UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE ENGENHARIA DE SOFTWARE

FRANCISCO GABRIEL TEIXEIRA MARINHO

JUNGLE: UM FRAMEWORK PARA A GERAÇÃO AUTOMÁTICA DE CÓDIGOS PARA APLICAÇÕES WEB

Itacoatiara – Amazonas
Julho – 2018

FRANCISCO GABRIEL TEIXEIRA MARINHO

JUNGLE: UM FRAMEWORK PARA A GERAÇÃO AUTOMÁTICA DE CÓDIGOS PARA APLICAÇÕES WEB

Monografia apresentada ao Instituto de Ciências Exatas e Tecnologia da Universidade Federal do Amazonas como parte dos requisitos necessários para a obtenção do título de Bacharel em Engenharia de Software

Orientadora: Prof. Dra. Odette Mestrinho Passos

Itacoatiara – Amazonas
Julho – 2018

FICHA CATALOGRÁFICA

Marinho, Francisco Gabriel Teixeira

M338j JUNGLE: Um Framework para a Geração Automática de Códigos para Aplicações Web / Francisco Gabriel Teixeira Marinho. 2018

91 f.: il. color; 31 cm.

Orientadora: Odette Mestrinho Passos

TCC de Graduação (Engenharia de Software) - Universidade Federal do Amazonas.

1. Desenvolvimento Dirigido por Modelos. 2. Geração Automática de Códigos. 3. Aplicações Web. 4. Arquitetura Dirigida por Modelos. I. Passos, Odette Mestrinho II. Universidade Federal do Amazonas III. Título

FRANCISCO GABRIEL TEIXEIRA MARINHO

JUNGLE: UM FRAMEWORK PARA A GERAÇÃO AUTOMÁTICA DE CÓDIGOS PARA APLICAÇÕES WEB

Monografia apresentada ao Instituto de Ciências Exatas e Tecnologia da Universidade Federal do Amazonas como parte dos requisitos necessários para a obtenção do título de Bacharel em Engenharia de Software.

Aprovado em 02 de Julho de 2018

BANCA EXAMINADORA

Profa. Dra. Odette Mestrinho Passos, Presidente
Universidade Federal do Amazonas

Prof. Lic. Alternei de Souza Brito, Membro
Universidade Federal do Amazonas

Prof. Dr. Raimundo da Silva Barreto, Membro
Universidade Federal do Amazonas

À minha família que foi fundamental para a minha formação acadêmica.

AGRADECIMENTOS

Agradeço ao Senhor Jesus Cristo e a Igreja Evangélica pela transmissão de valores e princípios que foram essenciais para o meu crescimento espiritual, pessoal e profissional durante esses cinco anos na universidade. Isso foi de suma importância para que eu pudesse enfrentar as adversidades com sabedoria, inteligência emocional e resiliência.

Agradeço aos meus pais, Zeliane Teixeira e Ricardo Marinho, primeiramente, por serem pessoas exemplares e, segundo, por me proporcionar uma base familiar inigualável, além de todo o apoio, amor, carinho e conselhos. Aos meus irmãos Ana Graziela e Davi Gustavo por serem os melhores irmãos do mundo. A minha família é a minha maior fonte de inspiração.

Agradeço ao Prof. Me. Bruno Bonifácio pela imensa humildade, generosidade e sabedoria repassadas durante o período de convivência na universidade, foi uma honra ser aprendiz de um mestre que, com um caráter, profissionalismo e didática excepcional, sempre treinou e incentivou seus alunos a lutarem pela realização de seus objetivos.

A Prof. Ma. Priscila Fernandes por ter me dado a oportunidade de mostrar o meu potencial me inserindo no meio de pessoas extraordinárias, capacitadas no mercado de trabalho e que colaboraram para a minha formação profissional.

A Prof. Dra. Odette Mestrinho Passos, minha orientadora, uma pessoa com caráter excepcional, com grande personalidade e uma habilidade extraordinária de trabalhar de maneira eficiente e eficaz, de forma dedicada e paciente. Sua orientação foi essencial para que uma ideia se tornasse realidade.

À minha namorada Evelym Vasconcelos por todo o companheirismo, amor e carinho incondicional demonstrado em todos os momentos. Seu incentivo e ajuda foram muito importantes durante os períodos mais complicados da minha trajetória acadêmica. Não posso deixar de agradecer também a sua família maravilhosa que sempre me acolheu com carinho desde o dia em que nos conhecemos.

Por fim, agradeço aos meus amigos, desde os amigos de infância, do ensino básico e da faculdade, amigos da coordenação acadêmica da UFAM os quais tive o privilégio de conhecer. Amigos do laboratório 312, os quais posso citar Christian Reis, Daniel da Silva, Christian Bacry, Daiane Oliveira, Álano Grana, Maik Elamide, Leonardo Jacauna, Rubber Rodrigues, Victor Lúcio, Elizabeth Chaves, Jean Ferreira, Alaise Guimarães e em especial, o meu grande amigo Ramon Breno, um gênio louco com quem convivo desde o meu ensino

médio no IFAM, há quase uma década. Essas pessoas me proporcionam a cada dia os melhores momentos da minha vida.

Estou sempre fazendo aquilo que não sou capaz, numa tentativa de aprender como fazê-lo.

Pablo Picasso

RESUMO

No contexto atual, é inevitável não perceber a grande demanda por softwares mais complexos, eficientes, eficazes e com desempenhos desejáveis, tornando-se uma tecnologia indispensável que ajuda a suprir as necessidades nos mais diversos ramos das atividades cotidianas. Deste modo, o Desenvolvimento Dirigido por Modelos torna-se uma estratégia viável para aumento da produtividade na criação de softwares ao utilizar a geração automática de código tendo os modelos como artefato principal do processo de desenvolvimento. Nesse sentido, o objetivo desse trabalho foi desenvolver um framework que possibilite a geração automática de códigos, a partir do Diagrama de Classe da Linguagem de Modelagem Unificada, de produtos de software direcionados para Aplicações Web. A metodologia consiste, primeiramente, em um levantamento bibliográfico para compor a fundamentação teórica e definir a proposta inicial desse trabalho. A partir disso, foi realizada a construção do software que inclui o levantamento de requisitos, a definição da documentação, assim como a implementação do software. Por fim, foi realizada uma avaliação de usabilidade e funcionalidade do framework Jungle. Os resultados dessas avaliações foram satisfatórios, de acordo com 83,3% dos participantes que concordaram totalmente que o Jungle pode auxiliar o desenvolvedor Web na construção de Aplicações Web, constatando a sua contribuição para o aumento da produtividade, portabilidade, qualidade e facilidade na construção e evolução de Aplicações Web.

Palavras-Chave: Desenvolvimento Dirigido por Modelos. Geração Automática de Códigos. Aplicações Web.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

CASE	*Computer Aided Software Engineering* (Engenharia de Software Assistido por Computador)
CIM	*Computacional Independent Model* (Modelo Computacional Independente)
CRUD	*Create, Retrieve, Update e Delete* *(*Criar, Recuperar, Atualizar e Excluir*)*
CSS	Cascading Style Sheets (Folhas de estilo em cascata)
DSLMs	*Domain Specific Modeling Languages* (Idiomas de modelagem específicos do domínio)
HTML	*HyperText Markup Language* (Linguagem de marcação de hipertexto)
IDE	*Integrated Development Environment* (Ambiente de desenvolvimento integrado)
MDA	*Model-Driven Architecture*
MDD	Model Driven Development (Desenvolvimento Dirigido por Modelo)
MDE	*Model Driven Engineering* (Arquitetura Dirigida por Modelo)
MDT	*Model Driven Testing* (Teste Dirigido por Modelo)
OMG	*Object Management Group* (Grupo de Gerenciamento de Objetos)
PHP	*PHP: Hypertext Preprocessor* (PHP: Pré-processador de hipertexto)
PIM	*Platform Independent Model* (Modelo Independente da Plataforma)
PSM	*Platform-Specific Model* (Modelo específico da plataforma)
UML	*Unified Modeling Language* (Linguagem de modelagem unificada)
W3C	*World Wide Web Consortium* (Consórcio World Wide Web)
XMI	*XML Metadata Interchange* (Intercâmbio de metadados XML)
XML	*eXtensible Markup Language* (Linguagem de Marcação extensível)

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

É notório a demanda crescente e acelerada pelo uso de software no cenário mundial, pois trata-se de uma tecnologia única, de grande importância e que supre as necessidades cotidianas das pessoas. Todavia, há tempos, jamais alguém poderia imaginar que o software seria hoje uma tecnologia indispensável no ramo da engenharia, ciência, entretenimento, atividades comerciais, entre outras (PRESSMAN e MAXIM, 2016).

Segundo Pressman e Maxim (2016), software é um conjunto de instruções que ao serem executadas fornecem características, funções e desempenho desejados. É importante frisar que o software possui categorias, como por exemplo, software de aplicação, software de sistema, software científico/de engenharia, software embarcado e software de aplicações para a web, sendo este último, de suma importância para esse trabalho.

O software está presente em todas as coisas que existem ao nosso redor (SOARES, 2015) e essas coisas nunca dependeram tanto de software como agora, em todas as facetas da atividade humana, podendo influenciar o comportamento de cada indivíduo, grupo e sociedade, causando mudanças que podem gerar impactos positivos e negativos para pessoas ou organizações (MEIRA, 2015).

Ninguém precisa ser graduado para aprender a programar um software, tanto que há tempos a disciplina de programação deveria ser assunto do ensino médio. Diferente dos anos 80 ou 90 onde o profissional deveria ter cursado, no mínimo, uma boa graduação para superar as expectativas do mercado, hoje qualquer pessoa de nível técnico ou até fundamental pode construir um software (MEIRA, 2015).

No entanto, a responsabilidade de quem projeta ou implementa um software é muito grande, diante dos impactos positivos ou negativos que o mesmo pode causar na vida de muitas pessoas. Fundamentalmente, softwares são máquinas abstratas, isso quer dizer que tomam por base máquinas físicas e as transformam em máquinas de finalidades diferentes. Deste modo, é a capacidade do software que torna possível vermos atualmente coisas incríveis que facilitam o trabalho do ser humano, como por exemplo aviões não tripulados (LEITE, 2015).

Assim sendo, algumas áreas surgem na computação com a intenção de colaborar com o desenvolvimento de software. Neste contexto, a Engenharia de Software é uma área recente que busca disponibilizar métodos, técnicas e ferramentas para a construção de software (SOARES, 2015).

Segundo Almeida (2014), o principal objetivo da Engenharia de Software é o desenvolvimento de produtos de software de qualidade. Para isso, ela possui atividades definidas e bem apresentadas para a sua construção que envolvem atividades que variam desde a sua concepção até a sua manutenção (SAMPAIO e IYODA, 2015).

Independente do indivíduo ser um arquiteto, um designer, projetista, engenheiro civil, construtor, ou qualquer outro, para que se construa algo, deve-se construir um modelo. De modo geral, na área da computação, modelos são feitos para apresentar uma descrição humana compreensível de características de um software (PRESSMAN E MAXIM, 2016 e ALMEIDA, 2014).

Segundo Neto, Costa e Oliveira (2010), há algumas décadas foram surgindo paradigmas para gerar software a partir da transformação automática de modelos em código fonte, diferenciando do modo convencional de produzir software. Deste modo, os estudos nessa área vêm crescendo muito recentemente e tem se intensificado visando transformar modelos abstratos em modelos mais concretos, denominados Desenvolvimento Dirigido por Modelos (MDD, do inglês *Model Driven Development*).

O MDD é um paradigma de desenvolvimento de software que tem como principal objetivo definir modelos como artefato principal do processo de desenvolvimento ao invés do código fonte (ALMEIDA, 2014).

Segundo Fernandes (2013), o MDD é uma abordagem onde a peça fundamental do desenvolvimento do software são modelos gráficos sendo esses, transformados em modelos de nível mais baixo para modelos de alto nível. Assim sendo, o nível de abstração nessa abordagem é bem maior, pois os modelos de software representam o domínio e a lógica do negócio em mais alto nível, e são convertidos automaticamente em códigos.

O MDD possui diversas variantes e uma delas é a Arquitetura Dirigida a Modelos (MDA, do inglês model-driven architecture), cuja característica principal é a utilização da linguagem UML para criar modelos gráficos que representem a estrutura e/ou o comportamento de um software. Ela é chamada dessa forma porque provê meios de usar

modelos de modo que se possa entender, projetar, construir e modificar o software (BUARQUE, 2009; FERNANDES 2013).

Segundo Sandri (2009) o MDA é uma abordagem padronizada e aberta que contempla padronizações de especificações do sistema, se baseando em modelos formais, independente da tecnologia utilizada na construção do software. A modelagem é formada de modelos gráficos que auxiliam o desenvolvedor a especificar, visualizar e construir a estrutura lógica e regras de interação entre partes de um sistema.

Neste trabalho foi utilizado como parâmetro de modelo gráfico a Linguagem de Modelagem Unificada (UML, do inglês *Unified Modeling Language*) que é uma linguagem criada para modelar sistemas computacionais por meio do Paradigma Orientado a Objetos. Atualmente é uma linguagem adotada mundialmente como um padrão para a fase de modelagem, principalmente na indústria da Engenharia de Software (GUEDES, 2011).

A UML não é uma linguagem de programação, ela é uma linguagem constituída de elementos gráficos que são utilizados para modelagem e que permitem representar os conceitos de paradigmas de programação orientados a objetos. Os elementos gráficos dessa linguagem permitem que se construa diagramas que representem as meias diversas perspectivas do sistema modelado (BEZERRA, 2015).

Uma outra característica importante da UML é que ela é independente de linguagens de programação ou de processo de desenvolvimento de software, ou seja, quando um sistema é devidamente modelado utilizando UML, esses modelos podem servir para qualquer tipo de linguagem de programação que será utilizado na fase de implementação do sistema, ou para qualquer forma de desenvolvimento de software que for adotada (BEZERRA, 2015).

Desse modo, este trabalho utilizou o conceito de MDD e MDA para o desenvolvimento de um framework que possibilite a transformação de modelos em alto nível de abstração em modelos mais concretos.

1.2 Justificativa

O ciclo de vida de um projeto de software possui as etapas de identificação do problema e da necessidade do usuário, estudo de viabilidade, análise e projeto de sistema, desenvolvimento, testes e manutenção (MARCZAK, 2016). Todavia, muitos problemas têm se agravado ao longo dos anos na etapa de desenvolvimento, sendo esses relacionados à

fatores como confiabilidade, manutenabilidade e principalmente a produtividade (SOMERVILLE, 2012).

Quanto ao problema mais evidente, isto é, a produtividade, é causado pela demanda reprimida, o tempo de desenvolvimento de um sistema e projetos não concluídos. Sobre a demanda, o problema ocorre devidos ao Backlog de aplicações, isto é, uma fila de espera na área de sistemas de uma empresa. Já o problema de tempo de desenvolvimento de um sistema ocorre devido a prazos não cumpridos que podem culminar em prejuízos tanto para o cliente quanto para o desenvolvedor (BUARQUE, 2009).

Para o problema de projetos não concluídos podem ter vários fatores como a falta de tempo para uma análise adequada, problemas técnicos e gerenciais, falta de recursos e inexperiência da equipe de desenvolvimento, sendo esse último evidenciados pelas dificuldades de alguns profissionais não possuírem conhecimentos suficientes para codificar. Assim sendo, uma alternativa para esses problemas seria a utilização do paradigma MDD (BUARQUE, 2009).

Segundo Sandri (2009), por se tratar de um paradigma que permite o desenvolvedor construir software desenhando as funcionalidades e recursos desejados de um sistema, ao invés de utilizar linguagens de programação, tornando “o desenho” no artefato central do processo desenvolvido ao invés do código fonte, o MDD está despertando cada vez mais a atenção da indústria e de comunidades de pesquisa.

O MDD representa um papel central na Engenharia de Software, pois nessa abordagem são evidenciados fatores como maior produtividade, portabilidade, menor custo, facilidade de evolução do software e maior qualidade, em relação à tecnologia orientada a objetos tradicional. Isso ocorre devido a ideia fundamental do MDD que é a transformação de modelos abstratos em modelos mais concretos, para que se obtenha o código do sistema. Além disso, o MDD possibilita que desde as fases iniciais de construção do software até as de manutenções futuras sejam realizados usando apenas modelos abstratos (BUARQUE, 2009).

Segundo Silva (2010), o MDD possui muitas vantagens em relação às abordagens tradicionais, dentre elas, podemos citar o aumento da legibilidade da representação visual de um sistema, consistência assegurada de componentes, especificações de informações não redundantes, geração de código automáticos, entre outras, são vantagens que incrementam a produtividade dos profissionais.

Almeida (2014) afirma que o MDD é uma opção que integra sistemas e ambientes, permitindo assim um alinhamento entre visão e negócio e visão de desenvolvimento. Assim sendo, essa abordagem ganhou uma importância muito grande ao longo dos últimos anos. Além do mais, o MDD possibilitou o surgimento de novas abordagens que se baseiam em suas práticas, dentre elas podemos citar o Teste Dirigido a Modelo (MDT, do inglês *Model Driven Testing*) que gera código de artefatos de teste de forma automática e o MDA que tem o foco nos modelos e transformações que levam a geração de código.

Assim sendo, as abordagens MDD e MDA são inter-relacionadas e elas aumentam a produtividade daqueles que possuem a tarefa de construir e manter um sistema de software (SANDRI, 2009).

Segundo Junior (2012), a engenharia de aplicações Web pode ser beneficiada pelo uso de práticas de MDD e MDA, pois o desenvolvimento nessa área é realizado geralmente de forma ágil e com publicações frequentes. Contudo, essas abordagens apresentadas pela comunidade acadêmica devem ser flexíveis para que se adaptem a esse contexto, pelo fator de apresentarem processos complexos que envolvem diversos modelos, linguagens de programação, *plug-ins* e ambientes de programação.

Segundo Neto, Costa e Oliveira (2010), estudos nessa área estão crescendo cada vez mais tendendo à construção de ferramentas capazes de transformar modelos efetivamente, alinhando objetivos dessas abordagens, elevando o nível de abstração na produção do software, trazendo ganhos com a geração automática de códigos.

Segundo Júnior (2012), a utilização de MDD no processo de desenvolvimento de software Web não é tão simples, pois automatizar a transformação entre modelos nesse tipo de aplicação não é trivial, uma vez que nesse contexto ferramentas prontas de modelos geralmente precisam de *IDEs* (ambientes de desenvolvimento integrado) e para que o desenvolvedor possa utiliza o MDD ele necessita conhecer várias linguagens.

Assim sendo, levando em consideração a popularização de serviços web e a demanda por softwares web mais complexos, a adoção do MDD e MDA podem ser benéficas para a aplicações web, assim como já são no desenvolvimento de aplicativos móveis e sistemas embarcados. Contudo, devem ser flexíveis e adaptáveis a esse contexto (JUNIOR, 2012).

1.3 Objetivos

Geral

Desenvolver um framework, denominado Jungle, que possibilite a geração automática de modelos de alto nível (diagrama de classe da UML) em modelos mais concretos (código fonte) de produtos de software direcionados para aplicações web, alinhado a boas práticas de Engenharia de Software.

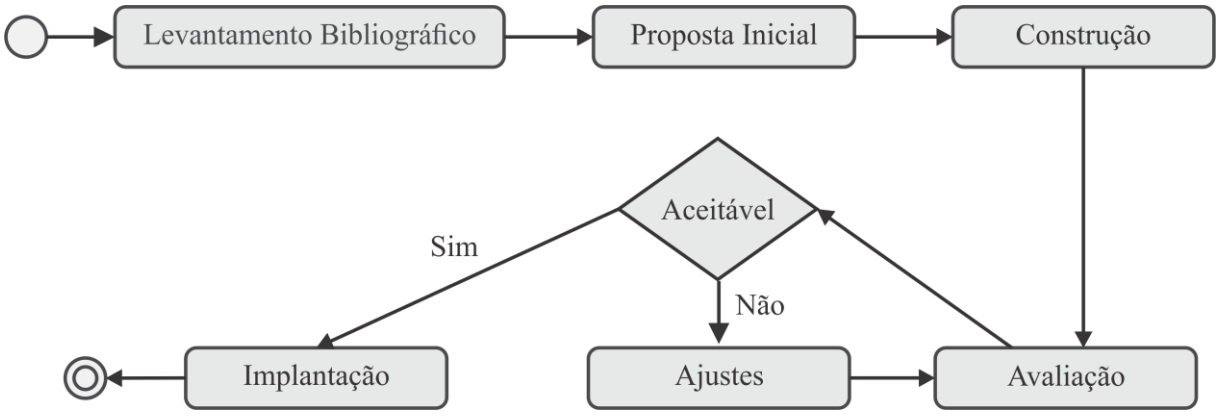
Específicos

- Estabelecer um processo de Desenvolvimento Dirigido por Modelos que propõe o uso da linguagem UML
- Construir um framework conforme a documentação gerada e realizar os testes de software.
- Avaliar a usabilidade e a funcionalidade do framework para atender as expectativas dos usuários.

1.4 Metodologia

A metodologia adotada foi dividida em seis etapas, segue o modelo apresentado abaixo na Figura 1.

Figura 1 - Fluxograma da metodologia



Fonte: O autor (2018).

Levantamento Bibliográfico: Nessa fase foi buscado os conceitos necessários para a composição da fundamentação teórica fornecendo assim uma base para o desenvolvimento do

framework. Nesta primeira fase, as pesquisas foram realizadas em livros, artigos científicos, sites e trabalhos de conclusão de curso relacionados com o tema.

Proposta Inicial: A partir dos resultados obtidos na primeira fase, foi definida a proposta do trabalho, que incluem os seguintes tópicos: MDD e MDA, linguagens de programação, Aplicações Web e UML.

Construção: Na terceira etapa do projeto foi realizada a construção do framework, baseada na modelo cascata. Seguindo essa metodologia, esta proposta seguiu as seguintes etapas.

- **Levantamento de Requisitos:** Foi realizado junto com profissionais da área da computação, por meio de entrevistas e aplicação de um questionário;
- **Modelagem:** É a representação de especificações de um sistema real por meio de diagramas (LAROZA e SEBRA, 2015), de forma a esclarecer as funcionalidades do framework, de acordo com os requisitos elicitados. Foi utilizado o software Astah Community 7.0, por intermédio da UML, para a elaboração dos seguintes diagramas: Casos de Uso, Classe, Sequência e Atividades.
- **Arquitetura:** é a descrição da estrutura ou estruturas dos sistemas, abrangendo todos os componentes de software, as propriedades externamente visíveis desse componente e as relações entre eles (PRESSMAN e MAXIM, 2016). Para modelar a arquitetura, foi utilizado a ferramenta CorelDraw X8.
- **Implementação:** Foram utilizadas algumas ferramentas como PhpStorm 2017.2 e Apache 3.2.2, além das Linguagens de Programação PHP 7.2.4 e Javascript. Os testes de software que foram realizados são: teste de sistema, teste funcional e teste unitário.

Avaliação: Foram realizadas as avaliações de Usabilidade e Funcionalidade, com desenvolvedores de software, que teve como meta verificar os critérios de usabilidade e funcionalidade do framework.

Ajustes: Após a avaliação foram feitos os ajustes finais na qual foram feitas as últimas correções para entrega definitiva do software para ser utilizados no seu referido ambiente.

Implantação: Nessa última fase, a versão final do framework foi implementada em um laboratório de pesquisa e desenvolvimento de software para ser utilizada por desenvolvedores.

1.5 Organização do Trabalho

O Capítulo 1 apresentou as principais características deste trabalho, descrevendo a contextualização, a justificativa, os objetivos e a metodologia adotada. Além desta Introdução, outros quatro Capítulos compõem o texto deste trabalho, organizados da seguinte forma:

- **Capítulo 2 – Fundamentação Teórica:** é apresentado todo o referencial teórico que fundamenta os conceitos de UML, Desenvolvimento Dirigido por Modelos, Linguagens de Programação e Aplicações Web, além dos trabalhos relacionados.
- **Capítulo 3 – Projeto do Framework Jungle:** este capítulo apresenta o levantamento de requisitos, os requisitos funcionais e não funcionais, além da modelagem dos diagramas e da arquitetura do sistema.
- **Capítulo 4 – Implementação do Framework Jungle:** este capítulo apresenta as telas do framework Jungle e da Aplicação Web gerada por ele, além do planejamento da avaliação de usabilidade e funcionalidade, bem como os resultados obtidos.
- **Capítulo 5 – Conclusão e Perspectivas Futuras:** este capítulo apresenta as considerações finais, as limitações deste trabalho e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Conceitos Relacionados

Nesta seção estão descritos os conceitos de UML, Desenvolvimento Dirigido por Modelos, Linguagens de Programação e Aplicações Web. Eles são de suma importância para a fundamentação teórica deste trabalho.

2.1.1 Linguagem Unificada de Modelagem (UML)

A Lei de Moore, declarada em 1965, é muito conhecida na computação, pois afirma que a densidade de um transistor dobra no período de 18 a 24 meses. Isso significa que há um crescimento exponencial na taxa de processamento de computadores. Desde modo, há um crescimento computacional acelerado, resultando assim em altas demandas por sistemas de software mais complexos (BEZERRA, 2015).

Segundo Thomas (2004) a modelagem pode ajudar a diminuir a complexidade de muitos problemas, tanto que é umas das técnicas mais utilizadas nos mais variados ramos do conhecimento. Desde modo, Bezerra (2015) afirma que diversas técnicas de modelagem de sistemas surgiram desde o surgimento dos primeiros computadores e tem evoluído muito até os dias atuais, visando minimizar problemas complexos do processo de desenvolvimento de software.

Várias técnicas eram utilizadas desde a década de 90 para a modelagem orientada a objetos, sendo que esse paradigma surgiu no início dessa década e ao final da mesma atingiu a sua maturidade. Durante esse período, era comum haver mais de uma técnica com diferentes notações gráficas, mesmo que estivessem descrevendo a mesma funcionalidade ou característica de um sistema (BEZERRA, 2015). A Tabela 1 mostra as principais propostas de técnicas de modelagem orientada a objetos durante a década de 90:

Tabela 1- Proposta de técnicas de modelagem orientada a objetos

Ano	Autor (Técnica)
1990	Shaler & Mellor
1991	Coad & Yourdon (OOAD- Objec-Oriented Analysis and Design)
1993	Grady Booch (Booch Method)
1993	Ivar Jacobson (OOSE - Objec-Oriented Software Engineering)
1995	James Rumbaugh et al. (OMT – Object Modeling Technique)
1996	Wirfs-Brock (Responsibility Driven Design)
1996	(Fusion)

Fonte: Bezerra (2014).

