Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Software para desenho de processos de negócios semanticamente descritos: uma aplicação em uma redação jornalística

Marcelo B. Fonseca

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Edison Ishikawa

Coorientador
Prof. Dr. Benedito Medeiros Neto

Brasília
2017

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Software para desenho de processos de negócios semanticamente descritos: uma aplicação em uma redação jornalística

Marcelo B. Fonseca

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Edison Ishikawa (Orientador)
CIC/UnB

Prof. Dr. Edgard Costa Oliveira
EPR/FT

Prof. Dr. Gheorghita Ghinea
Brunnel University

Prof. Dr. Márcio Victorino
FCI/UnB

Prof. Dr. Rodrigo Bonifácio de Almeida
Coordenador do Bacharelado em Ciência da Computação

Brasília, 07 de dezembro de 2017

Dedicatória

A minha família, amigos e colegas que me auxiliaram de alguma forma e me apoiaram durante a elaboração deste trabalho.

Agradecimentos

Ao meu orientador Edison Ishikawa pelo direcionamento no trabalho e por fornecer todo o seu apoio para a metodologia de pesquisa, capacitação teórica e prática.

Ao Ian Ferreira, que me auxiliou com a capacitação técnica necessária para a implementação do artefato.

Ao Renato da Fonseca que me ajudou nas revisões do texto.

Ao Professor Dr. Benedito Medeiros Neto pelo apoio e disponibilidade para ajudar durante o desenvolvimento da monografia.

Ao Professor M. Victorino que me ensinou e tirou dúvidas a respeito da anotação semântica.

Resumo

A modelagem de processos de negócios têm sido um foco recorrente nas organizações para aprimorar o fluxo de trabalho. Entretanto, atualmente as ferramentas de código aberto de modelagem de processos de negócios ainda não oferecem suporte à modelagem de processos usando a sua semântica, apesar de já existirem diversos trabalhos que formalizam a ontologia de processos e a sua anotação semântica. Com ferramentas semânticas é possível ampliar o poder da linguagem de processos de negócios e oferecer entendimento semântico sobre o significado dos elementos que compõem um processo. Dessa forma a semântica passa a ser entendida por máquina, o que amplia o auxílio na modelagem de novos processos. Neste contexto, este trabalho implementa uma ferramenta que auxilia a modelagem de novos processos em uma organização de comunicação, mais especifcamente em uma redação jornalística flexível, onde os processos podem ser criados ou reconfigurados *on-the-fly*. Para este fim, criou-se uma ontologia de domínio de uma redação jornalística, que pode ser classificada como leve, usando a metodologia 101 e a ferramenta *prótégé*. Também se utilizou o padrão BPMN para implementar um sistema de informação que sugerisse automaticamente o papel mais indicado para executar uma determinada tarefa num processo de produção de notícia de uma redação jornalística. Para alcançar este objetivo foi desenvolvido uma ferramenta para auxiliar na anotação semântica do domínio de aplicação para o qual os processos são voltados e enriquecer o poder da modelagem de processos no contexto semântico. De fato, seguido de regras de implementação, a ferramenta propostas evita alocação de tarefas à papéis indevidos.

Palavras-chave: Ontologia, processo de negócios, redação jornalística, jornalismo digital, fluxo de trabalho

Abstract

Business process modeling has been a recurring focus in organizations to improve it's workflow. However, today's business process modeling open-source tools still do not support process modeling using their semantics, although there are already several papers that formalize the process ontology and its semantic annotation. With semantic tools it is possible to extend the power of the business process language and offer semantic understanding about the meaning of the elements that make up a process. In this way the semantics becomes understood by machine, which increases the aid in the modeling of new processes. In this context, this work implements a tool that assists a modeling of new processes in a communication organization, more specifically in a flexible newsroom, in which process could be created or modified on the fly. To this end, it was created a domain ontology for newsroom classified as light, using the methodology 101 and *prótégé* modeling tool. The BPMN standard was also used to implement the information system that would automatically suggest the most appropriate role to perform a given task in a news production process of a newsroom. In order To achieve this goal, a tool was developed to aid in the semantic annotation of the application domain to which the processes are addressed and to enrich the power of process modeling in the semantic context. In fact, followed by implementation rules, the proposed tool avoids assigning tasks to improper roles.

Keywords: Ontology, business process, newsroom, digital journalism, workflow

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

API Application Programming Interface.

B2B Business-to-Business.

BPD Busines Process Diagram.

BPEL Business Process Execution Language.

BPM Business Process Management.

BPMN Busines Process Modelling Notation.

BPMNO BPMN Ontology.

CMS Content Management System.

CSP Communicating Sequantial Processes.

DSR Design Science Research.

EPC Event-Driven Process Chains.

JSON JavaScript Object Notation.

JVM Java Virtual Machine.

KMS Knowledge Management System.

MVC Model View Controller.

OMG Object Management Group.

OWL Ontology Web Language.

PNML Petri Net Markup Language.

RDF Resource Description Framework.

REST Representational State Transfer.

RoR Ruby on Rails.

SPARQL Protocol and RDF Query Language.

SQL Structured Query Language.

W3C World Wide Web Consortium.

WMS Workflow Management System.

WSMO Web Service Modeling Ontology.

XML eXtensible Markup Language.

Capítulo 1

Introdução

A tecnologia trouxe consigo novas ferramentas que permitem a troca de informações a uma velocidade cada vez mais rápida com um número maior de pessoas de uma forma cada vez mais ubíqua, o que possibilita a fácil acessibilidade a computadores, smartphones, redes sociais e internet cada vez mais veloz. O compartilhamento e divulgação das informações também se tornaram mais rápidas e simples. Neste novo ambiente, o jornal impresso perde espaço para outros meios digitais de divulgação. Neste cenário o processo da construção da notícia impressa tinha até um dia para a sua produção, com a digitalização da notícia as redações jornalísticas se viram obrigadas a acelerar a sua divulgação para horas ou minutos, a fim de não perder a novidade. Como apontado em [1], as organizações de comunicação enfrentam hoje um grande desafio devida a diminuição de leitores pagantes e pela concorrência impostas por diversas formas tecnológicas emergentes para produzir e divulgar notícias.

