Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

C837s

Costa do Prado, Rodolfo, 1988-
Segurança no FrameWeb : Adicionando Suporte a Controle de Acesso Via Papéis em um Método de Design de Aplicações Web Baseadas em Frameworks / Rodolfo Costa do Prado. - 2020.
64 f. : il.

Orientador: Vitor Estevão Silva Souza.
Dissertação (Mestrado em Informática) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Engenharia de Software. 2. Segurança de sistemas. 3. Engenharia de Sistemas. I. Estevão Silva Souza, Vitor. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 004

Rodolfo Costa do Prado

Segurança no FrameWeb: Adicionando Suporte a Controle de Acesso Via Papéis em um Método de Design de Aplicações Web Baseadas em Frameworks

Vitória, ES

2020

Rodolfo Costa do Prado

Segurança no FrameWeb: Adicionando Suporte a Controle de Acesso Via Papéis em um Método de Design de Aplicações Web Baseadas em Frameworks

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Orientador: Prof. Dr. Vítor E. Silva Souza

Vitória, ES

2020

Rodolfo Costa do Prado
Segurança no FrameWeb: Adicionando Suporte a Controle de Acesso Via Papéis em um Método de Design de Aplicações Web Baseadas em Frameworks/ Rodolfo Costa do Prado. – Vitória, ES, 2020-
63 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Vítor E. Silva Souza

Dissertação de Mestrado – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Programa de Pós-Graduação em Informática, 2020.

1. Frameweb. 2. Engenharia Web. I. Souza, Vítor Estêvão Silva. II. Universidade Federal do Espírito Santo. IV. Segurança no FrameWeb: Adicionando Suporte a Controle de Acesso Via Papéis em um Método de Design de Aplicações Web Baseadas em Frameworks

CDU 02:141:005.7

Rodolfo Costa do Prado

Segurança no FrameWeb: Adicionando Suporte a Controle de Acesso Via Papéis em um Método de Design de Aplicações Web Baseadas em Frameworks

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Trabalho aprovado. Vitória, ES, 15 de abril de 2020:

Prof. Dr. Vítor E. Silva Souza
Orientador

Prof. Dr. Sérgio Teixeira de Carvalho
Universidade Federal de Goiás

Prof. Dr. Davidson Cury
Programa de Pós-Graduação em Informática,
Universidade Federal do Espírito Santo

Vitória, ES
2020

Agradecimentos

Agradeço à minha mãe, por ter me criado sempre com amor por toda a minha vida. Ao meu pai pelo esforço para que nada me faltasse. Agradeço ao meu orientador, sem ele nada disso teria sido possível. E ao meu irmão, por sempre ser um exemplo para mim e

E Agradeço especialmente a minha esposa, pelo amor, apoio e companhia durante toda o andamento deste trabalho.

Resumo

Com a migração do mercado para a plataforma Web, devido a questões como manutenibilidade, escalabilidade e disponibilidade, tornou-se necessário o estudo das melhores práticas de engenharia de sistemas para esta plataforma. Foi criada então a Engenharia Web e, neste contexto, o método FrameWeb foi originalmente proposto em 2007. FrameWeb é um método para o desenvolvimento de Sistemas de Informação na plataforma Web, cuja arquitetura é baseada nos tipos de *frameworks* mais populares, como controladores Frontais, de Injeção de Dependência e de Mapeamento Objeto/Relacional. O objetivo do FrameWeb é alinhar a fase de *design* com a de implementação, sendo proposta a criação de uma série de modelos que ditam a configuração dos *frameworks* utilizados na codificação.

Neste contexto, é comum também o uso de *frameworks* de segurança, que proveem controle de acesso por meio de funcionalidades para autenticação e autorização e que podem ser reusadas se configuradas corretamente. O método FrameWeb, no entanto, não possui ainda suporte a este tipo de *framework*.

Neste trabalho, o método FrameWeb foi estendido para dar suporte a *frameworks* de segurança, permitindo que os desenvolvedores modelem as funcionalidades citadas anteriormente em modelos arquiteturais usando um editor gráfico e gerando código para a configuração do *framework* e artefatos relacionados. Como norteadora para as modificações ao método FrameWeb, foi usada a política de acesso a dados *Role-Based Acess Control* (RBAC), que sugere um controle de acesso baseado em papéis que cada usuário pode exercer dentro de uma aplicação e a cada papel é atribuído um conjunto de permissões para a execução das ações sobre os dados do sistema.

A proposta foi validada inicialmente usando o gerador de código e comparando os artefatos gerados automaticamente com projetos reais. Uma segunda avaliação foi feita com estudantes de graduação em Ciência da Computação, que receberam instruções de como operar a ferramenta e usaram-na para desenvolver aplicações Web. Após o desenvolvimento de seus projetos, os alunos receberam um formulário para registrar suas percepções de como a ferramenta auxilia o desenvolvimento. Foi percebido que o uso do FrameWeb com suporte aos *frameworks* de segurança auxilia na configuração dos mesmos porém, ainda são necessários esforços para facilitar o uso das ferramentas desenvolvidas.

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

Abstract

FrameWeb is a method for the development of Web-based Information Systems whose architectures are based on popular types of frameworks, such as Front Controller, Dependency Injection and Object/Relational Mapping frameworks. Also commonly used, Security Frameworks provide role-based access control through authentication and authorization features that can be reused if properly configured. In this work, we extend FrameWeb to support Security frameworks, allowing developers to model the aforementioned features in architectural design models using a graphical editor and generating code for the configuration of the framework and related artifacts.

The Role Based Access Control (RBAC) policy was used as a guideline for the extensions on the method. It proposes access control using roles that can be imbued to users inside an application, each role has a set of permissions for the execution of operations on system data.

The proposal was first validated by generating code based on models and comparing with artifacts from real projects. A second validation was made by students of Computer Science, that used the proposals of this work to develop web applications with security frameworks, then provided feedback on the experiment.

Keywords: Web Engineering, Frameworks, FrameWeb, Authentication, Authorization, Role-based Access Control, Code Generation

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UFO	Unified Foundational Ontology
RBAC	Role Based Access Control
WIS	Web-based Information System
JAAS	Java Authentication and Authorization Services
JSF	Java Server Faces
CDI	Context and Dependency Injection
API	Application Programming Interface
JPA	Java Persistence API
CSS	Cascading Style Sheets
DAO	Data Access Object
LDAP	Lightweight Directory Access Protocol
UML	Unified Modeling Language
AQL	Acceleon Query Language
XML	Extensible Markup Language
URL	Uniform Resource Locator

Sumário

1 Introdução

Há algumas décadas, a necessidade de uma abordagem mais disciplinada e de novos métodos e ferramentas para a construção de sistemas baseados na Web originou o novo campo de desenvolvimento e pesquisa, a Engenharia Web (MURUGESAN et al., 2001). Para atender a este chamado muitos autores criaram propostas de abordagens sistemáticas para o desenvolvimento Web. Juntamente com o nascimento da Engenharia Web, teve-se também a evolução de ferramentas para facilitar o desenvolvimento nesta plataforma, como *frameworks* construídos para solucionar questões comuns neste tipo de sistema. Neste contexto, o método FrameWeb (SOUZA; FALBO, 2007) foi proposto com objetivo de unir a Engenharia Web e a utilização dos *frameworks*.

FrameWeb, o *Framework-Based Design Method For Web Engineering* (Método de Projeto Web baseado em *Frameworks*). FrameWeb parte da premissa de que a maioria dos sistemas de informação baseados na Web (*Web-based Information Systems* ou WISs) são geralmente desenvolvidos em cima de uma infraestrutura sólida de *frameworks* como Controladores Frontais (ALUR; CRUPI; MALKS, 2003), que fazem a mediação da comunicação entre páginas Web (*front-end*) e os serviços da aplicação (*back-end*), mecanismos de Injeção de Dependência (FOWLER, 2004) para gerenciar dependências entre componentes do *back-end* e de Mapeamento Objeto/Relacional (BAUER; KING, 2004) para a comunicação com o banco de dados.

FrameWeb incorpora conceitos das categorias de *frameworks* citados anteriormente e um conjunto de modelos de design arquitetural, facilitando a comunicação no desenvolvimento e a documentação do projeto. Além disso, segue uma abordagem orientada a modelos (PASTOR et al., 2008), para permitir que desenvolvedores estendam o suporte do método para outras instâncias de *frameworks* dentro das categorias suportadas, provendo um editor gráfico (CAMPOS; SOUZA, 2017) para guiar desenvolvedores no uso da linguagem de modelagem do FrameWeb, e um gerador de código (ALMEIDA; CAMPOS; SOUZA, 2017) para facilitar o trabalho dos programadores, permitindo que se concentrem mais na lógica de negócio do que na infraestrutura.

Atualmente, o método suporta apenas as categorias de *frameworks* Controlador Frontal, Injeção de Dependência e Mapeamento Objeto/Relacional. Uma funcionalidade que é comumente implementada em WISs por meio de *frameworks* é autenticação e autorização. *Frameworks* de segurança como o Spring Security (https://projects.spring.io/spring-security/), Apache Shiro (https://shiro.apache.org) e o padrão JAAS (Java Authentication and Authorization Services), todos para a plataforma Java, implementam funcionalidades controle de acesso de maneira genérica, permitindo que desenvolvedores especifiquem o

código de acordo com o WIS a ser desenvolvido, aumentando o desempenho do desenvolvimento (menos código para escrever) e a confiança do código (os *frameworks* já foram exaustivamente testados).

Frameworks de segurança ainda não foram abordados pelo FrameWeb. A inclusão desta nova categoria é interessante, pois permite que desenvolvedores, na fase de projeto da arquitetura, já façam considerações sobre como será implementado o processo de autenticação da aplicação e a autorização das funcionalidades. Podem, ainda, se beneficiar da funcionalidade de geração de código, produzindo automaticamente boa parte da configuração de autenticação e autorização do *framework* escolhido.

1.1 Objetivos

O objetivo deste trabalho é a inclusão de suporte para *frameworks* de segurança ao FrameWeb. Assim, desenvolvedores poderão especificar o processo de autenticação de suas aplicações, representar em modelo quais funcionalidades do sistema a ser desenvolvido demandam autorização especial e definir como os atores do mundo real recebem acesso às mesmas. Um segundo objetivo é gerar automaticamente parte do código referente à segurança que foi definido na fase de *design*, graças à característica extensível de FrameWeb.

1.2 Método

Para alcançar os objetivos acima, foram realizadas as seguintes atividades:

1. Estudo sobre as melhores práticas quanto ao *design* da segurança em aplicações Web;
2. Análise dos principais *frameworks* de segurança para plataforma Web;
3. Modificação dos metamodelos do FrameWeb para suportar os novos conceitos identificados, propondo novos construtos nos diagramas já propostos por FrameWeb;
4. Implementação no Editor FrameWeb das modificações necessárias para a criação dos novos construtos propostos;
5. Implementação no Gerador de Código FrameWeb das modificações necessárias para a geração do código de segurança a partir dos modelos;
6. Validação da proposta feita, e das modificações nas ferramentas, por meio da comparação do código gerado com um já implementado para aplicações existentes;
7. Validação da proposta por meio de experimento com participantes voluntários, no qual a ferramenta é utilizada para o desenvolvimento de um aplicação.

1.3 Organização da Dissertação

O restante deste trabalho é organizado da seguinte forma: o Capítulo 2 introduz o referencial teórico que embasa a pesquisa que foi feita; o Capítulo 3 apresenta as propostas de mudança feitas ao FrameWeb para incluir o suporte aos *frameworks* de segurança; o Capítulo 4 detalha os experimentos feitos para avaliar o trabalho realizado; e no Capítulo 5 são apresentadas as considerações finais.

Os resultados deste trabalho foram publicados no artigo *Securing FrameWeb: Supporting Role-based Access Control in a Framework-based Design Method for Web Engineering* (PRADO; SOUZA, 2018), apresentado no 24º Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia).

2 Revisão da Literatura

Este capítulo apresenta o referencial teórico que embasa a pesquisa realizada neste trabalho. A Seção 2.1 discute o conceito de *Role-Based Access Control*, a Seção 2.2 apresenta o método FrameWeb, a Seção 2.3 descreve os *frameworks* de segurança; e a Seção 2.4 apresenta trabalhos relacionados ao desta dissertação.

2.1 Role-Based Access Control

Role-Based Access Control (RBAC) é um método para controle de acesso focado nos papéis que são executados pelas pessoas de uma determinada organização, sem se restringir a nenhuma política de segurança específica.

O objetivo para criação do *Role-Based Access Control* (RBAC) (FERRAIOLO; CUGINI; KUHN, 1995) foi simplificar o oneroso trabalho de criar e manter a configuração de segurança de uma organização. Na maioria das empresas, pessoas não possuem informação, os dados são liberados a certos indivíduos que desempenham um determinado papel dentro da empresa.

O conceito que serve como base do RBAC é a atribuição indireta de acesso a objetos para usuários. As autorizações para execução de ações sobre objetos são interpretadas como permissões que são agrupadas em papéis (*Roles*), que são desempenhados por indivíduos dentro da organização. Tais indivíduos são representados pelo conceito de Usuário (*User*). Sendo assim, um *User* agrega uma determinada quantidade de *Roles*, cada *Role* possui um conjunto de permissões que lhe dão direito de acesso a objetos. *Users* podem ser removidos de *Roles*, ou adicionados, com simplicidade, à medida que os processos de trabalho evoluem.

Devido à facilidade no controle de acesso proporcionado pelo uso do RBAC, é possível escalar a estrutura de concessão de acesso dentro de uma organização de forma prática. Além da escalabilidade, outra vantagem do RBAC é a sua aderência a políticas de acesso a dados como o acesso mínimo e a de separação de responsabilidades, descritos posteriormente nesta seção. Em termos de abstração, a abordagem se encaixa muito bem com a forma que as restrições de acesso são naturalmente formadas em uma organização.

A Figura 1 representa os 3 principais conceitos do padrão RBAC. Um *User* é uma pessoa, um *Role* é um conjunto de atividades funcionais, e uma Operação (*Operation*) representa um modo de acesso a um conjunto protegido de objetos. Vale notar que todos esses conceitos devem possuir identificadores únicos caso sejam implementados.

As funções a seguir descrevem as relações entre *Users*, *Subjects* (Que são sujeitos,

Figura 1 – Principais conceitos do RBAC e suas relações (FERRAIOLO; CUGINI; KUHN, 1995).

pessoas da organização) e *Roles* (FERRAIOLO; CUGINI; KUHN, 1995):

- *subject-user(s:subject)* = Os *Users* associados ao *Subject* "s".
- *authorized-roles(s:subject)* = Os *Roles* associados ao *Subject* "s".
- *role-members(r:role)* = Os *Users* autorizados a desempenharem o *Role* "r".
- *user-authorized-roles(u:user)* = Os *Roles* associados ao *User* "u".