Desde modo, em 1997 a OMG (do inglês *Object Management Group*), que segundo Verner e Puia (2004) é uma organização internacional, sem fins lucrativos que visa desenvolver a excelência tecnológica, mercadologicamente viável e independente de especificações proprietárias para a indústria de software, aprovou a UML como um padrão modelagem de sistemas de software. Essa linguagem surgiu da junção dos estudos de Booch, Rumbough e Jacobson, sendo uma linguagem unificada para a modelagem orientada a objetos. Ela possui um vocabulário e um conjunto de regras próprias, que permitem uma melhor visualização do sistema, da arquitetura, dos componentes e seus tipos de relacionamentos, tendo também incorporado as melhores características de cada uma das técnicas preexistente (BEZERRA, 2015).

A UML é uma linguagem-padrão de modelagem que é utilizada na indústria da Engenharia de Software a nível internacional. Assim sendo, muitos profissionais descrevem as características de um software, tais como suas funcionalidades, sua estrutura lógica, seu comportamento, dinâmica de processo, entre outras. Essas características são descritas na fase de modelagem, ou seja, antes da fase de codificação do software (GUEDES, 2011).

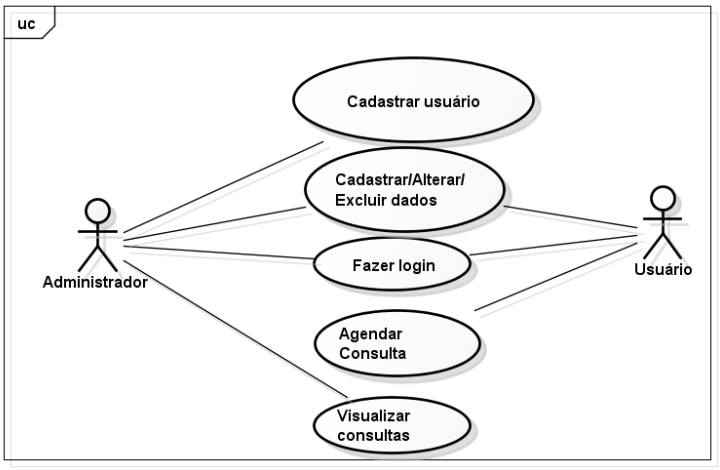
Segundo Mota, Machado e Ribeiro (2012), o que faz a UML ser a linguagem mais utilizada a cada dia que passa é a sua padronização e suas contínuas atualizações, além do mais, por intermédio de diagramas que possibilitam a modelagem de sistemas sob diversas perspectivas, é possível especificar as características e funcionalidades que o software deverá prover. Os diagramas são de fundamental importância para que todos os indivíduos envolvidos possam compreender a estrutura e o comportamento do software.

Guedes (2011) faz uma analogia afirmando que a construção de um edifício envolve diversas plantas, enfocada sob diversas formas, como layout dos andares, planta hidráulica e planta elétrica. Dessa forma, um software não é muito diferente, pois possui vários diagramas da UML que permitem analisar o sistema sob diversos aspectos, e ao final, cada diagrama complementa o outro.

Neste contexto, Guedes (2011) afirma que a versão UML 2.0 possui treze diagramas que podem representar a estrutura organizacional do sistema, o comportamento de um processo específico, a definição de um determinado algoritmo ou até mesmo as necessidades físicas de um sistema para que ele possa funcionar de maneira adequada. Veja abaixo as definições de Guedes (2011) com seus respectivos exemplos dos diagramas da UML:

Diagrama de Caso de Uso: é o diagrama mais geral e informal da UML e é utilizado, geralmente, na fase de levantamento e análise dos requisitos do sistema, mas é consultado durante todo o processo de modelagem, servindo, inclusive, de base para construção dos outros diagramas. De modo geral, ele apresenta uma linguagem simples e de fácil compreensão e é o único apresentado ao usuário para que ele possa ter uma ideia geral de como o sistema irá se comportar. Esse diagrama é composto de atores que utilizarão o software, e serviços, também conhecidos como casos de uso, isto é, as opções disponíveis do sistema para os atores, conforme exemplo apresentado na Figura 2.

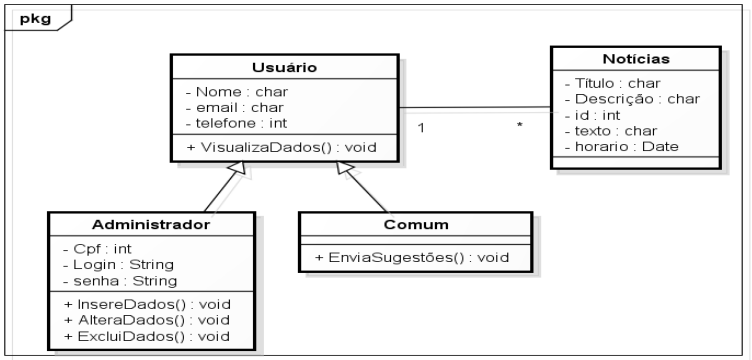
Figura 2 - Exemplo de Diagrama de caso de uso



Fonte: O autor (2018).

Diagrama de Classes: é o mais importante da UML, pois ele serve de apoio para a maioria dos outros diagramas. É nesse diagrama que é definido a estrutura das classes que serão utilizadas no sistema, definindo também os métodos e os atributos que cada classe deve possuir. Além disso, esse diagrama mostra os relacionamentos entre as classes e como trocam informações entre si. O diagrama de classe está mais detalhado na Seção 3.2.2 deste trabalho. A Figura 3 mostra um exemplo dele.

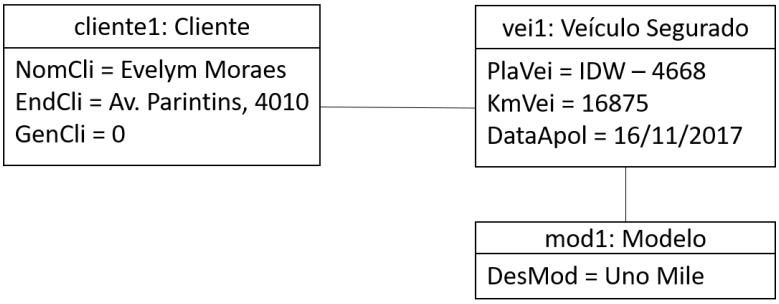
Figura 3 - Exemplo de Diagrama de Classes



Fonte: O autor (2018).

Diagrama de Objetos: é um complemento do diagrama de classes, pois mostra uma visão dos valores armazenados dos objetos do diagrama de classes em um determinado momento de execução de um processo de software, conforme exemplo apresentado na Figura 4.

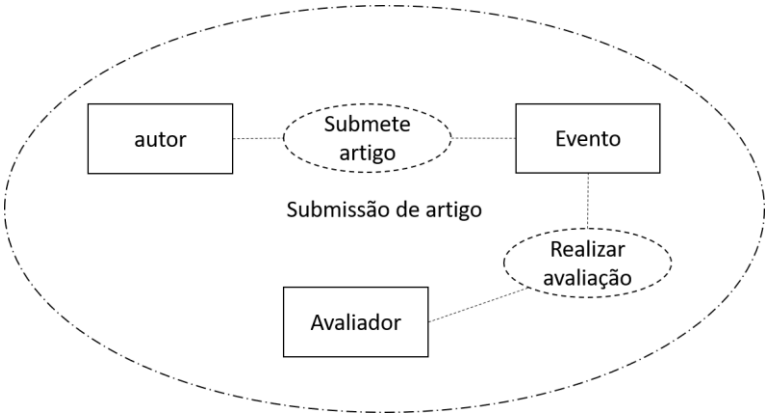
Figura 4 - Exemplo de Diagrama de objetos



Fonte: O autor (2018).

Diagrama de Estrutura Composta: esse diagrama descreve a estrutura interna de um classificador, como uma classe ou componente, além de detalhar como estas se comunicam e colaboram entre si. Além disso, ele ainda descreve uma colaboração em que um conjunto de instancias cooperam entre si para realizar uma determinada tarefa e, conforme exemplo apresentado na Figura 5.

Figura 5 - Exemplo de Diagrama de Estrutura Composta

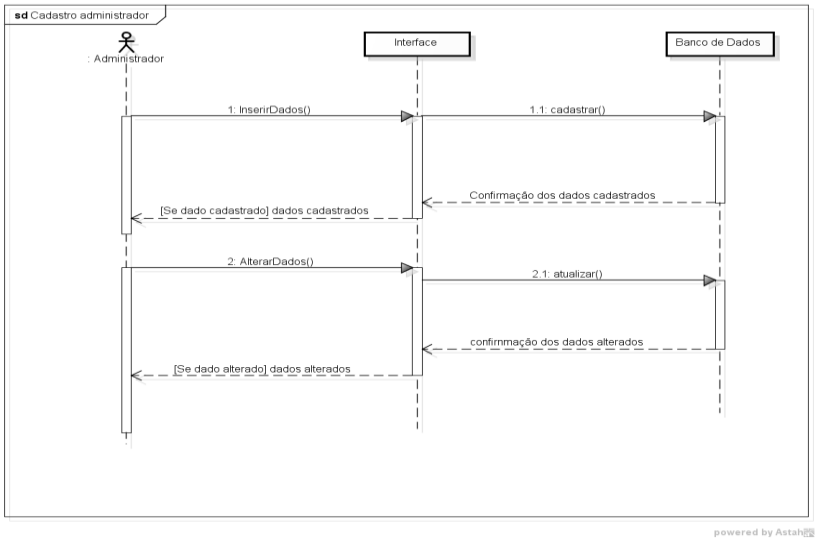


Fonte: O autor (2018).

Diagrama de Sequência: baseia-se em algum serviço definido no Diagrama de Casos de Uso e apoia-se no Diagrama de Classes para determinar os objetos das classes envolvidas em um processo. Esse diagrama se preocupa com a ordem temporal de como as mensagens são trocadas entre os objetos envolvidos em um determinado processo de software. De maneira geral, o Diagrama de Sequência identifica qual é o evento gerador do processo modelado, quem é o ator responsável por esse evento, como o processo deve se desenrolar e, por fim, ser

concluído por intermédio da chamada de métodos disparados por mensagens enviadas entre objetos, e, conforme exemplo apresentado na Figura 6.

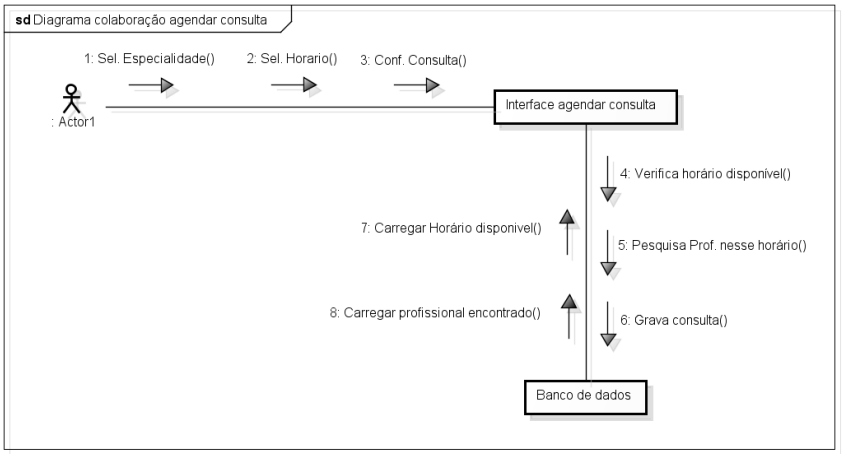
Figura 6 - Exemplo de Diagrama de Sequência



Fonte: o autor (2018).

Diagrama de Colaboração: também conhecido como Diagrama de Comunicação nas versões da UML anteriores, está amplamente associado ao Diagrama de Sequência, pois geralmente apresenta as mesmas informações desse diagrama, porém o que muda é que o Diagrama de Colaboração não se preocupa com a temporalidade do processo, ele enfatiza mais em como os objetos estão vinculados e quais são as mensagens trocadas entre eles durante o processo, e, conforme exemplo apresentado na Figura 7.

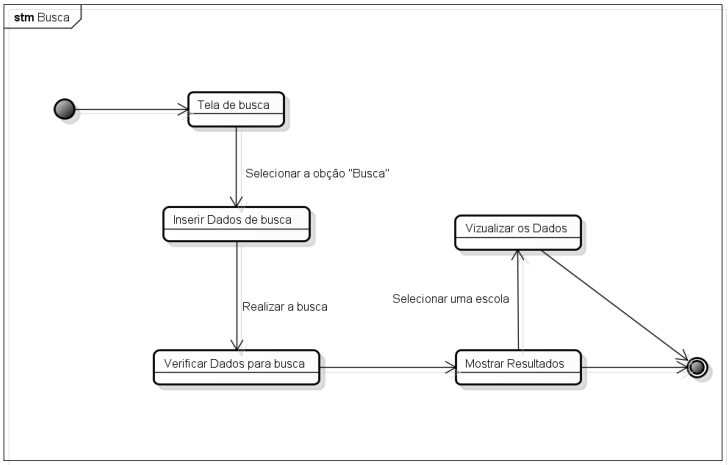
Figura 7 - Exemplo de Diagrama de Colaboração



Fonte: autor (2018).

Diagrama de Gráfico de Estados: também conhecido como Diagrama de Estados nas versões da UML anteriores, esse diagrama acompanha as mudanças sofridas por um objeto em um determinado processo. Ele acompanha os estados por que passa a instância de uma classe e pode também ser usados para representar os estados de um caso de uso em um determinado processo, conforme exemplo apresentado na Figura 8.

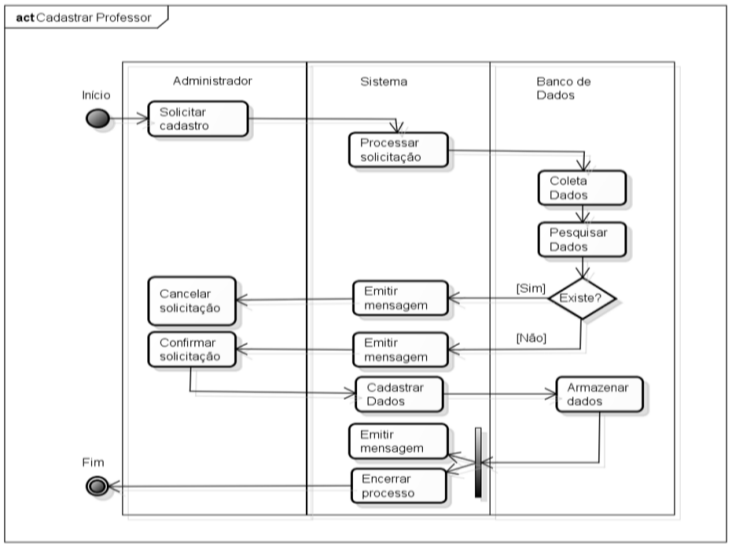
Figura 8 - Exemplo de Gráfico de Estados



Fonte: o autor (2018).

Diagrama de Atividades: preocupa-se em descrever todos os passos que devem ser percorridos para a conclusão de uma determinada atividade que, muitas vezes é representada por um método de um certo grau de complexidade. Assim sendo, o Diagrama de Atividades representa o fluxo de controle de uma atividade de um determinado processo, conforme exemplo apresentado na Figura 9.

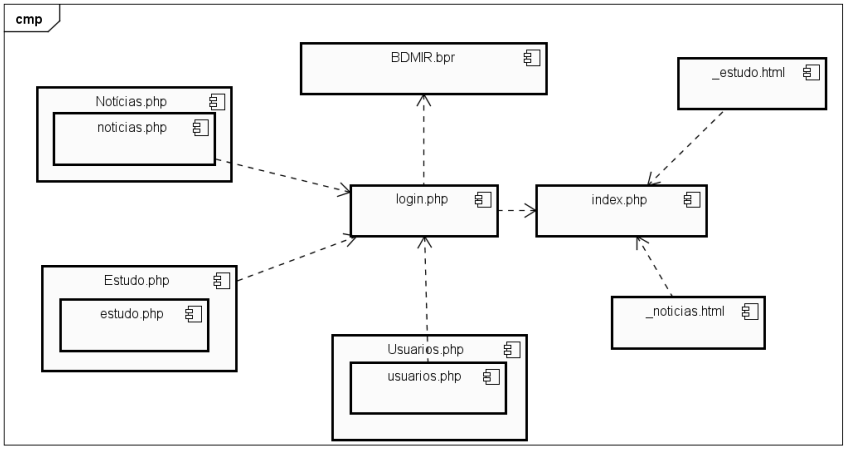
Figura 9 - Exemplo de Diagrama de Atividades



Fonte: o autor (2018).

Diagrama de Componentes: uma característica principal desse diagrama é que ele está muito relacionado às linguagens de programação que serão utilizadas para desenvolver o sistema modelado. Isso é evidente pelo fato desse diagrama representar os componentes do sistema que vai ser codificado em termos de módulos que podem ser de código-fonte, bibliotecas, formulários, entre outras. De maneira geral, o Diagrama de Componentes determina como os componentes estarão estruturados e irão interagir entre si para viabilizar o funcionamento adequado do sistema, conforme exemplo apresentado na Figura 10.

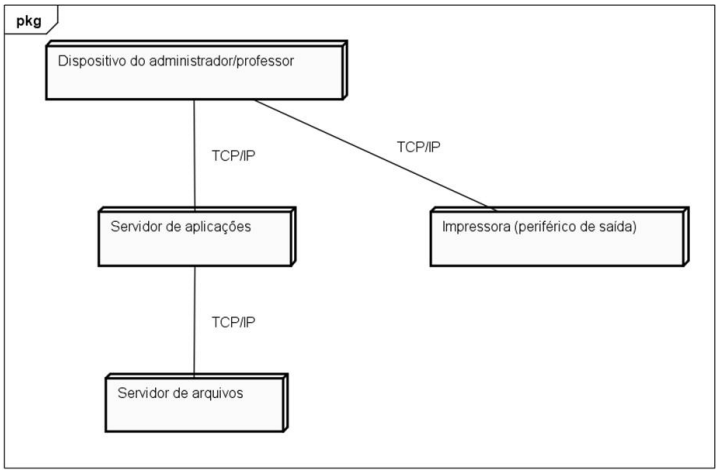
Figura 10 - Exemplo de Diagrama de Componentes



Fonte: o autor (2018).

Diagrama de Implantação: esse diagrama é utilizado para representar as características físicas, isto é, o aparato físico sobre o qual o sistema deverá funcionar. Isso inclui servidores, topologias, estações de trabalho, protocolos de comunicação, entre outros, são as necessidades de hardware do sistema modelado, conforme exemplo apresentado na Figura 11.

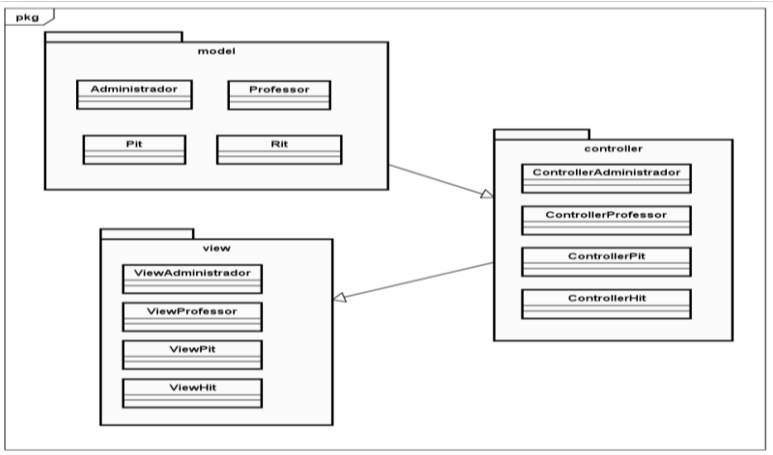
Figura 11 - Exemplo de Diagrama de Implantação



Fonte: o autor (2018).

Diagrama de Pacotes: esse diagrama representa os subsistemas englobados por um sistema de forma a determinar as partes que o compõem. Ele pode ser independente ou associado com os demais diagramas da UML, isso pode variar de acordo com o sistema modelado, conforme exemplo apresentado na Figura 12.

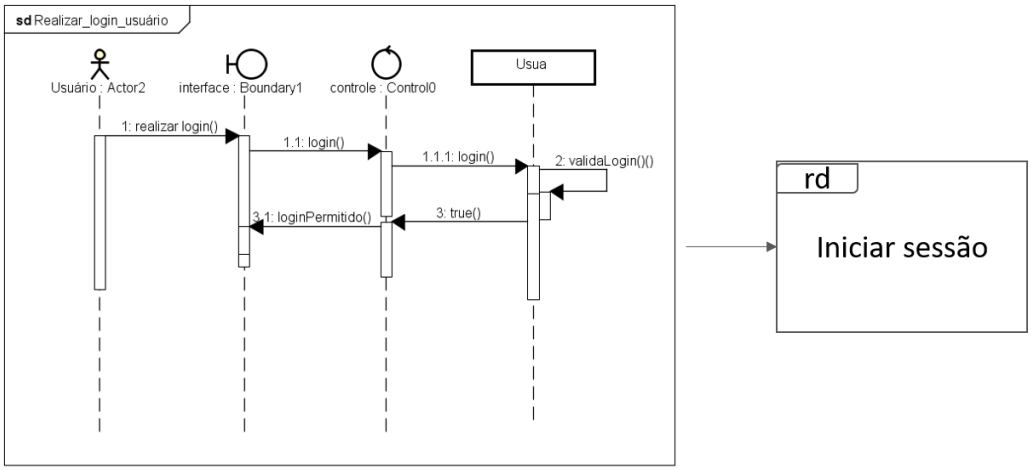
Figura 12 - Exemplo de Diagrama de Pacotes



Fonte: o autor (2018)

Diagrama de Interação Geral: esse diagrama é uma variação do Diagrama de Atividades, a diferença fundamental é que ele fornece uma visão geral dentro de um sistema ou processo de negócio, conforme exemplo apresentado na Figura 13.

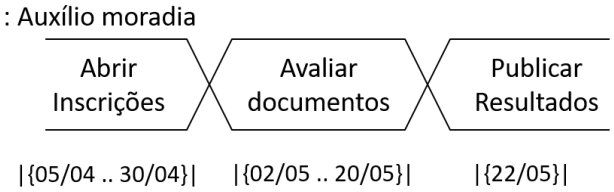
Figura 13 - Exemplo de Diagrama de Interação Geral



Fonte: o autor (2018)

Diagrama de Tempo: esse diagrama demonstra a mudança de estado de um objeto no tempo em resposta a eventos externos ao sistema. De modo geral, ele pode representar a mudança na condição de uma instancia de uma classe durante um determinado período de tempo, conforme exemplo apresentado na Figura 14.

Figura 14 - Exemplo de Diagrama de Tempo



Fonte: o autor (2018).

Melo (2004) afirma que o diagrama de componentes e o diagrama de implantação são diagramas de implementação, pois eles mostram aspectos inerentes à implementação física, como a estrutura dos componentes e a estrutura do sistema em tempo de execução. Vale ressaltar também que para o que Guedes (2011) chama de Diagrama de Gráfico de Estados, Melo (2004) chama de Diagrama de Máquina de Estados e Bezerra (2015) chama de Diagrama de Transições de Estados.

2.1.2 Desenvolvimento Dirigido por Modelos (MDD, do inglês Model-Driven Development)

O MDD é uma nova estratégia de desenvolvimento de software que utiliza os modelos como principal artefato do processo de criação de software. Os conceitos essenciais para o seu entendimento estão descritos abaixo, juntamente com o MDA que é uma variação do MDD e também será utilizado nesse trabalho.

As atividades de modelagem servem para minimizar a complexidade de diversos problemas, isso é perceptível desde a antiguidade, tanto que as engenharias utilizam a modelagem com uma técnica fundamental para permitir uma melhor compreensão, comunicação e construção em diversos sistemas (THOMAS, 2004). Assim sendo, o MDD surgiu no início dos anos 2000 com o intuito de otimizar o processo de desenvolvimento de software (VERNER e PUIA, 2004). Essa iniciativa adota uma estratégia diferenciada ao buscar o aumento da abstração na etapa de construção, mudando seus esforços para a criação de modelos em vez de programação ou codificação (BUARQUE, 2009).

Segundo Almeida (2014) o MDD é um padrão que tem os modelos como artefatos principais do desenvolvimento de software. Geralmente, os modelos utilizados são rigorosos durante todo o período de desenvolvimento e, com isso, a utilização do MDD pode trazer diversas vantagens, dentre elas, está a geração automática de código. No contexto da área de

Engenharia de Software, esse padrão visa trazer produtividade sem que haja perdas na qualidade dos produtos de software.

Vara e Marcos (2012) definem o MDD como uma abordagem que eleva o nível de abstração do desenvolvimento de software por utilizar modelos como classe principal de artefatos. Deste modo, o engenheiro de software não necessita programar, mas sim concentrar-se na criação de modelos de alto nível, evitando complexidades que poderiam ser geradas na codificação em diferentes plataformas.

Segundo Fernandes (2013) o MDD é uma abordagem que consiste em desenvolver software baseado em modelos gráficos, fazendo uma tradução automática deles para código. Esses modelos podem ser criados utilizando a UML ou Linguagens de Modelagens Específicas de Domínio (DSLMs, do inglês *Domain Specific Modeling Languagens*), sendo essas manipuladas computacionalmente.

Buarque (2009) afirma que o principal objetivo do MDD é a transformação de modelos, não a geração de código. A ideia principal encontra-se em transformar modelos em maiores níveis de abstração para modelos mais concretos, embora não haja consenso na comunidade acadêmica sobre qual é o modelo com maior nível de abstração, assim sendo o MDD é um tema que tem evoluído muito na área de Engenharia de Software.

