32   {% endfor %}
33
34   {% for method in methods %}
35   /** TODO: generated by FrameWeb. Should be documented. */
36   @Override
37   @RolesAllowed("{{class.PermissionName}}")
38   {{ method.Visibility.Name }} {% if method.MethodType is null %}void{% else %}{{
         method.MethodType.Name }}{% endif %} {{ method.Name }}({% for parameter in
        method.OwnedParameters %}{{ parameter.Type.Name }} {{ parameter.Name }}{% if
         loop.last == false %}, {% endif %}{% endfor %}) {
39     // FIXME: auto-generated method stub
40     return{% if method.MethodType is not null %} null{% endif %};
41   }
42   {% endfor %}
43
44 }
```

Listagem 3.3 – Exemplo de código gerado pelos templates para AuthRole

```
1
2 @Entity
3 public class Role extends Principal, Serializable, PersistentObjectSupport
    implements Comparable<Role> {
4   /** Serialization id. */
5   private static final long serialVersionUID = 1L;
6
7   /** TODO: generated by FrameWeb. Should be documented. false */
8   @NotNull
9   private String roleName;
10
11  /** TODO: generated by FrameWeb. Should be documented. */
12  @ManyToMany
13  private Set<Permission> Target;
14
15  /** TODO: generated by FrameWeb. Should be documented. */
16  @ManyToMany(mappedBy="Target")
17  private Set<User> Source;
18
19  /** Getter for roleName. */
20  public String getRoleName() {
21    return roleName;
22  }
23
24  /** Setter for roleName. */
25  public void setRoleName(String roleName) {
26    this.roleName = roleName;
27  }
28
29
30  /** Getter for Target. */
31  public Set<Permission> getTarget() {
32    return Target;
33  }
34
35  /** Setter for Target. */
36  public void setTarget(Set<Permission> Target) {
37    this.Target = Target;
38  }
39
40
41  /** Getter for Source. */
42  public Set<User> getSource() {
43    return Source;
44  }
45
46  /** Setter for Source. */
47  public void setSource(Set<User> Source) {
48    this.Source = Source;
49  }
50
51  /** @see java.lang.Comparable#compareTo(java.lang.Object) */
52  @Override
53  public int compareTo(Role o) {
54    // FIXME: auto-generated method stub
55    return super.compareTo(o);
56  }
57
58  public boolean equals(Object object) {
59    boolean flag = false;
60    if (object instanceof (Role)
61      flag = name.equals((((Role) object).getName());
62    return flag;
63  }
64 }
```

4 Avaliação da Proposta

Nesse capítulo está descrito resultados de avaliação da proposta da integração do suporte para os *frameworks* de segurança ao FrameWeb. Para avaliar o resultado dessa integração, foi utilizado um projeto desenvolvido por alunos de pós-graduação da UFES, na disciplina de Desenvolvimento Web e Web Semântica (DWWS) chamado MusicPlus, disponibilizada em um repositório GitHub.[1] Como o intuito das ferramentas FrameWeb é o auxilio aos desenvolvedores de software Web, foi feita uma analise tanto quantitativa quanto qualitativa.

O projeto MusicPlus consiste em um exemplo de aplicação desenvolvida em Java EE 7, com os *frameworks* JSF versão 2.2.12, PrimeFaces versão 6.2, CDI versão 1.1 e JPA versão 2.1, além do WildFly versão 19.0.0 como servidor de aplicações.

A aplicação MusicPlus permite fazer CRUD de músicas e possui um algoritmo simples de recomendação de música. Essa aplicação foi selecionada para ser utilizada como prova de conceito para a integração das ferramentas por ser uma aplicação simples e funcional e utiliza o JAAS (Java Authentication and Authorization Services) como *framework* de segurança.

Para esse projeto foi utilizado o FrameWeb Editor para criar seu modelo de Entidade, Navegação e Aplicação, que foram os modelos que sofreram alteração durante a integração.