Subject também deve possuir um identificador único e cada sujeito só pode estar associado a um único *User*. RBAC requer que se *authorized-roles(s)* = R e *subject-user(s)* = u então u deve estar associado com o conjunto de *Roles* R. Essa afirmação é verdadeira somente se:

- $\forall s : subject, u : user, R : Set(role), r : role, subjectUser(s) = u \wedge authorizedRoles(s) = R \wedge u \in roleMembers(r) \Rightarrow r \in R.$

Quando recebe autorização para desempenhar um *Role*, o *User* implicitamente está recebendo autorização para executar todas as *Operations* associadas a este *Role*. A relação entre *Roles* e *Operation* é definida pelas funções a seguir.

- *role-operations(r:roles)* = As *Operations* associadas ao *Role* "r".
- *operation-objects(op:operation)* = Os objetos aos quais a autorização "op" pode ser aplicada.

Para simplificar o trabalho de atribuição dos *Roles*, RBAC sugere o conceito de hierarquia. Sendo assim caso um *Role* X dentro da organização precise englobar as atividades dos *Roles* Y e Z, X pode ser definido como hierarquicamente superior aos outros dois. Vale notar que esta hierarquia é restrita à segurança e não deve simplesmente replicar a hierarquia administrativa da organização.

Na atribuição de um *User* a um *Role* o conceito de privilégio mínimo (*least privilege*) deve ser usado, ou seja, o usuário deve receber o mínimo de autorização possível para que

execute seu trabalho. Isso envolve identificar todas as atividades executadas pelo papel em questão na organização e, em seguida, buscar quais *Operations* estão envolvidas com cada atividade. Em casos de interseção de conjuntos de *Operations*, uma hierarquia de *Roles* deve ser usada.

Outro preceito a ser seguido para garantir a segurança é a divisão operacional de responsabilidade (*operational separation of duty*), segundo o qual nenhuma operação crítica da organização pode ser executada pela atuação de um único *Role*. Por exemplo a compra de algum insumo para o negócio pode ser acionada por um usuário mas deve ser autorizada por outro. Logicamente com a afirmação acima podemos concluir que nenhum *Role* pode possuir permissão para executar todas as *Operations* disponíveis.

Devido ao formato do modelo proposto, com as relações N x N entre *Users*, *Roles* e *Permissions*, o RBAC foi adotado neste trabalho como modelo básico para o controle de acesso no FrameWeb.

A Tabela 1 resume as regras para uma implementação correta do RBAC.

2.2 FrameWeb

FrameWeb (SOUZA; FALBO, 2007) é um método de Design para o desenvolvimento de Sistemas de Informação na plataforma Web, (*Web-based Information Systems*, WISs) que espera o uso de *framewoks* durante a fase de implementação. Sendo assim, o método busca introduzir alguns dos conceitos referentes aos *framewoks* que serão usados no ciclo de vida do software. Essa abordagem busca garantir um melhor direcionamento para a fase de implementação, pois os programadores irão utilizar os *framewoks* da maneira esperada pela equipe de *design*.

A abordagem propõe uma arquitetura básica que divide o sistema em três camadas principais (Apresentação, Lógica de Negócio e Acesso a Dados), para uma melhor integração com os três tipos de *framewoks* previstos: Controladores Frontais (ex.: *JavaServer Faces*/JSF), Injeção de Dependência (ex.: *Context and Dependency Injection for Java*/CDI) e de Mapeamento Objeto/Relacional (ex.: *Java Persistence API*/JPA). Essa arquitetura está representada na Figura 2.

As três camadas estão divididas em 5 pacotes, a saber:

- **Visão:** contém as páginas Web, folhas de estilo (CSS), scripts *client-side* e outros artefatos de interface;
- **Controle:** contém as classes controladoras, que lidam com as requisições feitas no pacote de visão, por meio da infraestrutura do *framework* controlador frontal, e faz as chamadas ao pacote de Aplicação;

Tabela 1 – Regras a serem seguidas para implementação completa de RBAC (FERRAIOLO; CUGINI; KUHN, 1995)).

Regra	**Descrição**	**Equação**
1. Hierarquia de *Roles*	Se um *subject* está autorizado a exercer um *Role* e este contém outro *Role*, então o *subject* também deve possuir o *Role* contido.	$\forall s : subject, r_{i,j} : roles : r_j \in authorized - roles(s) \wedge r_j > r_i \Rightarrow r_i \in authorized - roles(s)$.
2. Separação de Responsabilidade	Um *User* só está autorizado a exercer um *Role* se este não for mutualmente exclusivo com outro *Role* que ele já possua.	$\forall u : user, r_{i,j} : roles : i \neq j : u \in role - members(r_i) \wedge u \in role - members(r_j) \Rightarrow r_i \notin mutually - exclusive - authorization(r_j)$
3. Cardinalidade	A capacidade máxima de membros de um *Role* não pode ser superada.	$\forall r : roles : membership - limit(r) \geq number - of - members(r)$.
4. Autorização de *Role*	Um *subject* nunca pode ter um *Role* ativo para o qual ele não esteja autorizado.	$\forall s : subject, op : operation : active - roles(s) \subseteq authorized - role(s)$.
5. Execução de *Role*	Um *subject* só pode executar uma *operation* se tiver um *Role* ativo autorizado para tal.	$\forall s : subject, op : operation : exec(s, op) \Rightarrow active - roles(s) \neq \varnothing$
6. Separação dinâmica de responsabilidade	Um *subject* só pode receber um novo *Role* caso este não seja mutualmente exclusivo com nenhum de seus *Roles* atualmente ativos.	$\forall s : subject, r_{i,j} : roles : i \neq j : r_i \in active - roles(s) \wedge r_j \in active - roles(s) r_i mutually - exclusive - activation(r_j)$.
7. Autorização a *Operation*	Um *subject* só pode executar uma *Operation* se possuir algum *Role* ativo que esteja autorizado a tal.	$\forall s : subject, op : operation \exists r : roles : exec(s, op) \Rightarrow r \in active - roles(s) \wedge op \in role - operations(r)$.
8. Separação operacional de responsabilidade	Um *Role* só pode ser autorizado a executar uma *Operation* dentro de uma tarefa de negócio, caso este já não esteja autorizado a executar todas as outras operações envolvidas a tal tarefa.	$\forall s : subject, r : role, f : function : \neg(function - operations(f) \subseteq role - operations(r))$.
9. Autorização de acesso a objeto	Um *Subject* só pode acessar um objeto somente se possuir o *Role* com autorização de acesso a tal objeto.	$\forall s : subject, o : object : access(s, o) \Rightarrow \exists r : roles, op : operation : r \in active - roles(s) \wedge op \in role - operations(r) \wedge o \in operation - objects(op)$.

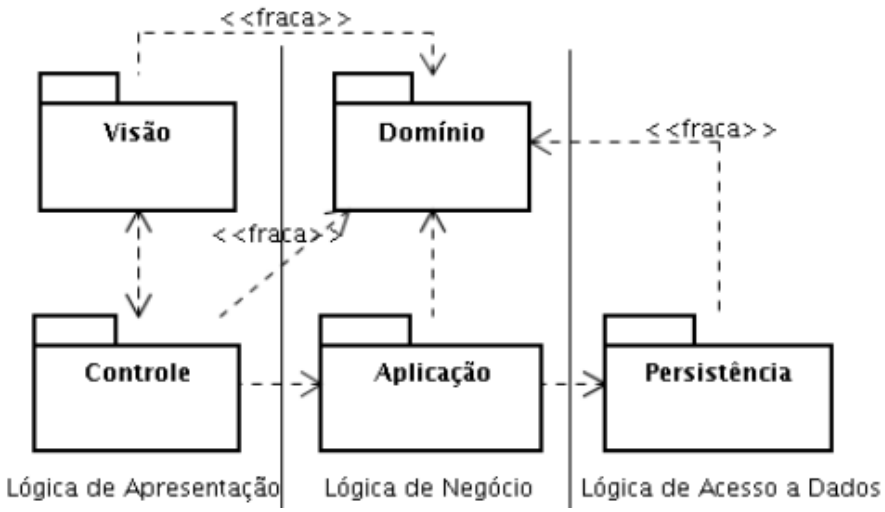

Figura 2 – Arquitetura padrão para WIS baseada no padrão arquitetônico *Service Layer* (SOUZA; FALBO, 2007).

- **Aplicação:** contém as classes responsáveis por implementar as funcionalidades do sistema, suas dependências com os pacotes de controle e persistência são preenchidas pelo *framework* de injeção de dependências. O pacote de aplicação manipula o pacote de Domínio e persiste as modificações por meio do pacote de Persistência;

- **Domínio:** contém as classes que representam o domínio do problema da aplicação e as anotações que vão guiar a persistência das mesmas pelo *framework* de mapeamento objeto/relacional;

- **Persistência:** contém as classes DAO (*Data Acess Object*), responsáveis por persistir no banco de dados as instâncias do pacote de Domínio por meio do *framework* objeto relacional.

Artefatos destes pacotes são representados em quatro diagramas, com objetivo de guiar (e gerar código para) a implementação da aplicação e configuração dos diferentes *frameworks* que serão usados.

Na abordagem de desenvolvimento orientado a modelos adotada pelo FrameWeb, os modelos pertencem a níveis de abstração baseados no que está sendo representado por seus elementos. O nível **M0** representaria diretamente os objetos de uma aplicação em tempo de execução. Já um modelo no nível de abstração **M1** representa as classes desta aplicação que são instanciadas pelos objetos no nível M0.

Os modelos propostos no FrameWeb, todos do nível **M1**, são:

- **Modelo de Persistência:** representa as classes do pacote de Persistência, nele podem ser identificados os métodos de acesso a dados que devem ser criados para cada classe do pacote de Domínio;

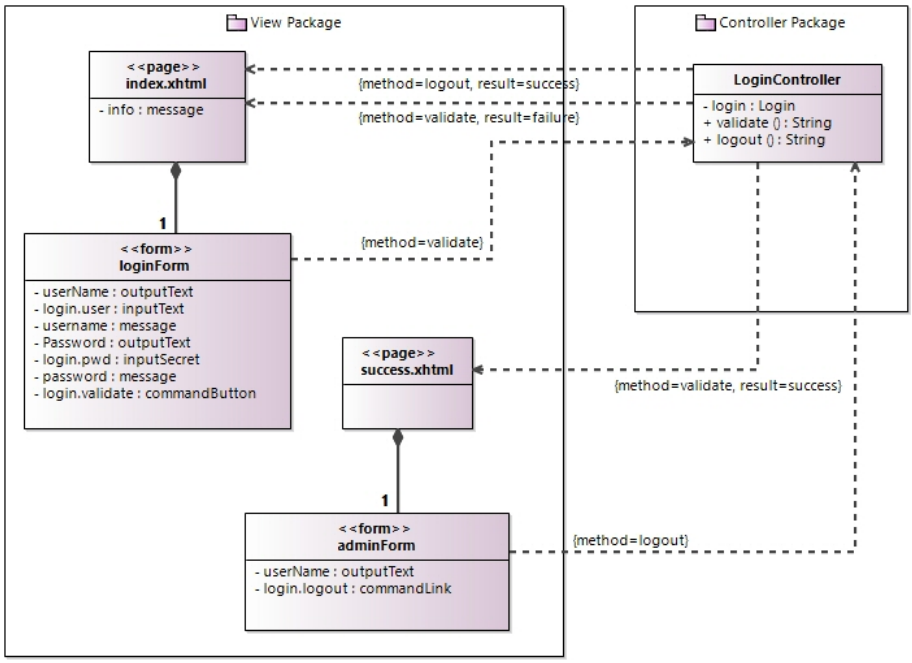

Figura 3 – Modelo de navegação construído com o editor FrameWeb (CAMPOS; SOUZA, 2017).

- **Modelo de Entidades:** representa as classes do pacote de Domínio e seus metadados, estes serão usados para o mapeamento Objeto/Relacional;
- **Modelo de Aplicação:** representa as classes do pacote de Aplicação e suas dependências com o pacote de Persistência. Também demostra quais classes de Controle possuem dependências para com classes de Aplicação;
- **Modelo de Navegação:** representa os componentes que formam a camada de apresentação, pacotes de Visão e Controle, e como estes interagem.

Seguindo uma abordagem *model-driven*, FrameWeb provê um metamodelo que define a linguagem abstrata usada nos modelos listados anteriormente, baseada na linguagem concreta da UML (MARTINS; SOUZA, 2015). Este metamodelo está no nível **M2**, pois seus elementos definem quais tipos de classes e as relações entre classes, que poderão existir nos modelos FrameWeb, ou seja, são instanciados pelos modelos do nível M1.

A Figura 3 apresenta um modelo de navegação,nível **M1**, criado para representar o login de uma aplicação implementada via JSF, extraída de um tutorial online e construída usando o *FrameWeb Editor* (CAMPOS; SOUZA, 2017), uma ferramenta CASE feita para o método FrameWeb.

O modelo representa duas páginas Web (estereótipo «page»), cada uma com um formulário (estereótipo «form»), ambas interagindo com uma classe controladora chamada LoginController. Existe uma relação de dependência de loginForm com LoginControler. Isto indica que quando este formulário é submetido o método validate() da classe LoginController

é acionado. O método a ser acionado é descrito no atributo method da relação. Pode-se ver também os respectivos pacotes que contém as entidades representadas no modelo. Dentro da representação de loginForm estão presentes alguns componentes JSF, como login.user e login.pwd que serão vinculados aos atributos user e pwd do objeto login, este que é um atributo de LoginController.

O método validate() possui dois possíveis resultados: success, que redireciona o usuário para uma página específica como podemos ver pela relação entre LoginController e success.xthml, e failure que resulta no retorno para a página index.xhtml, relação também exposta no modelo.

O Editor FrameWeb (CAMPOS; SOUZA, 2017) foi implementado usando o Eclipse Sirius e permite que o usuário crie modelos FrameWeb usando as definições de *frameworks* Web já existentes ou faça inclusão de características específicas de outras instâncias de *frameworks* dentre as categorias identificadas. Por exemplo, na Figura 3 foram usadas definições do *framework* JSF, como seus componentes visuais específicos. Desenvolvedores podem estender ainda mais os modelos adicionando especificações de outros *frameworks*, coso estes estejam dentro das categorias pré definidas.

Foi desenvolvido também o *FrameWeb Code Generator* (ALMEIDA; CAMPOS; SOUZA, 2017), que é capaz de gerar esqueletos de código para os diferentes artefatos modelados, como páginas Web e classes, usando modelos construídos no *FrameWeb Editor*. Este também é extensível pois são usados *templates* para geração do código, permitindo que interessados possam usar a ferramenta para construção de aplicativos em outras linguagens. As definições dos *frameworks* são guiadas pelo meta-modelo do FrameWeb (MARTINS; SOUZA, 2015), desenvolvido com intuito de garantir a formalização da linguagem usada nos diagramas FrameWeb.

Resumidamente, o FrameWeb é um método de desenvolvimento que propõe modelos que representam as classes de cada camada do WIS e como essas irão interagir entre si, considerando os *frameworks* adotados.

2.3 Frameworks de Segurança

Frameworks de segurança proveem uma infraestrutura reutilizável com uma série de funcionalidades relacionadas à segurança de uma aplicação, como autenticação, autorização, criptografia, controle de sessão etc. Neste trabalho nos focaremos na autenticação, verificação se um usuário é quem ele afirma ser, e autorização, verificação referente a permissão de um usuário para executar determinada ação.

