Support for Single Page Application Frameworks on FrameWeb

Pedro Henrique Brunoro Hoppe
Federal University of Espírito Santo (UFES)
Vitória, Brazil
pedrohoppe@gmail.com

Vítor Estêvão Silva Souza
Federal University of Espírito Santo (UFES)
Vitória, Brazil
vitor.souza@ufes.br

ABSTRACT

In the field of Web Engineering, many methods have been proposed to guide developers in designing and coding Web applications. The FrameWeb method is a model-driven approach that targets the development of systems that use certain kinds of frameworks in their architecture, proposing the use of models that incorporate concepts from these frameworks during design. Currently, the FrameWeb method does not consider SPA (Single Page Application) frameworks and, in recent years, they have gained a lot of popularity among developers. In this work, we propose to add support for SPA frameworks to FrameWeb. With our research, we have managed to update the FrameWeb meta-model so that its modeling language now supports SPA frameworks and their constructs. FrameWeb tools (graphical editor and code generator) also evolved to support the new elements. Experiments of modeling existing SPAs with this new version of FrameWeb, generating code from the models and comparing with the original, showed that, in average, around 69% of the HTML tags could be generated from the models. The support for SPA frameworks in FrameWeb allows developers to design and model their applications using constructs that relate to the frameworks used in practice, facilitating developer communication using the models and generating code to improve developer productivity.

CCS CONCEPTS

• **Software and its engineering** → **Software design engineering**; **Model-driven software engineering**; **Frameworks**; **Source code generation**; **Domain specific languages**; **Object oriented frameworks**; **System description languages**; • **Information systems** → *Web applications*.

KEYWORDS

Web Engineering, MDD, Software Engineering, WIS Frameworks, DSL, FrameWeb, Reuse, method, language, tools, SPA Frameworks

ACM Reference Format:
Pedro Henrique Brunoro Hoppe and Vítor Estêvão Silva Souza. 2023. Support for Single Page Application Frameworks on FrameWeb. In *Proceedings of the Brazilian Symposium on Multimedia and the Web (WebMedia '23), October 23–27, 2023, Ribeirão Preto, Brazil.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3617023.3617059

WebMedia '23, October 23–27, 2023, Ribeirão Preto, Brazil

ACM ISBN 979-8-4007-0908-1/23/10…$15.00
https://doi.org/10.1145/3617023.3617059

1 INTRODUÇÃO

Desde o início da Engenharia Web, muitos métodos que auxiliam a construção de aplicativos para a plataforma Web foram propostos. O método FrameWeb é um destes métodos. Ele auxilia no desenvolvimento de sistemas de informação Web, ou WISs (em inglês, *Web-based Information Systems*), que utilizam certas categorias de *frameworks* em sua infraestrutura (e.g., controladores frontais, mapeamento objeto/relacional, injeção de dependência) [28].

Para cada categoria de *framework* considerada pelo FrameWeb, o método propõe certos construtos em sua linguagem de modelagem [17], utilizados por seu editor gráfico [6] e gerador de código [7] para prover um esqueleto do WIS a partir de modelos baseados na UML, seguindo uma arquitetura padrão proposta pelo método, que incorpora as arquiteturas propostas por tais *frameworks*. O método visa auxiliar na comunicação entre os desenvolvedores, documentação da arquitetura do sistema e aumento da produtividade (e.g., por meio de geração de código).

Uma categoria de *frameworks* que se tornou estado-da-prática é a dos *frameworks* SPA (*Single Page Aplication* ou Aplicação de Página Única). Tais *frameworks* auxiliam na construção de aplicações Web cujas funcionalidades estão concentradas em uma única página ao invés de carregar uma nova página a cada operação. Gmail e Facebook são exemplos populares de aplicações SPA. Estes *frameworks* se concentram na parte conhecida como *front-end* da aplicação Web, ou seja, as páginas Web apresentadas como interface com o usuário (que executam do lado do cliente), solicitando serviços do *back-end* (lógica de negócio que executa do lado do servidor), por meio de chamadas HTTP, comumente seguindo o modelo de arquitetura REST (*Representational State Transfer*, ou Transferência Representacional de Estado).

Até o momento, o método FrameWeb não considerou a existência desta nova categoria de *frameworks* [28]. Isso possivelmente limita a aplicação do método, pois no caso do time de desenvolvimento escolher um *framework* SPA para implementação de um projeto de software, provavelmente não conseguirá representar os elementos arquiteturais desta categoria de *frameworks* nos modelos propostos pelo método, abandonando seu uso ou tentando adaptá-lo com soluções improvisadas e sem suporte de suas ferramentas.

Por esse motivo, este trabalho tem por objetivo levantar os elementos arquiteturais desta categoria de *frameworks* de modo que possam serem também representados pela linguagem de modelagem do método FrameWeb, analisar o nível de suporte que o método FrameWeb já proporciona a alguns destes elementos (na versão atual do método), propor uma evolução do método (novos construtos e ferramentas) de modo que possa dar suporte a *frameworks* SPA e avaliar tal proposta por meio de experimentos de modelagem de SPAs existentes. Com isso, espera-se melhorar o método FrameWeb, que visa a longo prazo dar suporte a um amplo conjunto de *frameworks* utilizados no desenvolvimento de

WISs [28], por meio de uma linguagem única capaz de modelar os conceitos independentemente do *framework* específico sendo utilizado e, assim, melhorar desenvolvimento e manutenção de *software*, como preconiza a abordagem MDD (*Model Driven Development* ou Desenvolvimento Orientado a Modelos) [5, 21].

O artigo é dividido da seguinte forma: a Seção 2 apresenta a fundamentação teórica deste trabalho. A Seção 3 mostra o resultado do estudo dos *frameworks* SPA e as propostas de melhorias sugeridas ao método, tanto em seu metamodelo como no editor gráfico e gerador de código. A Seção 4 mostra o resultado da avaliação das propostas sugeridas e implementadas no método FrameWeb. Por fim, as seções 5 e 6 mostram os trabalhos relacionados propostos nos últimos anos e as conclusões deste trabalho, respectivamente.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção apresentamos os conceitos básicos do Desenvolvimento Orientado a Modelos, o método FrameWeb e os *frameworks SPA*.

2.1 Desenvolvimento Orientado a Modelos

O desenvolvimento orientado a modelos (em inglês, *MDD – Model Driven Development*) é uma prática de desenvolvimento de *software* em que, ao invés de o desenvolvedor especificar tudo que um determinado programa deve fazer por meio de uma linguagem de programação, ele modela as funcionalidades necessárias e a arquitetura geral que o sistema deve ter [1] visando diminuir o *gap* entre o planejamento (i.e., a arquitetura) e o desenvolvimento de *software* (i.e., a codificação) [21]. A característica definidora do MDD é que o foco principal e os produtos do desenvolvimento de *software* são modelos, e não os códigos em si. Obviamente, se os modelos acabam meramente como documentação, eles são de valor limitado, porque documentação diverge muito facilmente da realidade. Consequentemente, uma premissa chave por trás do MDD é os programas serem gerados automaticamente de seus modelos correspondentes [26].