70% das redações jornalísticas na América Latina, norte da África e Oriente Médio reportam urgência para criar novos métodos para o fluxo de receita contra 44% nos EUA [8]. Dessa forma, a profissão fica comprometida por dois cenários. A independência das organizações de notícias de seus anunciantes fica comprometida [1] e o trabalho nas redações jornalísticas passa por uma crise. As condições de trabalho dos jornalistas estão se deteriorando com o aumento do desemprego, baixos salários e grande carga de trabalho.

Os jornais precisam mudar suas formas de trabalho para acompanhar o novo ritmo e a novas maneiras de entregar a notícia. Em [9], o problema geral se concentra na dificuldade de dar suporte e apoio ao trabalho da gestão e produção de notícias, em redações jornalísticas na contemporaneidade. Entretanto o problema tratado neste trabalho é a falta de semântica no mapeamento de processos na linguagem Busines Process Modelling Notation (BPMN). Desta forma, uma solução ao problema pode ser aplicado não apenas a redações jornalísticas mas para qualquer tipo de organização que faça uso de mapeamento de processos no padrão BPMN.

Como proposta de solução do problema, este trabalho implementa uma parte de processos de negócios flexíveis do *framework* semântico projetado para facilitar a colaboração na produção de conteúdo em uma redação jornalística. O trabalho propõe um modelo de *framework* semântico com *workflow* flexível projetado para facilitar na colaboração em uma redação jornalística. O *framework* busca explorar a utilização de ferramentas de *softwares* livres [10] e *softwares open-source* [11]. Este trabalho remete a implementação do módulo de workflow do framework proposto em [12, 13], o Workflow Management System (WMS).

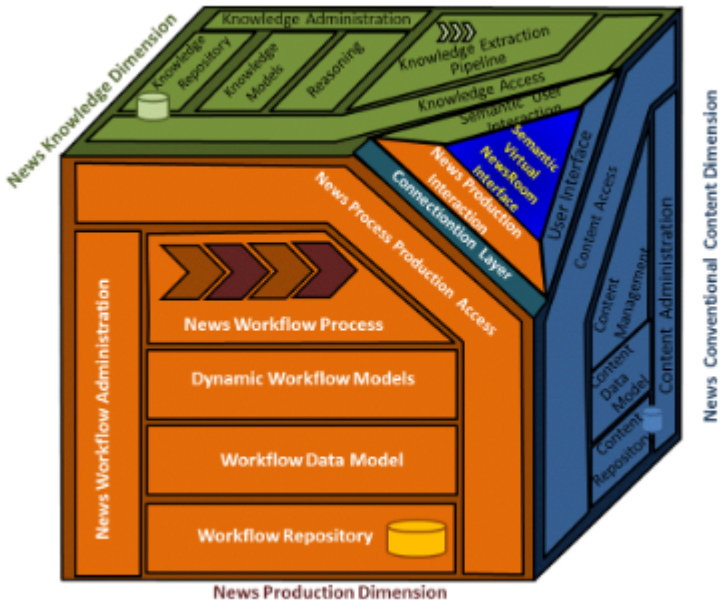

Figura 1.1: Modelo do framework semântico proposto em [1]

A Figura 1.1 representa a arquitetura do framework do projeto. O desenvolvimento do módulo deve ser orientada em virtude da comunicação entre módulos na arquitetura do framework semântico. A Figura 1.2 mostra módulo de WMS, a ser implementado neste trabalho.

O módulo WMS é responsável pela *workflow engine*, modelagem de processos e seu aprimoramento com ferramentas semânticas.

A modelagem de processos de negócios desempenha um papel crítico ao apoiar empresas a alcançarem seus resultados comerciais e melhorarem o entendimento do fluxo de trabalho desenvolvidos por elas. Muitas ferramentas foram desenvolvidas para modelagem de processos de negócios e apesar dos benefícios alcançados pela utilização destas, as mesmas também têm suas limitações. Neste trabalho, visamos utilizar a anotação se-