A pesquisa feita neste trabalho se concentrou na plataforma Java, mais especificamente em 3 *frameworks*: Spring Security, Apache Shiro e o padrão Java JAAS (Java

Listagem 2.1 – Parte do formulário de login usado como exemplo na documentação do Spring Security.

```
1 <form name="f" th:action="@{/login}" method="post">
2 <label for="username">Username</label>
3 <input type="text" id="username" name="username"/>
4 <label for="password">Password</label>
5 <input type="password" id="password" name="password"/>
6 <div class="form-actions">
7 <button type="submit" class="btn">Log in</button>
8 </div>
9 </form>
```

Listagem 2.2 – Parte do arquivo de configuração do Spring Security

```
1 <http auto-config="true" use-expressions="true">
2 <intercept-url pattern="/loginForm" access="permitAll" />
3 <intercept-url pattern="/user/login" access="permitAll" />
4 <intercept-url pattern="/**" access="hasAuthority('PERM_USER')" />
5 <form-login login-page="/loginForm" login-processing-url="/performLogin" username
    -parameter="username" password-parameter="password" default-target-url="/"/>
6 <logout logout-url="/logout" logout-success-url="/index"/>
7 </http>
```

Authentication and Authorization Services), pois estes foram, com uma larga vantagem, os mais pesquisados no Google em 2018. Esta verificação foi feita por meio da ferramenta Google Trends,[1] na qual termos de pesquisas podem ser comparados entre si quanto ao número de buscas realizadas pelos usuários do Google. Assumiu-se, então, que as ferramentas mais buscadas são as mais utilizadas pelo mercado.

A forma mais básica de autenticação fornecida por estes *frameworks* consiste na comparação entre as credenciais fornecidas por um usuário, com as credenciais que foram previamente salvas pela aplicação. Exemplo: um formulário de login no qual são informados o nome de usuário e sua senha e estes são verificados no banco de dados. Na configuração destes *frameworks* os nomes dos campos das credenciais são definidos e devem ser usados no formulário de login.

A Listagem 2.1 apresenta um simples formulário de login do Spring Security, que foi configurado com o conteúdo da Listagem 2.2. O arquivo de configuração especifica as URLs para o login e para o processamento do login, assim como os nomes a serem usados nos campos do formulário. Isso permite que o *framework* intercepte a requisição feita pelo formulário e extraia os valores de nome e senha de usuário. Após isso a autenticação é executada.

As credenciais informadas são comparadas com as provindas da aplicação, que podem ser definidas no arquivo de configuração do *framework*. Porém, em sistemas de tamanho relevante é mais comum existir alguma espécie de banco de dados ou integração

[1] <https://trends.google.com/trends/explore?date=today%205-y&q=Apache%20shiro,Spring%20Security,Jaas,Jguard,HDIV>

Listagem 2.3 – Class method with authorization annotation for Spring Security.

```
1 @PreAuthorize("hasAuthority('PERM_PERSON_DELETE')")
2 public void deletePerson(Person person);
```

LDAP,[2] por meio da qual o nome de usuário e senha podem ser guardados e acessados. Este conceito é chamado de *User Store* e os *frameworks* de segurança interagem com ela de variadas maneiras, mas a base de como as credenciais são verificadas é a mesma.

Além da autenticação, precisamos capacitar o *framework* a fazer a autorização. Em alguns exemplos, existentes na documentação das ferramentas de segurança citadas anteriormente, a autorização é feita através da simples associação entre o conceito de *User* ou *Role* com as classes e métodos presentes no código da aplicação. Essa associação direta vai de encontro ao o que é explicitado no modelo RBAC, que propõem alternativamente uma associação do *User* a uma lista de *Roles* e de cada *Role* a uma lista de *Permissions*. O conceito permission não é expressamente citado no RBAC, porém é usado nos *frameworks* de segurança para representar a permissão para execução de uma *Operation*.

Assim que a *User Store* é definida, precisamos informar que ações precisam de segurança na aplicação e quais permissões serão requeridas para execução de cada ação. Isso pode ser feito na camada de apresentação por meio da definição de permissões específicas para URLs (endereços das páginas Web). Porém, as boas práticas dizem que a segurança d eve ser feita nos serviços existentes na camada de Lógica de Negócio. Isso é feito pela associação direta de *Permissions* a classes e métodos do pacote de Aplicação, usando um arquivo de configuração ou anotações, este último sendo o mais comum pois o desenvolvedor consegue entender mais facilmente a estrutura do sistema de permissões simplesmente olhando o código-fonte da classe.

A Listagem 2.3 apresenta uma anotação de associação entre um método e uma *Permission*. somente usuários que possuem a permissão `PERM_PERSON_DELETE` podem executar este método. Acesso não autorizado geraria uma exceção, que poderia ser tratada pela camada de apresentação.

Assim como outros tipos de *frameworks*, a não ser que seja feita a documentação relativa à segurança, como a apresentada acima, no *design* arquitetural do sistema de informação, o trabalho de criar e codificar essas configurações fica sob responsabilidade do programador. No próximo capítulo será feita uma proposição de extensão ao método *FrameWeb* que permitirá aos arquitetos de software especificar a autenticação e autorização, bem como gerar parte do código relacionado ao uso dos *frameworks* de segurança automaticamente, a partir dos modelos especificados.

[2] *Lightweight Directory Access Protocol*, ou LDAP, é um protocolo de aplicação aberto, livre de fornecedor e padrão de indústria para acessar e manter serviços de informação de diretório distribuído sobre uma rede de Protocolo da Internet (IP). Fonte: <https://pt.wikipedia.org/wiki/LDAP>.

Conclui-se, então, que os *frameworks* de segurança considerados são capazes de prover ao programador ferramentas para fazer a autenticação dos usuários e implementação do modelo RBAC nas aplicações Web.

2.4 Trabalhos relacionados

Como mencionado no Capítulo 1, o método FrameWeb busca representar visualmente conceitos dos *frameworks* que irão ser usados na construção do sistema de informação Web (Web-based Information Systems – WISs). Os trabalhos anteriores no contexto do FrameWeb (MARTINS; SOUZA, 2015; ALMEIDA; CAMPOS; SOUZA, 2017; SOUZA; FALBO, 2007) não propuseram a inclusão de *frameworks* de segurança. Este trabalho busca preencher esta lacuna. Outros trabalhos na literatura propuseram a modelagem da funcionalidades de segurança para WISs, a seguir.

SecureUML (LODDERSTEDT; BASIN; DOSER, 2002) propõe uma linguagem de modelagem baseada na UML e no padrão RBAC para criar modelos que documentam a estrutura de autorização imaginada para uma aplicação. Nos modelos de SecureUML, instancias de *Roles* (vide Seção 2.1) são representadas e as *Permissions* descritas como uma relação entre um *Role* especifico e uma classe, associando esssa relação a uma ação especifica. Possíveis restrições de segurança podem ser adicionadas ao modelos através de UML *constraints* associadas a classes ou relações.

Os modelos criados usando SecureUML buscam representar uma fotografia do esquema de autorização de uma aplicação, não tem a intenção de auxiliar o desenvolvedor a criar este esquema nem propõe arquitetura para que tal logica de segurança seja modificada dinamicamente pelos usuários da ferramenta.

Pavlich-Mariscal, Michel e Demurjian (2007) propõem uma extensão da linguagem UML para modelagem dos aspectos de segurança de uma aplicação, sendo muito útil para criação dos requisitos de autorização de um sistema. Porém, também é uma representação estática da lógica de autorização.

UMLsec (JüRJENS, 2004), também é baseado na UML e cria modelos com intenção de simplificar o desenvolvimento de segurança de sistemas críticos, porém com um escopo um pouco maior, levando em consideração o ambiente em que a aplicação existe. O foco deste trabalho é na fase de *design* e na implementação dos WISs que usam *frameworks* de segurança.

Emig et al. (2007) propõem um meta-modelo para controle de acesso para arquiteturas de serviço Web que estende o meta-modelo do RBAC, porém não leva em consideração o uso de *frameworks* para lidar com a segurança.

Os trabalhos existentes na literatura que propuseram a modelagem de funciona-

lidades de segurança em WISs não preenchem a lacuna que este trabalho se propõe a preencher: um método para o *design* de WISs que utilizam *frameworks* de segurança, com uma linguagem de modelagem que incorpora os conceitos dos *frameworks* nos modelos de projeto, promovendo uma melhor comunicação entre desenvolvedores e facilitando o trabalho de implementação por meio de geração de código.

No próximo capítulo, apresentamos a proposta deste trabalho, por meio da adição dos conceitos de segurança ao FrameWeb.

3 Adição da Segurança ao FrameWeb

Neste capitulo, são apresentadas extensões do metamodelo do *FrameWeb* e de suas ferramentas associadas — *FrameWeb Editor* e *FrameWeb Code Generator* —, para que o método contemple conceitos de *Role-Based Access Control* (RBAC) e dos *frameworks* de segurança.

A Seção 3.1 apresenta os novos construtos adicionados à linguagem de modelagem de FrameWeb para que o método passe a ter suporte a *frameworks* de segurança. As Seções 3.2 e 3.3, respectivamente, descrevem como as ferramentas de edição e geração de código do FrameWeb foram estendidas para dar suporte aos novos construtos propostos.

3.1 Novos Construtos da Linguagem

A linguagem do *FrameWeb* foi estendida para especificar, em seus modelos, a configuração das propriedades de autenticação e autorização dos *frameworks* de segurança, usando os conceitos de RBAC. Conforme ilustrado na Seção 2.2, esta configuração envolve:

- As classes de domínio que representam *Users*, *Roles* e *Permissions* para a autenticação, como exemplificado na Listagem 2.1;
- Aspectos das páginas Web e formulários que iniciaram a requisição de autenticação, como exemplificado na Listagem 2.2;
- Que *Permissions* são requeridas para cada classe ou método de classe de serviço, como exemplificado na Listagem 2.3.

As definições supracitadas envolvem, respectivamente, o Modelo de Entidades, de Navegação e de Aplicação de *FrameWeb*. Como o desenvolvimento orientado a modelo foi incorporado ao método (MARTINS; SOUZA, 2015), para estender a linguagem foram feitas modificações nos metamodelos relativos a estes três diagramas. Como mencionado anteriormente, no contexto de desenvolvimento orientado a modelos, os modelos FrameWeb fazem parte da camada M1 e seus meta-modelos encontram-se na camada M2. Sendo assim, ao estender os modelos da camada M2, a linguagem usada pelos modeladores FrameWeb na camada M1 é incrementada com os elementos relativos aos *frameworks* de segurança.

A Figura 4 apresenta fragmentos dos três metamodelos, focado nas mudanças feitas para dar suporte a autenticação e autorização. As meta-classes originais são as apresentadas em verde, rosa e azul para os modelos de Entidades, Navegação e Aplicação

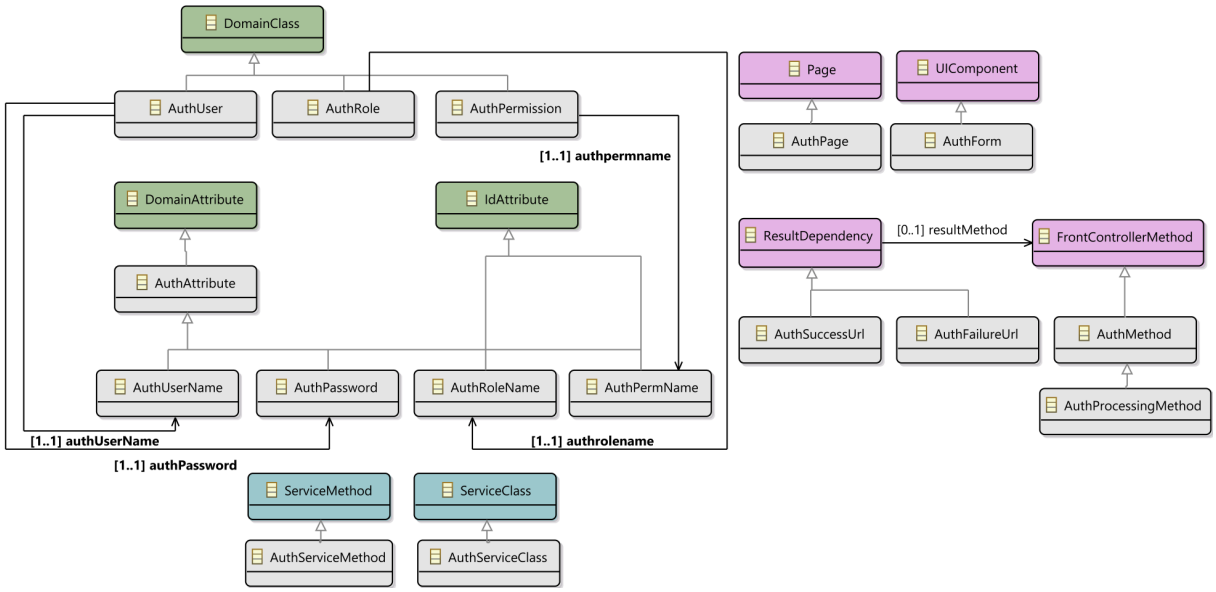

Figura 4 – Fragmentos dos meta-modelos de Entidades, Navegação e Aplicação, modificados para dar suporte aos *frameworks* de segurança.

respectivamente. As representadas em cinza são as adicionadas como resultado deste trabalho.

Os conceitos de RBAC são representados no modelo de entidades. Caso uma aplicação precise adotar RBAC, ela precisa de, no mínimo, implementar três entidades: *User*, *Role* e *Permission*, com associações n-para-n entre *Users* e *Roles* e entre *Roles* e *Permissions*. No metamodelo do *FrameWeb* (Figura 4) a meta-classe DomainClass foi estendida para permitir a adição de 3 novos construtos como classes de entidades: AuthUser, AuthRole e AuthPermission. A meta-classe AuthAttribute foi derivada de DomainAttribute para que atributos específicos de login sejam adicionados a essas classes: AuthUserName, AuthPassword, AuthRoleName and AuthPermName (nome da permissão). Os últimos dois também são extensões da meta-classe IdAttribute. Isso se deve ao fato de que os *frameworks* de segurança, para realizar a autorização, comparam os nomes das classes *Role* e *Permission* salvas pela aplicação com os nomes especificados nos arquivos de configuração ou nas *annotations*, então é fundamental que os nomes sejam identificadores únicos destas classes, evitando inconsistências na autorização.

A Figura 5 apresenta um Modelo de Entidades usando as novas meta-classes. Pode ser visto que foram usados como sintaxe concreta para representar *Users*, *Roles* e *Permissions* os seguintes estereótipos, respectivamente: «AuthUser», «AuthRole» and «AuthPermission». Para facilitar a visualização no editor, essas classes são representadas em cinza. É importante notar que elas podem ter qualquer nome que o designer desejar, já que sua função nos processos de autenticação e autorização é definida pelos estereótipos.

Enquanto a parte estrutural da autenticação é representada no Modelo de Entidades, seu comportamento é definido no Modelo de Navegação, como no exemplo da Figura 6.