O MDD ganhou bastante atenção tanto da indústria quanto da comunidade acadêmica devido ao seu potencial em aumentar a produtividade e qualidade do desenvolvimento de *software* [17]. No processo do MDD, modelos são especificados com uma sintaxe abstrata clara e regras bem definidas para a sua interpretação. Isso facilita o processamento pelas máquinas, o que por consequência permite a criação de ferramentas para modelagem, validação, transformação, etc. [17].

Conforme guia do *Object Management Group* (OMG),[1] modelo de sistema é uma descrição ou especificação de sua funcionalidade. Ele é geralmente composto por uma combinação de diagramas e textos, que podem ser escritos em uma linguagem de modelagem gráfica ou textual [16]. A OMG propõe uma infraestrutura que exemplifica a estrutura tradicional do MDD em que consiste em uma hierarquia de níveis de modelos, cada um (exceto o topo) sendo caracterizado como uma instância do nível acima. O nível mais baixo, M0, contém os dados do usuário, i.e., os dados que o *software* foi projetado para manipular. O próximo nível, M1, é dito que contém um modelo dos dados de usuário de M0. O nível M2 contém um modelo das informações de M1. Por se tratar de um modelo de um modelo, ele é comumente chamado metamodelo [1].

[1] https://www.omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf

Os modelos de um sistema de informação Web (*Web-based Information Systems* ou WIS) são definidos por uma linguagem gráfica que representa os elementos que compõem esse sistema. Essa linguagem utilizada por si só também é um modelo e, seguindo a ideia de hierarquia da OMG, o modelo que descreve o que cada componente da linguagem representa é o metamodelo da linguagem [1, 16], que podem também ser usados para definir novas linguagens [5].

"No nível do paradigma do MDD, pode-se trabalhar com linguagens para criação de modelos ou metamodelos, execução de transformações e geração de código. Porém, as linguagens podem e devem ser usadas para formalizar domínios. Nesse sentido o MDD pode ser usado na prática para o desenvolvimento de Linguagens Específicas de Domínio (*Domain Specific Languages* – DSL)" [16].

Linguagens Específicas de Domínio são linguagens de programação ou linguagens de especificação que visam um domínio de problema específico. Elas não são projetadas para providenciar recursos para resolver todo tipo de problema. Para esse caso usam-se Linguagens de Proposito Geral (*General Purpose Languages* – GPL) como as linguagens C e Java, por exemplo. Porém, por serem específicas para um determinado problema, será mais fácil e rápido utilizar a DSL do que uma GPL para resolver este problema [2].

Neste contexto, o método FrameWeb propõe uma DSL para modelar sistemas Web baseados em *frameworks*, como descrito a seguir.

2.2 Método FrameWeb

Reuso é algo comum no estado da prática do desenvolvimento de *software* como, por exemplo, o uso de bibliotecas, APIs, padrões de projeto e arquiteturais, etc. [12] Uma prática que se destaca neste contexto é o uso de *frameworks* que ajudam a evitar a contínua redescoberta e reinvenção de padrões arquiteturais básicos e componentes, reduzindo o custo e melhorando a qualidade de *software* por meio do uso de arquiteturas e *design* comprovados [17, 28].

Apesar de sua popularidade no uso do desenvolvimento de *software*, até recentemente nenhum método de Engenharia Web e linguagens de modelagem propostos na literatura considerou a existência de tais *frameworks* antes da fase de codificação do processo de *software*. Dado o quanto esses *frameworks* afetam a arquitetura de sistemas Web, isso motivou a proposta do *Framework-based Design Method for Web Engineering*, o FrameWeb [28].

O método FrameWeb tem por objetivo auxiliar desenvolvedores na construção de Sistemas de Informação baseados na Web (WISs), tirando proveito da fundação arquitetural oferecida por tais *frameworks* [28]. Ao longo do seu desenvolvimento, o método vem oferecendo suporte a diferentes categorias de *framework* e atualmente considera as seguintes categorias: controladores frontais, mapeamento objeto/relacional, injeção de dependência e segurança (autenticação e autorização). Tais *frameworks* levaram à definição de uma linguagem de modelagem [17] que propõe a construção de quatro tipos de modelos arquiteturais [28]: **Modelo de Entidades** (classes do domínio do problema e seus mapeamentos objeto/relacionais), **Modelo de Persistência** (classes responsáveis pela persistência das classes de domínio), **Modelo de Aplicação** (classes que implementam as lógicas de negócio do WIS e as relações de dependência entre outras camadas do projeto) e **Modelo de Navegação** (elementos que compõem a camada de apresentação, como páginas

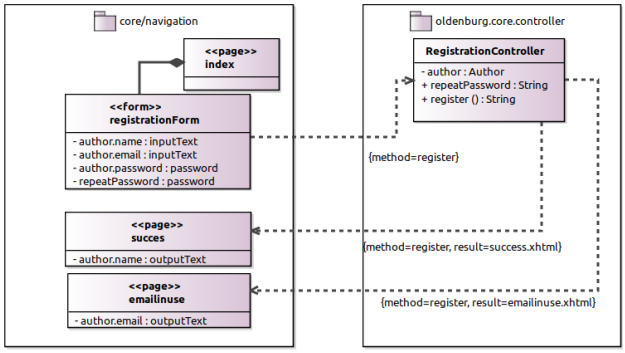

Figura 1: Exemplo de Modelo de Navegação [28].

HTML, *forms*, etc., além das classes controladoras responsáveis por gerenciar os elementos gráficos e a navegação entre eles).

O metamodelo do método FrameWeb é baseado no metamodelo UML. A partir disso, o projetista do *software* desenvolve todo o projeto por meio dos modelos mencionados acima (todos baseados no diagrama de classes), sendo cada um responsável por uma parte do *software*. Após terminada a modelagem, utilizando um editor gráfico [6], o código base do projeto pode ser gerado por meio de um gerador de código [7], ambos seguindo a abordagem MDD.

Devido ao escopo deste trabalho, nos limitaremos a mostrar como exemplo e de forma sucinta o Modelo de Navegação. No exemplo ilustrado pela Figura 1, temos a funcionalidade de cadastro de autores para um sistema de submissão de artigos. As páginas Web são armazenadas no caminho core/registration/, sendo que a página index contém o formulário registrationForm com os componentes inputText e os campos de senha (todos vem da biblioteca *PrimeFaces*,[2] utilizada neste projeto). Assim que o formulário é submetido, o controlador frontal copia os campos do formulário para os atributos de RegistrationController (observe que os campos com o prefixo author. se referem aos atributos do objeto author de RegistrationController ao qual são copiados os respectivos valores do formulário) e o método register() é invocado. Dependendo do resultado do processamento do método, o usuário será encaminhado à página de success ou emailinuse, as quais necessitam que o controlador frontal forneça dados de volta à *view* (author.name e author.email respectivamente).