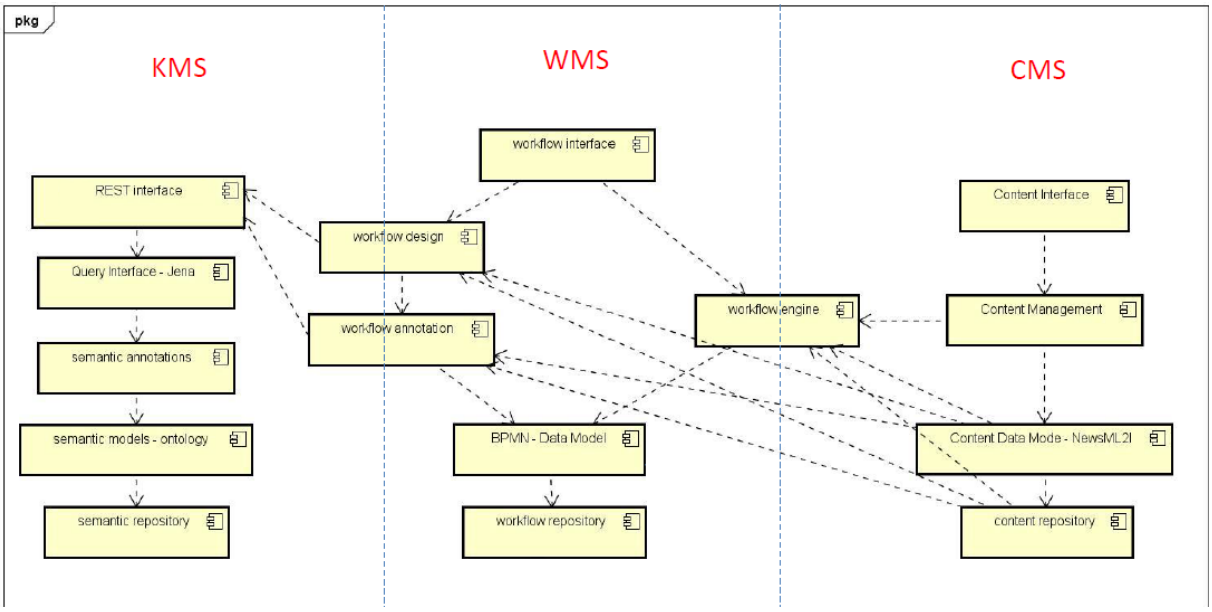

Figura 1.2: Comunicação do Módulo WMS com CMS e

mântica para auxiliar a modelagem de processos dentro de uma redação jornalística. De forma que, a anotação semântica ajude na modelagem de novos processos.

Utilizaremos como padrão e definição de processo de negócios a ferramenta de notação Business Process Model and Notation(BPMN) v.2.0 [14], uma linguagem gráfica para modelar diagramas de processo de negócios (BPD) especificado pela OMG[1].

1.1 Objetivos

1.1.1 Objetivos gerais

O objetivo geral deste trabalho é implementar uma ferramenta que auxilie na modelagem de processos em tempo de operação de uma redação jornalística para a produção de notícias. Isto permite flexibilizar os sistemas de informação que automatizam a produção de notícias, tornando-a mais flexível, e em consequência, mais adaptável à dinâmica do cenário atual. Esta implementação foi desenvolvida como um *framework* web de *workflow* que enriqueça a modelagem de processos com semântica levando-se em conta a ontologia do domínio de aplicação, que neste caso é uma redação jornalística.

1.1.2 Objetivos específicos

Para atingir o objetivo geral, os seguintes objetivos específicos foram definidos:

[1] `http://www.omg.org/`

1. Desenvolver uma aplicação de *workflow*;
2. Levantar dados dos papéis e fluxo de trabalho existentes em uma redação jornalística;
3. Modelar uma ontologia leve de domínio a partir dos dados levantados anteriormente para anotar semanticamente a tarefa no processo usando a ontologia;
4. Utilizar as bibliotecas necessárias para integrar a busca semântica ao projeto;
5. Apresentar a proposta de sugestão automática funcionando sobre um processo de notícia;

1.2 Estrutura do Documento

A estrutura do trabalho se encontra da seguinte maneira:

- **Capítulo 1:** Apresenta e contextualiza o problema, introduz à solução e explica a arquitetura do projeto;
- **Capítulo 2**: É descrito uma síntese da fundamentação teórica necessária para compreender o trabalho e uma visão geral de trabalhos relacionados de outros autores;
- **Capítulo 3**: Apresenta a metodologia utilizada para guiar o trabalho e descreve a pesquisa das tecnologias para desenvolvê-lo;
- **Capítulo 4**: Descreve a arquitetura e as etapas da implementação; Detalha os módulos e funcionamento do artefato;
- **Capítulo 5**: Apresenta os resultados obtidos e trabalhos futuros;

Capítulo 2

Revisão Bibliográfica

Este Capítulo descreve os principais métodos de anotação semântica e apresenta o referencial teórico necessário para entender o trabalho. Na Seção 2.1 introduzimos a definição de processo. Na subseção 2.1.1 é apresentado diferentes formas de representar um processo e suas classificações e na Subseção 2.1.2 introduzimos a linguagem gráfica BPMN. Na Seção 2.2 introduzimos a semântica web e as tecnologias que ela abrange. A subseção 2.2.1 introduz ao padrão RDF, Subseção 2.2.2 explica e exemplifica a representação da tripla-RDF, Subseção 2.2.3 explica a linguagem SPARQL, Subseção 2.2.4 provê algumas classificações sobre ontologia na literatura e como representá-las e a Subseção 2.2.5 explica a linguagem OWL e como representar uma ontologia em OWL. Na Seção 2.3 apresentamos maneiras de se abordar o enriquecimento de processos de negócios com semântica na literatura e a Seção 2.4 provê qual será o direcionamento deste trabalho.

2.1 Processos

Nesta Seção apresentamos formas de classificar e representar processos de negócios. Em seguida entramos em um maior nível de detalhes na linguagem BPMN que será utilizada para este trabalho.

2.1.1 Tipos de processos

Existem várias classificações para processos de negócios de acordo com suas características [15] e uma infinidade de notações para modelar processos operacionais (por exemplo, Petri Nets, BPMN, UML e EPCs) [16]. Essas notações têm em comum que os processos são descritos em termos de atividades.

A modelagem e análise de processos desempenha um papel central no gerenciamento de processos de negócios [16]. Portanto, a escolha da linguagem para representar os processos

de uma organização é essencial. Três classes de linguagens podem ser identificadas em [16]:

- **Linguagens formais:** os processos são estudados usando modelos teóricos. Os matemáticos têm usado cadeias de Markov, redes de enfileiramento e assim por diante para modelar processos. Os cientistas da computação têm usado máquinas de Turing, sistemas de transição, Petri Nets, lógica temporal e álgebras de processos para modelar processos. O que todas essas linguagens têm em comum, são que elas têm semântica inequívoca e permitem a análise.
- **Linguagens conceituais:** os usuários na prática geralmente têm problemas ao usar linguagens formais devido à rigorosa semântica (tornando impossível deixar as coisas intencionalmente vagas) e a natureza de baixo nível. Eles geralmente preferem usar linguagens de nível superior. Exemplos são BPMN, EPCs, diagramas de atividade UML, e assim por diante. Essas línguas são tipicamente informais; isto é, eles não têm uma semântica bem definida e não permitem análises. Além disso, a falta de semântica torna impossível executá-los diretamente.
- **Linguagens de execução:** linguagens formais tipicamente abstratos de "detalhes de implementação"(por exemplo, estruturas de dados, formulários e problemas de interoperabilidade) e as linguagens conceituais fornecem apenas uma descrição aproximada do comportamento desejado. Portanto, são necessários mais linguagens técnicas para a promulgação. Um exemplo é o Business Process Execution Language (BPEL). A maioria dos fornecedores fornece uma linguagem de execução proprietária. No último caso, o código-fonte da ferramenta implementada determina a semântica exata.