O MDD possui muitas características consideradas como vantajosas, dentre elas, podemos citar a especificação de informações não redundantes, a facilidade em verificação de erros, a aplicabilidade de restrições, a diminuição da complexidade ligada a plataformas, o aumento da legibilidade da representação visual, a geração automática de códigos, além de ter como grande benefício o ganho da produtividade em curto ou longo prazo (SILVA, 2010).

Quanto a classificação, Fernandes (2013) afirma que o MDD possui algumas variantes, como por exemplo DSLMs que são linguagens gráficas voltadas especificamente para determinado problema que pode ser relacionado, por exemplo, à medicina diagnostica, sistema embarcado ou indústria financeira.

Outra variante é o Teste Dirigido por Modelo (MDT, do inglês *Model Driven Testing*), que utiliza o conceito de MDD para a geração automática de artefatos de teste embasadas em regras de transformação pré-definidas em modelos de desenvolvimento. Isso pode trazer aumento da efetividade, confiabilidade e produtividade nos processos de teste, além de serem reaproveitados em diferentes plataformas (ALMEIDA, 2014).

A Engenharia Dirigida a Modelos (MDE, do inglês *Model Driven Engineering*) também é uma variante do MDD e engloba outras atividades dirigidas por modelos em todo o processo de Engenharia de Software, como por exemplo, a evolução de um sistema em um software legado (ALMEIDA, 2014).

Uma outra variante do MDD é a Arquitetura Dirigida por Modelos (MDA), que será utilizada nesse trabalho, pois ela propõe o uso da linguagem UML para a representação de modelos gráficos que expressam características e funcionalidades do software (FERNANDES, 2013).

O padrão MDA foi idealizado pela OMG, entidade também responsável pelo surgimento de algumas tecnologias importantes na indústria de software, dentre elas, podemos citar CORBA, XML, XMI, MOF, CWM e a UML (conforme explicado na Seção 2.1.1). O MDA visa atingir a meta de minimizar a complexidade, baixar os custos e acelerar a implantação de aplicações padronizadas (VERNER e PUIA, 2004).

Verner e Puia (2004) definem MDA como um framework de desenvolvimento de software, onde esse processo é dirigido por modelagem do sistema. O objetivo do MDA é a automatização de algumas desse processo incluindo a geração de código para uma determinada plataforma que for escolhida.

Buarque (2009) define MDA como uma abordagem de desenvolvimento de sistemas que utiliza os modelos para direcionar tudo que é relacionado ao processo de desenvolvimento de software, como por exemplo o projeto, a construção, a distribuição, a operação, a manutenção e a modificação do software.

Lucrédio (2009) afirma que o MDA é um conjunto de padrões utilizado para o desenvolvimento através do MDD. Ele apoia a construção de modelos e a transformação dos mesmo em código, mas para isso ele utiliza três conceitos: o Modelo Independente de Computação (CIM, do inglês *Computacional Independent Model*), o Modelo Independente de plataforma (PIM, do inglês *Platform Independent Model*) e o Modelo Específico de Plataforma (PSM, do inglês *Platform-Specific Model*).

O CIM, o PIM e o PSM são visões do sistema sob diferentes aspectos: (i) o CIM significa que não é necessário ter conhecimento de computação para entender o sistema, (ii) o PIM considera que o sistema seja independente da plataforma onde é implementado e (iii) o PSM é uma visão que considera detalhes específicos e relativos à plataforma em que o sistema é desenvolvido (LUCRÉDIO, 2009).

Segundo Almeida (2014) pode-se citar como uma grande característica do MDA a sua independência do fornecedor, além de ser uma abordagem aberta e padronizada. A OMG (2014) afirma que a portabilidade, a interoperabilidade e a reusabilidade são os principais objetivos do MDA.

Fernandes (2013) afirma que o MDA utiliza modelos gráficos da linguagem UML, que posteriormente são traduzidos de forma automática, em parte ou completamente, em código em uma determinada linguagem de programação.

Segundo Silva (2010) o MDD e MDA tornaram-se bastante populares devido a utilização da UML, pois os mesmos utilizam conceitos de programação, tais como classes ou valores de retorno como construtores de modelos. Desta maneira, desenvolvedores constroem modelos que podem ser facilmente traduzidos em código, além de descreverem o comportamento e a estrutura do software.

Como é possível observar, os conceitos de MDD se complementam, sendo, para este trabalho adotadas as ideias de cada um desses autores supracitados, pois os modelos são os principais artefatos do desenvolvimento de software tendo como intuito evitar complexidades na codificação, como afirmam Almeida (2014) e Vara e Marcos (2012). Além disso, foram utilizados modelos gráficos da UML, como dito por Fernandes (2013). Contudo, esse trabalho não visa passar uma ideia errônea de que com apenas um click obtém-se todas as transformações e o produto de software final, pois como Buarque (2009) afirma, o MDD não visa apenas gerar código, mas transformar os modelos.

No que se refere ao MDA, foi utilizada a definição de Lucrédio (2009), por ser uma das definições mais atuais e por se mostrar mais completa, ao usar conceitos de CIM, PIM e PSM, como ideias bases de abordagem.

2.1.3 Tecnologias de Desenvolvimento

A linguagem que qualquer computador pode entender é a linguagem de máquina. Cada computador tem a sua linguagem própria, natural e definida pelo design de hardware. Essa linguagem consiste em s*trings* de números, que em último caso são 0s e 1s, e ela é totalmente dependente da máquina (DEITEL e DEITEL, 2010). Tendo em vista que essa linguagem é complicada para os seres humanos, foram criadas as linguagens de programação com o intuito de tornar a criação de software uma tarefa mais viável e simples (K19, 2015).

Para este trabalho foram utilizadas as linguagens de programação PHP e Javascript. Também foram utilizados a linguagem de marcação HTML e a folha de estilo CSS, descritas a seguir.

2.1.3.1 PHP Orientado a Objetos

O PHP (*PHP: Hypertext Preprocessor*) é uma linguagem de programação criada em 1994 por Rasmus Lerdorf, e significava *Personal Home Page Tool.* Ela era formada por scripts escritos em linguagem C, com o intuito de monitorar o número de acesso a páginas dinâmicas na internet. Ao longo do tempo, Rasmus adicionou novos recursos que viabilizaram a interação com o banco de dados. Em 1995 o código-fonte do PHP foi liberado à comunidade da computação, possibilitando a junção de mais desenvolvedores ao projeto (DALL'OGLIO, 2015).

Em 1997, Gutmans e Zeev Suraski, dois estudantes que utilizaram o PHP em um projeto acadêmico, resolveram aprimorar o PHP em cooperação com Rasmus, criando assim o PHP 3, lançado oficialmente em junho de 1998. No final deste mesmo ano o PHP já ocupava cerca de 10% dos domínios da internet. Em maio de 2000 foi lançado o PHP 4, cujas melhorias focaram-se no suporte a diversos servidores web, mas ainda não tinha suporte à orientação a objetos. Assim sendo, em junho de 2004 foi lançado o PHP 5, dando suporte a orientação a objetos. Estima-se que o PHPOO é uma das linguagens de programação orientadas a objeto que mais cresce no mundo, sendo utilizado em mais de 80% dos servidores web existentes, sendo assim a linguagem mais utilizada para o desenvolvimento web (DALL'OGLIO, 2015).

Assim sendo o PHP é uma linguagem de programação utilizada em servidores web para a execução de tarefas que geram páginas dinâmicas de dados de uma base de dados da aplicação, processando tarefas complexas, garantindo segurança da aplicação Web, entre outras funcionalidades (CAELUM, 2017).

O PHPOO possibilita a utilização de uma ótica mais próxima do mundo real, lidando com estruturas que além de carregar dados, possuem um comportamento próprio, trocando mensagens entre si com o intuito de realizar uma tarefa. Isso significa que com o PHPOO é possível utilizar conceitos de Orientação a Objetos, como encapsulamento, herança, associação, agregação, polimorfismo etc. Por isso, o PHPOO versão atual 7.2.4, foi utilizado na construção do Framework proposto nesse trabalho.

2.1.3.2 HTML

HTML (do inglês, *HyperText Markup Language*) é uma linguagem de marcação de Hipertexto utilizada para o desenvolvimento de website. Ela é a linguagem base da internet, criada para ser facilmente entendida tanto por pessoas, quanto por máquinas (EIS, 2011).

Segundo Araújo (2015), o HTML é o código utilizado para a estruturação do conteúdo web, dando significado e propósito a ele. Pode haver uma universalidade de conteúdo, como por exemplo, parágrafos, tabelas de dados, vídeos, áudios, listas de pontos, entre outros. A W3C (2017) define HTML como uma linguagem criada para descrever a parte estrutural de páginas web, que são interpretadas e executadas pelo navegador. O HTML possibilita o desenvolvimento de páginas com recursos multimídia de forma estática.

A linguagem HTML é baseada em marcação porque os elementos que mostram as informações que a página exibe são marcados por marcações denominadas de *Tags*. Todas as *tags* são abertas e fechadas. Por exemplo, os títulos de artigos ou manchetes de sites são marcados com a *tag* H1 da seguinte forma: <h1> Texto do título </h1>. É através das *tags* que o desenvolvedor informa para o navegador o que é cada informação (EIS, 2011).

Para este trabalho foi utilizado a versão mais atual dessa linguagem, HTML 5 que, segundo a W3C (2017) recebeu mudanças significativas no que se refere a criação de código semântico, com elementos novos e exibição de conteúdos multimídia com imagens, vídeos e áudios, com o intuito de melhorar a navegação do usuário, padronizando a distribuição de conteúdo multimídia e melhorando a indexação de páginas por mecanismos de busca.

2.1.3.3 CSS

O CSS (do inglês, *Cascading Style Sheets*) foi criado em 1994 por Hakon Wium Lie, com o intuito melhorar a formatação de informações em páginas web. O que acontecia nessa época era a grande popularização do HTML, mas as qualidades e o domínio de controlar aparências para o documento foram sendo acrescentados aos poucos, pois o HTML não tinha a intenção de formatar documentos quando foi criado. Em 1995 o CSS foi apresentado à W3C que, posteriormente lançou o CSS1 em 1996, o CSS2 em 1998 (EIS, 2006).

O CSS é uma linguagem criada para descrever a forma como os elementos do HTML são apresentados, permitindo a definição de estilo da página, escolha de cores, posicionamento de layout, bordas, fontes, entre outros (W3C, 2017). No documento HTML, é possível descrever o código CSS usando o elemento style, mas como uma forma de deixar o

código mais organizado, geralmente os desenvolvedores incorporam arquivos CSS por intermédio do elemento link (JUNIOR, 2012).

CSS é independente do HTML, pois ele pode ser usado para formatar outros tipos de documentos, como XML, por exemplo. Além disso, o CSS possui uma sintaxe bastante simples de entender e aprender, que se baseia em três pontos: (i) seletor cuja função é selecionar quais elementos serão afetados pela definição dos valores das regras, (ii) regra que é um atributo específico de cada tipo de elemento, possuindo (iii) valores válidos (JUNIOR, 2012). Para esse trabalho foi utilizada a versão mais atual dessa linguagem, o CSS3, que está sendo definido anualmente pela W3C.

2.1.3.4 Javascript

O Javascript foi criado na década de 90 pela Netscape, mas inicialmente foi denominado Livescript, uma linguagem criada com o intuito viabilizar uma interação maior de usuários com as páginas do interpretadas pelo navegador Netscape, até então, o mais popular no início dessa mesma década (CAELUM, 2017).

O Javascript não é uma linguagem derivada ou relacionada com a linguagem Java, embora tenha o nome parecido. Ela recebeu esse nome porque na década de 90 a linguagem Java estava ganhando muito espaço no mercado de desenvolvimento de aplicações corporativas, então a Netscape com o intuito de popularizar Livescript, entrou em acordo com a empresa Sun, e rebatizando a linguagem Livescript para Javascript (CAELUM, 2017).

Segundo Araújo (2015) o Javascript é uma linguagem de programação utilizada para adicionar características interativas às páginas web. O Javascript é usado, por exemplo, quando coisas interessantes acontecem quando botões são pressionados, ou no armazenamento de dados em formulários, efeitos dinâmicos no estilo de elementos da página, animações, entre outros.

A W3C (2017) define o Javascript como uma linguagem responsável pala criação de scripts *cliente-side* que interagem com as páginas web e com o comportamento do navegador. No documento HTML, o script de um código Javascript é inserido dentro da *tag* <script>, para que o navegador saiba que se trata de um código de um script e não de HTML.

A Caelum (2017) define o Javascript como uma linguagem de *scripting*, isto é, uma linguagem de programação que possibilita ao desenvolvedor o controle de uma ou mais aplicações de terceiros. Neste contexto, o Javascript é utilizado para o controle de alguns

comportamentos de navegadores por intermédio de trechos de códigos enviados em páginas web.

Uma grande característica do Javascript e de outros tipos de linguagens *scripting* é que elas não dependem de compilação para serem executadas, por esse motivo, elas são linguagens facilmente interpretadas. No Javascript, assim como acontece com o HTML, o navegador ler, interpreta e executa o código linha por linha. Como há conversões automáticas durante as operações, o Javascript possui como outra característica uma grande tolerância a erros (CAELUM, 2017).

2.1.3.5 MySQL

A equipe que criou o Sistema de Gerenciamento de Banco de Dados (SGBD), teve a necessidade de um mecanismo que permitisse a conexão de tabelas criadas na linguagem SQL para um determinado fim. A princípio, o grupo iria utilizar o mSQL, mas logo perceberam que esta ferramenta não era rápida o suficiente para atender às necessidades do projeto. A melhor forma então foi criar uma solução própria, nascia assim o MySQL (HENRIQUE, 2013).

O MySQL foi criado por Michael Widenius na companhia suíça TcX. Por volta de 1979, Michael desenvolveu um banco de dados chamado UNIREG, sendo rescritos em várias linguagens. No ano de 1994, a empresa TcX iniciou a construção de aplicações baseadas na Web, possuindo como base o banco UNIREG, porém esse banco possuía muito custo de dados para se executar uma tarefa, para obter sucesso em uma aplicação para geração de páginas dinâmicas na Web (HENRIQUE, 2013).

Foi então que o desenvolvedor do banco UNIREG contatou o David Hughes criador do mySQL, para saber do interesse dele em unir os dois bancos. Sendo positivo o interesse de David, a empresa TcX resolveu desenvolver um novo banco, mas manteve ao máximo a compatibilidade com mSQL (HENRIQUE, 2013).

Então foi em maio de 1995 que, definitivamente, a primeira versão do MySQL foi lançada. Um dos parceiros da TcX sugeriu a distribuição do servidor na Internet, o objetivo disso era a utilização de um modelo pioneiro desenvolvido por Aladdin Peter Deutsch. O resultado foi uma maior flexibilidade em sem "copyright", que fez do MySQL mais difundido gratuitamente do que mSQL (HENRIQUE, 2013).

Nos dias de hoje, a MySQL AB desenvolve o programa. MySQL AB é a companhia dos fundadores e principais desenvolvedores do MySQL. Eles criaram-no porque precisavam de um banco de dados relacional que pudesse tratar grandes quantidades de dados em máquinas de custo relativamente barato (HENRIQUE, 2013).

Henrique (2013), define o MYSQL como um sistema de gerenciamento de banco de dados relacional multiencadeado, com código fonte aberto e nível corporativo. Além de ser um banco de dados, ele também é um gerenciador de banco de dados. Com este SGBD (Sistema Gerenciador de Banco de Dados), também pode ser utilizado para aplicações corporativas, o qual, necessitam de várias conexões simultâneas, que possibilita 101 conexões simultâneas.

Para quase todo tipo de aplicação, o MySQL é a solução robusta, considerando o seu baixo custo de propriedade. O MySQL oferece o melhor cenário de todos SGBD, executa em muitas plataformas, oferece um baixo TCO (custo total de propriedade) e é muito estável (HENRIQUE, 2013).

2.1.4 Aplicações Web

Pressman e Maxim (2016) definem essas aplicações Web como *WebApps*, uma das categorias de software que é centralizada em redes, abrangendo diversas aplicações. Essas *WebApps* podem ser um conjunto de arquivos de hipertexto interconectados, que podem forma ambientes computacionais sofisticados que ofereçam recursos especializados, funções computacionais e conteúdo para o usuário final.

Kappel et al. (2006) considera como aplicação Web um sistema de software formado por um conjunto de tecnologias baseados em padrões da W3C (do inglês, *World Wide Web Consortium*), que disponibiliza recursos específicos da Web, através de um cliente Web. Ou seja, nem tudo que for criado em HTML pode ser considerado Aplicação Web, por exemplo, páginas web estáticas, que não provê conteúdo e serviços, não são aplicações Web.

A definição que será utilizada para esse trabalho será a de Kappel et al. (2006), pois o Framework servirá não somente para a geração automática de código de páginas web, mas também a geração de código responsável pelas operações básicas que poderão prover serviços ao usuário Web, em linguagem Java ou PHP Orientada a Objetos, de acordo com o diagrama de classe da UML.

As principais características de aplicações web estão relacionadas com a sua utilidade, evolução e processo de desenvolvimento. Existe muitos fatores que as diferenciam das aplicações convencionais, dentre os quais podemos citar a integração entre vários conteúdos multimídias, a grande atenção que os desenvolvedores devem ter quanto à segurança e privacidade, usabilidade, acessibilidade, adequação a questões culturais e integração com outros tipos de sistemas (SCHWINGER e KOCH, 2006).

A utilização da abordagem MDD na criação de uma aplicação Web pode trazer muitos benefícios, dentre eles, podemos citar (i) o aumento da produtividade do desenvolvedor de aplicações Web e relação à maneira tradicional de desenvolvimento, (ii) qualidade das aplicações Web desenvolvidas devido a geração rápida de código, o reaproveitamento de código e facilidade na manutenção da aplicação e (iii) redução de esforços, tempo e custo no desenvolvimento de aplicações web, em ralação à maneira tradicional de desenvolvimento (JUNIOR, 2012).

2.2 Trabalhos Relacionados

Abaixo estão descritos os trabalhos de: (*i*) Câmara, Pinheiro e Almeida (2011), (*iii*) Lopes, Henkels e Fialho (2008), (*iii*) Castro (2010), (*iv*) Oliveira e Mählmann (2010) e (*v*) Deschamps e Grahl (2005). Esses trabalhos apresentam resultados relevantes para este trabalho, conforme descritos na Tabela 3 que mostra um comparativo da proposta com os trabalhos relacionados:

2.2.1 Câmara, Pinheiro e Almeida (2011)

O trabalho de Câmara, Pinheiro e Almeida (2011) teve como objetivo avaliar o tempo médio de desenvolvimento de algumas funcionalidades usuais entre projetos utilizando uma ferramenta para a geração de código a partir de diagrama de classe da UML. Além disso, eles verificaram se houve ou não um ganho real na produtividade no desenvolvimento de software.

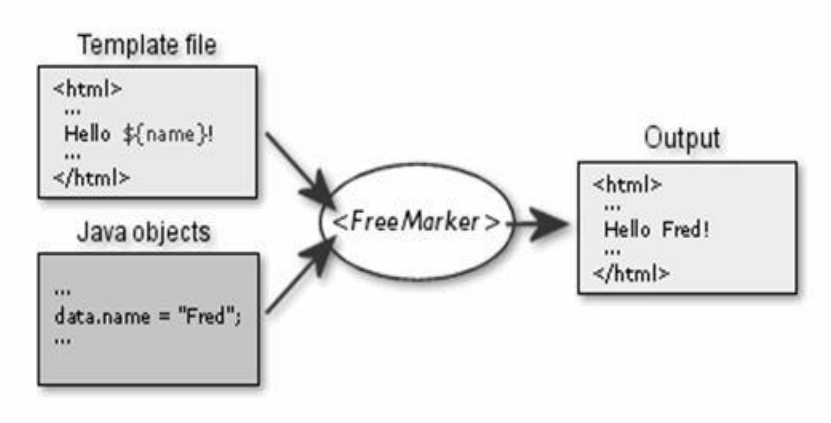
Para a construção da ferramenta, eles utilizaram a técnica de geração de código baseada em template que, segundo Lucas (2005 apud CÂMARA, PINHEIRO e ALMEIDA, 2011, p. 115) é menos formal que a técnica de geração de código baseada em modelos, pois esse tipo de geração de código recebe como entrada um conjunto de templates, onde o cada template é um conjunto de informações de texto corrido agregado a uma linguagem de programação. Já a técnica baseada em modelos é caracterizada pela entrada de informações

em forma de diagramas que, além de serem a principal base de dados para a geração do código, torna o processo de desenvolvimento um auxílio para a documentação do software.

Câmara, Pinheiro e Almeida (2011) utilizaram o ambiente de programação NetBeans IDE, pois viabiliza a construção de software em diversas linguagens de programação, além de oferecer suporte a diversas plataformas como Java, php, Ruby, Groovy, entre outras. A ferramenta criada por eles foi denominada Freemaker e foi projetada visando a geração de páginas Web em HTML.

O Freemaker apenas gera as páginas em formato de texto, pois ele tem como saída de seu processamento, texto com base em modelos. Os programadores incorporam seus projetos nele para que isso seja realizado, pois ele é uma biblioteca de classes, um pacote Java. (CÂMARA, PINHEIRO e ALMEIDA, 2011). A Figura 15 representa como é feito o processo de geração do código utilizando o Freemaker:

Figura 15 - Processo de geração do código utilizando freemaker



Fonte: Câmara, Pinheiro e Almeida (2011)

Como metodologia para avaliar o desempenho da ferramenta proposta por eles, Câmara, Pinheiro e Almeida (2011) elaboraram uma pesquisa na Universidade da Amazônia (UNAMA), na qual eles definiram as perguntas que seriam postas no questionário que foi aplicado a estudantes do sexo masculino ou feminino do curso de Ciência da Computação. Desse modo, o questionário foi construído com o intuito de avaliar o tempo médio que os programadores entrevistados levavam para criar funcionalidades comuns, como o CRUD (*Create, Retrieve, Update e Delete*), além de fazer um levantamento de quais ambientes de programação eles mais utilizavam e qual a linguagem de programação eles mais utilizavam para o desenvolvimento Web. O questionário teve cinco questões, sendo que as três últimas tiveram o intuito de analisar qual é o tempo necessário para se desenvolver o CRUD. A Tabela 2 apresenta os resultados identificados:

Tabela 2 - Questões de pesquisa e resultados obtidos

Pergunta	Resposta
Qual o ambiente de desenvolvimento que você mais utiliza?	Netbeans (55%), Eclipse (26%), Adobe DreamWaver (12%) e Outra (7%)
Qual a linguagem de programação para *Web* que você mais utiliza?	Java (69%), Php (17%), Outra (14%), Asp.net (0%)
Qual o tempo médio que você leva para desenvolver uma tela de cadastro?	Menos de 40min (45%), 40min até 1 hora (25%), Mais de 2 horas (16%) e de 1 até 2 horas (14%)
Qual o tempo médio que você leva para listar todos os registros de uma base de dados?	40min até 1 hora (34%), Menos de 40min (31%), Mais de 2 horas (14%) e de 1 até 2 horas (21%)
Qual o tempo médio que você leva para criar os métodos de alterar e excluir dos registros listados?	40min até 1 hora (33%), Menos de 40min (31%), Mais de 2 horas (19%) e de 1 até 2 horas (17%)

Adaptado de: Câmara, Pinheiro e Almeida (2011)

2.2.2 Lopes, Henkels e Fialho (2008)

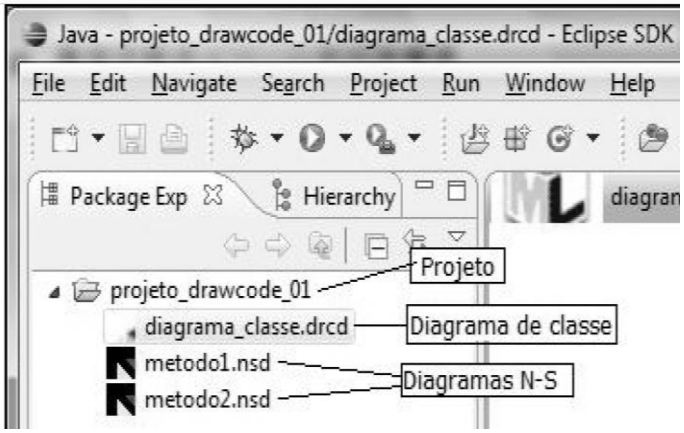
O trabalho de Lopes, Henkel e Fialho (2008) teve como objetivo principal mostrar uma ferramenta em forma de *plugin* para o ambiente *Eclipse* chamada Drawcode, demonstrando que ela pode ser uma grande aliada na construção de software utilizando o diagrama de classe da UML e Diagramas N-S, ao gerar código automático.

O Drawcode possibilita a edição de diagramas de classes e a codificação dos métodos de cada classe por intermédio de notações gráficas N-S. O diagrama N-S é uma das formas de representar a solução de um algoritmo estruturado, mas o trabalho utilizou ele com o intuído de perceber sua efetividade no auxílio à codificação de software Orientado a Objetos (LOPES, HENKEL e FIALHO, 2008).

Como metodologia para cumprir o objetivo proposto, eles fizeram um levantamento bibliográfico para explicar alguns conceitos relacionados ao trabalho, além de realizar pesquisas sobre ferramentas para a edição de diagramas de classes e diagramas N-S, e que tinham características que permitiam oferecer funcionalidades parecidas com a do Drawcode. Assim sendo, eles encontram 4 ferramentas em destaque: *Jupe, Topcased, NSDEditor e Structorizer*.

O Drawcode é um *plugin* para o ambiente *Eclipse*, então, para demonstrar como se utiliza o Drawcode, os autores criaram um projeto chamado projeto_drawcode_01, conforme fx, onde dentro dele criaram um diagrama de classe chamado diagrama_classe.drcd e dois diagramas N-S, chamados metodo1.nsd e metodo2.nsd, respectivamente, conforme é mostrado na Figura 16.