A Figura 2 mostra um fragmento do metamodelo do Modelo de Navegação do FrameWeb. Os elementos do modelo da Figura 1 são instâncias das metaclasses presentes no metamodelo. Por exemplo, index é instância de Page, RegistrationController é instância de FrontControllerClass e assim por diante. O metamodelo define a linguagem do FrameWeb, restringindo o que pode ou não ser usado nos modelos e guiando a construção de ferramentas como editor gráfico [6] e gerador de código [7].

O metamodelo do FrameWeb foi implementado com a ferramenta Eclipse Modeling Framework (EMF)[3] e seu editor gráfico com a ferramenta Sirius,[4] que utiliza o metamodelo como base para definir seus elementos gráficos. Para o gerador de código é utilizado o

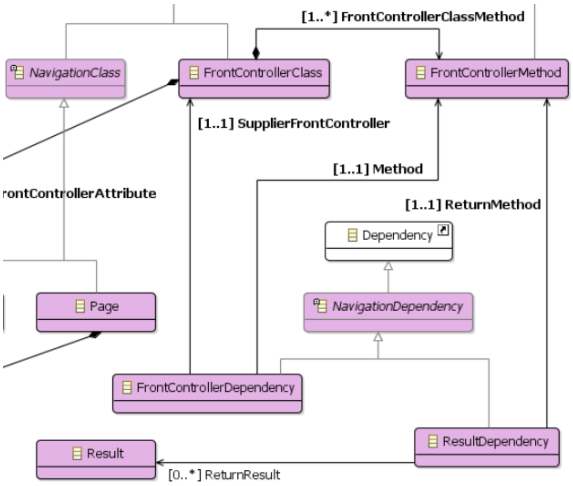

Figura 2: Fragmento do Metamodelo de Navegação [17].

template engine JTwig.[5] As implementações dos *templates* são separadas por linguagem de programação e *framework*, assim, conforme preconiza o método FrameWeb, uma linguagem única (DSL) é utilizada para implementar qualquer WIS e a geração de código é que se encarrega de gerar o código base para linguagens e *frameworks* específicos. A definição da linguagem e as ferramentas associadas estão disponíveis em um repositório de código-fonte aberto.[6]

2.3 Frameworks SPA

Um Aplicativo de Página Única (*Single Page Application*), ou SPA, é uma aplicação Web executada por meio de uma única página Web. Diferentemente de uma aplicação baseada no padrão MVC, numa SPA a lógica de apresentação é transferida do servidor para o cliente, não havendo portanto necessidade de requisições para atualização de páginas ao servidor, já que isso passa a ser gerido na página HTML por meio de código JavaScript. Desta forma, o desenvolvimento do *front-end* pode ser feito independentemente do *back-end*, interagindo com o servidor apenas por uma interface provida geralmente por meio do protocolo REST, o que permite uma maior flexibilidade no desenvolvimento [25].

Por auxiliarem no desenvolvimento deste tipo de aplicação, os *frameworks* SPA são hoje uma categoria de *frameworks* muito popular na indústria. No entanto, FrameWeb não considera atualmente tal nova categoria de *frameworks* [28].

Portanto, foi realizado um estudo com três *frameworks* populares dessa categoria, a saber: Angular, React e Vue.js,[7] por meio do desenvolvimento de pequenos projetos com cada um desses *frameworks*, a fim de capturar suas características comuns, i.e., os elementos arquiteturais que fazem parte da estrutura deste tipo de *framework*. Com isso, objetivou-se analisar o nível de suporte que o método FrameWeb já proporcionaria a alguns destes elementos e propor uma evolução do método (novos construtos e ferramentas) de modo que possa dar suporte a *frameworks* SPA.

[2]https://www.primefaces.org/
[3]https://www.eclipse.org/modeling/emf/
[4]https://www.eclipse.org/sirius/
[5]https://github.com/jtwig/jtwig
[6]https://github.com/nemo-ufes/FrameWeb
[7]https://angular.io/, https://reactjs.org/, https://vuejs.org/

As escolhas desses *frameworks* foram baseadas em sua popularidade de uso no mercado, de acordo com a pesquisa de 2021 realizada pelo popular site *StackOverflow*[8] com seus usuários sobre, dentre outras questões, as tecnologias (linguagens de programação, bancos de dados, *frameworks*, etc.) mais utilizadas por eles. Os três *frameworks* SPA escolhidos estão entre os 5 mais utilizados no quesito *Web frameworks*. O motivo de os *frameworks* *JQuery* e *Express*, respectivamente em segundo e terceiro lugares neste quesito, não terem sido incluídos é o fato de não se encaixarem na definição de *framework* SPA: o primeiro é um utilitário para codificação em JavaScript, enquanto o segundo é um *framework* usado no desenvolvimento do *back-end* das aplicações.

3 PROPOSTA

Nesta seção detalhamos nossa proposta para adicionar suporte aos *frameworks* SPA ao método FrameWeb.

3.1 Análise: FrameWeb e os *Frameworks* SPA

Como já mencionado, dentre os diagramas propostos por FrameWeb, este trabalho focou no Modelo de Navegação, visto que é este o modelo que se propõe a representar a camada de Apresentação de um WIS, contendo os elementos do *front-end* construídos com o auxílio dos *frameworks* SPA. Desde sua proposta original [27], este modelo propõe o uso de estereótipos da UML nas classes do diagrama (conforme exemplificado na Figura 1) para representar diferentes elementos que compõem a arquitetura dos *frameworks* MVC tradicionais (Controladores Frontais), conforme descreve a Tabela 1. Associações entre estes diferentes elementos representam as interações entre eles, conforme descreve a Tabela 2.

Após o estudo dos *frameworks* Angular, React e Vue.js observou-se que a principal característica destes *frameworks* é a presença de um elemento denominado *component*,[9] que desempenha tanto a função de *controller* (controlador) quanto de *view* (visão). Tal elemento não é encontrado nos *frameworks* MVC tradicionais e, portanto, não foi incluído na proposta original de FrameWeb. Além disso, constatou-se que estes *frameworks* de fato se "preocupam" apenas com a parte do *front-end*, mais especificamente com o controlador, a visão e a interação entre eles, que é feita por meio dos *components*.