[17] apresenta uma outra classificação de processos que se baseia nas características do processo:

- Processos fortemente emoldurados ou estruturados(Totalmente previsível, altamente repetitivo);
- Processos frouxamente emoldurados;
- Processos ad-hoc emoldurados;
- Processos totalmente não-moldurados ou não estruturados(Totalmente imprevisível, altamente não repetitivo);

Processos fortemente emoldurados são caracterizados como totalmente previsíveis e repetitivos. Eles podem ser repetidamente instanciados no tempo de execução.

Exemplos desta categoria são os processos de produção, administrativos e as transações bancárias que são executadas na sequência exata para cumprir as normas legais [17].

Processos frouxamente emoldurados correspondem a um processo no qual é possível representar o comportamento do processo e um conjunto de restrições a priori, de modo que o modelo do processo descreva a "maneira padrão de fazer as coisas"ao requerer adições, remoções ou geração de seqüência alternativa de atividades durante o tempo de execução [17].

Processos ad-hoc emoldurados não podem ser determinados em termos de lógica de processo explícita durante o tempo de modelagem devido a falta de conhecimento de domínio ou a complexidade de combinações de tarefas. Em vez disso, apenas os fragmentos estruturados podem ser identificados a priori e devidamente compostos por cada caso, enquanto as partes do processo que são indefinidas ou incertas só podem ser especificadas e incorporadas à medida que o processo evolui [17].

Processos totalmente não-emoldurados tem variabilidade suficiente de tal forma que nenhuma descrição do processo pode ser pré-definida. Como resultado, os participantes do processo precisam tomar decisões usando seus conhecimentos para criar atividades sob demanda. A criação de tais atividades é baseada em parâmetros específicos da situação cujos valores são determinados à medida que a execução do processo prossegue. Além de escolher atividades sob demanda, eles também decidem dinamicamente a ordem de execução de tais atividades [17].

Os processos em uma redação jornalística são predominantemente do tipo 2, 3 ou 4. Por isso a necessidade de oferecer suporte semântico ao jornalista para projetar processos sob demanda em uma redação jornalística flexível.

Na subseção seguinte apresentaremos a linguagem BPMN, que será a linguagem utilizada para a representação dos processos na implementação do trabalho.

2.1.2 Linguagem BPMN

O BPMN é uma linguagem gráfica e visa solucionar a comunicação empresarial por uma linguagem de notação intuitiva, através de processos complexos, considerada o padrão global para modelagem de processos pelo Object Management Group (OMG) em 2006.

O padrão de notação de Processo de Negócios BPMN fornece às empresas a capacidade de entender seus procedimentos comerciais internos em uma notação gráfica e dá às organizações a capacidade de comunicar estes procedimentos de forma padrão. Isso permite que as organizações se ajustem rapidamente a novas circunstâncias comerciais internas e B2B [14].

Um exemplo de processo modelado na linguagem BPMN é representado na Figura 2.1.

Tabela 2.1: Frequencia do uso dos padrões de processo por organizações em [7]

Process Standard	**2011**	**2013**	**2015**
BPMN	60%	60%	64%
ARIS EPC	14%	22%	18%
UML	14%	18%	17%
BPEL	12%	10%	18%
XPDL	5%	2%	4%

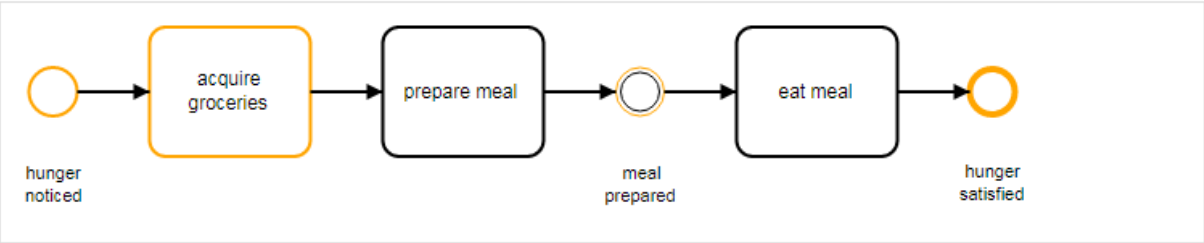

Figura 2.1: Exemplo de diagrama BPMN: Fome em [2]

Este diagrama mostra um processo simples desencadeado por alguém com fome. O resultado é que alguém deve fazer compras e preparar uma refeição. Depois disso, alguém vai comer a refeição e ter sua fome satisfeita [2]. Um diagrama modelado na notação BPMN pode ser chamado de Busines Process Diagram (BPD). Podemos dizer que a Figura 2.1 apresenta um evento de fome mapeado em um BPD.

Os blocos de atividades na Figura 2.1 podem ser chamado de tarefas. Ao nomear tarefas, tentamos aderir ao princípio de projeto orientado a objetos de usar o padrão [verbo] + [objeto]. Nós diríamos "adquirir mantimentos", por exemplo, não "primeiro cuidar da compra de mantimentos" [2]. Esse conceito vai ser importante para a etapa de desenvolvimento do trabalho. Mais exemplos e explicações da linguagem BPMN podem ser encontrados em [14, 2].