4.1 Modelo de Entidades

Na Figura 9, temos o diagrama do Modelo de Entidades do projeto Music Plus. Podemos ver em cinza os **AuthClasses**, que representam os novos construtos adicionados. As classes de verde serão geradas pelos *template* **EntityClassTemplate.twig** onde serão mapeadas suas classes e seus atributos, já o **AuthUser**, **AuthRole** e **AuthPermission** cada um possui um *template* diferente que mapeiam suas informações, pois dependem do *framework* de segurança que está sendo utilizado.

4.2 Modelo de Navegação

Na Figura 10, temos o diagrama do Modelo de Navegação do projeto Music Plus. Podemos observar que o controlador SessionController, que é onde será processado o login, possui um método do tipo **AuthProcessingMethod** representado com estereótipo «**AuthMethod**». Este funciona de forma análoga ao **FrontControllerMethod**, mas no

[1] <https://github.com/dwws-ufes/2020-MusicPlus>

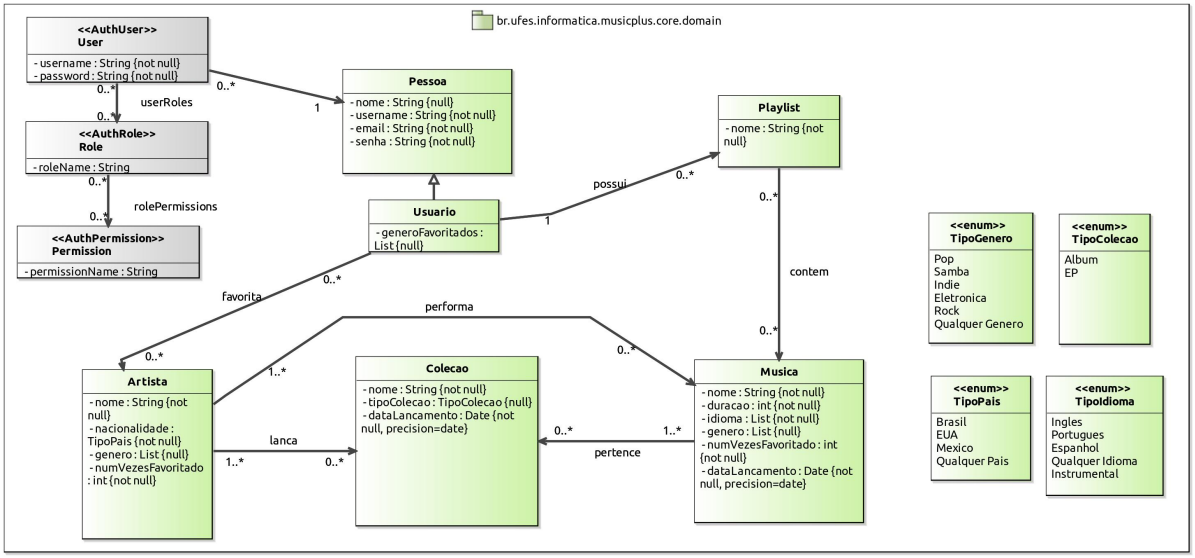