A Figura 3 mostra um exemplo de *component* implementado em Vue.js. Ele é parte de uma aplicação de criação de notas de texto (semelhante ao sistema de notas autoadesivas do Windows). A parte da visão é definida pela *tag* <template />, que define em seu conteúdo elementos HTML (e.g., e <button />) e referências a outros *components* (e.g., <lv-nova-nota /> e <lv-lista-notas />). Já a parte controladora é definida pela *tag* <script />, que em seu conteúdo traz código JavaScript que define o nome do *component* (lv-painel), bem como os diferentes aspectos de controle, como outros *components* utilizados, métodos definidos neste *component*, etc. Nota-se, também, a interação entre visão e controle, por exemplo

```
<template>
  <span>Painel logado com {{ $store.state.usuario.nome }}</span>
  <button @click="sair">Sair</button>
  <hr />
  <lv-nova-nota></lv-nova-nota>
  <lv-lista-notas></lv-lista-notas>
</template>

<script>
import LvNovaNota from "./LvNovaNota.vue";
import LvListaNotas from './LvListaNotas.vue'

export default {
  name: "lv-painel",
  components: {
    LvNovaNota,
    LvListaNotas
  },
  methods: {
    sair() {
      this.$store.dispatch("logarUsuario", { id: 0, nome: "", email: "" });
      this.$router.replace("/");
    },
  },
};
</script>
```

Figura 3: Exemplo de *component* do *framework* Vue.js.

por meio da interpolação {{ $store.state.usuario.nome }}, que preenche o conteúdo da *tag* com o nome do usuário ou o atributo @click="sair" da *tag* <button />, que aciona o método sair() definido no controlador ao se clicar no botão contido na visão.

Para efetivarem consultas ou modificações nos dados do sistema de informação Web, os *components* realizam chamadas por meio do protocolo HTTP, comumente seguindo o modelo de arquitetura REST, a serviços oferecidos pelo *back-end*. Isso é feito pela parte controladora de um *component*, intermediado pelo *framework* SPA, a um serviço que encontra-se fora do escopo desta categoria de *frameworks* (inclusive podem ser utilizados outros *frameworks* para implementação do *back-end*, como o já citado *Express*, ou até mesmo uma plataforma e linguagem de programação diferentes da utilizada no *front-end*). No exemplo da Figura 3, o método sair() chama o serviço logarUsuario, do *back-end*, passando alguns parâmetros como parte de sua lógica de controle.

Os demais *frameworks* SPA possuem construtos e funcionamento similares, o que nos levou à conclusão, a partir deste estudo, da necessidade de se propor alterações na linguagem de modelagem de FrameWeb para permitir o projeto de aplicações que utilizem este tipo de *framework*.

3.2 Alterações FrameWeb

Esta seção elenca as diversas propostas de alteração para o Modelo de Navegação de FrameWeb para inclusão de suporte aos *frameworks* SPA. As propostas serão ilustradas pelo exemplo da Figura 4, que modela a aplicação de notas de texto à qual o *component* exemplificado na Figura 3 faz parte.

Alteração 01: inclusão do novo estereótipo UML «partial» que pode ser usado em uma classe do modelo, para que esta represente um "pedaço" de página HTML que consiste da parte visão de um *component* de uma aplicação SPA. Classes com estereótipo «partial» (doravante denominadas *partials*) podem estar associadas a classes sem estereótipo (definidas a seguir), classes com estereótipo «form» (formulários HTML — funcionando da mesma forma que a associação entre páginas e formulários proposta originalmente) e outras classes «partial» (*partials* que se referem a *components* importados e reutilizados — vide Alteração 04). A Figura 4 ilustra

[8] https://insights.stackoverflow.com/survey/2021, o ano de referência é 2021, pois foi neste ano iniciada a nossa pesquisa.

[9] O termo *component* ou componente é usado em muitos contextos, com diferentes significados, na Engenharia de Software. Optamos, no entanto, por manter o termo utilizado na prática pelos usuários dos *frameworks* SPA, utilizando sempre sua grafia em inglês – *component* – para descrever, ao longo do artigo, este novo elemento arquitetural trazido por esta nova categoria de *frameworks*.

Tabela 1: Estereótipos UML que podem ser usados no Modelo de Navegação, segundo proposta original do FrameWeb [27].

Estereótipo	O que representa
(nenhum)	Uma classe de ação, para a qual o *framework Front Controller* delega a execução da ação.
«page»	Uma página Web estática ou dinâmica.
«template»	Um modelo (*template*) de uma página Web, processado por um *template engine* para produzir um documento HTML.
«form»	Um formulário HTML.
«binary»	Qualquer arquivo binário que pode ser exibido pelo navegador Internet (imagens, relatórios, documentos etc.).

Tabela 2: Associações do Modelo de Navegação e o que representam, segundo proposta original do FrameWeb [27].

De	Para	O que representa
Página/modelo	Classe de ação	Um link entre a página/modelo e a classe de ação. Quando o link é seguido, a ação é executada.
Formulário	Classe de ação	Os dados do formulário são enviados à classe de ação para processamento.
Classe de ação	Página/modelo	A página ou modelo são exibidos como resultado da execução de uma determinada ação.
Classe de ação	Arquivo binário	Um arquivo binário é exibido como resultado da execução de uma determinada ação.
Classe de ação	Classe de ação	Uma outra classe de ação é executada como resultado da execução de uma primeira.

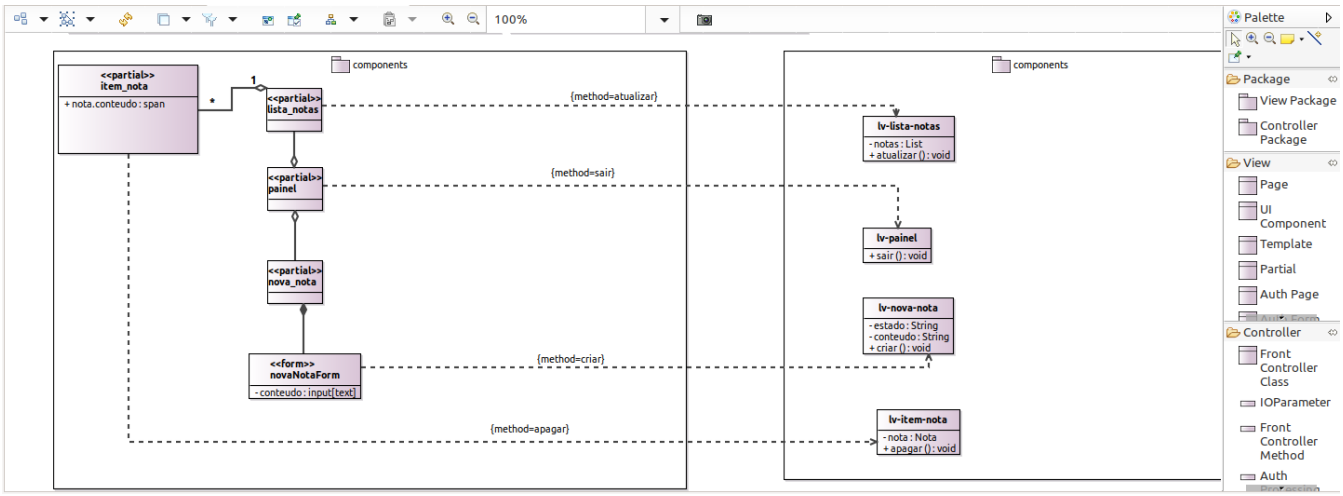

Figura 4: Exemplo de modelo de navegação com suporte aos *frameworks* SPA já incorporado ao Editor FrameWeb.

três *partials*, lista_notas, painel e nova_nota, e suas associações com outros componentes.