Figura 16 - Estrutura do Projeto Drawcode no ambiente Eclipse



Fonte: Lopes, Henkel e Fialho (2008)

Ao final, a geração de código é feita por meio de *templates*, que neste contexto são artefatos de software que possibilitam a junção de informações a um arquivo de marcações. Como resultado, o Drawcode se mostrou uma alternativa viável para ambientes educacionais, principalmente para pessoas em fases iniciais de formação, pois ele explora os benefícios dos conceitos de orientação a objetos aliados à notação gráfica (LOPES, HENKELS e FIALHO, 2008).

Um ponto negativo do Drawcode é que para utilizá-lo precisa-se ter o conhecimento de diagramas N-S. Além do mais, para construir o software é necessário construir tanto de diagrama N-S quanto diagrama de classe da UML. Com um ponto bastante positivo do Drawcode, podemos citar a grande contribuição que ele pode dar a desenvolvedores experientes, pois ele pode ser uma ferramenta que aumente a produtividade e a qualidade no desenvolvimento do software.

2.2.3 Castro (2010)

O trabalho de Castro (2010) teve como objetivo criar um gerador de código denominado BlueBox que otimiza a produção de empresas de desenvolvimento de software que utilizam o framework Iguassu para o desenvolvimento e criação de práticas de modelagem, visando a geração de código conforme o modelo Iguassu.

Quanto à metodologia desse trabalho, para a definição do tema foi realizado um Estudo de Caso, isto é, um modelo metodológico baseado na interpretação de dados primários que são coletados diretamente na origem do universo pesquisado, neste caso, no Departamento de Ciência da Computação da Universidade Federal de Lavras. Os dados

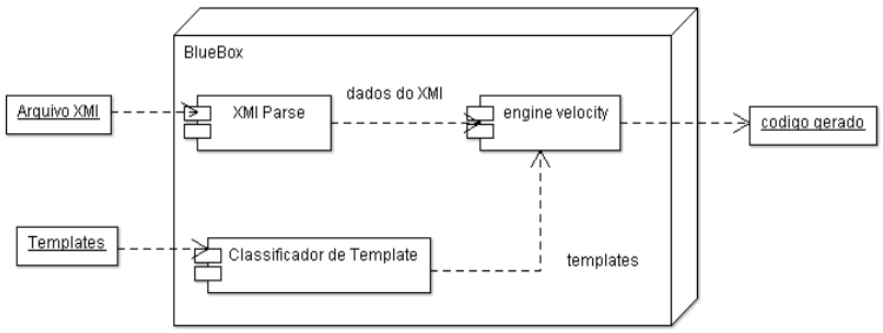
secundários foram levantados por meio de uma revisão bibliográfica em artigos técnicos, relatórios, normas técnicas, entre outras referências sobre o assunto (CASTRO, 2010).

Como estratégia para a criação do BlueBox, foi realizada uma pesquisa em portais eletrônicos, livros e artigos sobre a abordagem MDA relacionada à modelagem UML. Assim foi possível perceber o grande impacto que a UML pode ter na geração de arquivos de dados XMI, por isso que foi criado um procedimento de modelagem baseado nessa linguagem de modelagem. Depois disso, foi feito um estudo visando entender o comportamento das classes do framework Iguassu, visando criar templates que se adequassem ao mesmo, depois que os geradores automáticos de códigos fossem implementados (CASTRO, 2010).

Sobre o Framework Iguassu, Castro (2010) afirma que ele é uma implementação Java da arquitetura Iguassu, cujo modelo arquitetural é o MVC, que será discutido na próxima seção. Em relação à técnica de geração de código, o BlueBox utiliza o método baseado em templates (descrito na Seção 2.2.1), ele utiliza o arquivo XMI que é um padrão OMG criado para a troca de informações entre diversas ferramentas, e a tecnologia Velocity que realiza o referenciamento das informações das informações que são obtidas nos arquivos XMI (CASTRO, 2010).

Assim sendo, o BlueBox é formado por três componentes: (i) o XMI Parse, que obtém os dados do arquivo XMI, (ii) o classificador de Templates, cuja finalidade é classificar os templates conforme o seu comportamento durante a geração e (iii) o Engine Velocity que faz a geração de código com base em informações referenciadas pelo arquivo XMI nos templates. Da junção desses componentes é formada a arquitetura do BlueBox, conforme mostra a Figura 17.

Figura 17 - Arquitetura do BlueBox



Fonte: Castro (2010)

Essa ferramenta foi implantada na empresa Mitah Technologies, onde foi possível observar um grande aprimoramento na fase de produção na empresa. Houve grandes

resultados nos procedimentos de modelagem e na geração automática de código de aplicações feitas no framework Iguassu. Foi possível observar também que não houve mais necessidade de desenvolvedores desperdiçarem tempo com a implementação de operações básicas, diminuindo também gastos para a empresa durante a construção e manutenção do software (CASTRO, 2010).

2.2.4 Oliveira e Mählmann (2010)

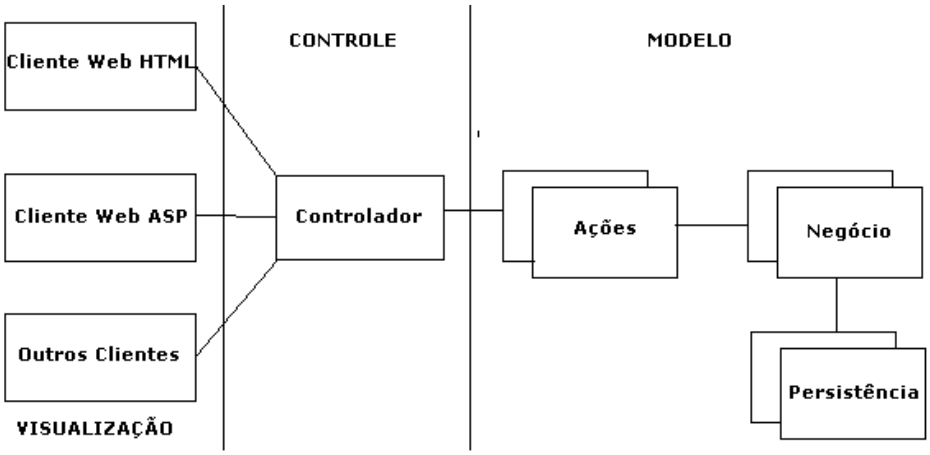
O trabalho de Oliveira e Mählmann (2010) teve principal objetivo desenvolver uma ferramenta compacta capaz de gerar código fonte em linguagem C#.NET, otimizada e flexível, viabilizando a geração da camada de negócio das aplicações, cujas regras pertinentes a ela serão encapsuladas. A estrutura de classes de um modelo de dados SQL Server serviu de base para que isso seja possível.

Como metodologia para o desenvolvimento desse trabalho, foi feito um levantamento de requisitos para a definição das funcionalidades necessárias para que atendam às necessidades do sistema. Após isso, foi realizado um estudo profundo sobre as metodologias baseadas em UML 2.0 que permitam a descrição da estrutura, o detalhamento e especificação dos requisitos levantados na fase de análise do projeto, envolvendo Diagramas de Casos de Usos, Diagramas de Sequência e Diagramas de Classe. Além do mais, como esse trabalho possui bases de dados relacionais, foi utilizado o Diagrama de Entidade e Relacionamento para a modelagem do banco de dados (OLIVEIRA e MÄHLMANN, 2010).

A técnica de geração de código nesse trabalho é baseada em *templates* (descrito na Seção 2.2.1), cujos mesmos são dinâmicos e divididos por funções. Para isso, quando o usuário se cadastra na ferramenta, ele informa o caminho físico que será utilizado para a exportação de seus projetos (OLIVEIRA e MÄHLMANN, 2010).

Para a construção da ferramenta foi utilizado o Padrão MVC (do inglês, *Model, View e Controller*), um modelo de três camadas físicas e é arquitetura padrão para sistemas corporativos com bases na *Web*. A Camada *Model* é onde estão as funções de validação das regras de negócio inerentes ao software, ou à entidade que a mesma representa. A Camada *View* é onde o *Model* é renderizado em uma forma que permita a interação com o usuário. A Camada *Controller* é onde ocorre o processamento de eventos, que geralmente são ações feitas pelo usuário e pode invocar alteração na camada *Model* (OLIVEIRA e MÄHLMANN, 2010). A Figura 18 mostra como ocorre a interação entre as três camadas MVC.

Figura 18 - Interação entre as Camadas MVC



Fonte: Oliveira e Mählmann (2010)

Deste modo, a ferramenta possui três camadas que são divididas em três projetos distintos denominados *Tool, Tool.Entity e Tool.Business*. O projeto *Tool* corresponde a camada *View,* e ele é responsável apenas pelas as interfaces da ferramenta. O projeto *Tool.Entity* corresponde a camada *Model,* e ele é responsável pelo tratamento de acessos à dados por meio de entidades que são referentes ao modelo de dados. O projeto *Tool.Business* corresponde à camada *Controller,* é responsável por encapsular as regras de negócio pertinente à entidade portadora do software. Esse projeto também trata todas as regras antes e após as operações de CRUD. A interação dessas três camadas constitui a arquitetura dessa ferramenta (OLIVEIRA e MÄHLMANN, 2010).

Ao criar 10 classes com diferentes características de um sistema de cadastro de controle patrimonial, como um exemplo para demonstrar o uso da ferramenta, pode-se observar uma grande diferença de tempo entre a criação manual e automatizada de cada classe criada. Houve uma produtividade significativa na codificação com a média de ganho de 49% em relação ao método manual (OLIVEIRA e MÄHLMANN, 2010).

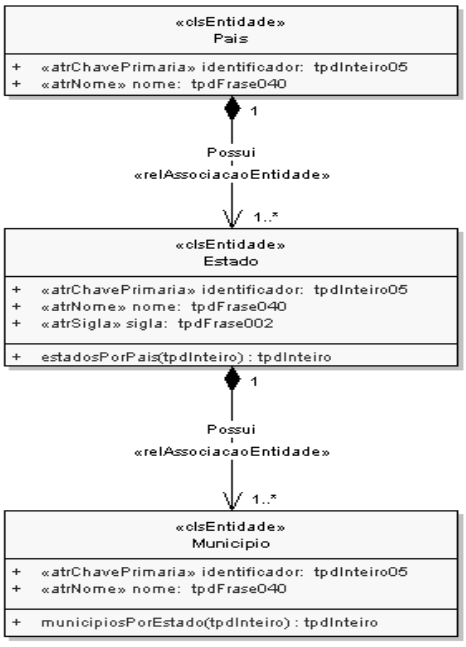
Contudo, chegou-se à conclusão de que é possível desenvolver uma ferramenta CASE objetiva e compacta capaz de gerar código fonte em linguagem C#.NET, que ajude a evitar problemas recorrentes do desenvolvimento de software, como por exemplo o retrabalho que pode casar grande impacto e desvios no cronograma do projeto, elevando os custos com mão de obra (OLIVEIRA e MÄHLMANN, 2010).

2.2.5 Deschamps e Grahl (2005)

O trabalho de Deschamps e Grahl (2005) teve como principal objetivo apresentar o desenvolvimento de uma ferramenta que auxilia o processo de desenvolvimento de software permitindo a especialização das informações que estão contidas em diagramas de classes da UML por arquitetura e idioma. O intuito da construção dessa ferramenta é a agregação de flexibilidade e personalização nesse processo, visto que por meio da estrutura de especialização é possível adicionar novas características ao diagrama de classe, por arquitetura e idioma (DESCHAMPS e GRAHL, 2005).

Como metodologia demonstrar a aplicabilidade da ferramenta para a geração do código foi realizado um estudo de caso, onde foi apresentado um exemplo utilizando um diagrama de classes de localizações criado através da ferramenta CASE *Enterprise Architect*, onde foi possível demonstrar a capacidade de especialização da ferramenta por arquitetura e idioma (DESCHAMPS e GRAHL, 2005). O diagrama de localizações possui três classes de entidade (“País”, “Estado” e “Município”), conforme observado na Figura 19.

Figura 19 - Diagrama de Localizações



Fonte: Deschamps e Grahl (2005)

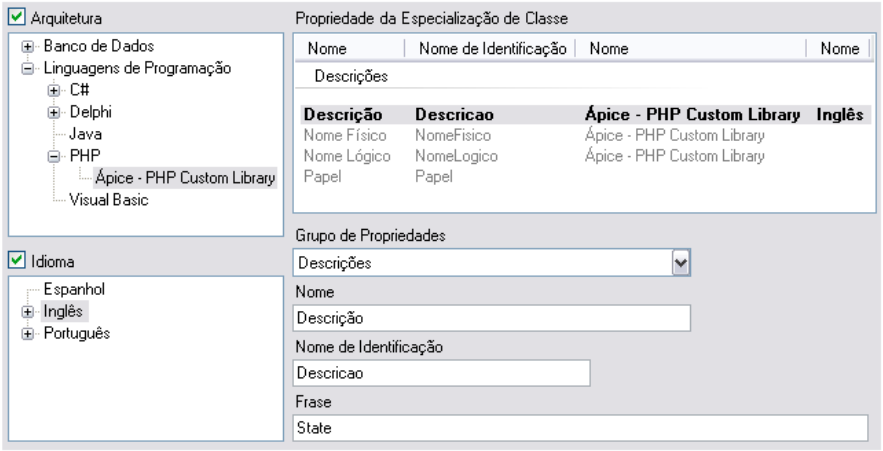
A ferramenta proposta pelo trabalho utiliza *plug-ins* para a geração de código ao invés de *scripts*, ao contrário das soluções disponíveis no mercado, pois Deschamps e Grahl (2005) afirmam que há aumento de velocidade na geração e ganho de recursos para a construção dos

mesmos. Para esse trabalho, eles utilizaram a plataforma Microsoft .NET, devido a possibilidade de utilizar diferentes linguagens de programação para a construção dos mesmos.

Segundo Deschamps e Grahl (2005), para o exemplo apresentado no estudo de caso, foram construídos dois *plug-ins*, um para a arquitetura Java e outro para a arquitetura PCL (do inglês, *PHP Custom Library*), que é uma biblioteca para o desenvolvimento PHP. É importante frisar que o *plug-in* para a geração de código para a arquitetura PCL realiza a geração de classes de interface permitindo a criação de um software completo para as operações CRUD, conforme apresentado a seguir na Figura 21. Na especialização do idioma, foram utilizados o português e o inglês. Nesse estudo de caso foram apresentadas várias figuras demonstrando a especialização de tipos de dados, classes de entidades, descrição e arquitetura. A Figura 20 mostra um exemplo da especialização da classe descrição para o idioma inglês.

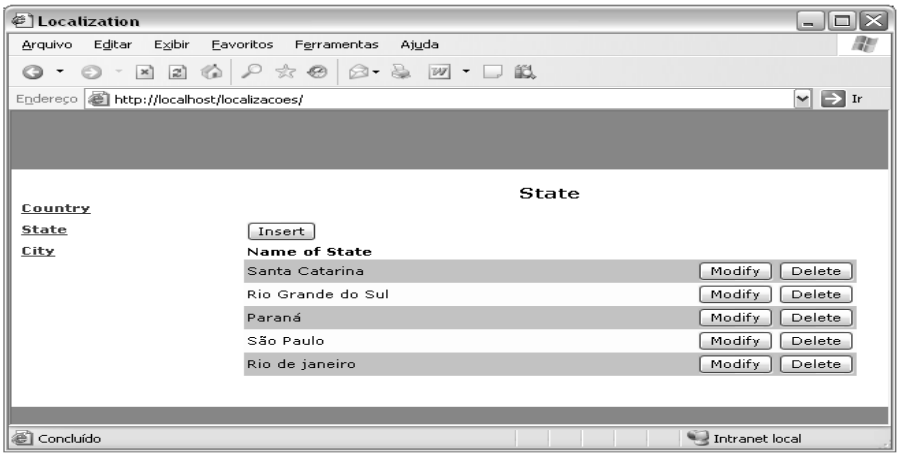
Como resultado, foi possível perceber que a ferramenta ajuda a agregar uma melhor padronização no processo de desenvolvimento de software, pois ela utiliza a linguagem de documentação e modelagem UML, além de usar boas práticas de Engenharia de software. Ela também diminui o tempo de implementação, sendo que esse é requisito fundamental para fechamento de negócios. Contudo, ela aumenta a qualidade dos softwares desenvolvidos, visto que os mesmos estarão mais flexíveis para manutenções futuras (DESCHAMPS e GRAHL, 2005).

Figura 20 - Especialização da propriedade "Descrição" para o idioma inglês



Fonte: Deschamps e Grahl (2005)

Figura 21 - Interface PCL gerada para a manutenção da classe "Estado"



Fonte: Deschamps e Grahl (2005)

2.2.6 Comparativo da Proposta com os Trabalhos Relacionados

A Tabela 3 mostra as principais características dos trabalhos relacionados com o intuído de mostrar um comparativo com a proposta desse trabalho.

Tabela 3 - Comparativo com os trabalhos relacionados

Trabalhos Relacionados	Comparativo com a Proposta
Câmara, Pinheiro e Almeida (2011)	O trabalho relacionado em questão, assim como este trabalho, tem o intuito de desenvolver uma ferramenta para a geração automática de código com base no Diagrama de Classe da UML. Porém, diferem na técnica utilizada: *templates* e modelos, respectivamente. No trabalho relacionado, a geração de código também é voltada para Aplicações Web, mas para isso a ferramenta precisa estar integrada com um IDE *Netbeans*, sendo o Java para Web a linguagem de programação em que o código é gerado. Diferente deste trabalho, a proposta não terá uma ferramenta que necessitará ser integrada a uma IDE, além do mais, o desenvolvedor poderá escolher qual a linguagem ele vai querer que o código seja gerado: Java ou PHP. **Principais assuntos abordados:** Geração de código usando *templates*, Diagrama de Classe, Aplicações Web, Java para Web e *Netbeans*.
Lopes, Henkels e Fialho (2008)	O trabalho relacionado teve o intuito de criar uma ferramenta para a geração automática de código a partir de Diagramas de Classe e Diagramas N-S. Mas para isso, foi utilizada a técnica baseada em *templates* e a ferramenta do resultado final desse trabalho é um *Plugin* do IDE *Eclipse*. Além do mais, foi utilizada o Diagrama N-S que trabalha com base na especificação dos dados do Diagrama de Classe. Este trabalho utilizará apenas o Diagrama de Classes, e a ferramenta gerada por ela não será um *Plugin*, mas sim um Framework, que utilizará os modelos como artefato principal para a geração das operações CRUD. **Principais assuntos abordados:** Geração de código usando *templates*, Diagrama de Classe, Diagrama N-S, *Plugin* e *Eclipse*.
Castro (2010)	O trabalho relacionado teve o intuito de criar uma ferramenta para a geração automática de código para otimizar a produção de software que utilizam o Framework Iguassu. Para isso, eles usaram a técnica baseada em *templates* e alguns conceitos do padrão MDA. A proposta não produzirá uma ferramenta para softwares que utilize um Framework específico, não vai ser baseada em t*emplates*, mas utilizará o padrão MDA, gerando códigos a partir de Diagrama de Classe da UML. **Principais assuntos abordados:** Geração de código usando *templates*, Diagrama de Classe, Framework Iguassu e MDA.
Oliveira e Mählmann (2010)	O trabalho relacionado teve o intuito de desenvolver uma ferramenta CASE geradora de códigos fontes na linguagem C#.NET. Mas para isso foi utilizada a técnica

Trabalhos Relacionados	Comparativo com a Proposta
	baseada em *templates* pré-definidos para a geração de classes, além de automatizarem a criação da camada de negócio arquitetada no padrão MVC. A proposta não utilizará C#.NET, mas sim o Java Web e PHP Orientada a Objetos. Além disso, a técnica de geração de código será baseada em Modelos, utilizando MDD e MDA, mas a aplicação Web será gerada no padrão MVC, assim como no trabalho relacionado. **Principais assuntos abordados:** Geração de classes usando *templates*, Padrão MVC e C# .NET.
Deschamps e Grahl (2005)	O trabalho relacionado teve o intuito de apresentar o desenvolvimento de uma ferramenta para a especialização do Diagrama de Classes por arquitetura e idioma. A geração de código é feita por meio de *plugins* e foi baseada com base tecnológica na ferramenta Microsoft .NET e as arquiteturas utilizadas são Java e PCL. Este trabalho não desenvolverá uma ferramenta que seja possível especializar arquitetura ou idioma e a geração de código não será feita por *plugins*. O framework Jungle possibilitará a construção de Diagramas de Classes e a escolha da linguagem que em o código fonte deverá ser gerado. Além disso, não será utilizado *plugins*, mas sim *scripts* para implementação das rotinas de geração. **Principais assuntos abordados:** Geração de classes usando *plugins*, Diagrama de Classes, Especialização de IMicrosoft .NET, Java e PCL.

Fonte: o autor (2018).

3 PROJETO DO FRAMEWORK JUNGLE

3.1 Levantamento dos Requisitos

O levantamento de requisitos é a etapa onde o problema que será aplicado no desenvolvimento do software é compreendido. O objetivo do levantamento de requisitos que tanto os usuários quanto os desenvolvedores possam tem a mesma visão do problema que será resolvido. É assim que são levantadas as necessidades dos usuários que influenciaram diretamente na construção das funcionalidades ou características do sistema. Essas necessidades levantadas são denominadas de requisitos (BEZERRA, 2015).

O levantamento de requisitos do framework Jungle foi dividido em três etapas: Planejamento (definição dos participantes e da instrumentação), Aplicação (o questionário/ entrevista foram realizados no local de pesquisa) e Análise (os dados foram extraídos e condensados).

3.1.1 Planejamento do Levantamento de Requisitos

- Definição dos participantes:

A coleta de dados dos requisitos foi realizada com pessoas que atendem os seguintes perfis:

- ✓ Ser profissional ou estudante da área da computação.
- ✓ Ter conhecimento sobre Diagramas da UML.
- ✓ Possuir alguma experiência na construção de aplicações Web.
- ✓ Possuir pelo menos uma experiência na utilização de frameworks na construção de aplicações Web.

- Definição da instrumentação:

Para realizar o levantamento de requisitos, primeiramente foi feita uma **entrevista** com um roteiro pré-definido (que contém questões a respeito do perfil profissional e sua opinião sobre o uso de frameworks de desenvolvimento web), e posteriormente, foi aplicado um **questionário** (que contém questões quanto a estrutura que um software deve ter para ser aplicado no contexto de Aplicações Web). Esses dois instrumentos estão descritos abaixo:

1 – Entrevista:

1. Qual é o seu nome?
2. Sexo
3. Qual é a sua idade?
4. Qual é o seu nível de escolaridade?
5. Qual é o seu nível de atuação?
6. Você já trabalhou com desenvolvimento de Aplicação Web?
7. Caso a resposta seja “sim”, há quanto tempo você trabalha com desenvolvimento de Aplicações Web?
8. É importante modelar o sistema, utilizando diagramas da UML, antes da fase de implementação?
9. Já utilizou algum framework para desenvolvimento de Aplicações Web?
10. Se já utilizou, quais foram?
11. Qual a linguagem para a Web que você mais prefere utilizar?
12. Qual é o tempo que você leva para desenvolver uma tela de cadastro?
13. Qual é o tempo médio que você leva para listar todos os registros de uma base de dados?
14. Qual é o tempo que você leva para criar os métodos de alterar e excluir os registros listados?
15. Quais funcionalidades você acha importante que um framework deva possuir?

2 – Questionário:

Nas questões abaixo, marque apenas uma opção de cada pergunta:

Verificação das características que devem incorporar o framework proposto, contendo opções prévias próprias de frameworks dessa natureza.

Imagine um framework onde será possível modelar Diagramas de Classe da UML de uma aplicação Web e a partir disso gerar o código com as operações básicas dessa aplicação. Em relação a este Framework, responda as questões abaixo:

1. Quais recursos você acha que um framework de geração de códigos a partir de diagramas de classe deverá possuir para a modelagem dos diagramas?

( ) Qualidade de código ( ) Segurança ( ) Documentação e materiais online ( ) Facilidade em manutenção do código ( ) Facilidade em modelagem dos diagramas ( ) Outro
Caso selecione "outro", informe quais: _____

Nas questões abaixo, marque apenas uma opção de cada pergunta:
2. O Framework deve informar o usuário o que está acontecendo no momento em que está sendo utilizado? ( ) Sim ( ) Não ( ) Depende Se marcou a opção "Depende", explique o porquê:
3. Quanto a linguagem utilizada para transmitir informações ao usuário, ela deve: ( ) Ser muito simples ( ) Ser simples ( ) Ser uma linguagem mais técnica ( ) Ser uma linguagem menos técnica ( ) Ser uma linguagem simples, mas um pouco técnica também ( ) Outros…
4. O Framework deve ter saídas de emergências, caso o usuário esteja em um estado indesejado? ( ) Sim ( ) Não ( ) Depende Se marcou a opção "Depende", explique o porquê:
5. O Framework deve permitir a geração automática de código: ( ) Quando houver pelo menos um diagrama modelado ( ) Quando todos os diagramas da aplicação forem modelados
6. Quanto à linguagem de programação do código gerado: ( ) Deverá ter uma linguagem padrão definida pelo Framework ( ) Deverá possuir mais de uma linguagem de programação disponível para que o desenvolvedor possa escolher a linguagem de sua preferência.
7.Você considera importante que o framework disponibilize uma opção para reaproveitamento de código durante a construção de um novo projeto? ( ) Sim ( ) Não
8. Ao longo da sua utilização, o Framework deve gerar um relatório em forma de texto (documento) a fim de mostrar o nome das classes, métodos, atributos e seus respectivos tipos? ( ) Sim ( ) Não
9. É importante que os diagramas de classe tenham uma estética parecida com a utilizada por Ferramentas CASE de modelagem disponível no mercado? ( ) Sim ( ) Não ( ) Em partes: Se marcou a opção "Em partes", explique quais:
10. O Framework deve ser padronizado em relação à sua estética, levando em consideração cores, botões ou ícones? ( ) Sim ( ) Não ( ) Um pouco

Nas questões abaixo, marque apenas uma opção de cada pergunta:
11. O Framework deve possuir opção de ajuda? ( ) Sim ( ) Não ( ) Depende Se marcou a opção "Depende", explique o porquê:
12. O Framework deve possuir cores: ( ) Primárias ( ) Secundárias ( ) Fortes ( ) Fracas
13. O Framework deve possuir uma fonte: ( ) Maiúscula ( ) Minúscula ( ) Arredondada ( ) Cursiva
14. O Framework deve possuir textos: ( ) Maiúsculos ( ) Minúsculos
15. Se você achar necessário, indique outras sugestões (de cores, nomes, funcionalidades etc)

3.1.2 Análise da Pesquisa

No total participaram 10 colaboradores que mais se adequaram ao perfil da pesquisa, sendo 5 estagiários (alunos da UFAM) e 5 profissionais atuantes na área da computação. Com relação à entrevista, a Tabela 4 e a Tabela 5 mostram, respectivamente, a caracterização dos participantes estagiários e profissionais.