Figura 9 – Modelo de entidades da projeto Music Plus

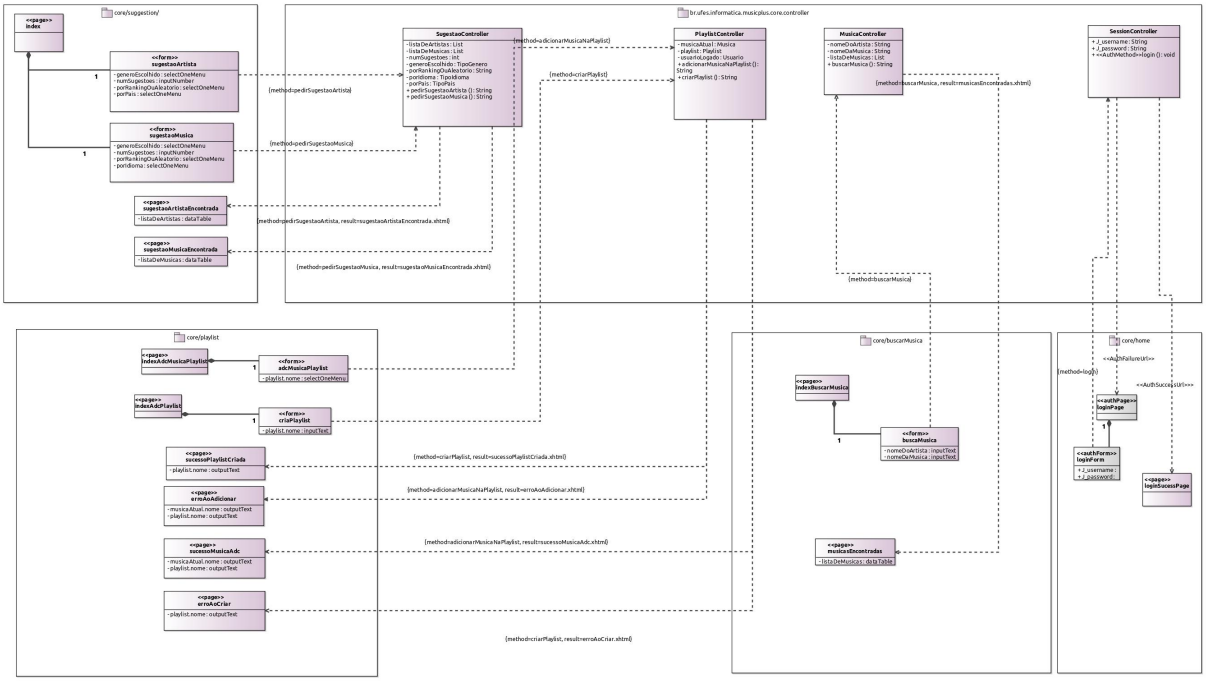

Figura 10 – Modelo de navegação da projeto Music Plus

seu *template* gera um código do login programático. O código é gerado pelo novo *template* AuthPage.twig e algumas modificações no FrontControllerClass.twig já existente.

4.3 Modelo de Aplicação

Na Figura 11, temos o diagrama do Modelo de Aplicação do projeto Music Plus. A adição da realização do **AuthServiceClass** para a interface de login é representada pelo LoginServiceBean, que gera o código do login programático, bem como de sua permissão. No

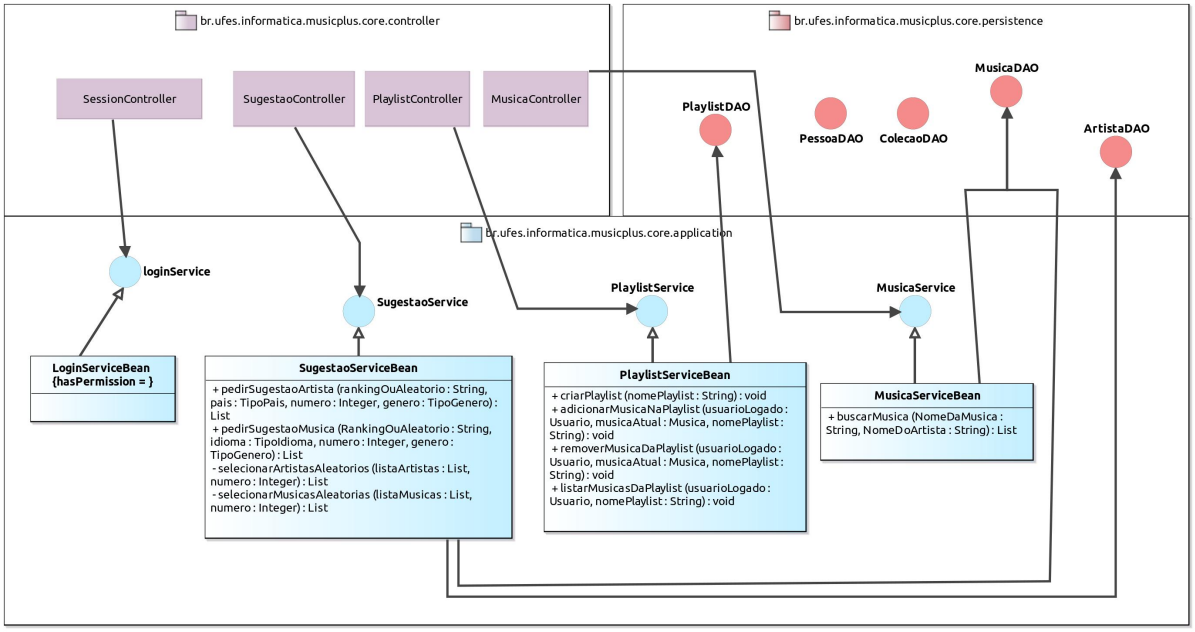

Figura 11 – Modelo de aplicação da projeto Music Plus

entanto, neste projeto os conceitos de RBAC não foram utilizados, portanto no diagrama o **permissionName** está representado como vazio. O código é mapeado para os *templates jTwig* AuthServiceClassTemplate.twig e AuthServiceInterfaceTemplate.twig.