A versão mais recente do BPMN disponível é 2.0 . O BPMN 2.0 consiste de um grande número de elementos do modelo. Desde a versão 1.0 o número de elementos têm aumentado continuamente [7]. Em [7] é apresentado um relatório que aponta o BPMN como a linguagem de modelagem de processos mais utilizada na indústria. O autor apresenta a tabela 2.1 dos padrões de processo usados, identificados pelo relatório.

2.2 Web Semântica

Evoluindo a já clássica e conhecida “Web de documentos”, o W3C ajuda no desenvolvimento de tecnologias que darão suporte à “Web dos dados”, viabilizando pesquisas como

num banco de dados. O objetivo final da Web de dados é possibilitar com que computadores façam coisas mais úteis e com que o desenvolvimento de sistemas possa oferecer suporte a interações na rede. O termo Web Semântica refere-se à visão do W3C da Web dos Dados Conectados. A Web Semântica dá às pessoas a capacidade de criarem repositórios de dados na Web, construírem vocabulários e escreverem regras para interoperarem com esses dados. A conexão de dados é possível com tecnologias como RDF, SPARQL e OWL [18]. Partindo deste princípio, temos como base que a Web pode nos trazer diversas informações, onde tanto pessoas como computadores sejam capazes de entender esses dados. Neste paradigma, a Web passa a ser vista como um grande banco de dados [19].

Exemplificando: imagine-se em busca de um determinado produto. Com nossas atuais tecnologias, deveríamos acessar um sistema de e-commerce ou ainda um site de busca, localizar o produto e, através do sistema disponibilizado, realizar o cálculo do frete e verificar os prazos de entrega, pois bem, esta não é a forma mais prática de se realizar este procedimento. A Web Semântica trata as informações disponíveis na Web como informações mesmo, ou seja, com estes recursos se você procurasse o produto, teria em suas mãos não somente os locais disponíveis, mas também o valor do frete e prazos de entrega de forma rápida e ágil [19].

Desta forma a web passa a ser encarada como uma grande base de dados. Nas subseções a seguir iremos descrever as tecnologias utilizadas para conectar semanticamente os dados.

2.2.1 RDF

O Resource Description Framework (RDF) é uma linguagem declarativa que fornece uma maneira padronizada de utilizar o XML para representar metadados no formato de sentenças sobre propriedades e relacionamentos entre itens na Web. Esses itens, chamados de recursos, podem ser virtualmente qualquer objeto(texto, figura, vídeos e outros), desde que possuam um endereço Web. O RDF foi projetado de modo a representar metadados de recursos Web de maneira legível e, sobretudo, processável por máquinas [20].

O RDF é um acrônimo comum na comunidade da web semântica porque ele forma um dos blocos de construção básicos para formar a web de dados semânticos. O que ele define é um tipo de banco de dados em grafo [3]. Podemos já estarmos familiarizados com o armazenamento de dados em banco de dados relacional, como MySQL, ou hierárquico, como XML. O armazenamento de dados RDF representa um banco de dados gráfico. Na Figura 2.2 é apresentado uma comparação entre os três modelos de banco de dados.

O RDF amplia a estrutura de ligação da web ao usar URIs para nomear a relação entre coisas, bem como as duas extremidades do *link* (conhecido como uma "tripla").

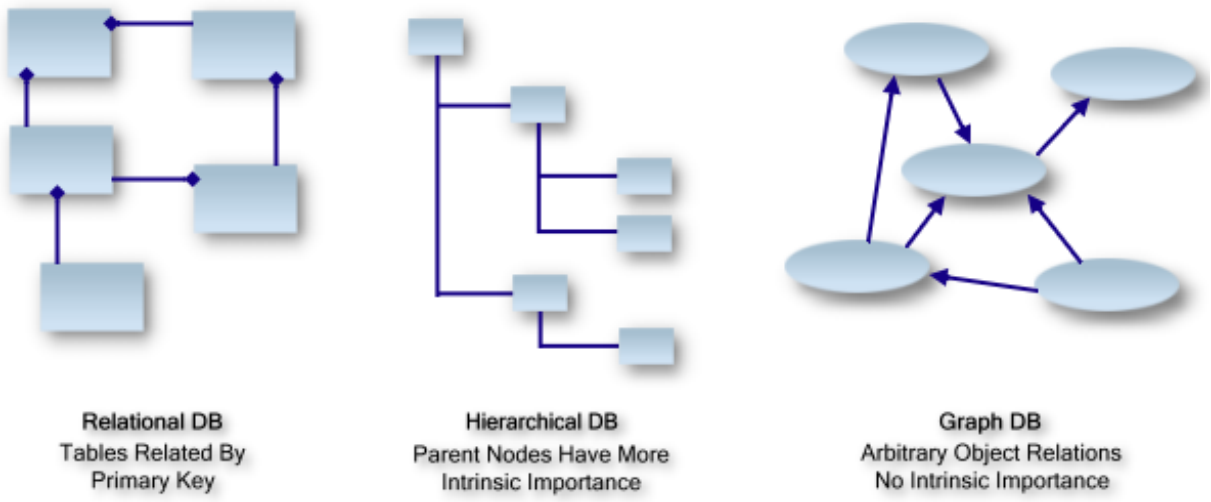

Figura 2.2: Modelos de bancos de dados

Usando este modelo simples, permite-se que os dados estruturados e semiestruturados sejam misturados, expostos e compartilhados em diferentes aplicativos [18].

2.2.2 Triplas RDF

Uma tripla RDF descreve a quebra da declaração nas suas três partes constituintes: o sujeito, o predicado e o objeto da declaração [3].