Tabela 4 - Resultado da entrevista (Estagiários)

Questão	Estagiário 1	Estagiário 2	Estagiário 3	Estagiário 4	Estagiário 5
2	Masculino	Masculino	Masculino	Masculino	Masculino
3	De 22 a 25	De 22 a 25	De 22 a 25	De 22 a 25	De 22 a 25
4	Ensino Superior Incompleto	Ensino Superior Incompleto	Ensino Superior Incompleto	Ensino Superior Incompleto	Ensino Superior Incompleto
5	Desenvolvedor Júnior	Desenvolvedor Trainee	Desenvolvedor Trainee	Desenvolvedor Trainee	Estudante
6	Sim, trabalho atualmente	Sim, trabalho atualmente	Sim, trabalho atualmente	Sim, trabalho atualmente	Sim, trabalho atualmente
7	1 ano	Mais de 2 anos e menos de 5 anos	Mais de 1 ano	Mais de 5 anos	Menos de 1 ano
8	Concordo plenamente	Não concordo, nem discordo	Concordo plenamente	Concordo plenamente	Concordo plenamente
9	Sim	Sim	Sim	Sim	Sim
10	Yii, Grails, Bootstrap 3 e 4	Yii e Grails	Yii e Angular JS	Yii e Grails	Grails e Rails
11	PHP	PHP	PHP	PHP	PHP
12	40min até 1h	Menos de 40min	40min até 1h	Menos de 40min	Menos de 40min
13	40min até 1h	40min até 1h	1h até 2hs	1h até 2hs	Menos de 40min
14	40min até 1h	1h até 2hs	1h até 2hs	Menos de 40min	Menos de 40min

Questão	Estagiário 1	Estagiário 2	Estagiário 3	Estagiário 4	Estagiário 5
15	Qualidade de código, documentação e materiais online e facilidade em Manutenção do Código	Segurança, documentação e matérias online e facilidade em manutenção do código	Qualidade de código, segurança, documentação e matérias online e facilidade em manutenção do código	Qualidade de código, segurança, documentação e matérias online, facilidade em manutenção do código e padrão de design	Qualidade de código e documentação e materiais online

Fonte: o autor (2018).

Tabela 5 - Resultado da entrevista (Profissionais)

Questão	Profissional 1	Profissional 2	Profissional 3	Profissional 4	Profissional 5
2	Prefiro não declarar	Feminino	Masculino	Masculino	Masculino
3	De 26 a 30	De 22 a 25	De 22 a 25	De 26 a 30	Mais de 30
4	Mestrado	Ensino Superior Completo	Ensino Superior Completo	Ensino Superior Completo	Ensino Superior Completo
5	Coordenador de Desenvolvimento	Analista de Teste Trainee	Desenvolvedor Júnior	Desenvolvedor Júnior	Desenvolvedor Sênior
6	Sim, trabalho atualmente	Sim, trabalho atualmente	Sim, há pouco tempo	Sim, trabalho atualmente	Sim, há muito tempo
7	Mais de 10 anos	Menos de 1 ano	Mais de 1 ano	Menos de 1 ano	Mais de 10 anos
8	Não concordo, nem discordo	Concordo parcialmente	Concordo plenamente	Concordo plenamente	Concordo parcialmente
9	Sim	Sim	Sim	Sim	Sim
10	Yii, Grails, Rails, ZEND, Cake PHP, Cogniter e Smart	Grails e Angular JS	Yii, Angular JS e Slim	Yii e Grails	Angular JS
11	Groovy e Python	Java	PHP	PHP	Asp.net
12	Funcionalidade = FrontEnd + Backend ? Depende da tecnologia :)	40min até 1h	5 minutos usando Yii	40min até 1h	Menos de 40min
13	Depende se for a funcionalidade (FrontEnd + Backend) ou consulta SQL	1h até 2hs	Menos de 40min	40min até 1h	Menos de 40min
14	Depende da Linguagem e da funcionalidade (FrontEnd + Backend)	40min até 1h	1h até 2hs	Menos de 40min	Menos de 40min
15	Qualidade de código, segurança, documentação e matérias online, facilidade em manutenção do código	Qualidade de código, segurança, documentação e matérias online, facilidade em manutenção do código	Qualidade de código, segurança, documentação e matérias online, facilidade em manutenção do código	Qualidade de código, segurança, documentação e matérias online, facilidade em manutenção do código	Qualidade de código, segurança, documentação e matérias online, facilidade em manutenção do código

Fonte: o autor (2018).

Pelo resultado obtido e mostrado nas tabelas de caracterização dos participantes foi possível observar que:

- 100% dos entrevistados têm experiência com desenvolvimento de Aplicações Web, sendo que 80% dos entrevistados trabalham atualmente com desenvolvimento dessas aplicações, 10% começaram a trabalhar com isso a pouco tempo e 10% há muito tempo.
- 60% dos entrevistados concordam plenamente que é importante modelar o sistema antes de implementá-lo.
- 100% dos entrevistados já utilizaram algum framework para desenvolvimento de Aplicações Web.
- 100% dos estagiários entrevistados preferem a linguagem PHP.
- 40% dos profissionais entrevistados preferem a linguagem PHP, 10% preferem Java, 10% preferem Asp.net e 10% preferem Groovy e Python.
- No total, 70% de todos os entrevistados preferem a linguagem PHP para o desenvolvimento de Aplicações Web.
- 70% dos entrevistados já utilizaram os frameworks Grails e Yii.
- Com relação às funcionalidades que um framework deva possuir, os entrevistaram citaram: documentação e materiais online (100%), qualidade de código (90%), facilidade em manutenção de código (90%), segurança (10%) e padrão de design (10%).

Com relação a verificação das características que devem incorporar o framework proposto, contendo opções prévias próprias de frameworks dessa natureza, foram obtidos, após a análise os resultados na Tabela 6, para todos os entrevistados. É importante frisar que foram considerados as respostas que alcançaram o maior índice na pesquisa.

Tabela 6 - Resultado do questionário

Questão	Resposta Escolhida	# de Resposta
1	Facilidade em manutenção do código	6
2	O Framework deve informar o usuário o que está acontecendo no momento em que está sendo utilizado	7
3	Quanto a linguagem utilizada para transmitir informações ao usuário, ela deve: ser simples	7
4	O Framework deve ter saídas de emergências, caso o usuário esteja em um estado indesejado	9
5	O Framework deve permitir a geração automática de código: Quando houver pelo menos um diagrama modelado	8
6	Quanto à linguagem de programação do código gerado: Deverá possuir mais de uma linguagem de programação disponível para que o desenvolvedor possa escolher a linguagem de sua preferência.	7

Questão	Resposta Escolhida	# de Resposta
7	Você considera importante que o framework disponibilize uma opção para reaproveitamento de código durante a construção de um novo projeto: sim	9
8	Ao longo da sua utilização, o Framework deve gerar um relatório em forma de texto (documento) a fim de mostrar o nome das classes, métodos, atributos e seus respectivos tipos: SIM	8
9	Os diagramas de classe devem ter uma estética parecida com a utilizada por Ferramentas CASE de modelagem disponível no mercado	8
10	O Framework deve ser padronizado em relação à sua estética, levando em consideração cores, botões ou ícones	9
11	O Framework deve possuir opção de ajuda	10
12	O Framework deve possuir cores: Primárias	6
13	O Framework deve possuir uma fonte: Minúscula e Arredondada	6
14	O Framework deve possuir textos: Minúsculos	8
15	Sugestões (de cores, nomes, funcionalidades etc): •O framework pode ter uma opção para gerar mais de um código possível para os diagramas •Quanto ao próprio framework podes usar padrões "estéticos" conhecidos como twitter bootstrap e/ou JQuery UI etc.	2

Fonte: o autor (2018).

Como resultado, pode-se concluir que a primeira versão do Framework deve ter as seguintes características de acordo com a opinião dos entrevistados:

- Emissão de avisos: informar o usuário o que está acontecendo no momento em que está sendo utilizado e saídas para possíveis emergências;
- Geração de código: quando houver pelo menos um diagrama;
- Código gerado: terá mais de uma linguagem de programação disponível (PHP, Java e Groovy), mas por padrão, estará selecionado PHP, sendo esta a preferência de 70% dos entrevistados.
- Estética: parecida com ferramentas CASE de modelagem disponíveis no mercado, contendo botões, cores e ícones padronizados;
- Linguagem verbal: simples;
- Cores: primárias;
- Fonte: minúscula e arredondada;
- Textos: minúsculos.

As demais características do framework estão descritas de forma mais detalhada na Seção 3.1.3, que fala dos requisitos funcionais, não funcionais e regras de negócio. Esses requisitos são essenciais para a modelagem dos diagramas gráficos da UML e da arquitetura do sistema.

3.1.3 Requisitos Funcionais e Não Funcionais

De acordo com McLaughlin, Pollice e West (2007) o termo requisito pode ser definido como uma necessidade única responsável por detalhar o que um determinado produto ou serviço deve ter ou fazer. No contexto da Engenharia de Software eles podem ser divididos em funcionais e não funcionais (FILHO, 2009).

Segundo Filho (2009) requisitos funcionais expressam os comportamentos que um programa ou sistema deve apresentar diante de certas ações dos usuários. Os requisitos não funcionais representam determinados aspectos de comportamento. De acordo com Pfleeger (2007) um requisito funcional é a descrição da interação do sistema e o ambiente, já o requisito não funcional descreve restrições do sistema que limitam opções do desenvolvedor no que se refere a criar soluções para o sistema. Para este trabalho, foram utilizadas as definições de McLaughlin, Pollice e West (2007) e Filho (2009). Assim, para a primeira versão do sistema foram definidos os requisitos funcionais (RF) e não funcionais (RNF), conforme mostrados na Tabela 7 e Tabela 8 , respectivamente:

Tabela 7 - Requisitos funcionais

ID	Descrição	Prioridade	Requisitos Relacionados
[RF01]	O Framework deve permitir a modelagem de diagramas de classes	Essencial	[RF02]
[RF02]	O Framework deve ter um botão "Classe" que, ao ser clicado, será mostrado uma caixa exibindo três compartimentos, onde o usuário terá que informar: (i) nome da classe, (ii) atributos e (iii) métodos da classe	Essencial	[RF07] [RF08][RF09]
[RF03]	O Framework deve ter um botão "Associação" que, ao ser clicado, será mostrado opções para o usuário escolher o tipo de associação que deseja fazer	Essencial	[RF04][RF05]
[RF04]	O Framework deve permitir associações com multiplicidade: zero ou um (0..1) e apenas um ou um (1..1)	Essencial	[RF03][RF05]
[RF05]	O Framework deve permitir associações do tipo Agregação e/ou Composição	Desejável	[RF03][RF04]
[RF06]	O Framework deve ter um botão "Generalização" que, ao ser clicado, permita que uma classe selecionada possa herdar métodos, atributos e associações de outra classe posteriormente selecionada (herança)	Desejável	[RF13][RF15] [RF17]
[RF07]	O Framework deve permitir que o usuário defina a **visibilidade de cada classe** por meio de um botão de seleção com as opções: public (+), private (-), protected (#) e package (~)	Essencial	[RF02] [RF08][RF09]
[RF08]	O Framework deve permitir que o usuário defina a **visibilidade de cada atributo e de cada método** da classe por meio de um botão de seleção com as opções: public (+), private (-), protected (#) e package (~)	Essencial	[RF02] [RF07][RF09]
[RF09]	O Framework deve permitir que o usuário defina o **tipo de cada atributo e de cada método** por meio de um botão de seleção com as opções: int, float, double, boolean, date e String.	Essencial	[RF02]
[RF10]	O Framework deverá ter um botão "Gerar Código", que ao ser clicado, as classes dos diagramas irão ser transformados em	Essencial	[RF17]

ID	Descrição	Prioridade	Requisitos Relacionados
	códigos na linguagem PHP (por padrão)		
[RF11]	O Framework deve possuir mais de uma linguagem de programação disponível para que o desenvolvedor possa escolher a linguagem de sua preferência, por meio de um botão de seleção com as opções: PHP (padrão), Java ou Groovy	Desejável	[RF14][RF15] [RF16][RF17]
[RF12]	O Framework deve possuir a opção "Ajuda"	Essencial	-
[RF13]	Ao clicar no botão "Gerar Código", o usuário poderá escolher em que diretório o projeto será armazenado	Desejável	-
[RF14]	O Framework deve gerar um relatório em forma de texto (documento) a fim de mostrar o nome das classes, métodos, atributos e seus respectivos tipos	Desejável	[RF02][RF07] [RF08][RF09] [RF11]
[RF15]	Framework deverá organizar automaticamente o projeto gerado de acordo com o padrão MVC	Essencial	[RF10]
[RF16]	O Framework deve permitir a geração automática do Banco de Dados do Projeto na Linguagem SQL	Essencial	[RF15][RF17]
[RF17]	O Framework deve permitir a geração automática das operações CRUD das classes em PHP, por padrão	Essencial	[RF10][RF15] [RF16][RF17]
[RF18]	O tipo de associação padrão dada pelo Framework será simples com multiplicidade 1 ou 1 (1..1)	Essencial	[RF04]
[RF19]	O Framework deve ter um botão "Excluir" que, ao ser clicado, permita qualquer classe, associação, atributo ou método selecionado seja excluído a qualquer momento durante a modelagem	Importante	[RF02] [RF03]

Fonte: o autor (2018).

Tabela 8 - Requisitos não funcionais

ID	Descrição	Categoria	Escopo	Prioridade	Requisitos Relacionados
[RNF01]	O software será desenvolvido utilizando a versão mais recente da linguagem PHP	Manutenabilidade	Sistema e Funcionalidade	Essencial	-
[RNF02]	Os diagramas de classe devem ter estética parecida com a utilizada por Ferramentas CASE de modelagem disponível no mercado	Usabilidade	Sistema	Importante	[RFN03]
[RNF03]	Framework deve ser padronizado em relação à sua estética, levando em consideração cores, botões ou ícones	Usabilidade	Sistema	Importante	[RFN02] [RFN04] [RFN05] [RFN06] [RFN09]
[RNF04]	O Framework deve possuir cores primárias	Usabilidade	Sistema	Desejável	[RFN03]
[RNF05]	O Framework deve possuir uma fonte minúscula e arredondada	Usabilidade	Sistema	Desejável	[RFN06]
[RNF06]	O Framework deve possuir textos minúsculos	Usabilidade	Sistema	Desejável	[RFN05]
[RNF07]	O Framework funcionará em desktops e notebooks	Portabilidade	Sistema	Essencial	-
[RNF08]	O Framework informará quando o usuário cometer um erro através de mensagens	Usabilidade	Sistema	Importante	-

ID	Descrição	Categoria	Escopo	Prioridade	Requisitos Relacionados
[RNF09]	Os botões “Associação”, “Generalização”, “Classe” e “Gerar Código” poderão estar em forma de ícones	Usabilidade	Sistema	Desejável	[RFN03]

Fonte: o autor (2018).

As regras de negócios são premissas e restrições aplicadas em operações comerciais de uma empresa. Isso significa que para o negócio funcionar da maneira esperada essas regras devem ser respeitadas e atendidas. No contexto da computação, sabe-se que o software tem o intuito de automatizar as atividades de uma empresa ou instituição através de funcionalidades, definidas nos requisitos funcionais. Assim sendo, as regras de negócio definem como serão realizados esses requisitos funcionais, isto é, como as necessidades e exigências serão atendidas (VENTURA, 2016). A Tabela 9 mostra as regras de negócio (RN) para que o framework funcione da maneira esperada.

Tabela 9 - Regras de negócios

ID	Descrição	Categoria	Prioridade	Requisitos Relacionados
[RN01]	O Framework deverá possuir consistência entre diagrama e banco de dados	Integridade e Segurança	Essencial	-
[RN02]	Os códigos deverão ser gerados quando pelo menos um Diagrama for modelado	Funcionalidade	Essencial	[RN04]
[RN03]	Apenas Diagramas de Classes poderão ser modelados no Framework	Funcionalidade	Essencial	-
[RN04]	As classes apenas irão ser geradas em PHP se o usuário não escolher outra linguagem de programação	Funcionalidade	Desejável	[RN02]

É importante frisar que a respeito da prioridade dos requisitos, foram adotadas as seguintes denominações:

- **Essencial:** quando o requisito é imprescindível para o funcionamento do sistema, são prioritários, sem eles o sistema não funciona.
- **Importante:** o requisito não é imprescindível para o funcionamento do sistema porque o sistema funciona mesmo sem ele, porém de forma não satisfatória, por isso ele deve ser implementado.
- **Desejável:** quando o requisito não compromete as funcionalidades básicas do sistema, pois o sistema pode funcionar de forma satisfatória sem ele. Esse requisito pode ser implementado em versões posteriores caso não haja tempo hábil para implementá-los para esta versão especificada.

3.2 Modelagem

Conforme explicado na Seção 2.1.1, a modelagem pode ajudar a diminuir a complexidade de muitos problemas. Pressman (2016) afirma que modelagem é uma fase na qual os engenheiros de software esboçam modelos para entender melhor as necessidades do software.

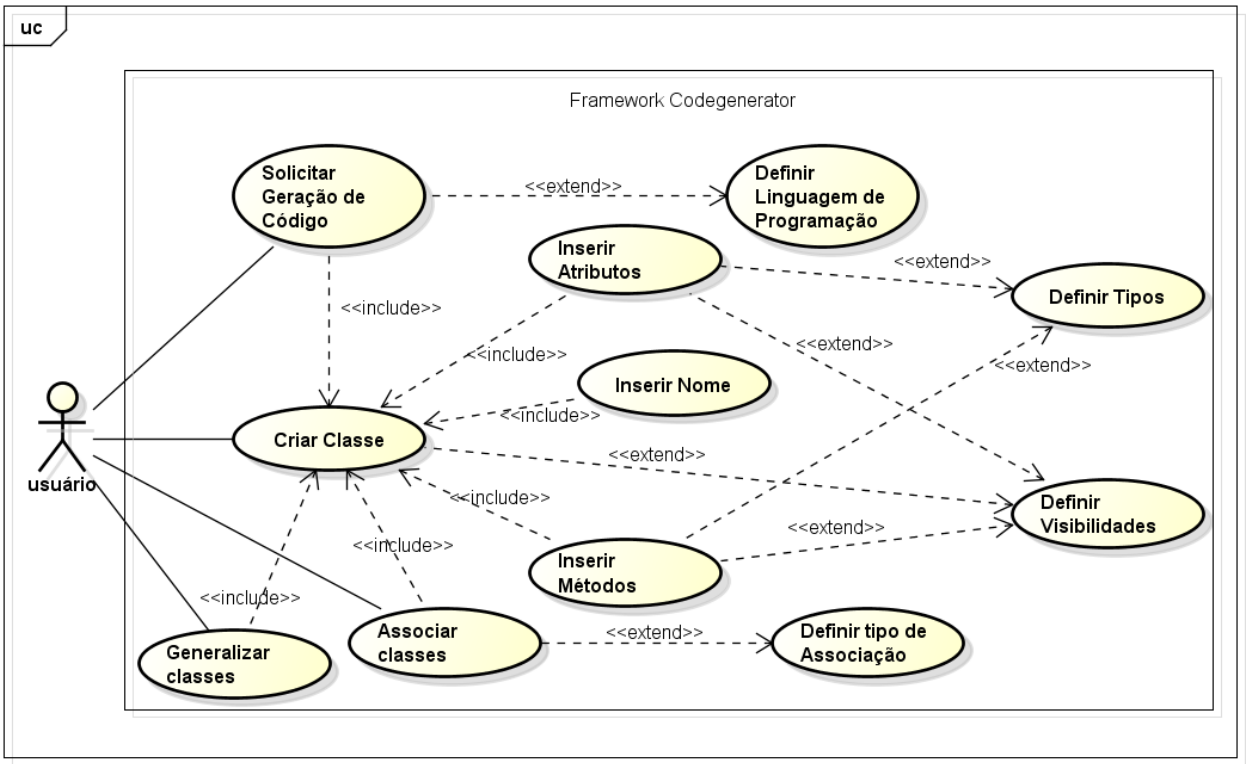
De acordo com Pressman (2016), no contexto da Engenharia de Software, podem haver duas classes de modelos: modelos de requisitos e modelos de projeto. Nos modelos de requisitos estão apresentados os requisitos do cliente. Nos modelos de projeto estão expressadas características do software que ajudam os desenvolvedores a compreender características mais concretas para que eles possam implementá-las. Nesse modelo pode-se encontrar detalhes dos componentes, interface de usuário e arquitetura do software. Nesse contexto, para este trabalho, foram esboçados os modelos de requisitos, onde se enquadram os diagramas da UML 2.0, e para o modelo de projeto, foi feita a arquitetura do software, visando oferecer especificação concreta para a construção do framework.

Assim sendo, a modelagem dos requisitos foi realizada baseando-se nas informações mostradas na Seção 3.1.3, utilizando os seguintes diagramas da UML 2.0, descritos na Seção 2.1.1: (*i*) Diagrama de Caso de Uso, (*ii*) Diagramas de Classes, (*iii*) Diagramas de Sequência e (*iv*) Diagramas de Atividades.

3.2.1 Diagrama de Caso de Uso

O Diagrama de Caso de Uso é bastante utilizado na fase de levantamento de requisitos, porque ele descreve de uma maneira simples, as expectativas que o usuário tem do software, em termos de funcionalidades e serviços. Assim sendo, a Figura 22 mostra o Diagrama de Caso de Uso modelado conforme os requisitos.

Figura 22 - Diagrama de Caso de Uso



Fonte: o autor (2018).

3.2.2 Diagramas de Classes

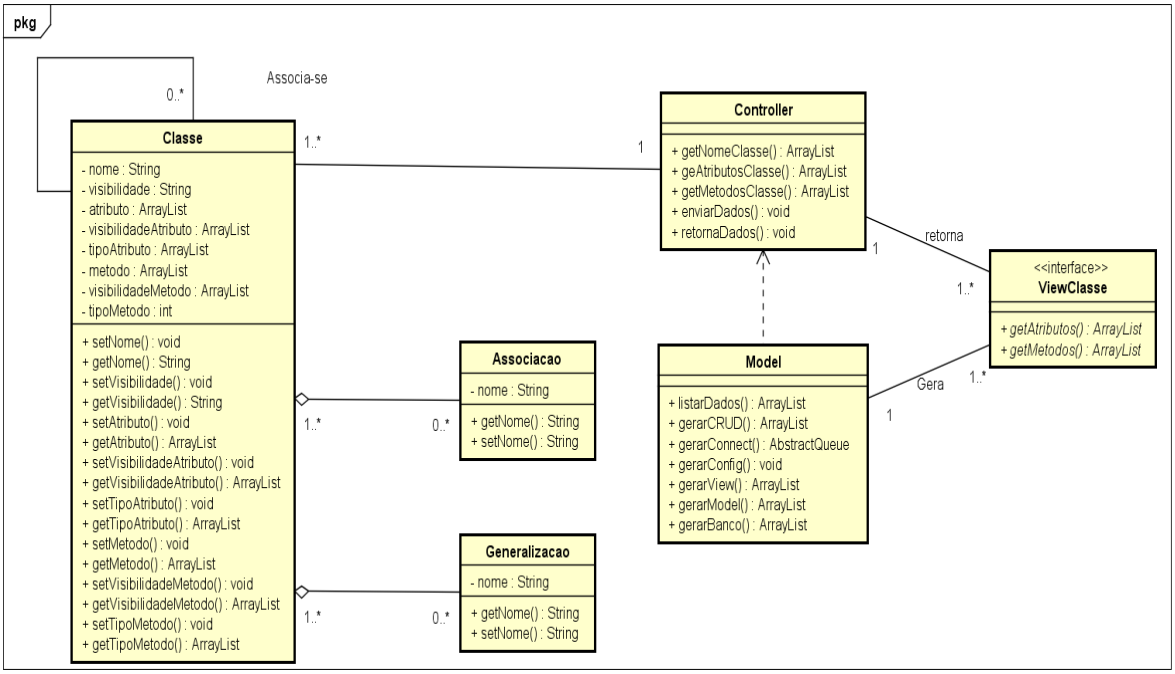
Bezerra (2015) afirma que de todos os diagramas da UML, esse diagrama é o mais rico em termos de notação, tanto que ele é usado desde o nível de análise até o nível de especificação. Segundo Guedes (2011) além de definir como as classes serão estruturadas, esse diagrama define também quais serão os métodos e os atributos que cada classe deverá possuir.

Esse diagrama define também como as classes estão relacionadas e como trocam informações entre si. Cada ocorrência de uma classe é denominada de objeto ou instância. São os objetos que se relacionam e é esse relacionamento que permite a troca de mensagens entre si, que ao final produzem as funcionalidades do sistema (BEZERRA, 2015; GUEDES, 2011).

Para representar os relacionamentos entre os objetos há um elemento no diagrama de classe denominado Associação. Geralmente, é representado por uma linha em seguimento de reta (BEZERRA, 2015). Mas existem tipos especiais de associação, dentre os quais podemos citar Agregação e Composição. A Agregação ocorre quando uma classe precisa ser complementada pelas informações contidas em determinados objetos de outra classe. O símbolo representado no diagrama de classe é um losango não preenchido na extremidade da

classe que contêm os objetos-todo. A composição ocorre quando uma classe precisa de outra para existir, assim sendo o símbolo representado no diagrama de classe é um losango preenchido na extremidade da classe que contêm os objetos-todo. Na Figura 23 (Diagrama de Classe do Framework) é possível perceber nos relacionamentos duas associações do tipo Agregação.