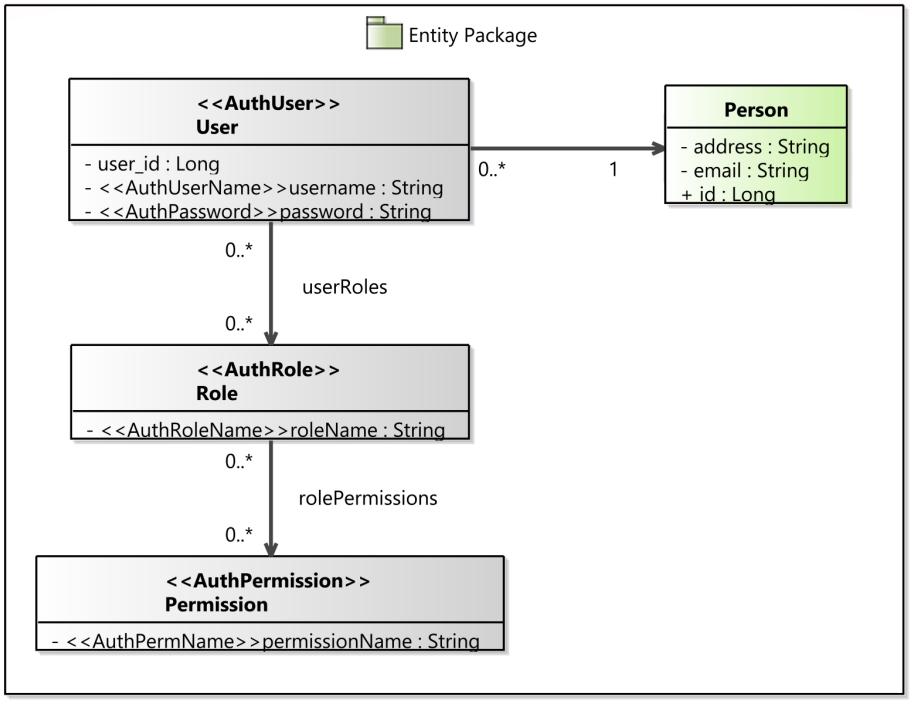

Figura 5 – Modelo de entidades com os construtos de autenticação e autorização.

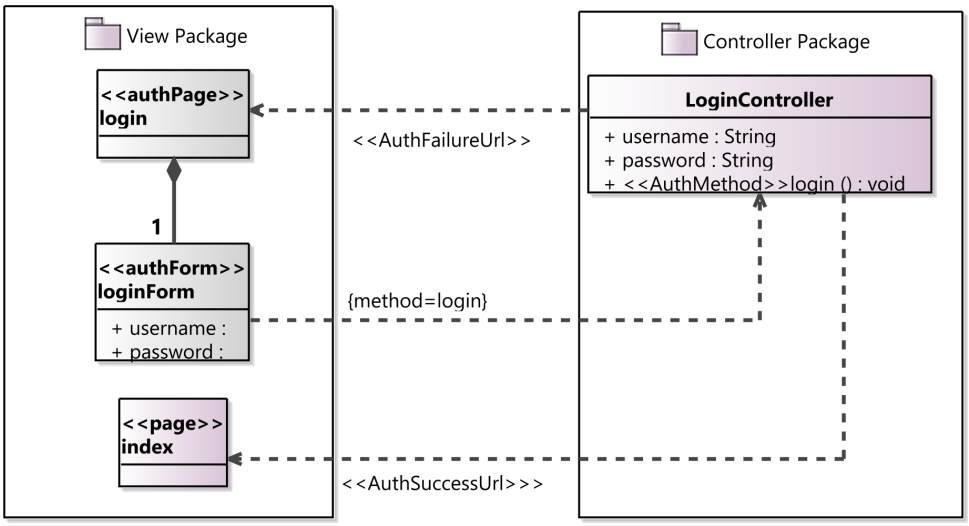

Figura 6 – Modelo de navegação com os construtos de autenticação e autorização.

O modelo representa uma página de login (estereótipo «authPage»), o formulário com os campos relativos as credenciais («authForm»), assim como as URLs de processamento («AuthMethod»), sucesso («AuthSuccessUrl») e falha («AuthFailureUrl») de login. No meta-modelo da Figura 4, as meta-classes AuthPage (página de login) e AuthForm (formulário de login) são derivadas de Page e UIComponent, respectivamente. As URLs para sucesso ou falha da autenticação são representadas como subclasses de ResultDependency, devido a sua dependência para com o método de processamento de login.

O processamento do login é executado pelo método que instancia a metaclasse AuthProcessingMethod, subclasse de FrontControllerMethod. A URL vinculada a tal método

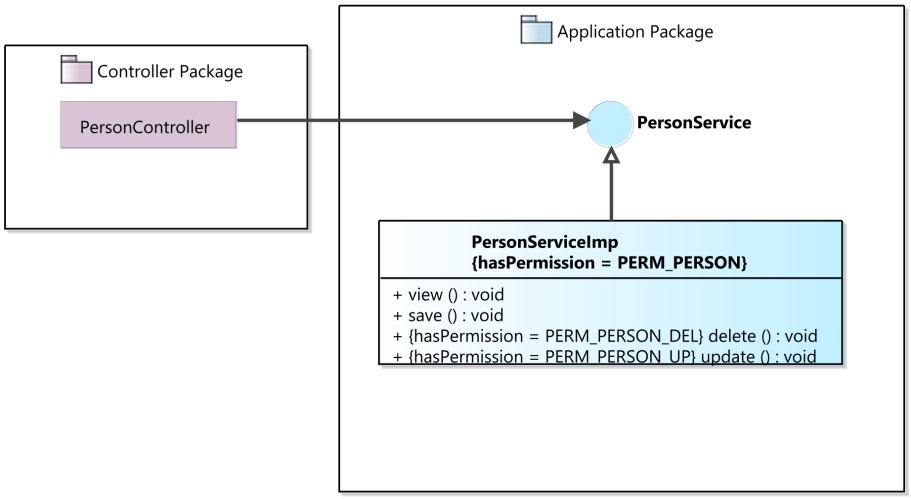

Figura 7 – Modelo de aplicação com os construtos de autenticação e autorização.

seria a *login processing URL* apresentada anteriormente na Listagem 2.2. Assim, o modelo se mantém relevante com uma implementação de segurança através de uma classe de serviço para prover credenciais e permissões, caso o desenvolvedor prefira customizar mais a segurança de sua aplicação.

Como explicado na Seção 2.3, a autorização é feita na camada de lógica de negócio pelas classes de serviço, ou seja, no Modelo de Aplicação do *FrameWeb*. A Figura 7 apresenta um modelo de aplicação com *Permissions* sendo representadas por restrições UML como linguagem concreta. A classe de serviço PersonServiceImp requer a permissão com nome de PERM_PERSON para ser acessada. Os métodos delete() e update() requerem PERM_PERSON_DEL e PERM_PERSON_UP respectivamente, além da permissão advinda da classe citada anteriormente. Isso foi feito por meio da extensão das meta-classes ServiceClass e ServiceMethod para AuthServiceClass e AuthServiceMethod respectivamente, como demostrado na Figura 4. Ambas subclasses possuem o atributo PermissionName, a fim de guardar o nome da permissão requerida para o acesso.

3.2 Suporte à Edição de Modelos

As mudanças do meta-modelo apresentadas na seção anterior também foram implementadas na ferramenta *FrameWeb Editor* (CAMPOS; SOUZA, 2017). As figuras 5–7 foram construídas já nesta nova versão do editor. A Figura 8 mostra o modelo da Figura 7 sendo construído dentro do editor.

Estender o editor FrameWeb significou criar *containers nodes* e *edges* para instanciar visualmente as novas meta-classes incorporadas ao meta-modelo do *FrameWeb*. Na Figura 9 pode-se ver uma porção da ferramenta Sirius na qual estes elementos são definidos.

Com essas adições, no entanto, superclasses já existentes no meta-modelo (ex.: Do-

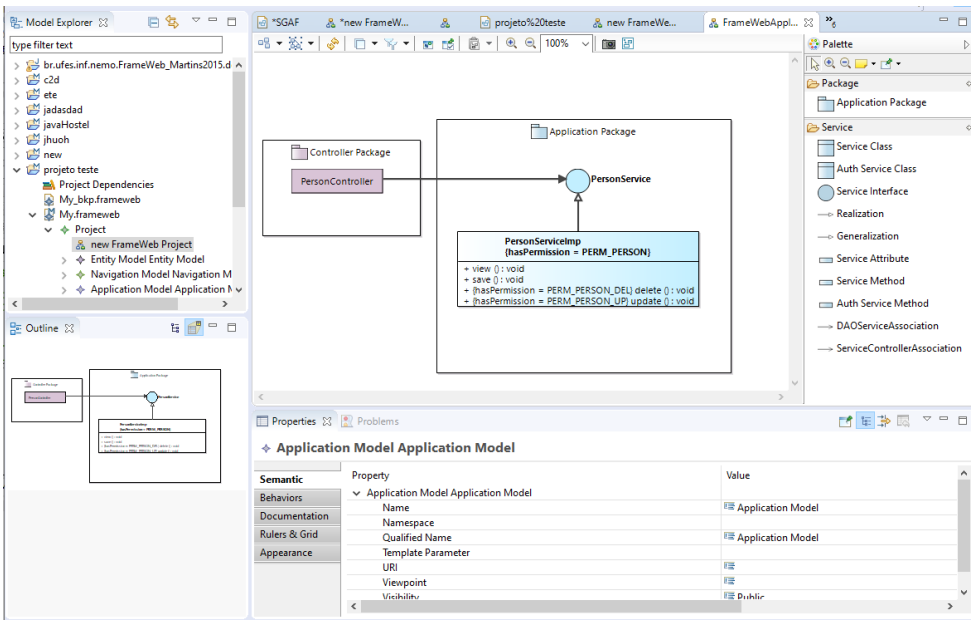

Figura 8 – Captura de tela do Editor FrameWeb, na qual pode-se ver um modelo de aplicação sendo construído e as permissões de acesso especificadas.

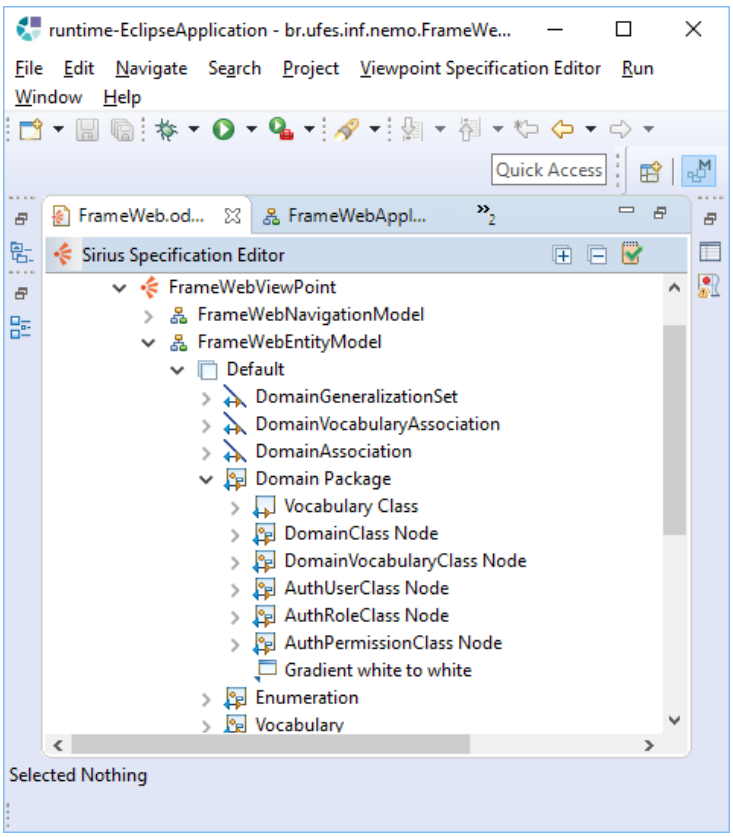

Figura 9 – Definição dos *nodes*, usados para representar os construtos no editor Sirius.

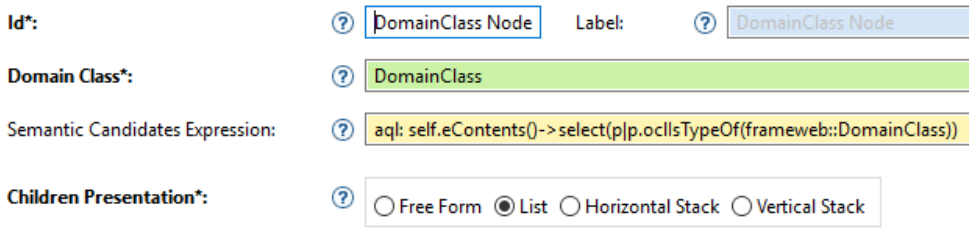

Figura 10 – Consulta AQL usada para filtrar os *nodes* da *DomainClass*.

mainClass) e subclasses propostas neste trabalho (ex.: AuthUser, AuthRole e AuthPermission) são instanciadas no mesmo modelo, fazendo com que a ferramenta crie dois *nodes*, um para a superclasse e outro para a subclasse, quando um dos novos construtos propostos neste trabalho fosse incluído no modelo. Para evitar isso, a linguagem Acceleo Query Language,[1] ao invés de expressões regulares, foi usada nos *nodes* relativos às superclasses para decidir que elementos serão apresentados. A expressão AQL pode ser vista no campo *Semantic Candidates Expression* presente na Figura 10. Isso garante que o *node* só seja apresentado se a superclasse for instanciada diretamente.

Os modelos com os construtos de autenticação e autorização, feitos na fase de design arquitetural, podem ser usados pelos desenvolvedores na configuração dos *frameworks* de segurança, pois os conceitos representados por eles são os mesmos adotados pelos principais *frameworks* de segurança (na plataforma Java) e podem prover uma direção para o uso dessas ferramentas em um sistema Web. Na próxima seção, descrevemos como esses modelos podem ser usados para gerar automaticamente artefatos de código para configuração. O código-fonte do editor entre outros artefatos podem ser acessadas no seguinte repositório: <https://github.com/nemo-ufes/FrameWeb>.

3.3 Geração de Código

O *FrameWeb Code Generator* (ALMEIDA; CAMPOS; SOUZA, 2017) traduz os modelos *FrameWeb* para código-fonte, produzindo artefatos que o desenvolvedor pode usar na construção de seus sistemas Web. O gerador de código navega nos aquivos `.frameweb` dos modelos (produzido pelo *FrameWeb Editor*) e extrai a informação requerida para criar classes de projeto, páginas e outros aquivos. Com as extensões descritas na Seção 3.1, a ferramenta também foi estendida para entender os conceitos de autenticação e autorização mínimos necessários para o uso dos *frameworks* de segurança considerados neste trabalho.

A Listagem 3.1 apresenta um trecho do arquivo `.frameweb` do modelo de navegação visto na Figura 6 no formato XML. Tomando a URL de processamento de login como exemplo, para conseguir esta informação a ferramenta procura pelas *tags* `packagedElements` com o tipo `FrontControllerClass` dentro do `packagedElement` com o tipo

[1] AQL, <http://www.eclipse.org/acceleo/documentation/>.