Como demonstrado em [3], é mais fácil primeiro ilustrar esses termos na forma de um gráfico simples. Veja o seguinte gráfico de dados na Figura 2.3 que descreve a cor de uma camiseta:

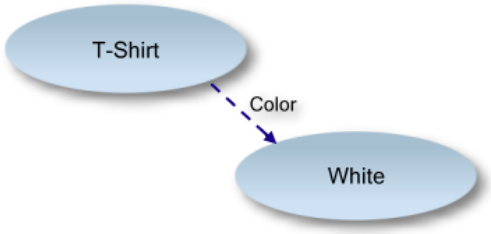

Figura 2.3: Exemplo de tripla RDF [3]

- Sujeito é **T-shirt**;
- Predicado (propriedade) é **color**;
- Objeto é **white**;

O sujeito denota o recurso e o predicado denota traços ou aspectos do recurso e expressa uma relação entre o sujeito e o objeto. Um sujeito em um documento RDF também pode ser referenciado como um objeto de uma propriedade em outra expressão RDF.

O RDF faz uso de sujeito em vez de objeto (ou entidade) em contraste com a abordagem típica de um modelo de entidade-atributo-valor no *design* orientado a objetos: Entidade (*T-Shirt*), atributo (*color*) e valor (*white*).

2.2.3 SPARQL

O SPARQL é uma linguagem de consulta RDF. Similarmente ao popular Structured Query Language (SQL), usado para buscar dados em banco de dados relacionais como MySQL[1], os dados RDF podem ser buscados utlizando sua própria linguagem de busca - Protocol and RDF Query Language (SPARQL). No entanto ela tem suas diferenças [3]. Os resultados das consultas SPARQL podem ser conjuntos de resultados ou gráficos RDF [18]. Mais informações a respeito da linguagem SPARQL podem ser encontradas em [3]. Na referência também se encontra um passo-a-passo e exemplos práticos da sintaxe.

2.2.4 Ontologia

Na última década, o tema Ontologia também tem sido estudado em diversas áreas, como: Linguagem e Cognição, Ciência da Informação e Ciência da Computação. Nessas áreas o termo ontologia está direta ou indiretamente relacionado ao tratamento e comunicação da informação ou do conhecimento [21].

Definir uma ontologia pode ser uma tarefa difícil devido às suas diversas interpretações e portanto, para este trabalho, utilizaremos a especificação de ontologia de Tom Gruber (1993), onde uma ontologia é uma especificação explícita de uma conceitualização [22]. Desta forma segundo Gruber, uma ontologia para a ciência da computação é a descrição de conceitos e relacionamentos que devem ser considerados por um agente ou por uma comunidade de agentes. Ele especifica ainda que uma ontologia é geralmente escrita como um conjunto de definições de um vocabulário formal.

Além disso, as ontologias podem ser divididas em dois grupos específicos: "ontologias leves" (lightweight ontologies) e "ontologias pesadas" *(heavyweight ontologies)*. As ontologias leves incluem conceitos, relações e instâncias. Já as ontologias pesadas contemplam todos os aspectos de uma ontologia leve acrescentando-se axiomas e restrições [23].

Em [20], apresenta-se um outro sistema de classificação que utiliza a generalidade da ontologia como critério de classificação proposta por Nicola Guarino [24]. Nesse sistema o autor identifica:

[1] `https://www.mysql.com/`

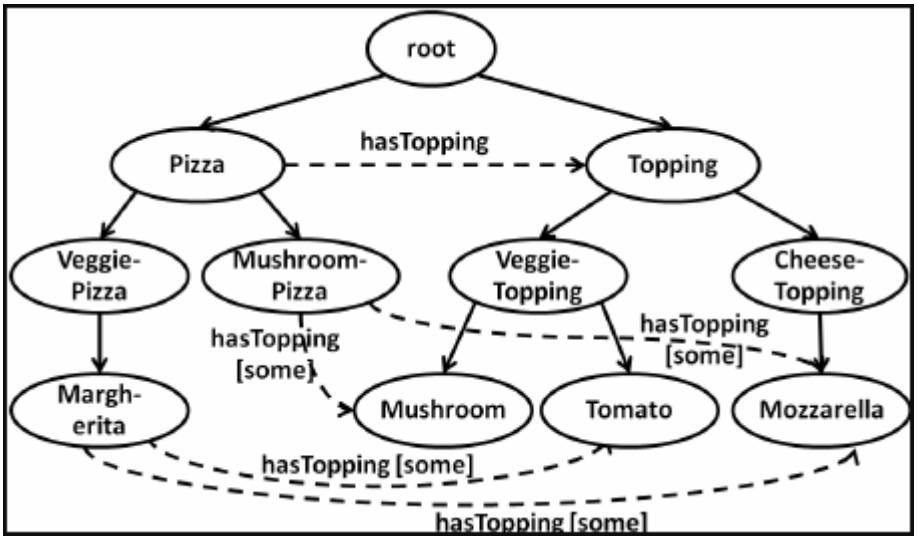

Figura 2.4: Representação da ontologia da pizza [4]

- **Ontologias de nível superior**: Descrevem conceitos muito genéricos, tais como espaço, tempo e eventos. Estes seriam, a princípio, independentes de domínio e poderiam ser reutilizados na confecção de novas ontologias. Exemplos de ontologias de alto nível são WordNet e Cye.
- **Ontologias de domínio**: Descrevem o vocabulário relativo a um domínio específico através da especialização de conceitos presentes na ontologia de alto nível;
- **Ontologias de tarefas**: Descrevem o vocabulário relativo a uma tarefa genérica ou atividade através da especialização de conceitos presentes na ontologia de alto nível;
- **Ontologias de aplicação**: São as ontologias mais específicas. Conceitos em ontologias de aplicação correspondem, de maneira geral, a papéis desempenhados por entidades do domínio no desenrolar de alguma tarefa;

Por uma ontologia ser de difícil compreensão, é apresentado um tutorial usando a ferramenta *protégé* [25] para modelar rapidamente uma ontologia da pizza[2], para fins didáticos. Um exemplo de modelagem da ontologia pode ser representada na Figura 2.4.