Alteração 02: classes sem estereótipo, que anteriormente representavam classes controladoras (cf. Tabela 1[10]) dos *frameworks* MVC tradicionais (Controladores Frontais), agora representam, quando utilizados os *frameworks* SPA, a parte controladora de um *component*. Desta forma, a linguagem FrameWeb se mantém a mesma independente do tipo de *framework* Web utilizado. Importante observar que o artefato de *back-end* chamado pelas partes controladoras dos *components* são representados apenas no Modelo de Aplicação de FrameWeb, o que também já ocorre quando são usados *frameworks* MVC tradicionais. As alterações aqui propostas se limitam ao Modelo de Navegação.

Uma importante observação sobre as alterações descritas acima é que um mesmo *component*, como por exemplo aquele ilustrado na Figura 3, é representado por dois elementos distintos no Modelo de Navegação: sua parte visão (*partial*) e sua parte controladora, associados entre si por meio de uma ou mais relações de dependência da UML, que representam chamadas de métodos (restrição method) do controlador por parte da visão. Na Figura 4, o *component* da Figura 3 é representado pelas classes painel e lv-painel (por convenção, o nome do *component* é usado como nome de sua parte controladora no diagrama) e a relação entre as duas representa a chamada do método sair(). Note que a representação da parte controladora de um *component* pode existir mesmo que não haja nenhuma chamada de método entre eles (neste caso, insere-se uma associação de dependência sem a definição da restrição method).

[10]Originalmente as classes de controle eram denominadas "classes de ação" pelo FrameWeb. A partir de suas primeiras evoluções, o termo "classe de controle" ou "classe controladora" passou a ser utilizado e é o termo utilizado neste artigo.

Tabela 3: Nova tabela de estereótipos para o Modelo de Navegação (atualiza a Tabela 1).

Estereótipo	O que representa
(nenhum)	Uma controladora de um *framework Front Controller* ou a parte controladora de um *component* de um *framework* SPA.
«page»	Uma página Web estática ou dinâmica.
«partial»	Parte de uma página HTML que é gerada em tempo de execução por meio de AJAX.
«form»	Um formulário HTML.
«binary»	Qualquer arquivo binário que pode ser exibido pelo navegador Internet (imagens, documentos, etc.).

Tabela 4: Nova tabela de associações para o Modelo de Navegação (atualiza a Tabela 2).

De	Para	O que representa
Página ou *partial*	Controladora	Um link (ou componente visual equivalente) entre a página ou *partial* e a controladora (classe ou parte controladora de um *component*). Quando o link é utilizado, um método da controladora é chamado.
Formulário	Controladora	Botão (ou componente visual equivalente) que, quando utilizado, envia os dados do formulário à controladora (classe ou parte controladora de um *component*) e um método da controladora é chamado.
Controladora	Página ou *partial*	A página ou *partial* é exibida como resultado da chamada de um determinado método na controladora.
Controladora	Arquivo binário	Um arquivo binário é exibido como resultado da execução da chamada de um determinado método na controladora.

Alteração 03: originalmente, quando uma página chamava um controlador de um *framework* MVC tradicional, havia uma dependência no sentido do controlador representando a chamada e outra no sentido contrário representando o retorno, i.e., para que página o usuário seria direcionado. No entanto, no caso de SPAs o mais comum, na prática, é que os controladores dos *components*, ao processar um método, retornem para o mesmo *partial*. Para esses casos, sugere-se não ser necessário fazer a associação de retorno para o *partial* original, apenas sendo necessário caso o retorno do método redirecione para outro *partial*. Na Figura 4, todas as chamadas de método retornam ao mesmo *partial*.

Ainda neste contexto, a navegabilidade entre os *components* de uma aplicação SPA pode ser feita por redirecionamento dentro da própria *partial* ou como o retorno de um método da parte controladora do *component*. A navegabilidade em *components* representa a geração, utilizando AJAX (*Asynchronous Javascript and XML*), do *partial* designado. A representação dessa navegabilidade permanece a mesma já descrita no Modelo de Navegação de FrameWeb.

Alteração 04: *components* trabalham também com a ideia de modularidade e um determinado *component* pode ser usado como parte de um *component* "maior" (ou mais complexo), que utiliza outros *components*, como "pecinhas de LEGO®" usadas para construir estruturas maiores. Por exemplo, na Figura 3 o *component* lv-painel utiliza os *components* lv-nova-nota e lv-lista. Propõe-se, portanto, o uso da relação de agregação da UML entre esses elementos. Na Figura 4, observa-se as relações de agregação que painel (*partial* de lv-painel) tem com lista_notas (*partial* de lv-lista) e nova_nota (*partial* de lv-nova-nota).

Alteração 05: por fim, com o estudo observou-se que existem bibliotecas externas aos *frameworks* com *components* já prontos para uso (por exemplo, o PrimeNG para Angular, o PrimeReact para React e o PrimeVue para Vue.js, produzidos pela mesma organização que desenvolveu o já citado PrimeFaces para JSF). Para esses casos e para o caso em que o desenvolvedor/modelador desejar abstrair o funcionamento de um determinado *component* (ou seja, colocá-lo como "caixa preta") propõe-se que tais *components* sejam representados não como classes "importadas" por meio de associações de agregação mas sim como atributos do *partial* onde serão utilizados. Por exemplo, no formulário novaNotaForm do *partial* nova_nota, o campo de texto simples (input [text]) conteudo poderia ser substituído pelo *component* Editor do PrimeVue.

3.3 Implementação

As propostas descritas anteriormente foram implementadas em FrameWeb por meio de alterações em seu metamodelo, que define sua linguagem de modelagem. Desta forma, serviram posteriormente de base para atualização das ferramentas FrameWeb (editor gráfico e gerador de código).

Como preparação para as alterações no metamodelo, foram atualizadas as tabelas 1 e 2 originais do FrameWeb. A Tabela 3 descreve o conjunto de estereótipos que podem agora ser utilizados no Modelo de Navegação, enquanto a Tabela 4 explica o que significam agora as associações entre os diferentes elementos deste modelo.

Além do suporte aos *frameworks* SPA, neste trabalho aproveitamos para atualizar estas definições relativas ao Modelo de Navegação nos seguintes sentidos: (1) foi retirado o elemento *template*, pois com o uso do método ao longo dos anos observou-se que o elemento página poderia representá-lo, sem necessidade de diferenciação no modelo; (2) o nome "Classe de Ação" usado pelo primeiro *framework* para o qual o FrameWeb proveu suporte foi substituído pelo termo "Controladora", podendo representar tanto uma classe controladora de um *framework* MVC tradicional quanto a parte controladora de um *component* de uma aplicação SPA; e (3) a inclusão dos *partials* contempla também os *frameworks* MVC tradicionais que possuam suporte a AJAX, já que os *partials* representariam trechos HTML da página principal (também HTML) geradas por requisição AJAX e os métodos operados sobre eles seriam feitos pelas classes controladoras tradicionais.