Listagem 3.1 – Trecho de um arquivo `.frameweb` de um modelo de navegação.

```
1 <compose xsi:type="frameweb:NavigationModel" name="Navigation Model">
2 <packagedElement xsi:type="frameweb:ControllerPackage" name="Controller Package">
3 <packagedElement xsi:type="frameweb:FrontControllerClass" name="LoginController">
4 <ownedOperation xsi:type="frameweb:AuthProcessingMethod" name="login"/>
5 </packagedElement>
6 </packagedElement>
7 <packagedElement xsi:type="frameweb:ViewPackage" name="View Package">
8 <packagedElement xsi:type="frameweb:AuthForm" name="loginForm">
9 <ownedAttribute xsi:type="frameweb:UIComponentField" name="username"/>
10 <ownedAttribute xsi:type="frameweb:UIComponentField" name="password"/>
11 <packagedElement/>
12 <packagedElement xsi:type="frameweb:AuthPage" name="login"/>
13 <packagedElement xsi:type="frameweb:Page" name="index"/>
14 </packagedElement>
15 <packagedElement xsi:type="frameweb:AuthFailureUrl" client="//@compose.1/
     Controller%20Package/LoginController" supplier="//@compose.1/View%20Package/
     login_page">
16 <resultDependencyConstraint result="null"/>
17 </packagedElement>
18 <packagedElement xsi:type="frameweb:AuthSuccessUrl" client="//@compose.1/
     Controller%20Package/LoginController" supplier="//@compose.1/View%20Package/
     index">
19 <resultDependencyConstraint result="null"/>
20 </packagedElement>
21 </compose>
```

Listagem 3.2 – Trecho do código XML gerado para configuração do Spring Security.

```
1 <http auto-config='true' use-expressions = "true">
2 <intercept-url pattern="/FW_AUTH_LOGIN_PAGE" access="permitAll" />
3 <intercept-url pattern="/FW_AUTH_LOGIN_PROC_URL" access="permitAll" />
4 <form-login login-page="/FW_AUTH_LOGIN_PAGE" login-processing-url="/
     FW_AUTH_LOGIN_PROC_URL" username-parameter="FW_AUTHAT_USERNAME" password-
     parameter="FW_AUTHAT_PASSWORD" default-target-url="/FW_AUTH_LOGIN_SUCC_URL"
     authentication-failure-url="/FW_AUTH_LOGIN_FAIL_URL"/>
5 <logout logout-url="/logout" logout-success-url="/FW_AUTH_LOGIN_PAGE"/>
6 </http>
```

`ControllerPackage`, o que resulta em uma lista de *Controllers*. Dentro de cada `FrontControllerClass`, o tipo `AuthProcessingMethod` é procurado. Quando encontrado a propriedade nome do `AuthProcessingMethod` é usada para construir a URL de processamento de login. A mesma abordagem é usada para conseguir a URL de login, buscando o tipo `AuthPage`. Para as URLs de sucesso e falha, suas respectivas páginas são encontradas na propriedade *client* que ambas herdam da superclasse ResultDependency. Por exemplo, o code generator busca por um `packagedElement` com o tipo `AuthSuccessUrl` e lê a propriedade `client` para construir a URL de sucesso de autenticação.

A Listagem 3.2 apresenta um *template* que serve de entrada para a ferramenta gerar o arquivo de configuração XML para o *framework* Spring Security. O gerador de código substitui tags definidas (e.g., `FW_AUTH_LOGIN_PROC_URL`) com a informação extraída do modelo (e.g., a URL de processamento), como explicado acima. No *template* já é especificado que as URLs de login e de processamento de login são acessiveis para usuários não autenticados, para que a autenticação seja possível.

Listagem 3.3 – Trecho do arquivo `.frameweb` para um Modelo de Entidades.

```
1 <compose xsi:type="frameweb:EntityModel" name="Entity Model">
2 <packagedElement xsi:type="frameweb:DomainPackage" name="Entity Package">
3 <packagedElement xsi:type="frameweb:AuthUser" name="User">
4 <ownedAttribute xsi:type="frameweb:AuthUserName" name="username"/>
5 <ownedAttribute xsi:type="frameweb:AuthPassword" name="password"/>
6 <ownedAttribute xsi:type="frameweb:IdAttribute" name="user_id"/>
7 </packagedElement>
8 <packagedElement xsi:type="frameweb:AuthRole" name="Role">
9 <ownedAttribute xsi:type="frameweb:AuthRoleName" name="roleName" visibility="
      private" type="//@compose.3/Entity%20Package/String"/>
10 </packagedElement>
11 <packagedElement xsi:type="frameweb:AuthPermission" name="Permission">
12 <ownedAttribute xsi:type="frameweb:AuthPermName" name="permissionName"/>
13 </packagedElement>
14 <packagedElement xsi:type="frameweb:DomainClass" name="Person">
15 <ownedAttribute xsi:type="frameweb:IdAttribute" name="id"/>
16 <ownedAttribute xsi:type="frameweb:DomainAttribute" name="address"/>
17 <ownedAttribute xsi:type="frameweb:DomainAttribute" name="email"/>
18 </packagedElement>
19 </packagedElement>
```

Listagem 3.4 – Trecho do código XML gerado para configuração do Spring Security, pode-se ver a consulta SQL para aquisição de credenciais e permissões.

```
1 <authentication-manager>
2 <authentication-provider>
3 <jdbc-user-service
4 data-source-ref="dataSource"
5 users-by-username-query="select FW_AUTH_USER.FW_AUTHAT_USER_USERNAME,
      FW_AUTH_USER.FW_AUTHAT_USER_PASSWORD, 1 from FW_AUTH_USER where FW_AUTH_USER.
      FW_AUTHAT_USER_USERNAME=?"
6 authorities-by-username-query="SELECT DISTINCT FW_AUTH_USER.
      FW_AUTHAT_USER_USERNAME, FW_AUTH_ROLE_FW_AUTH_PERM.
      FW_AUTH_PERMs_FW_AUTHAT_PERM_NAME
7 FROM FW_AUTH_USER, FW_AUTH_USER_FW_AUTH_ROLE, FW_AUTH_ROLE_FW_AUTH_PERM
8 where FW_AUTH_USER.FW_USER_ID = FW_AUTH_USER_FW_AUTH_ROLE.FW_AUTH_USER_FW_USER_ID
       and
9 FW_AUTH_ROLE_FW_AUTH_PERM.FW_AUTH_ROLE_FW_AUTHAT_ROLE_ROLENAME =
      FW_AUTH_USER_FW_AUTH_ROLE.FW_AUTH_ROLEs_FW_AUTHAT_ROLE_ROLENAME and
      FW_AUTH_USER.FW_AUTHAT_USER_USERNAME = ?" />
10 </authentication-provider>
11 </authentication-manager>
```

A Listagem 3.3 mostra parte do arquivo `.frameweb` do modelo de entidades. Aqui a ferramenta busca pelos tipos `AuthUser`, `AuthRole` e `AuthPermission`. Dentro do tipo `AuthUser` é feita a busca pelas tags `ownedAttribute` com o tipo `AuthUserName` e `AuthPassword` para fazer a comparação com as credenciais recebidas na requisição de autenticação. `AuthRole` e `AuthPermission` e seus atributos `AuthRoleName` e `AuthPermName` serão usados para conseguir a lista de permissões relacionada a essas credenciais. A Listagem 3.4 apresenta um template para o arquivo de configuração do Spring Security em uma implementação XML, retirado de (MULARIEN, 2010), que usa a informação extraída do modelo de entidades para gerar o código do artefato.

A geração do código para o modelo de aplicação segue os mesmos princípios, as *tags* para as permissões são buscadas no arquivo do modelo e substituídas no *template* de

código. Diferentes *templates* podem ser desenvolvidos para geração de código para outros *frameworks* como Apache Shiro, JAAS, etc. A implementação descrita neste trabalho está disponível no repositório publico <https://github.com/Rodolfocostapr/CodeGenRodolfo.git>.

Com as modificações feitas ao metamodelo e ao editor FrameWeb, o arquiteto da aplicação pode agora especificar em seus modelos a página de login, as páginas de sucesso e falha de autenticação e as permissões necessárias para execução dos métodos das classes de serviço. Também pode ser descrito no modelo de entidades o padrão RBAC, caso este seja adotado para a aplicação. Com as mudanças no gerador de código o trabalho de implementação dessas configurações é simplificado.

4 Avaliação do Trabalho

Este capítulo descreve os esforços e resultados de avaliação da proposta de inclusão de suporte para *frameworks* de segurança ao FrameWeb. A Seção 4.1 descreve a avaliação quantitativa em que códigos de configuração dos *frameworks* de segurança gerados a partir dos modelos FrameWeb foram comparados com códigos originalmente desenvolvidos manualmente. A Seção 4.2 relata os resultados de uma avaliação qualitativa, feita por meio de questionários com desenvolvedores no contexto de uma disciplina de Desenvolvimento Web na universidade.

4.1 Avaliação quantitativa

Para fazer a avaliação do trabalho, foram usados projetos feitos por estudantes de graduação e pós-graduação de nossa universidade, no contexto de um curso de desenvolvimento Web, de modo a medir o quanto os modelos FrameWeb ajudam os desenvolvedores a implementar o RBAC em seus projetos. Neste capítulo, reportamos os resultados de três projetos específicos: SGAF[1] (publicação de avaliação de filmes), C2D[2] (avaliação de pesquisadores baseado em suas publicações) e S2C-VV[3] (gerenciamento de atividades religiosas).

A metodologia usada foi: (1) o nível de segurança original da aplicação foi identificado; (2) modelos foram criados usando o Editor FrameWeb; (3) os modelos foram usados para gerar código para uso do *framework* de segurança; (4) as linhas de código geradas foram comparadas com as já presentes na aplicação a fim de verificar a cobertura do código gerado; e (5) o número de ajustes feitos nó código gerado para implementar um nível mínimo de RBAC foi medido. Todas estas etapas foram executadas pelo autor deste trabalho, sem a participação dos alunos que desenvolveram as aplicações.

A Tabela 2 mostra o resultado da avaliação, com as primeiras colunas informando o nível de segurança presente na aplicação; a coluna **Auth?** indica se o projeto original implementou autenticação ou autorização, a coluna **RBAC?** informa se verdadeiro RBAC foi implementado, de acordo com a definição apresentada na Seção 2.1.

As demais colunas mostram a comparação entre o código gerado com o original. A coluna **Gen** informa o número de linhas automaticamente geradas a serem usadas na configuração do *framework* de segurança. A coluna **Not** indica quantas linhas relacionadas a funcionalidades de segurança estavam presentes nos projetos originais mas que não

[1] <https://github.com/dwws-ufes/2017-SGAF>
[2] <https://github.com/dwws-ufes/2017-C2D>
[3] <https://github.com/dwws-ufes/2017-S2C-VV>

Tabela 2 – Resultado da avaliação usando projetos de desenvolvimento Web

Project	**Auth?**	**RBAC?**	**Gen**	**Not**	**Adj**	**%**
`SGAF`	Yes / Yes	No	37	7	2	84%
`C2D`	No / Yes	No	20	7	6	74%
`S2C-VV`	No / No	No	37	4	2	90%

foram geradas automaticamente pelo gerador FrameWeb, como definições de segurança de banco de dados, configurações *hash* etc. A coluna **Adj** retrata quantas linhas foram geradas, mas precisavam de algum tipo de ajuste manual, como indicação de nome de permissão, requerido pelo *framework* nas implementações com JAAS. Finalmente, a coluna % apresenta a porcentagem de linhas de código geradas automaticamente em relação ao total requerido para implementar a segurança nesses projetos.

Esses resultados nos dizem que o FrameWeb foi capaz de gerar a maior parte do código necessário para implementação das funcionalidades de segurança baseada nas extensões feitas aos seus modelos, facilitando o trabalho dos desenvolvedores neste aspecto. Além disso, o código gerado implementa um RBAC verdadeiro, o que não estava presente em nenhum dos projetos originais.

4.2 Avaliação qualitativa

Para validar este trabalho de forma mais qualitativa, foi elaborada uma segunda avaliação por meio de um experimento para verificar o uso do Editor FrameWeb e do gerador de código, estendidos com os conceitos de segurança propostos neste trabalho, por estudantes de graduação e pós-graduação de nossa universidade. O experimento ocorreu no contexto da mesma disciplina de Desenvolvimento Web do experimento relatado anteriormente, porém desta vez os alunos utilizaram diretamente a ferramenta no início do desenvolvimento de seus projetos. Foi ministrada aos participantes uma única aula na qual foi explicado o uso do editor, para criação dos modelos, e do gerador de código. Também foi disponibilizado aos participantes um tutorial de como criar os modelos e de como utilizar o gerador de código.

Ao todo, 21 alunos participaram do experimento. A Tabela 3 resume o nível de conhecimento dos participantes nos assuntos relevantes ao experimento. Pode-se observar que a grande maioria possuía um nível de conhecimento baixo. Já na Tabela 4 pode-se ver a forma com que os participantes adquiriram tais conhecimentos.

A Tabela 5 agrega as respostas dos participantes em relação a quanto o FrameWeb ajudou na integração do código da aplicação Web aos *frameworks* utilizados. Vemos que, para a maioria dos participantes, o método e o editor FrameWeb ajudaram na utilização dos *frameworks*, sendo destacadas as atividades de **Integração entre páginas**

Tabela 3 – Nível de conhecimentos dos participantes nos assuntos relevantes ao experimento. Células vazias indicam que a opção não existia no formulário.

Assunto	**Nenhum**	**Baixo**	**Medio**	**Alto**
`Desenvolvimento Web`		15(72%)	3(14%)	3(14%)
`Frameworks Java`		17(81%)	3(14%)	1(5%)
`Método FrameWeb`	10(48%)	9(43%)	2(10%)	0(0%)
`Modelagem de Software`		13(65%)	4(20%)	3(15%)

Tabela 4 – Formas de aquisição dos conhecimentos dos participantes sobre os assuntos relevantes ao experimento: desenvolvimento Web (DW) e *frameworks* Java (FJ) para desenvolvimento Web.

Forma de aquisição	**Total DW**	**% DW**	**Total FJ**	**% FJ**
`Disciplina(s)`	9	43%	7	33%
`Curso(s)`	1	5%	0	0%
`Estágio/Trabalho`	10	48%	10	48%
`Outro(s)`	8	38%	8	38%

Tabela 5 – Resultado da pesquisa em relação ao quanto o FrameWeb auxiliou na integração de seu código aos *frameworks* usados. A: Ajudou Muito; B: Ajudou; C: Neutro; D: Ajudou Pouco; E: Não Ajudou; F: Não Implementei.