4.4 Resultados

Como pode ser observado nos modelos apresentados, foi possível representar os elementos de segurança do projeto graficamente. Após a parte da modelagem, tais elementos puderam também ser processados a partir dos modelos pelo gerador de código. A Tabela 1 mostra a porcentagem, em termos de linha de código gerado pelo *FrameWeb Code Generator* vs. o original produzido pelos estudantes da disciplina de Desenvolvimento Web e Web Semântica (DWWS). Apenas comparamos o SessionController e o LoginServiceBean, porque é neles que existe uma sobreposição do código do login programático.

O *FrameWeb Code Generator* foi capaz de gerar a maior parte do código necessário para implementação das funcionalidades de segurança, nesse caso o login. O SessionController original possui código relacionado à solução e, se desconsiderarmos esta parte, conseguimos aproximadamente 55% sendo gerados automaticamente. Já o LoginServiceBean, foi capaz de gerar aproximadamente 70%. Mostrando que a ferramenta consegue agregar para o desenvolvedor que usar o JAAS como *framework* de segurança.

Tabela 1 – Resultado da avaliação usando Music Plus

Arquivo	**Nºlinhas Original**	**Nºlinhas Geradas**	**Porcentagem**
SessionController	215	68	33.8%
LoginServiceBean	76	58	69,84%

5 Conclusão

Este capítulo apresenta as conclusões obtidas a partir do trabalho de implementação da integração do suporte aos *frameworks* de segurança às ferramentas FrameWeb.

O objetivo deste trabalho foi implementar o suporte dos *framework* em termos do editor gráfico e do gerador de código, contribuindo com a evolução do *plugin* FrameWeb desenvolvido para Eclipse, assim auxiliando desenvolvedores Web a implementar o método FrameWeb de uma maneira mais eficiente e simples.

Depois de todo o estudo realizado, foi possível trazer a extensão do metamodelo, que trouxe consigo uma forma de representar aspectos comuns dos *frameworks* de seguranças mais populares como, Spring Security, Apache Shiro e o JAAS (Java Authentication and Authorization Services). A autenticação foi feita de forma programática, e a autorização foi feita por RBAC, como a geração de códigos do mesmo.

É fundamental destacar que essas mudanças foram feitas em cima de um *plugin* existente que já deu estrutura para a adequação dos novos construtos e para os *templates* de geração de código. Foi bem interessante colocar em prática os estudos sobre MDD, e conhecer ferramentas e linguagens dentro desse domínio, como o AQL, alem de por em pratica conhecimentos adquiridos durante o curso.

Os *templates* desenvolvidos para a tecnologia Java e podem ser melhorados e como o método FrameWeb é bastante versátil, podem ser feitos outros experimentos, testando em outras linguagens e outros *frameworks* de segurança.

Como trabalhos futuro, poderia expandir para outros tipos de *framework* de segurança e verificar se o método e a ferramenta suportam *framework* de segurança que não são RBAC, como ABCA (Attribute Based Access Control),[1] grupos de usuários, OAuth,[2] etc.