As modificações feitas no metamodelo do FrameWeb, que usa o Eclipse Modeling Framework (EMF), foram as seguintes:

- Para representar um *partial*, foi adicionada a metaclasse Partial como subclasse de Page, por se comportar exatamente como esta última, com a adição de suas particularidades no campo semântico (descritos anteriormente na Seção 3.2);
- Para representar a associação de agregação entre *partials*, foi criada a metaclasse NavigationAggregationAssociation como subclasse de Association da UML2;
- NavigationAggregationAssociation é acompanhada pelos elementos NavigationAggregationSource e NavigationAggregationTarget, que representam respectivamente o todo e a parte da agregação. Estes herdam de NavigationProperty (do metamodelo original de FrameWeb), que herda de Property, da UML2, meta-classe que define o atributo de cardinalidade da relação utilizada para a associação de agregação.

Após realizadas as alterações no metamodelo do FrameWeb, foram também implementadas as devidas modificações para o Modelo de Navegação no editor gráfico, a saber: a adição do elemento Partial como uma classe do modelo e a possibilidade de adição de atributos dispostos dentro do Partial; adição do elemento que representa a relação de agregação de dois *partials* com o símbolo característico de losango não preenchido. O resultado pode ser observado no Modelo de Navegação ilustrado na Figura 4, já apresentado anteriormente na Subseção 3.2 para ilustrar as alterações propostas.

Por fim, foram implementadas também modificações no gerador de código, de modo que fosse possível realizar uma avaliação das propostas deste trabalho, descritas a seguir. Todas as modificações encontram-se disponíveis no repositório de código do FrameWeb.

4 AVALIAÇÃO

Para avaliar o suporte aos *frameworks* SPA, realizamos um experimento em laboratório (i.e., experimentação feita pelos próprios autores da proposta com exemplos fictícios) com o objetivo de verificar: (1) se uma aplicação SPA simples pode ser devidamente modelada utilizando FrameWeb e suas ferramentas; (2) se estas ferramentas conseguem gerar código esqueleto para a aplicação modelada; e (3) quanto código, em média, consegue ser gerado automaticamente a partir do modelo, liberando o desenvolvedor de fazê-lo manualmente.

Para isso, o primeiro autor deste artigo utilizou o método FrameWeb e suas ferramentas, agora modificadas para incluir a nova categoria de *frameworks*, para modelar 7 sistemas de informação Web (WIS) implementados por desenvolvedores diferentes dos autores deste artigo e que utilizaram um dos três *frameworks* SPA mencionados na Seção 2.3. Os WISs foram implementados nos últimos anos como trabalhos de uma disciplina sobre Programação Web, oferecida anualmente a estudantes de graduação e pós-graduação em nossa universidade. Feito o modelo FrameWeb, executamos a geração de código e comparamos o total de *tags* HTML geradas com o número de *tags* codificadas manualmente nas implementações feitas pelos(as) estudantes e calculamos a razão (número de *tags* geradas divido por número de *tags* codificadas manualmente) em porcentagem da parte *view* dos *components*.

Com um total de 91 *components* modelados e gerados pelo gerador de código, obtivemos uma média de 69% e mediana de 66,7% do total de *tags* gerados pelo gerador de código. Com relação à parte *controller* do *component*, o gerador de código gera apenas as assinaturas dos métodos e dos atributos, cabendo ao programador desenvolver sua lógica. Os arquivos com os projetos, modelos e resultados estão disponíveis em https://nemo.inf.ufes.br/projetos/frameweb/.

Consideramos os resultados destes experimentos satisfatórios, pois foi possível demonstrar que o método e as ferramentas FrameWeb podem ser utilizados para modelar aplicações SPA simples e gerar uma quantidade razoável de código, considerando que toda a parte de lógica da aplicação precisa necessariamente ser implementada manualmente por um desenvolvedor. No entanto, apesar de conseguimos um resultado de 69% de *tags* gerados, são necessários novos experimentos para estimar a quantidade de esforço economizada pela geração de código.

Analisamos as ameaças à validade dos experimentos à luz do arcabouço proposto por Wohlin et al. [29], a saber:

- *Conclusion Validity*: a avaliação foi feita com base em sete projetos de desenvolvimento de aplicações SPA, todos desenvolvidos em contexto acadêmico (*low statistical power*), o experimento foi conduzido pelo próprio autor da proposta, em busca de um resultado positivo (*fishing*), e os resultados dependem da modelagem dos sistemas na linguagem FrameWeb, que pode ser diferente se repetida por outra pessoa (*reliability of treatment implementation*). Para mitigar tais ameaças, todo o material (projetos, modelagem, resultados) foi disponibilizado de forma pública para que possam ser avaliados pelo leitor interessado;
- *Construct Validity*: apenas o número de *tags* geradas a partir dos modelos FrameWeb foi medido (*mono-method bias*), e o experimento foi feito pelo próprio autor da proposta, sendo sujeito da própria avaliação que conduzia (*interaction of testing and treatment*). Para mitigar esta última ameaça, utilizamos projetos de *software* desenvolvidos por outras pessoas como base para a modelagem FrameWeb;
- *External Validity*: os projetos de *software* usados como base foram todos desenvolvidos na academia, portanto com características diferentes daqueles desenvolvidos na indústria (*interaction of setting and treatment*). No entanto, de acordo com [15], em certas condições a diferença entre experimentos com estudantes e profissionais é pequena.

5 TRABALHOS RELACIONADOS

Nos últimos anos, foram propostas diversas abordagens baseadas no Desenvolvimento Orientado a Modelos (MDD) de modo a facilitar o processo de desenvolvimento de software em diversos contextos e também com relação a *frameworks* SPA.

Em [14] os autores propõem uma ferramenta de modelagem de consumo de API's REST no contexto de aplicações SPA. Utilizam-se do arcabouço proposto pelo MDD para desenvolverem um artefato chamado Query Service que é responsável por gerar de forma automática uma "interface" única para consumo de serviços. Este Query Service é definido por um metamodelo construído pelos autores. Eles propõem que os usuários de sua ferramenta modelem sua aplicação com a DSL do Query Service e, após isso, é gerado o esqueleto de código na linguagem específica do framework SPA escolhido, cabendo ao desenvolvedor codificar as funções. Este trabalho se concentra apenas em modelar e fornecer uma interface que

consome os serviços REST na aplicação SPA, enquanto o método FrameWeb preconiza o desenvolvimento de toda a aplicação e, com as melhorias propostas neste trabalho, também as partes *controller* e *view* da parte do cliente.

Outros trabalhos [4, 11, 13] também propõem modelarem serviços RESTful Web e transformá-los em código, porém eles focam na parte do servidor ao invés do cliente. Em [9] os autores propõem um *framework* e um processo para desenvolvimento de serviços Web com a arquitetura REST. Para isso, propõem uma DSL para modelagem de aspectos do serviço e, então, processos de transformações de modelos são utilizados para transformar de um nível de modelo mais abstrato para um mais específico dependente de plataforma. Eles desenvolveram ferramentas também para darem apoio ao método proposto. Este trabalho se concentra apenas em modelar serviços Web REST, não focando em WISs.