Atividade	**A**	**B**	**C**	**D**	**E**	**F**
Mapeamento objeto/relacional das classes de domínio	1 (8%)	6 (50%)	4 (33%)	1 (8%)	0 (0%)	0 (0%)
Classes DAO que realizam persistência	2 (17%)	6 (50%)	2 (17%)	1 (8%)	1 (8%)	0 (0%)
Injeção de dependência nas classes de controle e aplicação	2 (17%)	1 (8%)	6 (50%)	1 (8%)	2 (17%)	0 (0%)
Integração entre páginas Web e classes controladoras	4 (33%)	4 (33%)	2 (17%)	2 (17%)	0 (0%)	0 (0%)

Web e classes controladoras, que foi a mais facilitada pelo método e a **Injeção de dependência nas classes de controle e aplicação** a menos facilitada.

A Tabela 6 resume as respostas quanto ao uso dos modelos como forma de documentação. Os resultados também foram positivos e destacaram-se os modelos de Entidades e de Navegação como muito úteis.

Já a Tabela 7 versa sobre o uso das ferramentas na implementação de autenticação e autorização no padrão RBAC. Como vemos na tabela, os resultados do experimento não foram positivos nesta questão. Podemos ver que praticamente metade dos participantes ou

Tabela 6 – Resultado da pesquisa em relação aos modelos e quanto os participantes consideram como forma de documentação do que foi implementado. A: Muito Útil; B: Útil; C: Neutro; D: Pouco Útil; E: Nada Útil.

Modelo	**A**	**B**	**C**	**D**	**E**
`Modelo de Entidades`	7 (58%)	4 (33%)	1 (8%)	0 (8%)	0 (0%)
`Modelo de Persistência`	3 (25%)	3 (25%)	4 (33%)	1 (8%)	1 (8%)
`Modelo de Aplicação`	3 (25%)	5 (42%)	4 (33%)	0 (0%)	0 (0%)
`Modelo de Navegação`	7 (58%)	4 (33%)	1 (8%)	0 (0%)	0 (0%)

Tabela 7 – Resultado da pesquisa em relação a utilização das ferramentas para implementação dos conceitos de autenticação e autorização. A: Ajudou Muito; B:Ajudou; C:Neutro;D Ajudou Pouco; E Não Ajudou; F:Não Implementei RBAC ou não sei o que é. H:Não Usei RBAC nos Modelos

Modelo	**A**	**B**	**C**	**D**	**E**	**F**	**H**
Editor no Dev. Autenticação e Autorização	0(0%)	0(0%)	1(9%)	0(0%)	1(9%)	3 (27%)	6(55%)
Editor na Implementação do padrão RBAC	0(0%)	0(0%)	0(0%)	1(9%)	1(9%)	2 (18%)	7(64%)
Gerador de Código no Dev. da Autenticação e Autorização	0(0%)	0(0%)	0(0%)	0(0%)	2(18%)	4(36%)	5(45%)
Gerador de Código na implementação do padrão RBAC	0(0%)	0(0%)	0(0%)	1(9%)	1(9%)	4(36%)	5(45%)

não implementou autenticação e autorização ou não utilizou RBAC nos modelos FrameWeb criados. Sobre o editor, dos que o utilizaram para desenvolver os conceitos de segurança, os resultados ficaram entre: Neutro, Ajudou Pouco ou Não Ajudou. O gerador de código obteve resultados semelhantes nesses quesitos.

Pode-se ver na tabela que a maioria dos participantes não fez implementação do modelo RBAC em suas aplicações, o que limita esta avaliação do trabalho realizado.

Alguns outros fatores podem ter impactado nos resultados da avaliação qualitativa:

- Por se tratar de uma ferramenta em desenvolvimento, a usabilidade da mesma ainda não foi priorizada. Impactando na experiência de uso pelos alunos;
- Não foi explicitado aos participantes o ganho em adicionar os conceitos de RBAC aos modelos. Se fossem fornecidos modelos base, já com RBAC, para que os participantes fizessem a evolução dos mesmos existiria incentivo no uso da ferramenta;

- Para garantir a entrega do projeto para a disciplina cursada, muitos participantes priorizaram a inclusão de outras funcionalidades mais cruciais para suas aplicações do que autenticação e autorização;

- Como o número de linhas de código de segurança geradas não escala de acordo com o tamanho da aplicação, pois a maioria faz parte da configuração do *framework* de segurança, talvez os participantes acharam melhor escrever diretamente o código do que utilizar modelos e geradores de código;

- Como a ferramenta depende de *templates* de código, para um programador que pretende desenvolver uma única aplicação de maneira isolada, realmente não existe muito ganho na utilização das ferramentas, pois o ganho advém da reutilização dos *templates* em mais de um projeto;

- O controle de `Roles` e `Permissions` do RBAC só é interessante em aplicações grandes e com muitos usuários exercendo muitas atividades diferentes, o que não era o caso dos projetos desenvolvidos pelos participantes.

4.2.1 Ameaças a Validade do Estudo

De acordo com Wohlin et al. (2012), alguns fatores podem representar ameaças à validade de um estudo experimental. Apresentamos, a seguir, uma análise de tais fatores em relação aos experimentos conduzidos para avaliar as propostas deste trabalho:

- **Validade de Conclusão:** ameaças à validade de conclusão referem-se à capacidade de se conseguir a conclusão correta sobre a relação entre forma do experimento e seu resultado. Neste trabalho uma forma de ameaça à validade de conclusão relevante e que precisa ser considerada é o baixo poder estatístico, pois o número de participantes que deram *feedback* sobre o FrameWeb com autenticação e autorização foi muito pequeno. A ameaça conhecida como *"Fishing"* também é relevante. Trata-se de quando a medição é feita na busca de um resultado especifico desejável, como este trabalho foi feito por indivíduos envolvidos na construção e evolução do método FrameWeb, resultados positivos são mais satisfatórios. Sobre a heterogeneidade do grupo, esta ameaça é parcialmente remediada por se tratar de um grupo que participa de uma mesma disciplina em um curso de graduação;

- **Validade Interna:** ameaças à validade interna são fatores que possam afetar a variáveis independentes sem a ciência do pesquisador, quando não existe um grupo de controle no experimento. A ameaça à validade interna mais significativa neste trabalho é a "Mortalidade". Vários participantes do grupo inicial não voltaram para preencher o formulário de *feedback*;

- **Validade de Construto:** este tipo de validade aborda a influência dos construtos usados no experimento sobre o resultado. Uma ameaça neste sentido é a explicação inadequada das ferramentas e dos formulários. Para remediar este fator foi ministrada uma aula explicando a ferramenta e foi também disponibilizado um tutorial online;

- **Validade Externa:** ameaças à validade externa são condições que limitem a capacidade de se generalizar o resultado do experimento ao mercado de desenvolvimento de software. Neste trabalho o fato de que todos os participantes eram estudantes de graduação pode ameaçar a validade externa. Porém ve-se na Tabela 4 que um número considerável destes possui experiência profissional.

5 Considerações Finais

Neste trabalho, foi proposta uma extensão da linguagem de modelagem FrameWeb para contemplar o uso de *frameworks* de segurança. Após a fase de pesquisa foi percebido que os *frameworks* mais usados na plataforma Web são Spring Security, Apache Shiro e o padrão JAAS (Java Authentication and Authorization Services). Estes *frameworks* possuem um processo de autenticação similar, portanto foi possível estender os meta-modelos do FrameWeb para representar as estruturas básicas necessárias para o uso destes.

Quanto à autorização, foi adotado o padrão RBAC, devido à sua capacidade de capturar e manter um controle de papéis e permissões similar ao que existe na realidade de qualquer organização. Os três *frameworks* escolhidos foram capazes de representar os conceitos do padrão. Assim sendo, também foram feitas extensões ao meta-modelo FrameWeb para que o RBAC pudesse ser usado em aplicações desenvolvidas com o uso do método.

Essas modificações foram trazidas para as ferramentas já existentes relacionadas ao FrameWeb: seu editor e seu gerador de código. Foi possível então usar o editor para criar modelos com intuito de guiar o uso do *framework* de segurança, bem como gerar automaticamente, a partir dos modelos, parte do código de configuração e uso de tais *frameworks*. Desta forma, os objetivos elencados no Capítulo 1 foram completamente atendidos.

Como pontos fortes do trabalho, a comunicação da fase de *design* com a fase de implementação foi melhorada e a codificação também foi favorecida pela geração de código. Outro ganho foi a adoção do RBAC como padrão, o que permite o desenvolvimento de aplicações mais aderentes ao dinamismo do controle de acesso do mundo real.

Nas avaliações iniciais os resultados pareciam promissores. Porém, após um experimento com voluntários que testaram as ferramentas no desenvolvimento da aplicação, ficou claro que ainda é necessária uma evolução considerável, tanto no editor quanto no gerador de código.

Como trabalhos futuros, portanto, podem ser estabelecidos *templates* de modelos, já com RBAC desenhado, para que um esqueleto minimo de segurança já esteja pronto, para que o desenvolvedor evolua sobre o mesmo, tendo em vista que este código minimo realmente tende a ser bastante similar para aplicações ainda pequenas.

Existem também muitas outras funcionalidades de segurança que precisam ainda ser consideradas no método, como OAuth,[1] hierarquia de papéis, grupos de usuários, etc.

[1] https://oauth.net/

Como todos os *frameworks* considerados são na plataforma Java EE, é importante avaliar como os metamodelos e o editor FrameWeb representariam as funcionalidades de segurança em outra linguagem de desenvolvimento Web. Um estudo sistemático e a proposta de uma ontologia para *frameworks* de segurança — nos moldes, por exemplo, do que foi feito para *frameworks* de mapeamento objeto/relacional (ZANETTI; AGUIAR; SOUZA, 2019) — é indicado para garantir uma cobertura mais completa para este tipo de *framework*.

Sobre o editor FrameWeb e o gerador de código construídos neste trabalho, tais ferramenta devem ser agregada às ferramentas FrameWeb oficiais[2] para que a geração de código seja feita sem a utilização de nenhuma aplicação externa, melhorando a experiência do usuário. Após essa melhoria poderia ser feito um segundo experimento sobre a utilização do editor para desenvolvimento de uma aplicação com *frameworks* de segurança.

[2] <https://github.com/nemo-ufes/FrameWeb/wiki/ToolsTutorial01>.

Referências

ALMEIDA, N. V. de; CAMPOS, S. L.; SOUZA, V. E. S. A Model-Driven Approach for Code Generation for Web-based Information Systems Built with Frameworks. In: *Proc. of the 23rd Brazilian Symposium on Multimedia and the Web*. Gramado, RS, Brazil: ACM, 2017. p. 245–252. Citado 4 vezes nas páginas 11, 20, 23 e 30.

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 11.

BAUER, C.; KING, G. *Hibernate in Action*. 1. ed. [S.l.]: Manning, 2004. ISBN 9781932394153. Citado na página 11.

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 5 vezes nas páginas 7, 11, 19, 20 e 28.

EMIG, C. et al. An access control metamodel for web service-oriented architecture. In: *2nd International Conference on Software Engineering Advances - ICSEA 2007*. [S.l.: s.n.], 2007. ISBN 0769529372. Citado na página 23.

FERRAIOLO, D.; CUGINI, J.; KUHN, D. R. Role-based access control (RBAC): Features and motivations. *Proceedings of 11th Annual Computer Security Application Conference*, pages, p. 241–248, 1995. Disponível em: <http://brutus.ncsl.nist.gov/groups/SNS/rbac/documents/ferraiolo-cugini-kuhn-95.pdf>. Citado 5 vezes nas páginas 7, 8, 14, 15 e 17.

FOWLER, M. *Inversion of Control Containers and the Dependency Injection pattern, http://www.martinfowler.com/articles/injection.html (last access: September 29th, 2016)*. 2004. Citado na página 11.

JüRJENS, J. *UMLsec: Extending UML for Secure Systems Development*. 2004. Citado na página 23.

LODDERSTEDT, T.; BASIN, D. A.; DOSER, J. Secureuml: A uml-based modeling language for model-driven security. In: *Proceedings of the 5th International Conference on The Unified Modeling Language*. London, UK, UK: Springer-Verlag, 2002. (UML '02), p. 426–441. ISBN 3-540-44254-5. Disponível em: <http://dl.acm.org/citation.cfm?id=647246.719477>. Citado na página 23.

MARTINS, B. F.; SOUZA, V. E. S. A Model-Driven Approach for the Design of Web Information Systems based on Frameworks. In: *Proc. of the 21st Brazilian Symposium on Multimedia and the Web*. [S.l.]: ACM, 2015. p. 41–48. Citado 4 vezes nas páginas 19, 20, 23 e 25.

MULARIEN, P. *Spring Security 3: Secure your web applications against malicious intruders with this easy to follow practical guide*. [S.l.]: Packt Publishing Ltd, 2010. Citado na página 32.

MURUGESAN, S. et al. Web engineering: A new discipline for development of web-based systems. *Web Engineering*, v. 2016, p. 3–13, 2001. Citado na página 11.

PASTOR, O. et al. Model-driven development. *Informatik-Spektrum*, v. 31, p. 394–407, 2008. Citado na página 11.

PAVLICH-MARISCAL, J.; MICHEL, L.; DEMURJIAN, S. Enhancing uml to model custom security aspects. In: *Proceedings of the 11th International Workshop on Aspect-Oriented Modeling (AOM@ AOSD'07)*. [S.l.: s.n.], 2007. Citado na página 23.

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. ISBN 9781450358675. Citado na página 13.

SOUZA, V. E. S.; FALBO, R. A. FrameWeb - A Framework-based Design Method for Web Engineering. In: *Proc. of the 2007 Euro American Conference on Telematics and Information Systems*. [S.l.]: ACM, 2007. Citado 5 vezes nas páginas 7, 11, 16, 18 e 23.

WOHLIN, C. et al. *Experimentation in software engineering*. [S.l.: s.n.], 2012. v. 9783642290442. 1–236 p. ISSN 0098-5589. ISBN 9783642290442. Citado na página 38.

ZANETTI, F. L.; AGUIAR, C. Z. d.; SOUZA, V. E. S. Representação Ontológica de Frameworks de Mapeamento Objeto/Relacional. In: *Proc. of the 12th Seminar on Ontology Research in Brazil (ONTOBRAS 2019)*. Porto Alegre, RS, Brasil: CEUR, 2019. Citado na página 41.

Apêndices

Formulário de Perfil do Participante

Pesquisa: *Securing FrameWeb*: Adição de Suporte a *Role-based Access Control* em um Método de Desenvolvimento para Engenharia Web Baseado em Frameworks. / Aplicação de técnicas de Modelagem Conceitual no desenvolvimento de métodos e ferramentas de Engenharia de Software.

Estudo de Caso: avaliação dos conceitos de segurança adicionados ao FrameWeb através de uma análise qualitativa de seu impacto no ciclo de desenvolvimento. / Avaliação da abordagem FrameWeb e de sua ferramenta CASE FrameWeb Editor no projeto e implementação de aplicações Web no contexto de uma disciplina universitária.

Questões

1) Qual seu nível de conhecimento sobre **desenvolvimento Web**?

[ ] Alto (mais de 3 anos) [ ] Médio (1 a 3 anos) [ ] Baixo (menos de 1 ano)

2) Como você adquiriu conhecimento sobre **desenvolvimento Web**? Indique uma ou mais opções.