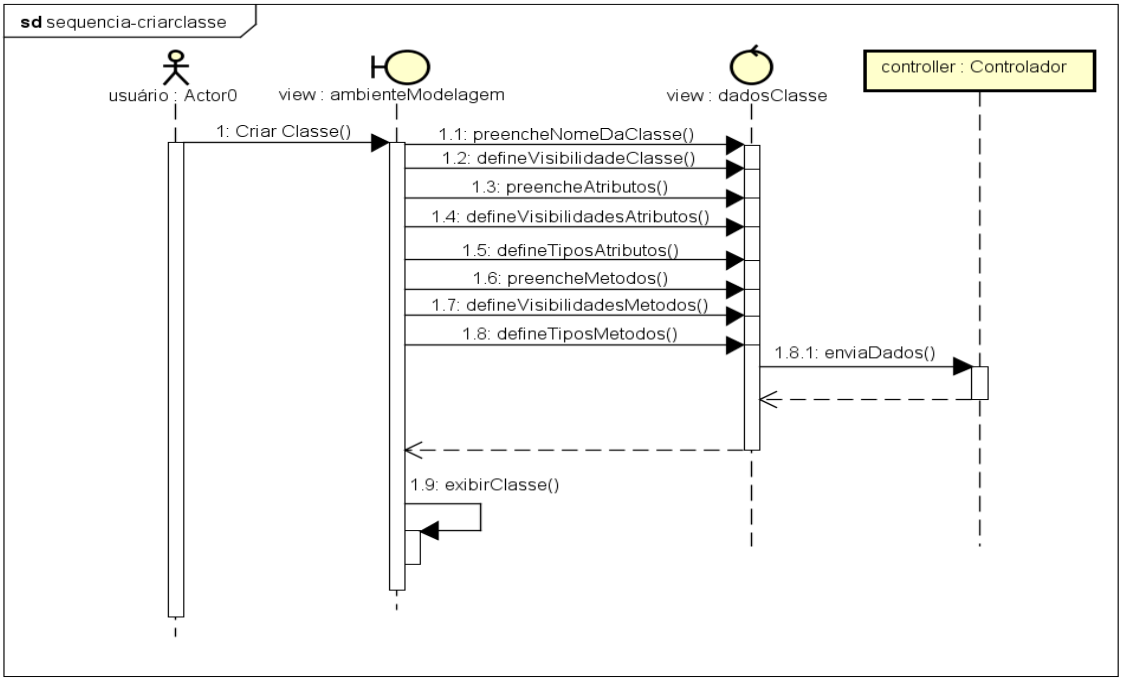
Figura 23 - Diagrama de Classes do Framework



3.2.3 Diagramas de Sequência

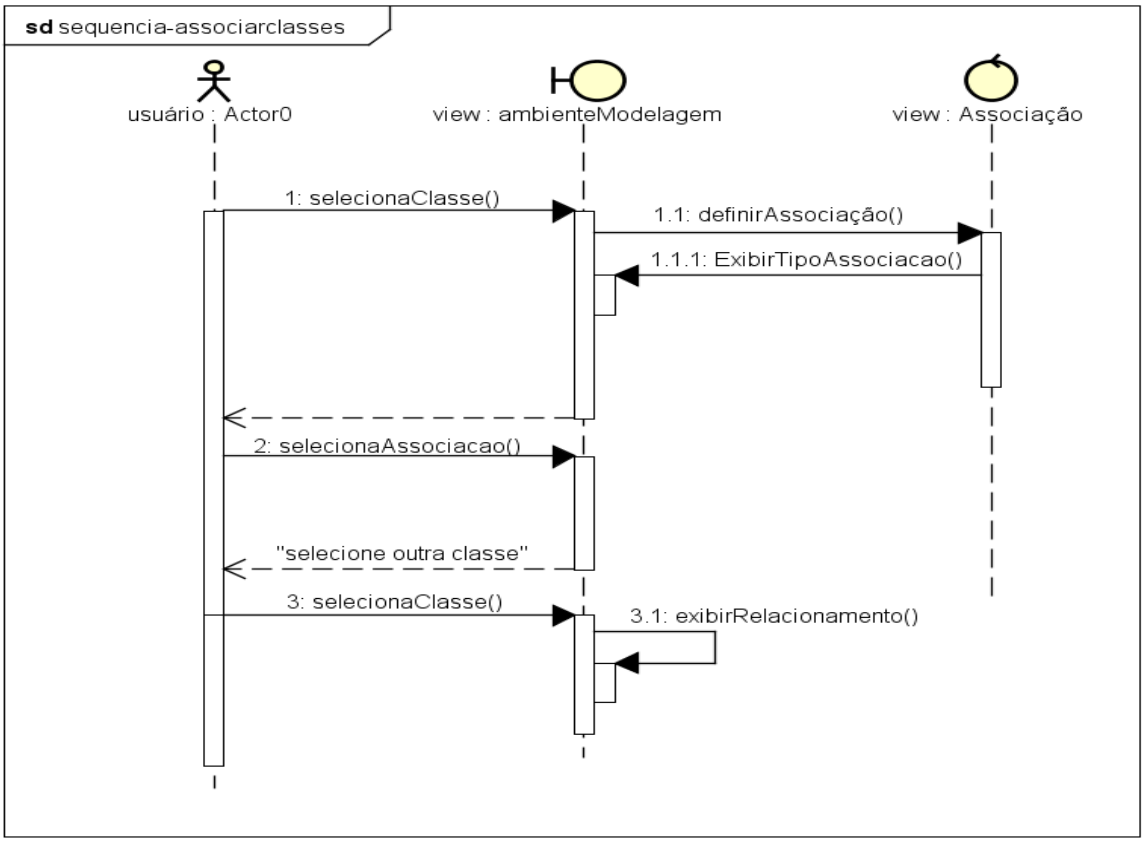
Os Diagramas de Sequência apresentam especificações comportamentais do sistema, preocupando-se com a ordem temporal das trocas de mensagens entre os objetos, identificando, geralmente em forma de atores, os eventos responsáveis por gerar os processos modelados, determinando como o processo deve se comportar até que o mesmo seja devidamente concluído, por meio de métodos disparados por mensagens e objetos (FILHO, 2012; GUEDES, 2011). Assim sendo, foram modelados os Diagramas de Sequência “Criar Classe” (Figura 24), “Associar Classes” (Figura 25), e “Gerar Código” (Figura 26)

Figura 24 - Diagrama de Sequência "Criar Classe"



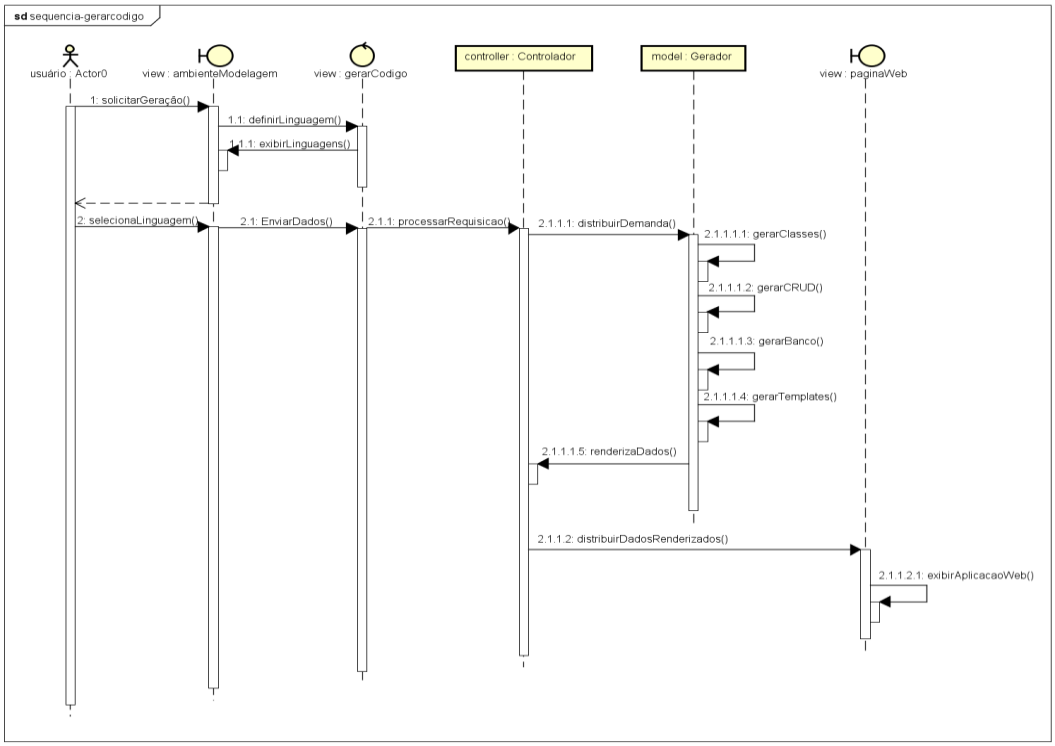
Fonte: o autor (2018).

Figura 25 - Diagrama de Sequência "Associar Classes"



Fonte: o autor (2018).

Figura 26 - Diagrama de Sequência "Gerar Código"

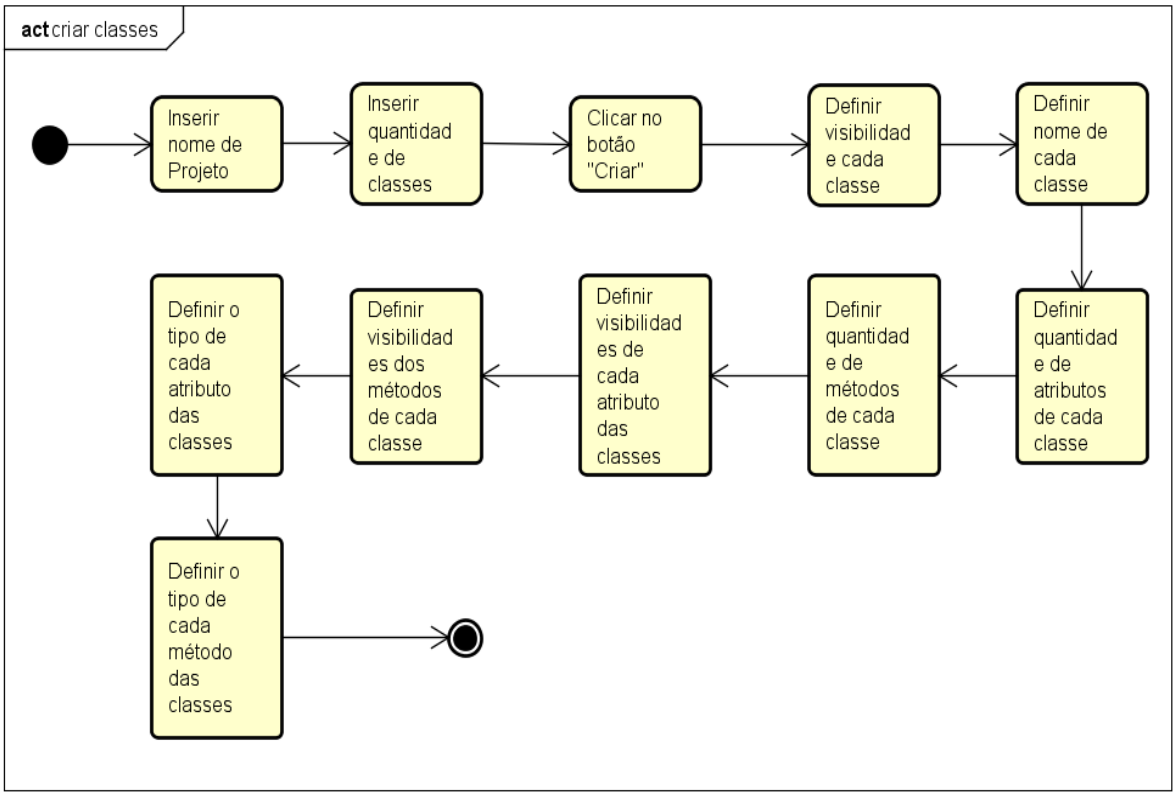


Fonte: o autor (2018).

3.2.4 Diagramas de Atividade

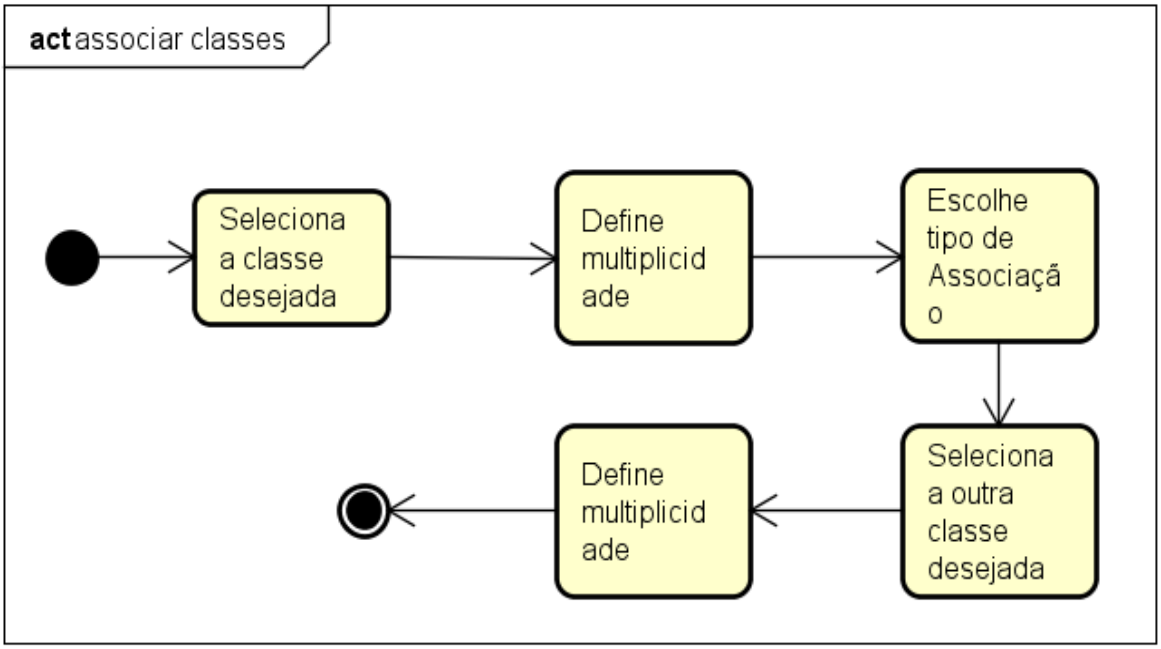
Os Diagramas de Atividades descrevem a ordem em que as atividades são realizadas e as dependências existentes entre elas. Esse diagrama apresenta as entradas e as saídas de atividades do fluxo de trabalho. Assim sendo, foi modelado os Diagramas de Atividades: “Criar classe” (Figura 27), “Associar classes” (Figura 28) e “Gerar código” (Figura 29).

Figura 27 - Diagrama de Atividade "Criar classe"



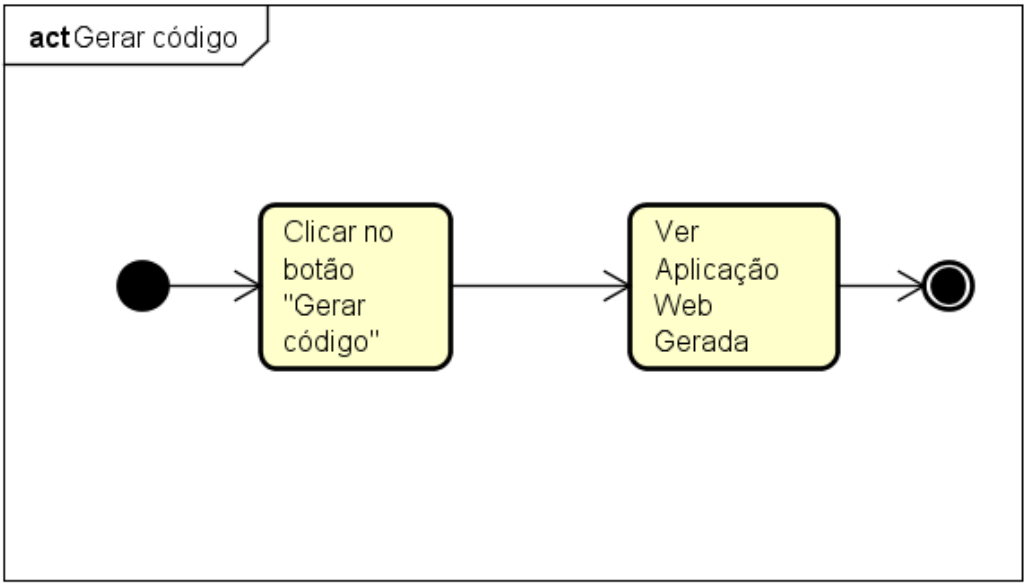
Fonte: o autor (2018).

Figura 28 - Diagrama de Atividade "Associar Classes"



Fonte: o autor (2018).

Figura 29 - Diagrama de Atividades "Gerar Código"



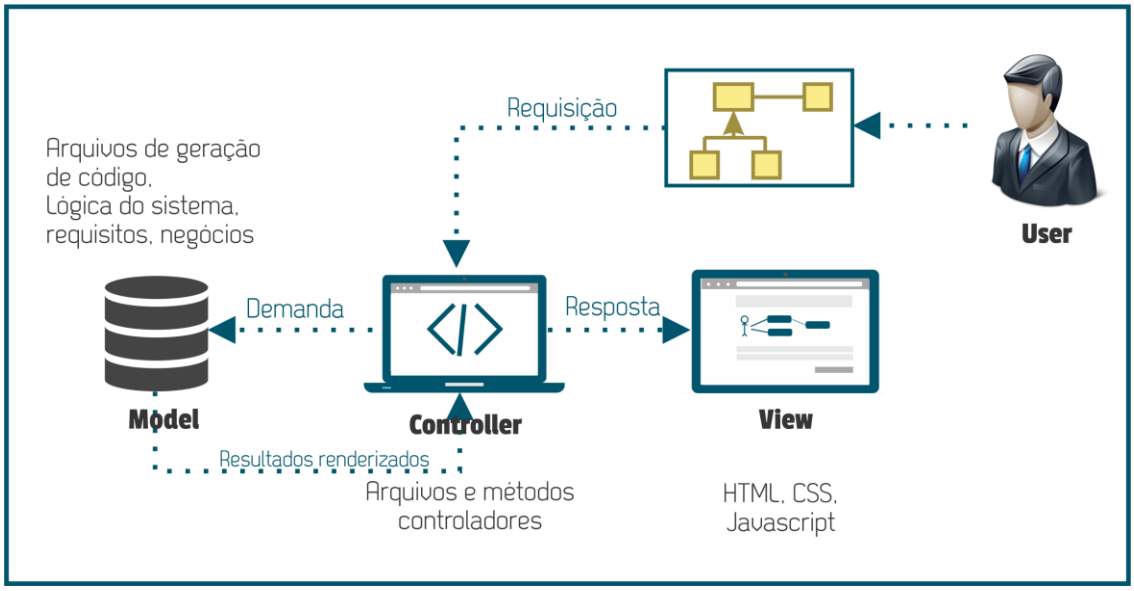
Fonte: o autor (2018).

3.2.5 Arquitetura

Pfleeger (2007) ressalta que além de ser importante para a implementação e os testes do sistema, uma arquitetura completa pode influenciar na rapidez e na eficiência da realização de possíveis alterações e manutenções no mesmo. Para Pressman (2016) a arquitetura de software é uma estrutura que apresenta os componentes do software, as propriedades externamente visíveis e o relacionamento entre eles.

Conforme explicado na Seção 2.1.2, o MDD utiliza os modelos como os principais artefatos no desenvolvimento do software, tornando-se uma estratégia diferenciada em virtude do alto nível de abstração na etapa de implementação. Desse modo, o usuário precisa interagir com uma interface, com características de um ambiente que possibilite a modelagem de diagramas de classe, que sejam transformados em códigos, de uma maneira bastante organizada. Assim, o padrão arquitetural utilizado nesse trabalho foi o MVC (descrito na Seção 2.2.4), por ser bastante vantajoso em termos de reaproveitamento de código e regras, facilidade de manutenção, facilidade na implementação de camadas de segurança, facilidade de atualização da interface da aplicação etc. A arquitetura de software que, segundo Pressman (2016) faz parte do modelo de projeto, foi desenhado com embasamento no padrão MVC, conforme mostrado na Figura 30.

Figura 30 - Arquitetura do sistema



Fonte: o autor (2018).

- **Controller:** é onde estão os métodos responsáveis pela interação entre as camadas View e Model. É nessa camada que os eventos ou ações dos usuários são processados. Por exemplo, quando o usuário cria as classes, métodos, atributos, definindo visibilidades e tipos na camada View, o Controller pega esses dados, processa e envia para a camada Model, que aplica a lógica definida, devolve ao Controller, que consequentemente envia os dados processados ao usuário pela camada View.
- **Model:** É onde estão os arquivos responsável pela lógica do sistema, a implementação dos requisitos funcionais e não funcionais e regras de negócio. É nessa camada que guarda os arquivos de geração de código das classes, CRUDs e banco de dados.
- **View**: camada onde estão todas as interfaces do framework, desde a interface que possibilita a modelagem de diagramas de classes, até a interface que apresenta as telas do sistema gerado a partir do diagrama.

4 IMPLEMENTAÇÃO DO FRAMEWORK JUNGLE

A codificação do Framework Jungle foi feita utilizando o IDE PhpStorm 2017.2 utilizando a linguagem de programação PHP Orientado a Objetos na versão 7.2.4, por existir diversas documentações disponíveis no levantamento bibliográfico, além de possuir uma diversos tutoriais e instruções na comunidade PHP e também por ser muito simples no desenvolvimento de aplicações utilizando os conceitos de orientação a objetos.

O PhpStorm apresenta uma interface simples, além agregar grande suporte para outros tipos de linguagens fundamentais para o desenvolvimento de Aplicações Web, tais como javascript, CSS, HTML. O PhpStorm é pago, todavia, foi utilizado uma licença de estudante cedida pela empresa JetBrains, gratuita por um período de um ano.

Além do PhpStorm, foi utilizada outras tecnologias, como o servidor web Apache versão 3.2.2 e o SGBD phpMyAdmin versão 4.8.0, além do framework Bootstrap 3.3.7, que auxiliam na construção de interfaces amigáveis por intermédios de arquivos CSS e Javascripts prontos. A Aplicação Web gerada pelo Jungle possui um padrão de interface totalmente baseada no Bootstrap 3.3.7.

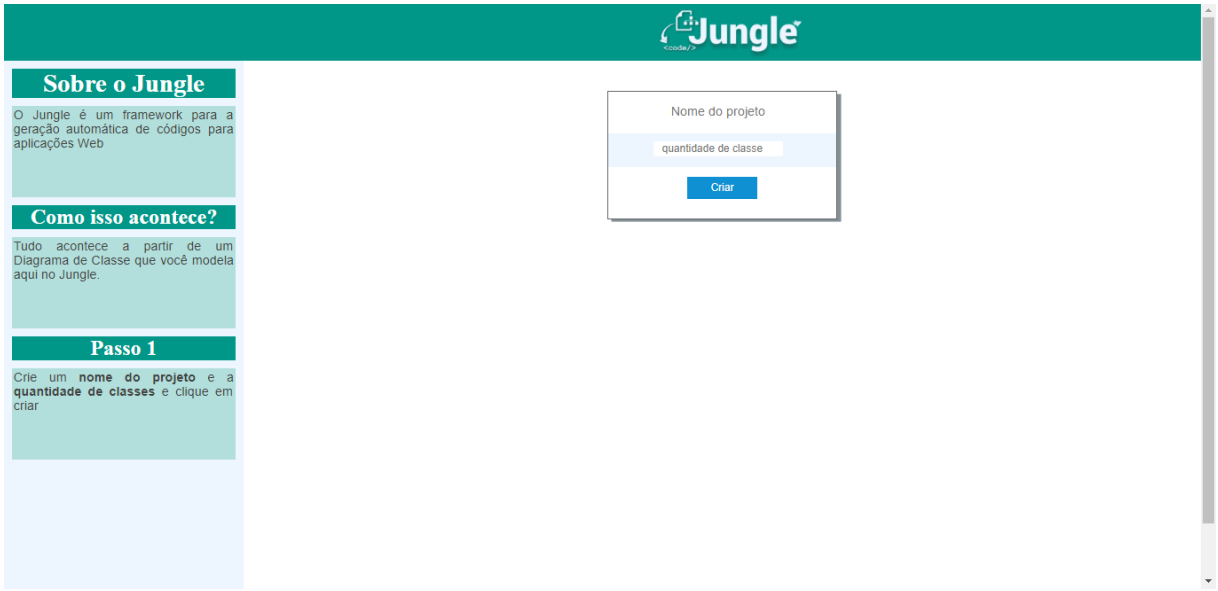
4.1 Telas do Framework

4.1.1 Tela Criar Projeto

O framework Jungle foi desenvolvido de acordo com o levantamento de requisitos. A tela inicial apresenta informações a respeito do que é o framework, como ele funciona e as instruções que o usuário deve fazer nessa primeira tela.

Nessa tela são mostrados dois campos: nome do projeto e quantidade de classes. Após preenchidos, o botão “Criar” deve ser clicado para ir ao segundo passo, conforme é mostrado na Figura 31.