2.2.5 OWL

Ontology Web Language (OWL) é uma linguagem para definir e instanciar ontologias na Web [18]. Assim como o RDF, a linguagem OWL é uma especificação da W3C. Uma ontologia OWL pode descrever suas classes, respectivas propriedades e seus relacionamentos.

[2]tutorial ontologia da pizza:`https://protegewiki.stanford.edu/wiki/Protege4Pizzas10Minutes`

Desta forma, uma classe em OWL é uma classificação de indivíduos em grupos que compartilham características em comuns. Se um indivíduo é um membro de uma classe, ele diz a um leitor de máquina que se encontra sob a classificação semântica dada pela classe OWL [3]. Mais detalhes e um passo-a-passo sobre sua sintaxe podem ser encontrados em [3].

2.3 Anotação Semântica de processos

Nesta seção apresentamos uma visão geral da anotação semântica de processos em questão do problema resolvido, possibilidades de solucioná-los e os trabalhos existentes.

Os processos podem ser ambíguos e as linguagens gráficas para representá-los não fornecem informações sobre o contexto em que o processo se encontra. O desenvolvimento de ferramentas para a anotação semântica visam solucionar esses problemas. As ferramentas dão suporte para o desenvolvimento de análises mais inteligentes, como o auxílio com sugestões de palavras e atividades ou correções do modelo de processo.

Há diversas direções de pesquisas relacionadas a anotação semântica de processos de negócios que apareceram nos últimos anos. Entre elas, se destacam as soluções utilizando ontologias e Petri Nets.

2.3.1 Ontologia

Adicionar atributos semânticos em um processo de negócios retirados de uma ontologia se tornou uma necessidade para prover e integrar serviços [26]. Portanto, com ontologia podemos dar uma visão mais abrangente fazendo uso do conceito de aplicação do domínio do processo.

O trabalho [27] analisa diversas abordagens para enriquecer processos de negócios com semântica usando ontologias e são levantados cinco questões que um modelador de processos de negócios deve prover e que podem ser solucionadas explorando a semântica do contexto. São elas:

- **Compatibilidade de entidade:** Saber identificar as entidades presentes no processo e quais as suas permissões.
- **Inspeção:** Identificar se o processo se encontra estruturalmente correto.
- **Otimização:** Realocar, melhorar completamente ou parcialmente um processo.
- **Reuso de processo:** Reuso de processo em diferentes domínios.
- **Aprimoramento de processo:** Implementar ou substituir uma ou mais das tarefas definidas pelo significado de serviços online para melhorar o processo.

Todas estas questões podem ser implementadas por um enriquecimento semântico da modelagem do processo [28].

A anotação semântica de processos de negócios usando ontologia vem sendo abordada em trabalhos de diversas maneiras. Neste contexto destaca-se o projeto SUPER [29], financiado na Europa. O objetivo do projeto foi criar ferramentas para a semantização de processos de negócios descrevendo modelos de processos usando conceitos da ontologia [30]. O SUPER providenciou um *framework* muito complexo para modelagem, gerenciamento e enriquecimento semântico dos processos de negócios, com foco no BPMN e na WSMO para fornecer a semântica [27]. O resultado do projeto é composto por diversas ontologias. Em [29] são encontradas algumas informações do projeto e em [30] é descrito sua arquitetura e integração das diversas ontologias utilizadas no projeto.

Similarmente, no trabalho [30] é proposto enriquecer *frameworks* de código aberto de execução de processo com semântica. O autor compara as tecnologias disponíveis de *workflow engine* **Camunda**, **jBPM** e **Activiti** para a linguagem Java. Todas utilizam a linguagem BPMN e são apresentadas como boas candidatas para adicionar anotação semântica aos processos.

O trabalho [31] descreve uma descrição ontológica formal da linguagem BPMN. A ontologia proposta (a ontologia BPMN ou BPMNO) fornece uma classificação de todos os elementos da BPMN em um documento OWL. A ontologia se encontra disponível na web[3]. Trabalhos como o [32], foram construídos com a utilização da BPMNO desenvolvida em [31].

O trabalho [7] fornece uma ferramenta para capturar e anotar processos usando a **Semantic MediaWiki** [33], uma extensão gratuita de código aberto do **MediaWiki**, como uma plataforma colaborativa e disponibiliza uma versão demonstrativa na web. o Cognitive Process Designer[4]. O Cognitive Process Designer usa a API **MediaWiki** para atualizar a página wiki se um elemento for criado, alterado ou excluído. O **bpmn-js** [34] é usado como kit de ferramentas de renderização BPMN 2.0 e modelador web [7].

O trabalho [35], de mesmo autor, utiliza da anotação semântica de processos do trabalho anterior [7] para calcular a semelhança entre os processos atuais e o processo alvo, a fim de poder quantificar a variedade e ver como os diferentes processos se comportam em termos de diferentes aspectos e dimensões da qualidade do serviço.

2.3.2 Petri Nets

O mapeamento do BPMN utilizando Petri Nets [36] fornece uma análise estática do processo. O uso de Petri Nets permite validar a modelagem do processo e realizar correções.

[3] `http://dkm.fbk.eu/bpmn-ontology`

[4] `https://www.mediawiki.org/wiki/Extension:Cognitive_Process_Designer`

Desta maneira, a proposta de mapeamento não apenas serve para o propósito de retirar a ambiguidade da construção do BPMN, mas para prover um fundamento para verificação estática da semântica correta do modelo BPMN [37].