Por fim, a partir do uso das ferramentas FrameWeb e com o *feedback* de seus usuários finais, poder-se-ia dar continuidade ao desenvolvimento da ferramenta em termos do gerador de código e editor gráfico.

[1] <https://csrc.nist.gov/Projects/Attribute-Based-Access-Control>.
[2] <https://oauth.net/>.

Referências

ALMEIDA, N. V. de; CAMPOS, S. L.; SOUZA, V. E. S. A Model-Driven Approach for Code Generation forWeb-based Information Systems Built with Frameworks. In: *Proc. of the 23rd Brazilian Symposium on Multimedia and the Web (WebMedia 2017)*. Gramado, RS, Brazil: ACM, 2017. p. 245–252. Citado 3 vezes nas páginas 10, 14 e 17.

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 (WFA/WebMedia 2017)*. Gramado, RS, Brazil: SBC, 2017. p. 199–203. Citado 3 vezes nas páginas 10, 14 e 17.

FERRAIOLO, D. F.; CUGINI, J. A.; 'KUHN, D. R. *Role-Based Access Control (RBAC): Features and Motivations*. Gaithersburg MD 20899, 1995. Citado 2 vezes nas páginas 6 e 15.

MARTINS, B. F. *Evolução do Método FrameWeb para o Projeto de Sistemas de Informação Web Utilizando uma Abordagem Dirigida a Modelos*. Vitória, ES, Brazil, 2016. Citado na página 15.

OLSINA, L.; LAFUENTE, G.; ROSSI, G. *Specifying quality characteristics and attributes for websites*. Argentina, 2001. 266–278 p. Citado na página 12.

PASTOR, O. et al. Model-driven development. *Informatik-Spektrum*, Springer, v. 31, n. 5, p. 394–407, 2008. ISSN 01706012. Citado 2 vezes nas páginas 11 e 15.

PRADO, R. C. do. *Segurança no FrameWeb: Adicionando Suporte a Controle de Acesso Via Papéis em um Método de Design de Aplicações Web Baseadas em Frameworks*. Vitória, ES, Brasil, 2020. Citado 5 vezes nas páginas 6, 10, 16, 18 e 19.

PRADO, R. C. do; SOUZA, V. E. S. Securing FrameWeb: Supporting Role-based Access Control in a Framework-based Design Method for Web Engineering. In: *Proc. of the 24th Brazilian Symposium on Multimedia and the Web (WebMedia '18)*. Salvador, BA, Brazil: ACM, 2018. p. 213–220. Citado na página 10.

PRESSMAN, R. S. *Software Engineering - A Practitioner's Approach*. 8. ed. [S.l.]: McGraw-Hill, 2014. Citado na página 10.

SILVA, L. R. M. *Integração do Editor de Modelos de FrameWeb à IDE Eclipse*. Vitória, ES, Brasil, 2019. Citado 2 vezes nas páginas 10 e 14.

SOUZA, V. E. S. The FrameWeb Approach to Web Engineering: Past, Present and Future. In: ALMEIDA, J. P. A.; GUIZZARDI, G. (Ed.). *Engineering Ontologies and Ontologies for Engineering*. 1. ed. Vitória, ES, Brazil: NEMO, 2020. cap. 8, p. 100–124. ISBN 9781393963035. Disponível em: <http://purl.org/nemo/celebratingfalbo>. Citado 5 vezes nas páginas 6, 10, 13, 16 e 17.

SOUZA, V. E. S. et al. Requirements-driven software evolution. *Computer Science - Research and Development*, Springer, v. 28, n. 4, p. 311–329, 2013. Nenhuma citação no texto.

SOUZA, V. E. S.; MYLOPOULOS, J. Designing an adaptive computer-aided ambulance dispatch system with Zanshin: an experience report. *Software: Practice and Experience (online first: http://dx.doi.org/10.1002/spe.2245)*, Wiley, 2013. Nenhuma citação no texto.