Figura 31 - Tela Criar Projeto

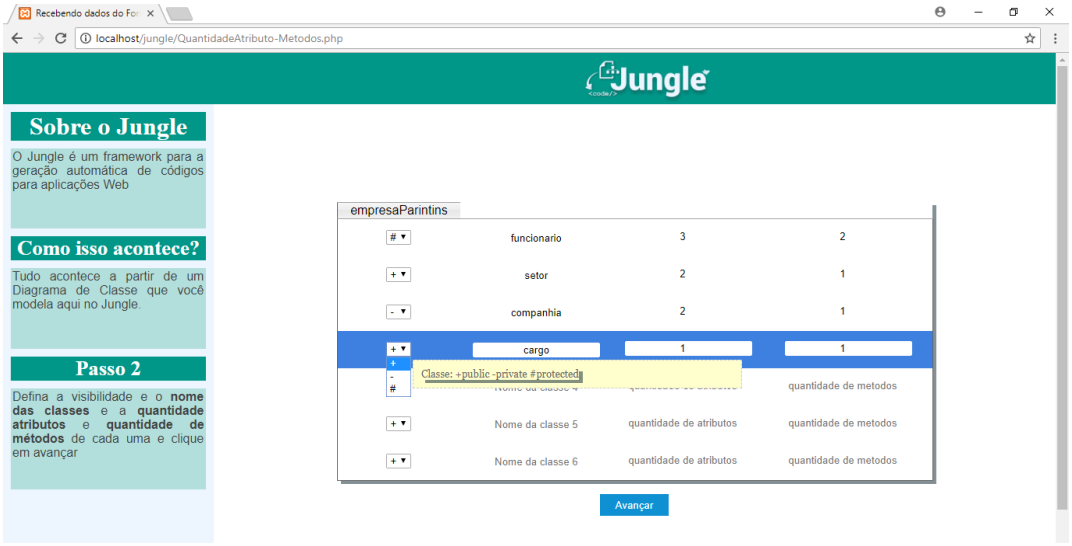


Fonte: o autor (2018).

4.1.2 Tela de Definição de Classes

Na tela de definição das classes há uma tabela com quatro colunas: (i) visibilidade: onde tem um botão de seleção com as opções (a) *public* (representado por "+"), (b) *private* (representado por "-") e (c) *protected* (representado por "#"). Quando o usuário passar o mouse por cima dos símbolos, há uma mensagem que diz o nível de restrição no qual o símbolo representa, (ii) nome da classe: o usuário define o nome de cada classe na coluna (iii) atributos: é definida a quantidade de atributos para cada classe e (iv) métodos: é definida a quantidade de métodos, conforme é mostrado na Figura 32.

Figura 32 - Tela de definição das classes

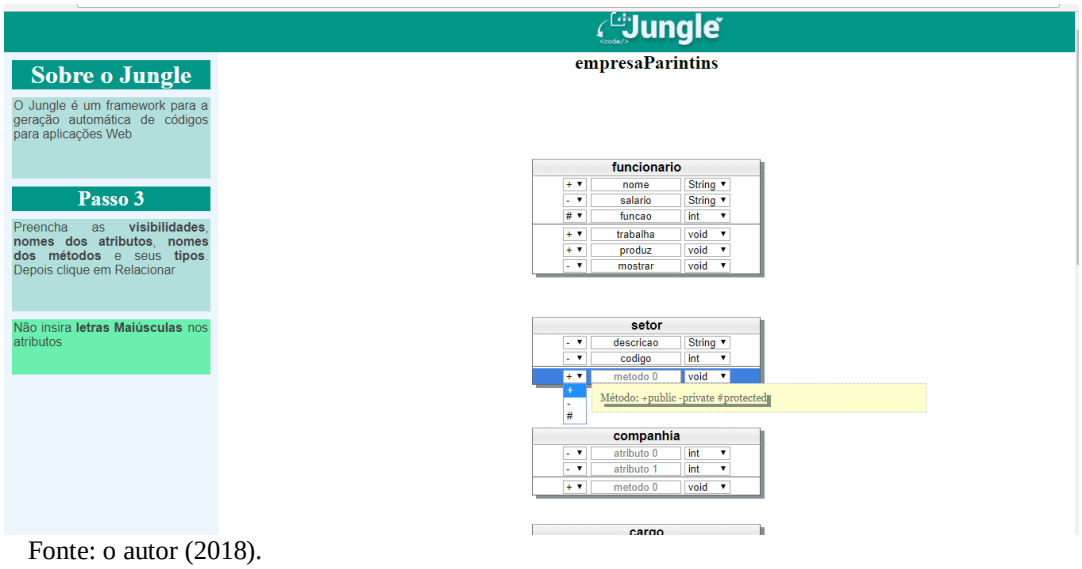


Fonte: o autor (2018).

4.1.3 Tela de Definição de Métodos e Atributos

Na tela de definição de métodos e atributos (Figura 33) é mostrado as classes em formas de pequenas tabelas semelhantes ao do diagrama de classes, com quantidade de campos “métodos” e “atributos” de acordo com o que foi definido na tela anterior. Para cada atributo ou método que o usuário definir, deve-se informar a sua visibilidade (*public*, *private* ou *protected*) e o seu tipo (int, string ou float). Isso é muito importante para a definição das regras de negócios que serão aplicadas na hora de gerar o banco de dados, para que ele seja consistente com o diagrama de classes. Após tudo definido o usuário deve clicar no botão “Relacionar”.

Figura 33 - Tela de definição de métodos e atributos

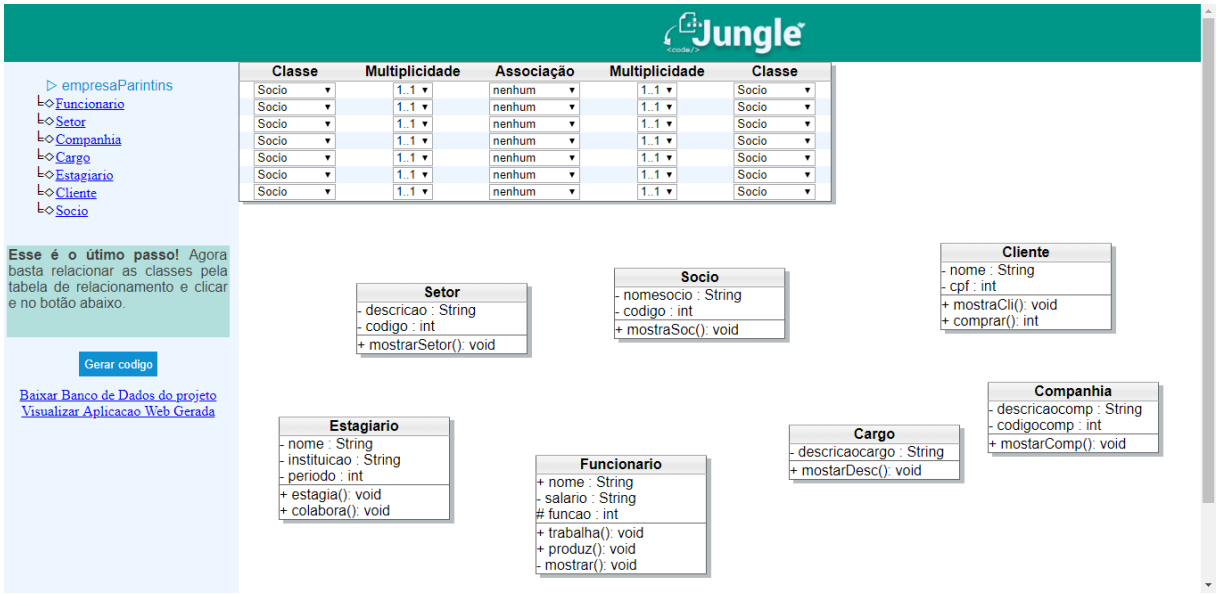


Fonte: o autor (2018).

4.1.4 Tela de Relacionamento entre Classes

A última tela (Figura 34) é onde são mostradas as instâncias do diagrama de classes, além de uma tabela onde devem ser definidos os relacionamentos do tipo (a) associação simples, (b) agregação e (c) composição, além da multiplicidade de cada classe relacionada. No menu lateral esquerdo é mostrado a estrutura das classes e um botão “Gerar Código” que deve ser pressionado quando o usuário finalizar o relacionamento. Após clicar neste botão, será mostrado nesta tela dois links: (i) Baixar Banco de Dados do projeto e (ii) Visualizar Aplicação Web Gerada, que serão melhor detalhados nas seções abaixo.

Figura 34 - Tela de Relacionamento entre classes



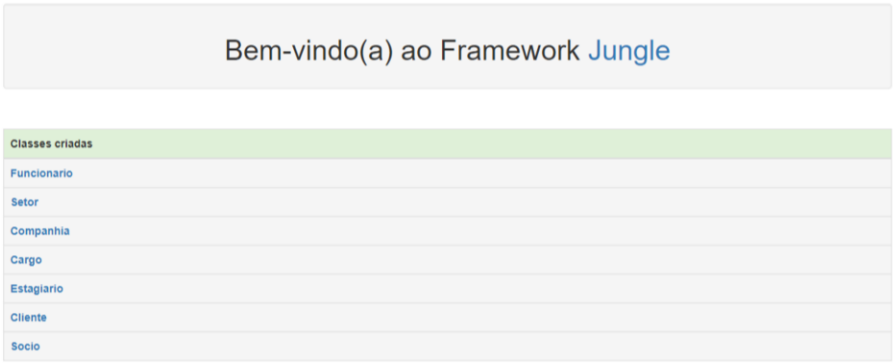
Fonte: o autor (2018).

4.2 Aplicação Web Gerada pelo Jungle

4.2.1 Tela: Visualizar Aplicação Web Gerada

No link denominado “Visualizar Aplicação Web Gerada” quando clicado abre uma nova aba no navegador com a página inicial da Aplicação Web, gerada pelo framework Jungle, contendo uma lista com os nomes das classes (em forma de links) criadas pelo usuário, conforme mostrado na Figura 35.

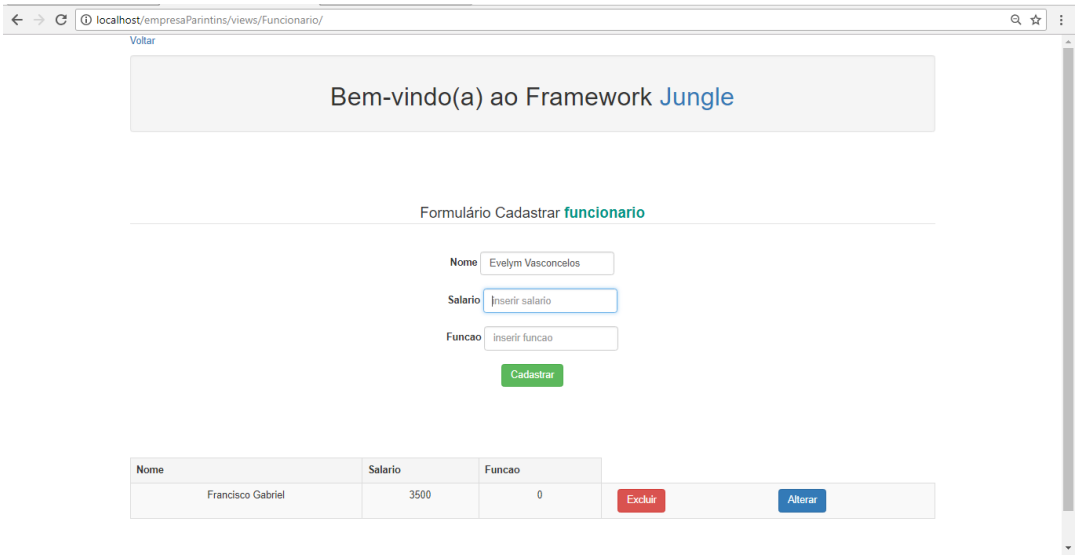
Figura 35 - Página inicial da Aplicação Web



Fonte: o autor (2018).

Ao clicar em qualquer uma das opções dos nomes das classes, o usuário é redirecionado para a tela de cadastro de dados, onde pode ser realizado as seguintes operações: visualizar, cadastrar, alterar e excluir, como mostrado na Figura 36 (neste exemplo, o usuário clicou na opção "funcionário" mostrado na Figura 35, derivado da classe "Funcionário" mostrado na Figura 34).

Figura 36 - Tela para cadastro de dados



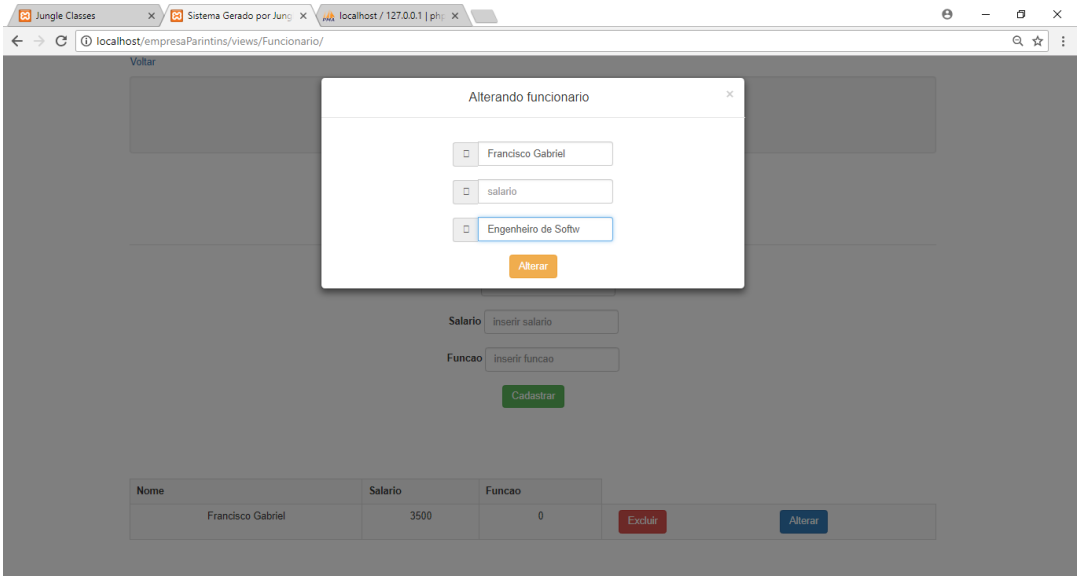
Fonte: o autor (2018).

É importante ressaltar que para que as operações citadas acima sejam realizadas sem erro, é necessário importar o banco de dados gerado pelo framework, conforme discutido na seção abaixo.

Os dados cadastrados na Aplicação Web são salvos diretamente no banco de dados. Caso, o usuário queira alterar algum dado, ele deve clicar no botão "Alterar" que carrega um script com um formulário de alteração daquele determinado dado. No exemplo da Figura 37, na página "funcionário", quando o usuário deseja fazer uma alteração, os dados são

carregados dentro de uma tela “Alterando funcionário” que tem os campos com os dados atuais carregados diretamente do banco de dados. Após ele atualizar os dados desejados, basta finalizar clicando no botão laranja “Alterar”.

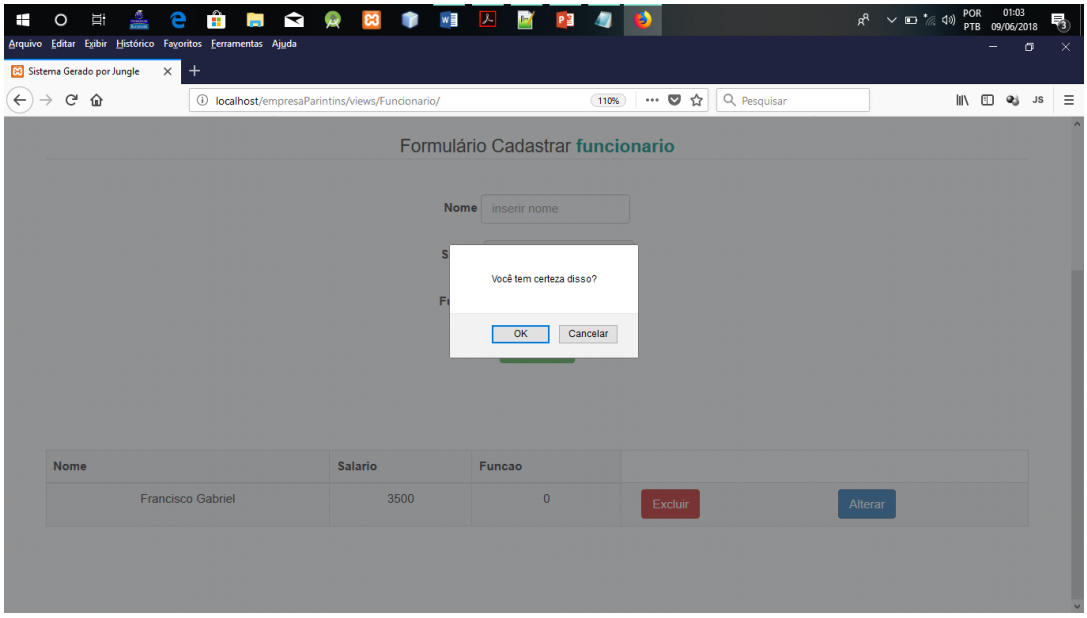
Figura 37 - Tela para atualização de dados



Fonte: o autor (2018).

Para o usuário excluir dados da tabela, basta ele clicar no botão “Excluir” na linha que possuir os dados que ele deseja deletar. Ao clicar em “Excluir”, é carregado um script de confirmação com a pergunta “Você tem certeza disso? ”, com as opções: (i) “ok”: para confirmar exclusão e (ii) “cancelar” para desistir da exclusão. Isso é mostrado na Figura 38.

Figura 38 - Tela de exclusão de dados

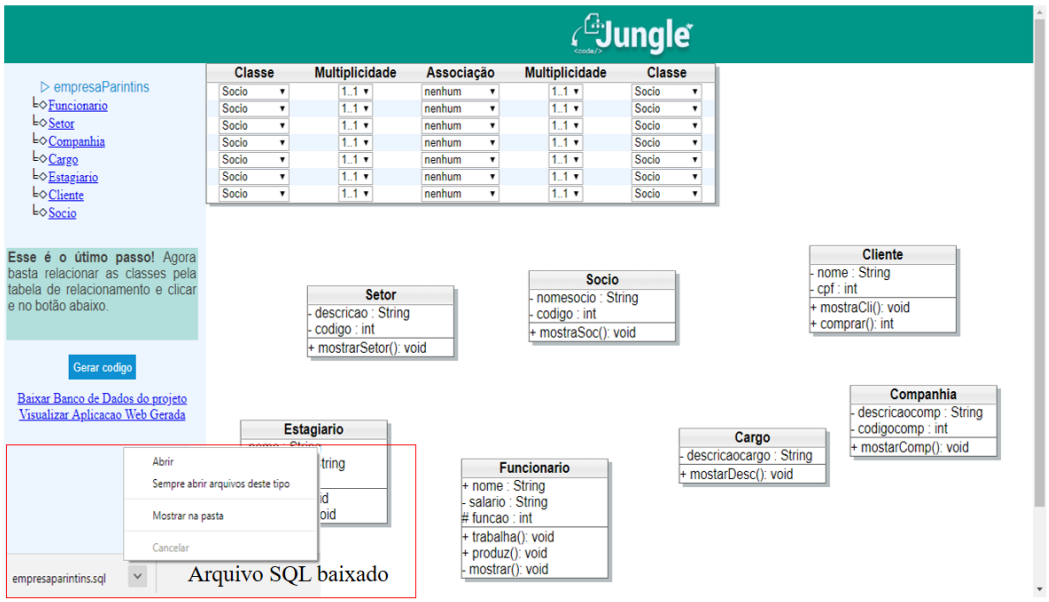


Fonte: o autor (2018).

4.2.2 Tela: Baixar Banco de Dados do Projeto

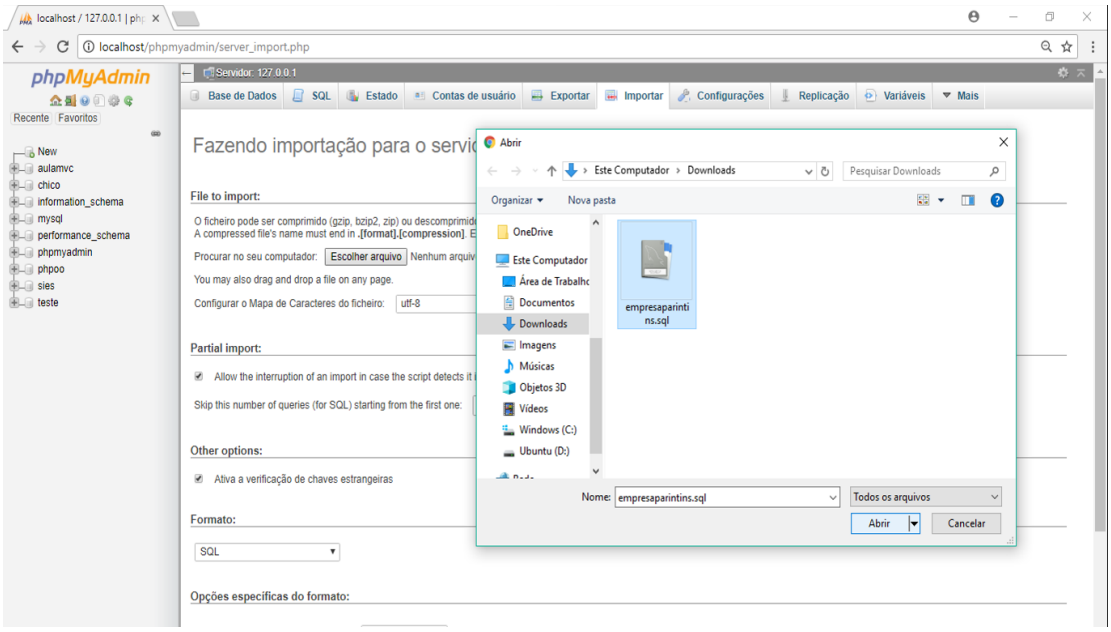
No link denominado “Baixar Banco de Dados do Projeto” é possível baixar o banco de dados criado na pasta do projeto. Dessa forma, é gerado um arquivo com extensão .sql que deve ser importado para o SGBD PhpMyAdmin. A Figura 39 mostra o arquivo SQL baixado e na Figura 40 é possível observar o procedimento de importação no PhpMyAdmin.

Figura 39 - Download do arquivo SQL



Fonte: o autor (2018).

Figura 40 - Importação do Banco de Dados

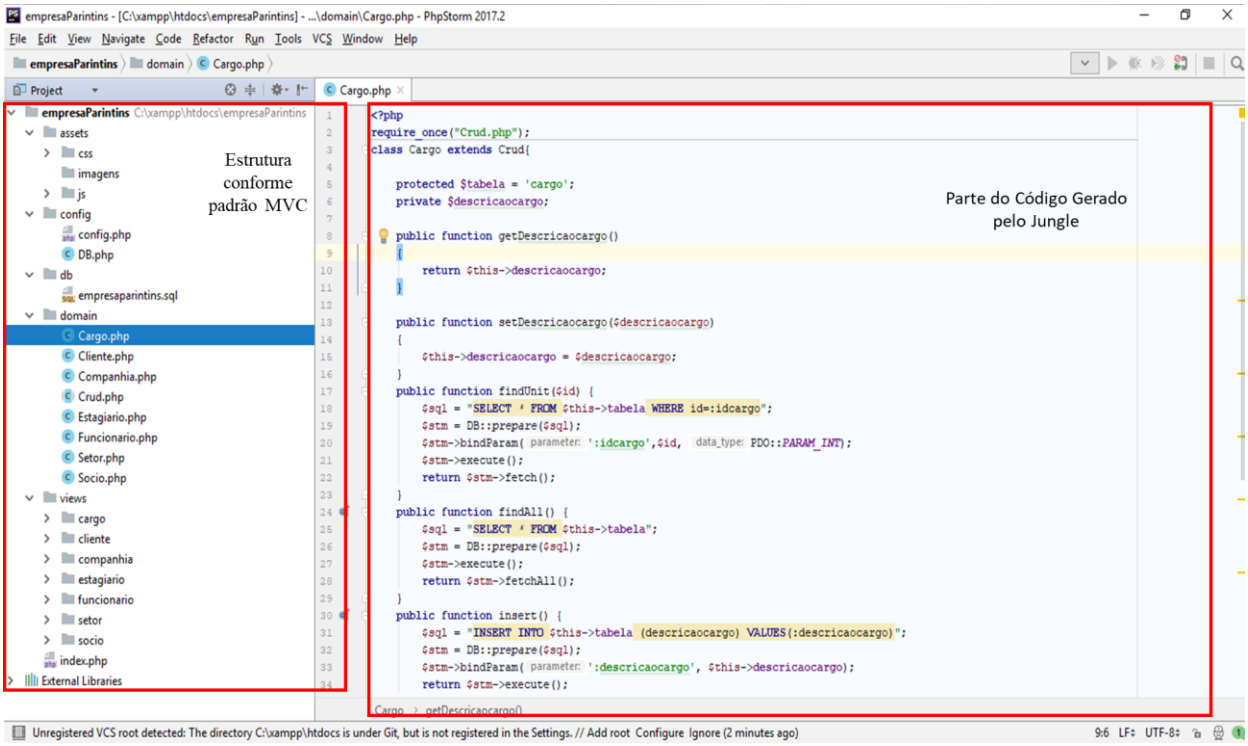


Fonte: o autor (2018).

4.2.3 Estrutura do Projeto Gerado pelo Jungle

O projeto gerado no Jungle é organizado conforme o padrão arquitetural MVC descrito nas Seções 2.2.4 e 3.2.5. A Figura 41 mostra um exemplo desse padrão para o projeto "empresaParintins" que possui diretórios com os modelos de domínio na pasta "domain", representando a camada *Model*. Cada modelo de domínio tem uma *view* com o seu nome no diretório "views", representando a camada *Views.* Os arquivos da camada *Controller* estão dentro da pasta "assets/js" possuindo scripts controladores. Espera-se que utilizando esse padrão haja bastante vantagem em termos de reaproveitamento de código e regras, além de facilidades de manutenção, de implementação de camadas de segurança, de atualização da interface da aplicação, etc.

Figura 41 - Estrutura de projeto no PhpStorm



Fonte: o autor (2018).

4.3 Avaliação de Funcionalidade e Usabilidade

A avaliação de funcionalidade visa encontrar as não conformidades do software em relação aos requisitos do sistema. Essa avaliação é realizada de forma rápida e eficiente, por meio de um conjunto de atividades de testes. Isso é essencial para o aumento da confiança sobre o software, além de poder garantir a qualidade do produto, principalmente se as

funcionalidades estiverem de acordo com especificações e requisitos do sistema (BRUNO, SILVA e ALVES, 2018).