Pando et al. [20] propõem uma ferramenta para modelagem de sistemas chamada PlantUMLGen, baseada em uma ferramenta chamada PlantUML, a qual utiliza princípios do MDD para modelar aplicativos MVC e depois o esqueleto de código é gerado para o *framework* Laravel. O objetivo dos autores é incentivar alunos de *design* de *software* a aplicarem técnicas de MDD. Este trabalho se diferencia no nosso na questão de se tratar uma ferramenta de ensino para o incentivo de alunos e não focando em produção industrial, por isso é bem limitada se comparada com a versão atual do método FrameWeb no suporte para o desenvolvimento de WIS. Além disso, não dá suporte a aplicações do tipo SPA, proposta deste trabalho. Já em [24] é proposta uma metodologia de desenvolvimento de APIs REST baseada em diagramas UML para modelagem e transformações de modelos para a linguagem de modelagem RAML, específica para modelagem de API's, focando somente nessa parte do projeto e não em um WIS completo, como em FrameWeb.

Em edições passadas do WebMedia, há propostas de abordagens e ferramentas baseadas em MDD, porém em outros contextos. Em [23], Ponciano et al. apresentam uma ferramenta baseada na ideia do MDD chamada EasyContext para desenvolvimento de regras contextuais para dispositivos móveis. Já Sousa et al. [8] apresentam uma abordagem baseada em MDD para o desenvolvimento de aplicações mulsemedia. Em [18], também se utiliza a ideia do MDD para mulsemedia. Bezerra e Souza [3] apresentam uma abordagem MDD para o desenvolvimento de UI (*User Interface*) reativa e personalizável dentro do contexto da internet das coisas. Pinto et al. [22] utilizam a ideia por trás do MDD para apresentarem uma abordagem para gerar *User Interface Prototypes* (UIPs) automaticamente a partir de especificações de requisitos Agile escritas em Concordia. Em [19], os autores propõem uma abordagem MDD para desenvolvimento de sistemas de recomendação.

Dentro do contexto do FrameWeb, em [10] propõe-se adicionar suporte aos *frameworks* de segurança por meio de controle de acesso por papel (*role*, em inglês) ao metamodelo, editor e gerador de código FrameWeb. A proposta é bastante similar à trazida neste artigo, porém trata de uma outra categoria de *frameworks*.

Com relação à geração de código provida pelo método FrameWeb, a indústria também possui soluções nessa área. A plataforma Angular possui uma ferramenta CLI (*command-line interface*)[11] que consegue gerar um esqueleto básico de um novo projeto Angular e também consegue gerar esqueletos de componentes e serviços, porém não gera métodos previamente especificados e nem atributos, algo que o FrameWeb faz. O VueJS também possui *Toolchain* chamado "create-vue"[12] que gera apenas o *scaffolding* do projeto e gerencia eventuais dependências que o projeto possa ter (como, por exemplo, adição de *plugins*). De forma semelhante o React possui o "create-react-app"[13] que também gera *scaffolding* do projeto e também gerencia eventuais dependências. O ASP.NET Core[14] também possui uma ferramenta de geração de projetos React e Angular baseados respectivamente no "create-react-app" e "Angular CLI" com a adição de fazer a interface com o ASP.NET Core utilizado como *backend*.

A principal diferença dessas ferramentas com o método FrameWeb é que este provê suporte para projeto arquitetural de *software* utilizando técnicas de MDD, possui suporte para múltiplos *frameworks*, tanto MVC quanto SPA (este último proposto neste trabalho), além de a geração de código ser mais específica, gerando assinatura de métodos, de atributos e, no caso dos SPA, das *tags* HTML especificadas no modelo de navegação. FrameWeb provê uma linguagem de modelagem unificada que pode ser adaptada para gerar código para diferentes *frameworks*, sejam eles atuais, legados ou que venham a ser propostos no futuro.

6 CONCLUSÕES

Neste trabalho foram apresentadas as melhorias propostas ao método FrameWeb de forma que este passe a englobar a categoria de *frameworks* SPA. Foram apresentadas as propostas de alteração na linguagem FrameWeb, as modificações implementadas em seu metamodelo, bem como as alterações realizadas no editor gráfico e, por fim, no gerador de código, permitindo-nos avaliar preliminarmente a factibilidade e a utilidade de nossas propostas.

Acreditamos que este trabalho contribui para a ampliação do uso do método FrameWeb e, consequentemente, para o uso do Desenvolvimento Orientado a Modelos (MDD) no contexto da Engenharia Web. O suporte aos *frameworks* SPA auxilia tanto os desenvolvedores como os projetistas na construção e manutenção de WIS que utilizem tais *frameworks*, bem como na comunicação entre as equipes (o que é preconizado pelo MDD).

Algumas limitações deste trabalho nos indicam possíveis caminhos para trabalhos futuros, como, por exemplo: (i) aprimorar o gerador de códigos de modo a incluir mais *frameworks* SPA; (ii) experimentar o método em contextos reais de produção, avaliando-o com profissionais em desenvolvimento de sistemas de informação Web; (iii) construir uma ontologia dos *frameworks* SPA de modo a estabelecer com mais precisão os elementos arquiteturais relevantes desta categoria de *frameworks* e garantir que FrameWeb proveja um suporte completo a eles; (iv) realizar mais testes de validação como, por exemplo, entrevistas com desenvolvedores de *frameworks* SPA; (v) melhorar a usabilidade das ferramentas FrameWeb para facilitar sua adoção por desenvolvedores; (vi) estudar a possibilidade de usar outra linguagem além da UML como base para a DSL do método FrameWeb com o intuito de se há melhor representatividade dos elementos.

[11] Disponível em https://angular.io/cli

[12] Disponível em https://github.com/vuejs/create-vue

[13] Disponível em https://github.com/facebook/create-react-app

[14] Disponível em https://learn.microsoft.com/pt-br/aspnet/core/client-side/spa/angular e https://learn.microsoft.com/pt-br/aspnet/core/client-side/spa/react

ACKNOWLEDGMENTS

Este trabalho contou com apoio da FAPES (T.O. 1022/2022).

REFERÊNCIAS

[1] C. Atkinson and T. Kuhne. 2003. Model-driven development: a metamodeling foundation. *IEEE Software* 20, 5 (2003), 36–41. https://doi.org/10.1109/MS.2003.1231149

[2] Lorenzo Bettini. 2016. *Implementing Domain Specific Languages with Xtext and Xtend - Second Edition* (2nd ed.). Packt Publishing.