[ ] Disciplina(s) Qual(is)? _____

_____

[ ] Curso(s) Qual(is)? _____

_____

[ ] Estágio/Trabalho

[ ] Outro(s) Qual(is)? _____

3) Qual seu nível de conhecimento sobre **frameworks para desenvolvimento Web**?

[ ] Alto (mais de 3 anos) [ ] Médio (1 a 3 anos) [ ] Baixo (menos de 1 ano)

4) Como você adquiriu conhecimento sobre **frameworks para desenvolvimento Web**? Indique uma ou mais opções.

[ ] Disciplina(s) Qual(is)? _____

_____

[ ] Curso(s) Qual(is)? _____

_____

[ ] Estágio/Trabalho

[ ] Outro(s) Qual(is)? _____

5) Quais **frameworks para desenvolvimento Web** você já utilizou em ao menos 1 projeto de desenvolvimento de software?

_____

_____

_____

6) Qual o seu nível de conhecimento sobre o **método FrameWeb**?

[ ] Alto (Já utilizei o método e o FrameWeb Editor na elaboração de modelos de sistemas)

[ ] Médio (Já conheço o método mas nunca o utilizei na elaboração de modelos de sistemas)

[ ] Baixo (Conheço apenas o que foi apresentado na aula da disciplina Desenvolvimento Web e Web Semântica sobre o método e o FrameWeb Editor)

[ ] Nenhum (Nunca tive contato com o método e nem com o FrameWeb Editor)

7) Qual seu nível de conhecimento sobre **modelagem de software** no contexto de **projeto arquitetural de sistemas**?

[ ] Alto (mais de 3 anos) [ ] Médio (1 a 3 anos) [ ] Baixo (menos de 1 ano)

8) Como você adquiriu conhecimento sobre **modelagem de software** no contexto de **projeto arquitetural de sistemas**? Indique uma ou mais opções.

[ ] Disciplina(s) Qual(is)? _____

_____

[ ] Curso(s) Qual(is)? _____

_____

[ ] Estágio/Trabalho

[ ] Outro(s) Qual(is)? _____

9) Qual das descrições abaixo se aplica a você **atualmente** (marque todas que se aplicam)?

[ ] Aluno de graduação em Ciência da Computação ou Engenharia de Computação

[ ] Aluno de graduação em outro curso de Engenharia

[ ] Aluno de Mestrado em Informática

[ ] Aluno de Doutorado em Ciência da Computação

[ ] Funcionário de empresa que trabalha com desenvolvimento de software

[ ] Estagiário de empresa que trabalha com desenvolvimento de software

[ ] Sócio de empresa que trabalha com desenvolvimento de software

[ ] Profissional liberal, atualmente trabalhando com desenvolvimento de software

Formulário de Feedback do Participante

Pesquisa: *Securing FrameWeb*: Adição de Suporte a *Role-based Access Control* em um Método de Desenvolvimento para Engenharia Web Baseado em Frameworks. / Aplicação de técnicas de Modelagem Conceitual no desenvolvimento de métodos e ferramentas de Engenharia de Software.

Estudo Experimental: avaliação dos conceitos de segurança adicionados ao FrameWeb através de uma análise qualitativa de seu impacto no ciclo de desenvolvimento. / Avaliação da abordagem FrameWeb e de sua ferramenta CASE FrameWeb Editor no projeto e implementação de aplicações Web no contexto de uma disciplina universitária.

1) Qual ferramenta você utilizou para criar os modelos FrameWeb do Trabalho 1 da disciplina de DWWS 2018/2?

[ ] FrameWeb Editor **sem** os elementos de segurança/RBAC (código-fonte disponível em https://github.com/nemo-ufes/FrameWeb/)

[ ] FrameWeb Editor **com** os elementos de segurança/RBAC (código-fonte disponível em https://github.com/Rodolfocostapr/Experimento-Frameweb-Sec/)

[ ] Ferramenta UML genérica. Justifique o motivo do FrameWeb Editor não ser utilizado: _____
_____

Sobre o Método FrameWeb em geral

2) Criar modelos FrameWeb ajudou na implementação da integração do seu código com os frameworks considerados pelo método?

a) Mapeamento objeto/relacional das classes de domínio: [ ] Não implementei

[ ] Ajudou muito [ ] Ajudou [ ] Neutro [ ] Ajudou pouco [ ] Não ajudou

b) Classes DAO que realizam persistência: [ ] Não implementei

[ ] Ajudou muito [ ] Ajudou [ ] Neutro [ ] Ajudou pouco [ ] Não ajudou

c) Injeção de dependência nas classes de controle e aplicação: [ ] Não implementei

[ ] Ajudou muito [ ] Ajudou [ ] Neutro [ ] Ajudou pouco [ ] Não ajudou

d) Integração entre páginas Web e classes controladoras: [ ] Não implementei

[ ] Ajudou muito [ ] Ajudou [ ] Neutro [ ] Ajudou pouco [ ] Não ajudou

Justifique:_____
_____

_____
_____
_____
_____
_____

3) Para cada modelo gerado a partir do uso do FrameWeb indicado abaixo, indique o quão útil você o considera como forma de documentação do que foi implementado:

a) Modelo de Entidades:

[ ] Muito útil [ ] Útil [ ] Neutro [ ] Pouco útil [ ] Nada útil

b) Modelo de Persistência:

[ ] Muito útil [ ] Útil [ ] Neutro [ ] Pouco útil [ ] Nada útil

c) Modelo de Aplicação:

[ ] Muito útil [ ] Útil [ ] Neutro [ ] Pouco útil [ ] Nada útil

d) Modelo de Navegação:

[ ] Muito útil [ ] Útil [ ] Neutro [ ] Pouco útil [ ] Nada útil

Justifique:_____
_____
_____
_____
_____
_____
_____
_____

Sobre os elementos de segurança/RBAC de FrameWeb

4) Criar modelos FrameWeb com elementos de segurança/RBAC ajudou no desenvolvimento das funcionalidades de Autenticação e Autorização da aplicação?

[ ] Ajudou muito [ ] Ajudou [ ] Neutro [ ] Ajudou pouco [ ] Não ajudou

[ ] Não implementei Autenticação e Autorização na aplicação

[ ] Não usei elementos de segurança/RBAC nos modelos FrameWeb

Justifique:_____
_____
_____
_____
_____

5) Criar modelos FrameWeb com elementos de segurança/RBAC ajudou na implementação do padrão RBAC (*Role Bases Access Control*) na aplicação?

[ ] Ajudou muito [ ] Ajudou [ ] Neutro [ ] Ajudou pouco [ ] Não ajudou

[ ] Não implementei o padrão RBAC ou não sei o que é o padrão RBAC

[ ] Não usei elementos de segurança/RBAC nos modelos FrameWeb

Justifique:_____
_____
_____
_____
_____

6) O uso do Gerador de Código auxiliou no desenvolvimento da Autenticação e Autorização da aplicação?

[ ] Ajudou muito [ ] Ajudou [ ] Neutro [ ] Ajudou pouco [ ] Não ajudou

[ ] Não implementei Autenticação e Autorização na aplicação

[ ] Não usei elementos de segurança/RBAC nos modelos FrameWeb

Justifique:_____
_____
_____
_____
_____

7) O uso do Gerador de Código auxiliou na implementação do padrão RBAC (*Role Based Access Control*) na aplicação?

[ ] Ajudou muito [ ] Ajudou [ ] Neutro [ ] Ajudou pouco [ ] Não ajudou

[ ] Não implementei o padrão RBAC ou não sei o que é o padrão RBAC

[ ] Não usei elementos de segurança/RBAC nos modelos FrameWeb

Justifique:_____
_____
_____
_____
_____

Sobre o FrameWeb Editor e o Gerador de Código em geral

8) O uso do gerador de código ajudou na implementação da integração do seu código com os frameworks considerados pelo método?

a) Mapeamento objeto/relacional das classes de domínio: [ ] Não gerei código

[ ] Ajudou muito [ ] Ajudou [ ] Neutro [ ] Ajudou pouco [ ] Não ajudou

b) Classes DAO que realizam persistência: [ ] Não gerei código

[ ] Ajudou muito [ ] Ajudou [ ] Neutro [ ] Ajudou pouco [ ] Não ajudou

c) Injeção de dependência nas classes de controle e aplicação: [ ] Não gerei código

[ ] Ajudou muito [ ] Ajudou [ ] Neutro [ ] Ajudou pouco [ ] Não ajudou

d) Integração entre páginas Web e classes controladoras: [ ] Não gerei código

[ ] Ajudou muito [ ] Ajudou [ ] Neutro [ ] Ajudou pouco [ ] Não ajudou

Justifique:_____
_____
_____
_____
_____
_____
_____
_____

9) Considerando uma escala que vai de 0 (zero, nenhuma capacidade) a 10 (dez, total capacidade), como você avalia o FrameWeb Editor quanto a (responda apenas se usou alguma das duas versões do FrameWeb Editor):

5) Como você avalia entre Ótimo, Bom, mediano, Ruim ou Inexistente o Editor Frameweb quanto a Maturidade (Capacidade do produto de software de evitar falhas decorrentes de defeitos no software)?

[ ] Ótimo [ ] Bom [ ]Mediano [ ]Ruim [ ]Inexistente

6) Como você avalia entre Ótimo, Bom, mediano, Ruim ou Inexistente o Editor Frameweb quanto a Tolerância a falhas (Capacidade do produto de software de manter um nível de desempenho especificado em casos de defeitos no software ou de violação de sua interface especificada)?

[ ] Ótimo [ ] Bom [ ]Mediano [ ]Ruim [ ]Inexistente

7) Como você avalia entre Ótimo, Bom, mediano, Ruim ou Inexistente o Editor Frameweb quanto a Recuperabilidade (Capacidade do produto de software de restabelecer seu nível de desempenho especificado e recuperar os dados diretamente afetados no caso de uma falha.)?

[ ] Ótimo [ ] Bom [ ]Mediano [ ]Ruim [ ]Inexistente

8) Como você avalia entre Ótimo, Bom, mediano, Ruim ou Inexistente o Editor Frameweb quanto a Inteligibilidade (Capacidade do produto de software de possibilitar ao usuário compreender se o software é apropriado e como ele pode ser usado para tarefas e condições de uso específicas.)?

[ ] Ótimo [ ] Bom [ ]Mediano [ ]Ruim [ ]Inexistente

9) Como você avalia entre Ótimo, Bom, mediano, Ruim ou Inexistente o Editor Frameweb quanto a Apreensibilidade (Capacidade do produto de software de possibilitar ao usuário aprender sua aplicação.)?

[ ] Ótimo [ ] Bom [ ]Mediano [ ]Ruim [ ]Inexistente

10) Como você avalia entre Ótimo, Bom, mediano, Ruim ou Inexistente o Editor Frameweb quanto a Operacionalidade (Capacidade do produto de software de possibilitar ao usuário operá-lo e controlá-lo.)?

[ ] Ótimo [ ] Bom [ ]Mediano [ ]Ruim [ ]Inexistente

11) Como você avalia entre Ótimo, Bom, mediano, Ruim ou Inexistente o Editor Frameweb quanto ao Comportamento em relação ao tempo (Capacidade do produto de software de fornecer tempos de resposta e de processamento, além de taxas de transferência, apropriados, quando o software executa suas funções, sob condições estabelecidas.)?

[ ] Ótimo [ ] Bom [ ]Mediano [ ]Ruim [ ]Inexistente

12) Como você avalia entre Ótimo, Bom, mediano, Ruim ou Inexistente o Editor Frameweb quanto a Utilização de recursos (Capacidade do produto de software de usar tipos e quantidades apropriados de recursos, quando o software executa suas funções sob condições estabelecidas.)?

[ ] Ótimo [ ] Bom [ ]Mediano [ ]Ruim [ ]Inexistente

13) Como você avalia entre Ótimo, Bom, mediano, Ruim ou Inexistente o Gerador de Código quanto a Maturidade (Capacidade do produto de software de evitar falhas decorrentes de defeitos no software)?

[ ] Ótimo [ ] Bom [ ]Mediano [ ]Ruim [ ]Inexistente

14) Como você avalia entre Ótimo, Bom, mediano, Ruim ou Inexistente o Gerador de Código quanto a Tolerância a falhas (Capacidade do produto de software de manter um nível de desempenho especificado em casos de defeitos no software ou de violação de sua interface especificada)?

[ ] Ótimo [ ] Bom [ ]Mediano [ ]Ruim [ ]Inexistente

15) Como você avalia entre Ótimo, Bom, mediano, Ruim ou Inexistente o Gerador de Código quanto a Recuperabilidade (Capacidade do produto de software de restabelecer seu nível de desempenho especificado e recuperar os dados diretamente afetados no caso de uma falha.)?

[ ] Ótimo [ ] Bom [ ]Mediano [ ]Ruim [ ]Inexistente

16) Como você avalia entre Ótimo, Bom, mediano, Ruim ou Inexistente o Gerador de Código quanto a Inteligibilidade (Capacidade do produto de software de possibilitar ao usuário compreender se o software é apropriado e como ele pode ser usado para tarefas e condições de uso específicas.)?

[ ] Ótimo [ ] Bom [ ]Mediano [ ]Ruim [ ]Inexistente

17) Como você avalia entre Ótimo, Bom, mediano, Ruim ou Inexistente o Gerador de Código quanto a Apreensibilidade (Capacidade do produto de software de possibilitar ao usuário aprender sua aplicação.)?

[ ] Ótimo [ ] Bom [ ]Mediano [ ]Ruim [ ]Inexistente

18) Como você avalia entre Ótimo, Bom, mediano, Ruim ou Inexistente o Gerador de Código quanto a Operacionalidade (Capacidade do produto de software de possibilitar ao usuário operá-lo e controlá-lo.)?

[ ] Ótimo [ ] Bom [ ]Mediano [ ]Ruim [ ]Inexistente

19) Como você avalia entre Ótimo, Bom, mediano, Ruim ou Inexistente o Gerador de Código quanto ao Comportamento em relação ao tempo (Capacidade do produto de software de fornecer tempos de resposta e de processamento, além de taxas de transferência, apropriados, quando o software executa suas funções, sob condições estabelecidas.)?

[ ] Ótimo [ ] Bom [ ]Mediano [ ]Ruim [ ]Inexistente

20) Como você avalia entre Ótimo, Bom, mediano, Ruim ou Inexistente o Gerador de Código quanto a Utilização de recursos (Capacidade do produto de software de usar tipos e quantidades apropriados de recursos, quando o software executa suas funções sob condições estabelecidas.)?