A avaliação de usabilidade visa verificar o desempenho de um software no que se refere a sua eficiência e eficácia quanto a interface homem-computador, tendo em vista a obtenção de indícios que demostrem o grau de satisfação do usuário, identificando também possíveis problemas de usabilidade durante a realização de atividades para que os mesmos sejam posteriormente corrigidos (MIRANDA e MORAES, 2003).

Segundo Miranda e Moraes (2003), essa técnica de projeto participativo tem muitas vantagens, dentre elas, podemos citar a (i) possibilidade de trabalhar com protótipos ao invés da interface final, (ii) facilidade de utilização devido esse procedimento ter um baixo custo e poder ser aplicado a qualquer pessoa e (iii) feedback do que deverá ser reformulado ou mantido.

4.3.1 Planejamento da Avaliação

- Definição dos Participantes

Foram convidados profissionais e estudantes da área da computação para avaliar o Framework Jungle. Os estudantes dos cursos de Engenharia de Software e Sistema de Informação do ICET/UFAM se disponibilizaram a fazer a avaliação de usabilidade, pois eles atendiam ao perfil de avaliador definido nessa pesquisa que era:

- ✓ Possuir conhecimento sobre Diagramas da UML.
- ✓ Possuir alguma experiência na construção de Aplicações Web.
- ✓ Possuir pelo menos uma experiência na utilização de frameworks na construção de Aplicações Web.

- Definição da Instrumentação

Foi utilizado um questionário de avaliação dividido em três etapas: (i) perfil do participante, (ii) lista de atividades das funcionalidades e (iii) questionário de usabilidade (veja a Figura 43). Antes da aplicação do questionário, os participantes tiveram que assinar um termo de consentimento livre e esclarecido (veja a Figura 42).

Figura 42 - Termo de consentimento livre e esclarecido

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Avaliação do Software "Jungle: Um Framework que possibilita a geração automática de códigos para Aplicações Web"

Prezado (a) Participante,

Meu nome é Francisco Gabriel Teixeira Marinho, sou aluno da Universidade Federal do Amazonas e curso Engenharia de Software. Estou realizando uma pesquisa como resultado do meu Trabalho de Conclusão de Curso, sob a orientação da Profa. Dra. Odette Mestrinho Passos, cujo **objetivo é o desenvolvimento de um framework que possibilite a geração automática de modelos de alto nível (diagrama de classe da UML) em modelos mais concretos (código fonte) de produtos de software direcionados para aplicações web, alinhado a boas práticas de Engenharia de Software.**

Sua participação envolve em utilizar em utilizar o Framework Jungle e responder um questionário para avaliar esse software. Todo esse processo levará cerca de 20 minutos. Sua contribuição é muito importante para a conclusão final desse trabalho.

A sua participação é voluntária e você poderá desistir a qualquer momento. Na publicação dos resultados desta pesquisa sua identidade será mantida no mais rigoroso sigilo. Serão omitidas todas as informações que permitem identificá-lo(a).

☐ Aceito em participar deste estudo

Em: ____/____/____ Assinatura: _____

Fonte: O autor (2018).

Figura 43 – Questionário de Avaliação

Poder Executivo
Ministério da Educação
Universidade Federal do Amazonas
Instituto de Ciências Exatas e Tecnologia

UFAM

AVALIAÇÃO DO JUNGLE

Descrição: Framework que possibilita a geração automática de códigos para Aplicações Web.

Objetivo: O objetivo do Jungle é possibilitar a geração automática de modelos de alto nível (diagrama de classe da UML) em modelos mais concretos (código fonte) de produtos de software direcionados para aplicações web, alinhado a boas práticas de Engenharia de Software.

Esta avaliação possui um tempo máximo de 20 minutos e é composta de 3 etapas: 1) Perfil do participante, 1) Lista de Atividades e 2) Questionário de Avaliação.

1) PERFIL DO PARTICIPANTE

1) Você é um(a):
( ) Profissional da área da computação ( ) Estudante da área da computação

2) Você possui conhecimento sobre Diagramas da UML?

_____

3) Você possui alguma experiência na construção de Aplicações Web?

_____

4) Você possui pelo menos uma experiência na utilização de frameworks na construção de Aplicações Web ?

_____

Poder Executivo
Ministério da Educação
Universidade Federal do Amazonas
Instituto de Ciências Exatas e Tecnologia

2) LISTA DE ATIVIDADES DAS FUNCIONALIDADES

Realize as atividades a seguir ao abrir o Jungle no navegador usando a url "localhost/jungle"

1° Passo: Crie um projeto definindo a quantidade de classes que desejar

2° Passo: Defina em cada classe:

- A visibilidade da classe
- O nome da classe
- Quantidade de atributos
- Quantidade de métodos

3° Passo: Preencha em cada classe:

- Visibilidades dos atributos
- Nomes dos atributos
- Tipos dos atributos
- Visibilidades dos Métodos
- Nomes dos Métodos
- Tipos dos Métodos

4° Passo: Relacione as classes e clique em **"Gerar código"**

5° Passo: Para visualizar a sua Aplicação Web, clique no link "Visualizar Aplicacao Web Gerada"

6° Passo: Para usar sua aplicação, você deve importar o banco de dados gerado no phpMyAdmin (use a url:"localhost/phpmyadmin/"para acessar no navegador), você pode baixar ele no link "Baixar Banco de Dados do projeto"

7° Passo: Cadastre, Visualize Altere e Exclua dados em quantas classes você desejar na sua Aplicação Web.

ERROS: _____

_____

_____

_____

_____

_____

Poder Executivo
Ministério da Educação
Universidade Federal do Amazonas
Instituto de Ciências Exatas e Tecnologia

UFAM

3) QUESTIONÁRIO DE USABILIDADE

Responda as questões a seguir de acordo com a legenda:

1 – Discordo Totalmente 2 – Discordo 3 – Neutro
4 – Concordo 5 – Concordo Totalmente

Questões	1	2	3	4	5
1) A interface do Jungle é amigável					
2) O Jungle é bem sucedido na realização das funções que se propõe a realizar					
3) O Jungle pode auxiliar o desenvolvedor web na construção de Aplicações Web					
4) Foi fácil modelar o Diagrama de Classes no Jungle					
5) Foi fácil gerar códigos no Jungle					
6) O Jungle consegue ter clareza nos comandos facilitando sua utilização					
7) A Aplicação Web gerada pelo Jungle permite o cadastro, a visualização, a alteração e a exclusão de dados					
8) Achei este framework muito inconsistente					
9) Imagino que os desenvolvedores aprenderiam a usar o Jungle rapidamente					
10) Eu me senti confortável com este framework					
11) Foi fácil encontrar a informação que eu precisava					
12) A organização de informações na tela do Jungle é clara					
13) Estou satisfeito(a) com o funcionamento do Jungle					

O que pode ser acrescentado em uma versão futura que possa contribuir para a melhoria do Jungle?

_____
_____
_____
_____
_____
_____
_____

Desenvolvedor (a): _____

Data da Avaliação: ____ / ____ / ____

Fonte: o autor (2018).

4.3.2 Resultado da Avaliação

O Framework Jungle foi instalado e avaliado no notebook de um projeto da UFAM denominado Projeto Jovens Doutores (PJD). Após as configurações necessárias, o participante recebeu o questionário de avaliação com a lista de atividades que ele deveria fazer usando o framework. Assim sendo, cada participante modelou o diagrama de classe e

testou a Aplicação Web em um período que durou de 5 a 10 minutos, variando conforme a quantidade de classes que ele criou. No total, participaram da avaliação 6 alunos desenvolvedores, sendo 3 deles estagiários em empresas ou instituições que trabalham com desenvolvimento web. Os resultados obtidos estão descritos abaixo.

- **Perfil dos Participantes**

A Tabela 10 apresenta o perfil dos participantes. Pode-se perceber que 100% eram estudantes da área da computação, 83.3% possuíam conhecimentos sobre diagramas da UML e 16.6% possuía pouco conhecimento. 100% possuíam alguma experiência na construção de Aplicações Web e 100% disseram que possuíam pelo menos uma experiência na utilização de frameworks na construção de Aplicações Web.

Tabela 10 - Perfil dos participantes

Questões	Aluno1	Aluno2	Aluno 3	Aluno 4	Aluno 5	Aluno 6
Questão 1	Estudante	Estudante	Estudante	Estudante	Estudante	Estudante
Questão 2	Sim	Pouco	Sim	Sim	Sim	Sim
Questão 3	Sim	Sim	Sim	Sim	Sim	Sim
Questão 4	Sim	Sim	Sim	Sim	Sim	Sim

Fonte: o autor (2018).

- **Lista de Atividades das Funcionalidades**

Com relação as atividades das funcionalidades, tudo funcionou de modo esperado. Entretanto, 3 dos 6 participantes fizeram comentários a respeito dos erros/falhas encontrados, os quais citaram: (i) “A opção alterar não funcionou na aplicação web”, (ii) “Não alterou” e (iii) “Tudo funcionou da melhor forma, a partir da criação das classes gerou perfeitamente o código. Muito show! ”.

- **Questionário de Usabilidade**

A Tabela 11 mostra o resultado obtido na avaliação de usabilidade seguido do número de respostas por questões.

Tabela 11 - Número de respostas do questionário de usabilidade

Questões	Discordo Totalmente	Discordo	Neutro	Concordo	Concordo Totalmente
1) A interface do Jungle é amigável	-	-	1	2	3
2) O Jungle é bem sucedido na realização das funções que se propõe a realizar	-	-	-	3	3
3) O Jungle pode auxiliar o desenvolvedor web na construção de Aplicações Web	-	-	-	1	5
4) Foi fácil modelar o Diagrama de Classes no Jungle	-	-	2	3	1

Questões	Discordo Totalmente	Discordo	Neutro	Concordo	Concordo Totalmente
5) Foi fácil gerar códigos no Jungle	-	-	-	2	3
6) O Jungle consegue ter clareza nos comandos facilitando sua utilização	-	-	1	4	1
7) A Aplicação Web gerada pelo Jungle permite o cadastro, a visualização, a alteração e a exclusão de dados	-	-	1	2	3
8) Achei este framework muito inconsistente	6	-	-	-	-
9) Imagino que os desenvolvedores aprenderiam a usar o Jungle rapidamente	-	-	1	2	3
10) Eu me senti confortável com este framework	-	-	-	3	3
11) Foi fácil encontrar a informação que eu precisava	-	-	1	5	-
12) A organização de informações na tela do Jungle é clara	-	-	3	2	1
13) Estou satisfeito(a) com o funcionamento do Jungle	-	-	-	3	3

Fonte: o autor (2018).

Como resultado, pode-se concluir que a primeira versão do Framework teve um bom desempenho de acordo com a opinião dos avaliadores, porque ele possui as seguintes características:

- Consistência: 100% dos participantes da avaliação não acharam o framework inconsistente, marcando a opção discordo totalmente;
- Facilidade em usar: 83,3% dos participantes concordaram que foi fácil encontrar a informação que precisavam;
- Facilidade em gerar código: 66,67% dos participantes concordaram totalmente que foi fácil gerar códigos no Jungle;
- Clareza nos comandos: 66.67% dos participantes concordaram que o Jungle consegue ter clareza nos comandos facilitando sua utilização.

Além disso, é possível perceber que 66.67% dos entrevistados ficaram neutros sobre a afirmação que a organização de informações na tela do Jungle é clara. Isso demostra que deve haver muitas melhoras nas versões posteriores, no que tange a interface e a organização das informações na tela. Todavia, no que tange as funcionalidades do framework, o resultado foi satisfatório, pois 83,3% dos avaliadores concordaram totalmente que o Jungle pode auxiliar o desenvolvedor Web na construção de Aplicações Web.

A respeito do que poderia ser acrescentado em uma versão futura que possa contribuir para a melhora do Jungle, 66.67% sugeriram algumas melhorias. Dentre as sugestões, pode-se citar:

- Criar um botão para baixar o banco de dados.
- O Jungle poderia importar o banco de dados automaticamente no SGBD.
- Melhorar design do framework e criar um botão de ajuda que redirecione o usuário para uma maior quantidade de informações referentes ao framework.
- Permitir que os relacionamentos entre as classes sejam feitos de forma mais dinâmica.

Após a aplicação da lista de atividades da funcionalidade, foram realizadas algumas alterações no código-fonte referentes a algumas falhas reportadas durante a avaliação dos 2 primeiros participantes que não conseguiram realizar a atividade de alteração de dados na Aplicação Web. Em função disso, nas 4 últimas avaliações todos os participantes conseguiram realizar as atividades das funcionalidades sem qualquer erro ou falhas.

5 CONCLUSÃO E PERSPECTIVAS FUTURAS

5.1 Considerações Finais

O MDD é uma abordagem de desenvolvimento de software que define a modelagem como a artefato principal do processo de desenvolvimento ao invés do código fonte. Não é à toa que MDD está despertando cada vez mais a atenção da indústria e de comunidades de pesquisa, pois além de permitir que o desenvolvedor construa software desenhando as funcionalidades e recursos desejados de um sistema, ao invés de utilizar linguagens de programação, tornando “o desenho” no artefato central do processo desenvolvido ao invés do código fonte, ele ainda agrega produtividade, portabilidade, menor custo, facilidade de evolução do software, geração automática de código e maior qualidade.

Neste sentido, o objetivo desse trabalho foi desenvolver um framework que possibilite a geração automática de modelos de alto nível (diagrama de classe da UML) em modelos mais concretos (código fonte) de produtos de software direcionados para aplicações web, alinhado a boas práticas de Engenharia de Software.

Para se ter uma base do que deveria ser implementado foi realizado um levantamento de requisitos com estudantes e profissionais da área da computação, atuantes em projetos de desenvolvimento tanto na universidade quanto em empresas. Eles puderam opinar sobre a concepção do framework.

No Framework Jungle é possível modelar diagramas de classe e gerar códigos de forma automática. A criação das classes, métodos e atributos são realizadas por meio de telas com formulários. O relacionamento ainda é feito por meio de uma tabela, não de forma muito dinâmica nessa primeira versão, mas eficiente para geração de códigos que envolvem o banco de dados, classes de domínio, métodos controladores, views, etc. Dependendo da quantidade de classes que o usuário desejar criar no seu diagrama, a geração de código ocorre em questão de segundos, obtendo um resultado satisfatórios ao final da avaliação.

Após o desenvolvimento do Framework Jungle foi realizada a sua avaliação de funcionalidade e usabilidade. Como resultado, foi possível perceber que o framework é relevante para o auxílio de desenvolvedores de Aplicações Web por ter consistência entre diagrama de classes e banco de dados, além de gerar códigos de forma rápida.

Dessa maneira, podemos concluir que o desenvolvimento Web pode ser muito beneficiado por meio da utilização da abordagem MDD, agregando valores como produtividade, portabilidade, manutenabilidade e geração automática de códigos. Entretanto, a utilização dessa técnica deve ser acompanhada de boas práticas de Engenharia de Software e programação consistente.

5.2 Limitações

As limitações deste trabalho estão relacionadas, principalmente, a três itens: (1) A modelagem de diagramas poderia ser feita de maneira mais dinâmica, sem a necessidade de preenchimento de formulários, (2) a avaliação de usabilidade poderia ter um número maior de participantes, incluindo professores, alunos e profissionais atuantes em empresas de desenvolvimento de Aplicações Web, agregando novas ideias em termos de usabilidade e funcionalidades do software e (3) está relacionado ao número restrito de funcionalidades nessa primeira versão do framework. Por exemplo, na parte de relacionamento ainda não há uma opção para generalização de classes, além disso, a multiplicidade disponível é muito pouca, além das alguns defeitos que estão sendo resolvidos nas associações.

5.3 Trabalhos Futuros

Como trabalhos futuros, pretende-se desenvolver funcionalidades mais robustas ao framework no que tange à fase de modelagem. Estão sendo realizadas novas funcionalidades que visam facilitar a modelagem do diagrama de classes, trazendo características mais dinâmicas na parte de criação das classes, métodos, atributos, e nos relacionamentos entre classes deixando essas características mais próximas possíveis das ferramentas CASE presentes no mercado de software.

Outra funcionalidade que será implementada está relacionada à geração automática de códigos. Nas versões posteriores do framework Jungle, espera-se que seja possível a geração da Aplicação Web em outras linguagens de programação. Atualmente, por padrão, o projeto é gerado na linguagem PHP Orientado a Objetos, mas utilizando essa mesma abordagem, nas versões posteriores o usuário poderá ter mais opções de escolha como Java, Python ou Groovy, podendo até reaproveitar projetos feitos no Jungle importando-os.

Com essas novas funcionalidades, espera-se que o Jungle facilite ainda mais o trabalho dos desenvolvedores de Aplicações Web, utilizando esse conceito prático que é o MDD e o MDA, aliado a boas práticas de programação e Engenharia de Software.

REFERÊNCIAS

ALMEIDA, C. Qualitas: **Um Modelo de Processo de Desenvolvimento de Software Orientado a Modelos**. Dissertação (Mestrado em Ciência da Informação) – Universidade Federal de Sergipe, 2014.

BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML**. Rio de Janeiro: Elsevier, 2015.

BRUNO, E; SILVA, P; ALVES, T. **Testes Funcionais de Software**. 2012. Disponível em: < https://www.devmedia.com.br/testes-funcionais-de-software/23565>. Acesso em 09 jun. 2018.

BUARQUE, A. **Desenvolvimento de Software Dirigido por Modelos: Um Foco em Engenharia de Requisitos**. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Pernambuco, 2009.

CAELUM (2017). **Desenvolvimento Web com HTML, CSS e Javascript**. 14 nov. 2017 Disponível em: < https://www.caelum.com.br/apostila-html-css-javascript/>. Acesso em: Acesso em 14 nov. 2017

CÂMARA, A; PINHEIRO, R; ALMEIDA, P. **Geração Automática de Código com Base no Diagrama de Classe da UML**. Traços, Belém, v.12, n. 27, p. 113-127, jun. 2011.

CASTRO, L. **Procedimentos de Modelagem e Uma Ferramenta de Geração Automática de Código**. Monografia (Bacharel em Ciência da Computação) – Universidade Federal de Lavras, 2010.

DALL'OGLIO, P. PHP: **Programando com Orientação a Objetos**. São Paulo: Novatec, 2015.

DEITEL, P; DEITEL, H. **Java: Como Programar**. Tradução de Edson Furmankiewicz. 8. ed. São Paulo: Pearson Prentice Hall, 2010.

DESCHAMPS, A; GRAHL, E. **Ferramenta para Geração de Código a partir da Especialização do Diagrama de Classes**. In: SEMINÁRIO DE COMPUTAÇÃO, 14., 2005, Blumenau. **Anais...** Blumenau: FURB, 2005. p. 57 – 68.

EIS, D. O **Básico: o que é HTML?.** Tableless, 21 jan. 2011. Disponível em: <https://tableless.com.br/o-que-html-basico/>. Acesso em: 10 nov. 2017.

_____. **Uma Breve História do CSS**. Tableless, 11 jan. 2006. Disponível em: <https://tableless.com.br/o-que-html-basico/>. Acesso em: 10 nov. 2017.

FERNANDES, S. **Catálogo de Modelos de Computação para o Desenvolvimento de Linguagens Específicas de Modelagem de Domínio**. Tese (Doutorado em Engenharia) – Universidade de São Paulo, 2013.

FILHO, W. **Engenharia de Software: Fundamentos, Métodos e Padrões**. 3. ed. São Paulo: LTC, 2009.

GUEDES, G. **UML 2: Uma Abordagem Prática**. 2. ed. São Paulo: Novatec, 2011.

HENRIQUE, D.R. **Introdução a Banco de Dados – MySQL (2013)**. Disponivel em: <https://programadoresdofuturo.files.wordpress.com/2013/06/introduc3a7c3a3o-ao-mysql.pdf>. Acessado em: 15.nov 2017.

JÚNIOR, M. MDWA: **Uma Abordagem Guiada por Modelos para Desenvolvimento de Software Web**. Dissertação (Mestrado em Ciência da Informação) – Universidade Federal de São Carlos, 2012.

KAPPEL, G., PRÖLL, B., REICH, S., RETSCHITZEGGER, W. **An Introduction to Web Engineering**. In: Web Engineering: The Discipline of Systematic Development of Web Applications, John Wiley & Sons, 2006, ISBN: 978-0470015544.

K19. **Desenvolvimento Web com HTML, CSS e Javascript**. 14 jun. 2015. Disponível em: < www.k19.com.br>. Acesso em: 10 nov. 2017

LAROZA, J.; SEABRA, R. REA-UML: **Recurso Educacional Aberto para Ensino da UML**. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 26., 2015, Maceió. **Anais...** Maceió: CBIE-LACLO, 2015. p. 11 – 20.

LEITE, J. Software Consciente. ***Revista da Sociedade Brasileira de Computação***, v. 28, n. 3, p. 49 – 52, 2015.

LENGSTORF, J. PHP **Orientado a Objetos para Iniciantes**. Envato, 23 dez. 2011 Disponível em <https://code.tutsplus.com/pt/tutorials/object-oriented-php-for-beginners--net-12762> Acesso em: 10 nov. 2017

LOPES, M.; HENKELS, A.; FIALHO, F. **Drawcode: Uma Ferramenta para a Geração de Código a partir de Diagramas de Classe e Diagramas N-S**. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO, 28., 2008, Belém. **Anais...** Belém: SBC, 2008. p. 69 – 78.

LUCREDIO, D. **Uma Abordagem Orientada a Modelos para Reutilização de Software**. Tese (Doutorado em Ciências da Computação e matemática Computacional) – Universidade de São Paulo, 2009.

MARCZAK, S. A Interdisciplinaridade em Equipes de Desenvolvimento de Software. **Revista da Sociedade Brasileira de Computação**, v. 31, n. 2, p. 52 – 56, 2016.

MEIRA, S. Sistemas de Informação e Engenharia de Software: Cadê as Escolas? **Revista da Sociedade Brasileira de Computaçã***o*, v. 28, n. 3, p. 11 – 15, 2015.

MELO, E. **Desenvolvendo Aplicações com UML 2.0.** Rio de Janeiro: Brasport, 2004.

MIRANDA, F.; MORAES, A. **Avaliação da Interface de um Site de Comércio Eletrônico Através da Técnica Cooperativa**. In: 2º USIHR, 2003. Rio de Janeiro. Anais...Rio de Janeiro: 2º USIHR, 8P.

MOTA, D.; MACHADO, M.; RIBEIRO, F. **A Importância da UML no Desenvolvimento de Softwares**. In: MOSTRA CIENTÍFICA, 10., 2012, Catalão. **Resumos...** Catalão: CESUC, 2012.

NETO, V.; COSTA, S.; OLIVEIRA, J. **Lições Aprendidas sobre Desenvolvimento Dirigido por Modelos a partir da Refatoração de uma Ferramenta**. In: ENCONTRO ANUAL DE COMPUTAÇÃO, 10., Catalão: ENACOMP, 2010.

OLIVEIRA, A.; MÄHLMANN, L. **Ferramenta CASE (Gerador de Código .NET).** Universidade Luterana do Brasil, 29 nov. 2010. Disponível em: <http://www.ulbra.inf.br>. Acesso em: 30 out. 2017.

OMG. **MDA Guide Revision 2.0**. Jun. 2014. Disponível em: < http://www.omg.org/cgi-bin/doc?ormsc/14-06-01.pdf > Acesso em: 10 nov. 2017

OTAVIO. **Introdução ao HTML**. 9. maio 2017. Disponível em: <https://developer.mozilla.org/pt-BR/docs/HTML/Introduction>. Acesso em: 14 nov. 2017

PFLEEGER, S. **Engenharia de Software – Teoria e Prática.** 2. Ed. São Paulo: Prentice Hall, 2007.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software: Uma Abordagem Profissional**. 8. ed. Porto Alegre: AMGH, 2016.

SAMPAIO, A.; IYODA, J. Engenharia de Sistemas de Sistemas. **Revista da Sociedade Brasileira de Computação**, v. 28, n. 3, p. 44 – 48, 2015.

SANDRI, A. **Viabilidade de Construção de Software com MDD e MDA**. Centro Universitário La Salle, 2009. Disponível em: <www.andresandri.com.br/artigos/MDA/>. Acesso em: 22 ago. 2017

SCHWINGER, W., KOCH, N. **Modeling Web Applications,** In: Kappel, G., Pröll, B., Reich, S., Retschitzegger, W. (eds), Web Engineering: The Discipline of Systematic Development of Web Applications, John Wiley & Sons, 2006, ISBN: 978-0470015544.

SILVA, A. **Uma Abordagem Dirigida por Modelos para Desenvolvimento de Middlewares Auto-Adaptativos para Transmissão de Fluxo de Dados Baseado em Restrições de QoS**. Dissertação (Mestrado em Sistemas e Computação) – Universidade Federal do Rio Grande do Norte, 2010.

SOARES, S. Tudo é Software: Qual é a Importância da Engenharia de Software para o Mercado e para a Geração de Conhecimento? **Revista da Sociedade Brasileira de Computação,** v. 28, n. 3, p. 7 – 10, 2015.

SOMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Education -BR, 2011.

THOMAS, D. **MDA: Revenge of the modelers or UML utopia?** IEEE Software, v. 21, n. 3, p. 15–17, 2004.

VARA, J.; MARCOS, E. **A Framework for Model-driven Development of Information Systems: Technical decisions and lessons learned**. Journal of Systems and Software, v. 85, n. 10, p. 2368-2384, 2012.

VENTURA, P. **O que é Regra de Negócio?** 2016. Disponível em <http://www.ateomomento.com.br/o-que-e-regra-de-negocio/> Acesso em 04 maio 2018

VERNER, C.; PUIA, H. **Melhoramento no Desenvolvimento de Software com a Utilização do MDA.** Santa Catarina, v. 40, 2004.

W3C. **Cascading Style Sheets.** 2017. Disponível em: < https://www.w3.org/Style/CSS/> Acesso em 14 nov. 2017

_____. **World Wide Web Consortium**. 2017. Disponível em: <http://www.w3c.org> Acesso em 11 nov. 2017