[3] Jagni Dasa Horta Bezerra and Cidcley Teixeira de Souza. 2019. A Model-Based Approach to Generate Reactive and Customizable User Interfaces for the Web of Things. In *Proceedings of the 25th Brazillian Symposium on Multimedia and the Web* (Rio de Janeiro, Brazil) *(WebMedia '19)*. Association for Computing Machinery, New York, NY, USA, 57–60. https://doi.org/10.1145/3323503.3360631

[4] Rodrigo Bonifácio, Thiago M Castro, Ricardo Fernandes, Alisson Palmeira, and Uirá Kulesza. 2015. NeoIDL: A Domain-Specific Language for Specifying REST Services.. In *SEKE*. 613–618.

[5] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. 2017. *MDSE Principles*. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-031-02549-5

[6] Silas Louzada Campos and Vítor E. S. Souza. 2017. FrameWeb Editor: Uma Ferramenta CASE para suporte ao Método FrameWeb. In *Anais do 16º Workshop de Ferramentas e Aplicações, 23º Simpósio Brasileiro de Sistemas Multimedia e Web (WFA/WebMedia 2017)*. SBC, Gramado, RS, Brazil, 199–203.

[7] Nilber V. de Almeida, Silas L. Campos, and Vítor E. S. Souza. 2017. A Model-Driven Approach for Code Generation forWeb-based Information Systems Built with Frameworks. In *Proc. of the 23rd Brazilian Symposium on Multimedia and the Web (WebMedia 2017)*. ACM, Gramado, RS, Brazil, 245–252. https://doi.org/10.1145/3126858.3126863

[8] Marcelo Fernandes de Sousa, Raoni Kulesza, Carlos André Guimarães Ferraz, and Matheus Lima. 2019. A Generative Software Development Approach for Mulsemedia Application Domain. In *Proceedings of the 25th Brazillian Symposium on Multimedia and the Web* (Rio de Janeiro, Brazil) *(WebMedia '19)*. Association for Computing Machinery, New York, NY, USA, 29–36. https://doi.org/10.1145/3323503.3360290

[9] Amirhossein Deljouyi and Raman Ramsin. 2022. MDD4REST: Model-Driven Methodology for Developing RESTful Web Services.. In *MODELSWARD*. 93–104.

[10] Rodolfo Costa do Prado and Vítor E. S. Souza. 2018. 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)*. ACM, Salvador, BA, Brazil, 213–220. https://doi.org/10.1145/3243082.3243092

[11] Hamza Ed-Douibi, Javier Luis Cánovas Izquierdo, Abel Gómez, Massimo Tisi, and Jordi Cabot. 2016. EMF-REST: generation of RESTful APIs from models. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. 1446–1453.

[12] William B. Frakes and Kyo Kang. 2005. Software reuse research: Status and future. *IEEE Transactions on Software Engineering* 31, 7 (2005), 529–536.

[13] Florian Haupt, Dimka Karastoyanova, Frank Leymann, and Benjamin Schroth. 2014. A Model-Driven Approach for REST Compliant Services. In *2014 IEEE International Conference on Web Services*. 129–136. https://doi.org/10.1109/ICWS.2014.30

[14] Adrian Hernandez-Mendez, Niklas Scholz, and Florian Matthes. 2018. A Model-driven Approach for Generating RESTful Web Services in Single-Page Applications.. In *MODELSWARD*. 480–487.

[15] Martin Höst, Björn Regnell, and Claes Wohlin. 2000. Using Students as Subjects – A Comparative Study of Students and Professionals in Lead-Time Impact Assessment. *Empirical Software Engineering* 5 (2000), 201–214.

[16] Beatriz Franco Martins. 2016. *Evolução do Método FrameWeb para o Projeto de Sistemas de Informação Web Utilizando uma Abordagem Dirigida a Modelos*. Technical Report. Dissertação de Mestrado, Programa de Pós-Graduação em Informática, Universidade Federal do Espírito Santo, Vitória, ES, Brasil.

[17] Beatriz F. Martins and Vítor E. S. Souza. 2015. 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 (WebMedia 2015)*. ACM, Manaus, AM, Brazil, 41–48. https://doi.org/10.1145/2820426.2820439

[18] Douglas P. Mattos and Débora C. Muchaluat-Saade. 2018. MultiSEM: A Mulsemedia Model for Supporting the Development of Authoring Tools. In *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web* (Salvador, BA, Brazil) *(WebMedia '18)*. Association for Computing Machinery, New York, NY, USA, 109–116. https://doi.org/10.1145/3243082.3243114

[19] Yuri Oliveira, Leonardo Silveira, and Cidcley Souza. 2018. A Model-Driven Approach to Evolve Recommender Systems. In *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web* (Salvador, BA, Brazil) *(WebMedia '18)*. Association for Computing Machinery, New York, NY, USA, 169–172. https://doi.org/10.1145/3243082.3267457

[20] Brian Pando and Jose Castillo. 2022. PlantUMLGen: A tool for teaching Model Driven Development. In *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, 1–6.

[21] Oscar Pastor and Juan Carlos Molina. 2007. *Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling*. Springer Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-71868-0

[22] Thiago Delgado Pinto, Willian Inacio Gonçalves, and Pablo Veiga Costa. 2019. User Interface Prototype Generation from Agile Requirements Specifications Written in Concordia. In *Proceedings of the 25th Brazillian Symposium on Multimedia and the Web* (Rio de Janeiro, Brazil) *(WebMedia '19)*. Association for Computing Machinery, New York, NY, USA, 61–64. https://doi.org/10.1145/3323503.3360639

[23] Tarik Ponciano, Davi Tabosa, Windson Viana, Paulo Duarte, and Rafael Carmo. 2020. A Generative Approach for Android Sensor-Based Applications. In *Proceedings of the Brazilian Symposium on Multimedia and the Web* (São Luís, Brazil) *(WebMedia '20)*. Association for Computing Machinery, New York, NY, USA, 33–40. https://doi.org/10.1145/3428658.3430976

[24] Davide Rossi. 2016. UML-based Model-Driven REST API Development.. In *WEBIST (1)*. 194–201.

[25] Emmit Scott. 2015. *SPA Design and Architecture: Understanding Single Page Web Applications* (1st ed.). Manning Publications Co., USA.

[26] Bran Selic. 2003. The pragmatics of model-driven development. *IEEE software* 20, 5 (2003), 19–25.

[27] Vítor E. S. Souza. 2007. *FrameWeb: um Método baseado em Frameworks para o Projeto de Sistemas de Informação Web*. Master's thesis. Universidade Federal do Espírito Santo. http://portais.ufes.br/PRPPG/ext/mono.php?progpess=2032&curso=9&prog=30001013007P0

[28] Vítor E. S. Souza. 2020. The FrameWeb Approach to Web Engineering: Past, Present and Future. In *Engineering Ontologies and Ontologies for Engineering* (1 ed.), João Paulo A. Almeida and Giancarlo Guizzardi (Eds.). NEMO, Vitória, ES, Brazil, Chapter 8, 100–124. http://purl.org/nemo/celebratingfalbo

[29] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media.