[ ] Ótimo [ ] Bom [ ]Mediano [ ]Ruim [ ]Inexistente

	2. Como adquiriu conhecimento sobre desenvolvimento web?			
1. Qual seu nível de conhecimento sobre desenvolvimento web	2.1 Disciplina(s)	2.2 Curso	2.3 Estágio/Trabalho	2.4 Outro
Baixo (menos de 1 ano)				
Baixo (menos de 1 ano)	X			
Baixo (menos de 1 ano)	X			
Médio (1 a 3 anos)			X	X
Baixo (menos de 1 ano)	X		X	
Baixo (menos de 1 ano)	X		X	
Alto (Mais de 3 anos)			X	X
Alto (Mais de 3 anos)			X	X
Baixo (menos de 1 ano)				
Baixo (menos de 1 ano)	X			
Baixo (menos de 1 ano)			X	
Médio (1 a 3 anos)			X	
Baixo (menos de 1 ano)	X			
Baixo (menos de 1 ano)				X
Baixo (menos de 1 ano)				X
Baixo (menos de 1 ano)	X			
Baixo (menos de 1 ano)	X			X
Baixo (menos de 1 ano)				X
Médio (1 a 3 anos)	X		X	
Baixo (menos de 1 ano)		X	X	
Alto (Mais de 3 anos)			X	X
Total:	9	1	10	8
Alto (Mais de 3 anos)	X			
Médio (1 a 3 anos)				
Baixo (menos de 1 ano)				

	4. Como você adquiriu conhecimento sobre Frameworks Java para desenvolvi		
3 Qual seu nível de conhecimento sobre Frameworks Java para desenvolvimento web?	4.1 Disciplina	4.2 Curso	4.3 Estágio/Trabalho
Baixo (menos de 1 ano)			
Baixo (menos de 1 ano)	X		
Baixo (menos de 1 ano)			
Baixo (menos de 1 ano)			X
Baixo (menos de 1 ano)	X		X
Baixo (menos de 1 ano)	X		
Alto (Mais de 3 anos)			X
Alto (Mais de 3 anos)			X
Baixo (menos de 1 ano)			
Baixo (menos de 1 ano)	X		
Baixo (menos de 1 ano)			X
Baixo (menos de 1 ano)			X
Baixo (menos de 1 ano)	X		
Baixo (menos de 1 ano)			
Baixo (menos de 1 ano)			
Baixo (menos de 1 ano)	X		
Baixo (menos de 1 ano)			
Baixo (menos de 1 ano)			X
Médio (1 a 3 anos)	X		X
Baixo (menos de 1 ano)			X
Alto (Mais de 3 anos)			X
	7	0	10

mento web?			
4.4 Outro(s)	5 Quais frameworks para desenvolvimento Web você já utilizou em ao menos 1 projeto de desenvolvimento de software	6 Qual o seu nível de conhecimento sobre o método FrameWeb	7 Conhecimento sobre modelagem de software no contexto de projeto arquitetural de sistemas
	Nenhum	Baixo	Baixo (menos de 1 ano)
	Nenhum	Nenhum	Baixo (menos de 1 ano)
		Baixo	Baixo (menos de 1 ano)
X	Code Igniter	Médio	Médio (1 a 3 anos)
	React.js	Baixo	
	Nenhum	Nenhum	Médio (1 a 3 anos)
X	Cake php; zend	Baixo	Alto (Mais de 3 anos)
X	Grasili; servlet 2.5	Nenhum	Alto (Mais de 3 anos)
		Nenhum	Médio (1 a 3 anos)
	Laravel; JSF	Nenhum	Baixo (menos de 1 ano)
	bootstrap; code igniter; jquery	Nenhum	Baixo (menos de 1 ano)
	Laravel	Baixo	Baixo (menos de 1 ano)
	nenhum	Nenhum	Baixo (menos de 1 ano)
X	Nenhum	Nenhum	Baixo (menos de 1 ano)
	Nenhum	Nenhum	Baixo (menos de 1 ano)
	jsf	Baixo	Baixo (menos de 1 ano)
X	spring	Baixo	Baixo (menos de 1 ano)
X	nenhum	Nenhum	Baixo (menos de 1 ano)
	Laravel hibernate ORM JPA	Médio	Médio (1 a 3 anos)
X	Hibernate; Spring Jquery	Baixo	Baixo (menos de 1 ano)
X	laravel	Baixo	Alto (Mais de 3 anos)
8			
		Alto	
		Médio	
		Baixo	
		Nenhum	

Conhecimento sobre modelagem de software Curso				Qual das descrições ab			
8.1 Disciplinas	8.2 Curso	8.3 Estágio/Trbalho	8.4 Outro(s)	9.1 Aluno de graduação em Ciência da Computação ou Engenharia de Computação	9.2 Aluno de graduação em outro curso de Engenharia	9.3 Aluno de Mestrado em Informática	9.4 Aluno de Doutorado em Ciência da Computação
					X		
X	X						
X						X	
X				X			
					X		
X		X		X			
		X				X	
		X	X			X	
X				X			
X				X			
X				X			
		X		X			
X				X			
X				X			
					X		
X					X		
X				X			
X				X			
X						X	
X				X			
X		X				X	
15	1	5	1	11	4	5	0

aixo se aplica a você atualmente (marque todas que se aplicam)?			
9.5 Funcionário de empresa que trabalha com desenvolvimento de software	9.6 Estagiário de empresa que trabalha com desenvolvimento de software	9.7 Sócio de empresa que trabalha com desenvolvimento de software	9.8 Profissional liberal, atualmente trabalhando com desenvolvimento de software
			X
			X
	X		
	X		
	X		
		X	
0	3	1	2

Para cada modelo gerado a partir d

Criar modelos FrameWeb ajudou na implementação da integração do seu código com os frameworks considerados pelo método?

Colunas2	Colunas3	Colunas4	Colunas5	Colunas6	Colunas7
1 Qual ferramenta você utilizou para criar os modelos FrameWeb do Trabalho 1 da disciplina de DWWS 2018/2?	2a Mapeamento objeto/relacional das classes de domínio	2b Classes DAO que realizam persistência	2c Injeção de dependência nas classes de controle e aplicação	2d Integração entre páginas Web e classes controladoras	3a Modelo de Entidades
Com RBAC	Neutro	Ajudou	Ajudou	Ajudou Muito	Útil
Com RBAC	Ajudou Pouco	Ajudou	Não Ajudou	Ajudou	Neutro
Sem RBAC	Neutro	Neutro	Neutro	Ajudou Pouco	Útil
Sem RBAC	Neutro	Neutro	Neutro	Neutro	Muito Útil
Sem RBAC	Ajudou	Ajudou Muito	Neutro	Ajudou Muito	Muito Útil
Sem RBAC	Ajudou	Ajudou Muito	Neutro	Ajudou Muito	Muito Útil
Com RBAC	Neutro	Ajudou	Ajudou Muito	Ajudou Muito	Muito Útil
Sem RBAC	Ajudou	Ajudou	Neutro	Ajudou	Muito Útil
Com RBAC	Ajudou	Ajudou	Ajudou Muito	Ajudou	Útil
Com RBAC	Ajudou Muito	Ajudou Pouco	Ajudou Pouco	Ajudou	Muito Útil
Sem RBAC	Ajudou	Ajudou	Neutro	Neutro	Útil
Com RBAC	Ajudou	Não Ajudou	Não Ajudou	Ajudou Pouco	Muito Útil

Sem RBAC
Com RBAC
Outro Editor

Ajudou Muito
Ajudou
Neutro
Ajudou Pouco
Não Ajudou
Não Implementei

Muito Útil
Útil
Neutro
Pouco Útil
Nada Útil

o uso do FrameWeb indicado abaixo, indique o quão útil você o considera como forma de documentação do que foi implementado

Colunas8	Colunas9	Colunas10	Colunas11	Colunas12	Colunas13
3b Modelo de Persistência	3c Modelo de Aplicação	3d Modelo de Navegação	4 Criar modelos FrameWeb com elementos de segurança/RBAC ajudou no desenvolvimento das funcionalidades de Autenticação e Autorização da aplicação?	5 Criar modelos FrameWeb com elementos de segurança/RBAC ajudou na implementação do padrão RBAC (Role Bases Access Control) na aplicação?	6 O uso do Gerador de Código auxiliou no desenvolvimento da Autenticação e Autorização da aplicação?
Útil	Útil	Muito Útil	Não Ajudou	Não Ajudou	Não Ajudou
Útil	Útil	Muito Útil	Neutro	Ajudou Pouco	Não Ajudou
Pouco Útil	Neutro	Útil	Não Implementei RBAC	Não Usei RBAC nos Modelos	Não Implementei RBAC
Neutro	Neutro	Útil	Não Usei RBAC nos Modelos	Não Usei RBAC nos Modelos	Não Usei RBAC nos Modelos
Muito Útil	Muito Útil	Muito Útil	Não Usei RBAC nos Modelos	Não Usei RBAC nos Modelos	Não Implementei RBAC
Muito Útil	Muito Útil	Muito Útil	Não Usei RBAC nos Modelos	Não Usei RBAC nos Modelos	Não Usei RBAC nos Modelos
Muito Útil	Muito Útil	Muito Útil	Não Implementei RBAC	Não Implementei RBAC	Não Implementei RBAC
Neutro	Útil	Muito Útil	Não Usei RBAC nos Modelos	Não Usei RBAC nos Modelos	Não Usei RBAC nos Modelos
Neutro	Útil	Útil	Não Implementei RBAC	Não Implementei RBAC	Não Implementei RBAC
Neutro	Neutro	Muito Útil	Não Usei RBAC nos Modelos	Não Usei RBAC nos Modelos	Não Usei RBAC nos Modelos
Útil	Útil	Neutro			
Nada Útil	Neutro	Útil	Não Usei RBAC nos Modelos	Não Usei RBAC nos Modelos	Não Usei RBAC nos Modelos

Ajudou Muito
Ajudou
Neutro
Ajudou Pouco
Não Ajudou
Não Implementei RBAC
Não Usei RBAC nos Modelos

O uso do gerador de código ajudou na implementação da integração do seu código com os frameworks considerados pelo método?

Colunas14	Colunas15	Colunas16	Colunas17	Colunas18	Colunas19
7 O uso do Gerador de Código auxiliou na implementação do padrão RBAC (Role Based Access Control) na aplicação?	8a Mapeamento objeto/relacional das classes de domínio:	8b Classes DAO que realizam persistência	8c Injeção de dependência nas classes de controle e aplicação	8d Integração entre páginas Web e classes controladoras	9a Capacidade do produto de software de evitar falhas decorrentes de defeitos no software
Não Ajudou	Não Gerei Código	Não Gerei Código	Não Gerei Código	Não Gerei Código	Ruim
Ajudou Pouco	Ajudou Pouco	Não Gerei Código	Não Gerei Código	Não Gerei Código	Mediano
Não Usei RBAC nos Modelos	Não Gerei Código	Não Gerei Código	Não Gerei Código	Não Gerei Código	Ruim
Não Usei RBAC nos Modelos	Não Gerei Código	Não Gerei Código	Não Gerei Código	Não Gerei Código	Ótimo
Não Implementei RBAC	Não Gerei Código	Não Gerei Código	Não Gerei Código	Não Gerei Código	Bom
Não Implementei RBAC	Não Gerei Código	Não Gerei Código	Não Gerei Código	Não Gerei Código	Bom
Não Implementei RBAC	Neutro	Neutro	Neutro	Neutro	Mediano
Não Usei RBAC nos Modelos	Não Gerei Código	Não Gerei Código	Não Gerei Código	Não Gerei Código	Bom
Não Implementei RBAC	Ajudou	Ajudou	Ajudou Muito	Ajudou Muito	Bom
Não Usei RBAC nos Modelos	Não Gerei Código	Não Gerei Código	Não Gerei Código	Não Gerei Código	Mediano
					Bom
Não Usei RBAC nos Modelos	Não Gerei Código	Não Gerei Código	Não Gerei Código	Não Gerei Código	Mediano

Ajudou Muito
Ajudou
Neutro
Ajudou Pouco
Não Ajudou
Não Gerei Código

Ótimo
Bom
Mediano
Ruim
Inexistente

Considerando uma escala que vai de 0 (zero, nenhuma capacidade) a 10 (dez, total capacidade), como você avalia o FrameWeb Editor quanto a (responda apenas se usou alguma das duas versões do FrameWeb Editor):

Colunas20	Colunas21	Colunas22	Colunas23	Colunas24	Colunas25
9b Capacidade do produto de software de manter um nível de desempenho especificado em casos de defeitos no software ou de violação de sua interface especificada	9c Capacidade do produto de software de restabelecer seu nível de desempenho especificado e recuperar os dados diretamente afetados no caso de uma falha	9d Capacidade do produto de software de possibilitar ao usuário compreender se o software é apropriado e como ele pode ser usado para tarefas e condições de uso específicas	9e Capacidade do produto de software de possibilitar ao usuário aprender sua aplicação	9f Capacidade do produto de software de possibilitar ao usuário operá-lo e controlá-lo.	9g Capacidade do produto de software de fornecer tempos de resposta e de processamento, além de taxas de transferência, apropriados, quando o software executa suas funções, sob condições estabelecidas
Ruim	Bom	Bom	Mediano	Mediano	Ótimo
Bom	Bom	Ruim	Ruim	Ruim	Mediano
Mediano	Mediano	Ruim	Mediano	Ruim	Ruim
Ótimo	Mediano	Mediano	Mediano	Ruim	Ótimo
Bom	Bom	Mediano	Ótimo	Bom	Ótimo
Bom	Bom	Mediano	Ótimo	Mediano	Ótimo
Mediano	Ótimo	Bom	Mediano	Bom	Bom
Bom	Bom	Ótimo	Ótimo	Bom	Bom
Mediano	Bom	Mediano	Bom	Bom	Ótimo
Mediano	Ruim	Ruim	Mediano	Ruim	Ótimo
Mediano	Mediano	Bom	Mediano	Bom	Mediano
Bom	Mediano	Mediano	Mediano	Mediano	Bom

Considerando uma escala que vai de 0 (zero, nenhuma capacidade) a 10 (dez, total capacidade), como você avalia o Gerador de Códi

Colunas26	Colunas27	Colunas28	Colunas29	Colunas30	Colunas31
9h Capacidade do produto de software de usar tipos e quantidades apropriados de recursos, quando o software executa suas funções sob condições estabelecidas	10a Capacidade do produto de software de evitar falhas decorrentes de defeitos no software	10b Capacidade do produto de software de manter um nível de desempenho especificado em casos de defeitos no software ou de violação de sua interface especificada	10c Capacidade do produto de software de restabelecer seu nível de desempenho especificado e recuperar os dados diretamente afetados no caso de uma falha.	10d Capacidade do produto de software de possibilitar ao usuário compreender se o software é apropriado e como ele pode ser usado para tarefas e condições de uso específicas	10e Capacidade do produto de software de possibilitar ao usuário aprender sua aplicação
Bom					
Bom	Mediano	Mediano	Mediano	Ruim	Inexistente
Mediano		Ruim	Mediano	Mediano	Bom
Bom		Ruim	Mediano	Mediano	Bom
Ótimo					
Ótimo					
Ótimo					
Ótimo					
Ótimo					
Bom					
Bom					
Bom					

go quanto a (responda apenas se usou Gerador de Código):

Colunas32	Colunas33	Colunas34
10f Capacidade do produto de software de possibilitar ao usuário operá-lo e controlá-lo	10g Capacidade do produto de software de fornecer tempos de resposta e de processamento, além de taxas de transferência, apropriados, quando o software executa suas funções, sob condições estabelecidas	10h Capacidade do produto de software de usar tipos e quantidades apropriados de recursos, quando o software executa suas funções sob condições estabelecidas
Ruim	Bom	Mediano
Mediano	Mediano	Mediano
Mediano	Mediano	Mediano