A escolha de usar Petri Nets como alvo para o mapeamento é motivada pela disponibilidade de técnica de análise estática eficiente. Assim, o mapeamento proposto não só serve para desambiguar as construções do núcleo da BPMN, mas também fornece uma base para verificar estaticamente a correção semântica dos modelos BPMN [38]. Evidências anedóticas sugerem que os usuários de BPMN às vezes produzem modelos com erros semânticos que poderiam ser detectados usando a tecnologia de verificação existente [38]. Após o mapeamento de um processo para Petri net, pode-se verificar por erros estáticos no Diagrama. Como a existência de *deadlocks* e estados mortos.

O trabalho realizado em [38] fez uso de mapeamento de BPMN usando Petri Nets para suportar a reivindicação como técnica de análise estática eficiente. Os autores implementaram uma ferramenta que se traduz entre a serialização XML de Modelos BPMN suportados por uma ferramenta BPMN existente e o Petri Net Markup Language (PNML) para mapear o BPMN na Petri Net. A ferramenta desenvolvida no trabalho se encontra disponível para uso na web[5].
Seus recursos incluem a capacidade de definir:

- Subprocessos que podem ser executados simultaneamente várias vezes;
- Subprocessos que podem ser interrompidos como um resultado de exceções;
- Fluxo de mensagens entre os processos. As interações entre esses recursos são uma fonte adicional de erros semânticos;

2.3.3 Communicating Sequential Processes

Assim como o Petri Nets, o Communicating Sequantial Processes (CSP) [39] é usado em associação com o BPMN. No entanto, eles apenas suportam um subconjunto da notação BPMN ou apenas avaliam estaticamente e validam o modelo original [27]. O trabalho [40], apresenta uma solução visando mostrar como um subconjunto da BPMN pode receber uma semântica de processo no CSP.

2.3.4 Outras possibilidades de semantização

A maioria dos Frameworks de código aberto fornecem suporte à interface REST. No trabalho [41] o autor propõem a linguagem SAWADL(Semantic Annotation for Web Application Description Language) que permite a semantização da REST *web service*.

[5] `http://is.tm.tue.nl/staff/rdijkman/bpmn.html`

2.4 Direcionamento do trabalho

Os trabalhos referentes a anotação semântica com processos de negócios são recentes e em consequência disso, não há muitas aplicações de código aberto que tiram proveito desta abordagem.

Neste trabalho seguiremos uma aproximação semelhante aos trabalhos [32, 42, 43, 30] e [7] mas de maneira bem mais simples. Iremos elaborar uma ontologia leve de domínio para anotar processos semanticamente e auxiliar na modelagem de processos uma redação jornalística.

Nesse contexto, o trabalho a ser desenvolvido consiste em criar uma ferramenta para auxiliar a modelagem de diagramas de processo. A ferramenta será integrada com semântica para fornecer apoio durante a modelagem, com o uso da ontologia. A ontologia será modelada de forma suficientemente genérica e flexível para poder ser aplicada em outras organizações jornalísticas.

O uso de uma ontologia de domínio permite a pesquisa de informações sobre os papéis das pessoas envolvidas no processo. Desta forma, é possível associar ao elemento BPMN, informações correspondentes, como o papel exato na organização, informações pessoais, possíveis links com outras funções e assim por diante [27]. Assim, o foco do trabalho consiste de implementar uma ferramenta de *workflow* que tenha integrado a modelagem de processo de negócios dentro do padrão BPMN, e em conjunto com uma ontologia de domínio, anotar o processo semanticamente para auxiliar na modelagem de processos com uma sugestão automática do papel mais apropriado para uma determinada tarefa.

O conceito de especificação de uma tarefa BPMN apresentada na Subseção 2.1.2 será uma direção para modelagem da ontologia e a organização das triplas RDF de maneira a identificar a tarefa de um papel da redação jornalística. Entraremos em mais detalhes a respeito deste padrão de implementação no final do Capítulo 3 e no Capítulo 4.

A escolha de se usar a anotação com ontologia se dá para conseguir abranger futuramente outros trabalhos partindo de um consenso da anotação semântica. Como contribuição, esperamos elaborar uma ontologia de domínio simples o suficiente para auxiliar na modelagem de um processo de produção da notícia.

Capítulo 3

Metodologia

Este capítulo descreve quais foram os procedimentos necessários para implementar o protótipo do *framework* web proposto no objetivo geral. Este trabalho utiliza o método de pesquisa Design Science Research (DSR) [5]. A DSR também conhecida como *constructive research*, é uma abordagem metodológica que consiste em construir artefatos que trazem benefícios às pessoas. As metodologias propostas por ela visam preencher a lacuna existente entre pesquisa e a prática [5]. Desta forma, podemos assegurar uma boa direção do trabalho na elaboração do artefato proposto. A metodologia pode ser ilustrada pela Figura 3.1.

Como apontado em [5], a metodologia visa produzir conhecimento científico e ajudar organizações a resolver problemas reais, o que torna o seu uso um guia para este trabalho já que, como recomendado pelo DSR, o projeto visa inserir alguma mudança em um sistema e possibilitar uma melhora de desempenho. Assim, o resultado do estudo deste artefato tem uma natureza prescritiva. A elaboração do trabalho segue as seguintes etapas:

1. Definição do problema;
2. Revisão da literatura;
3. Pesquisa de ferramentas;
4. Pesquisa dos papéis na redação jornalistica;
5. Modelagem de processos BPMN;
6. Workflow Engine;
7. Modelagem da Ontologia;
8. Consulta SparQL;

O item 1 encontra-se no Capítulo 1, o item 2 é abordado no capítulo 2 e os itens 5 ao 8 serão abordados no Capítulo 4. Os itens 3 e 4 serão abordados neste Capítulo.

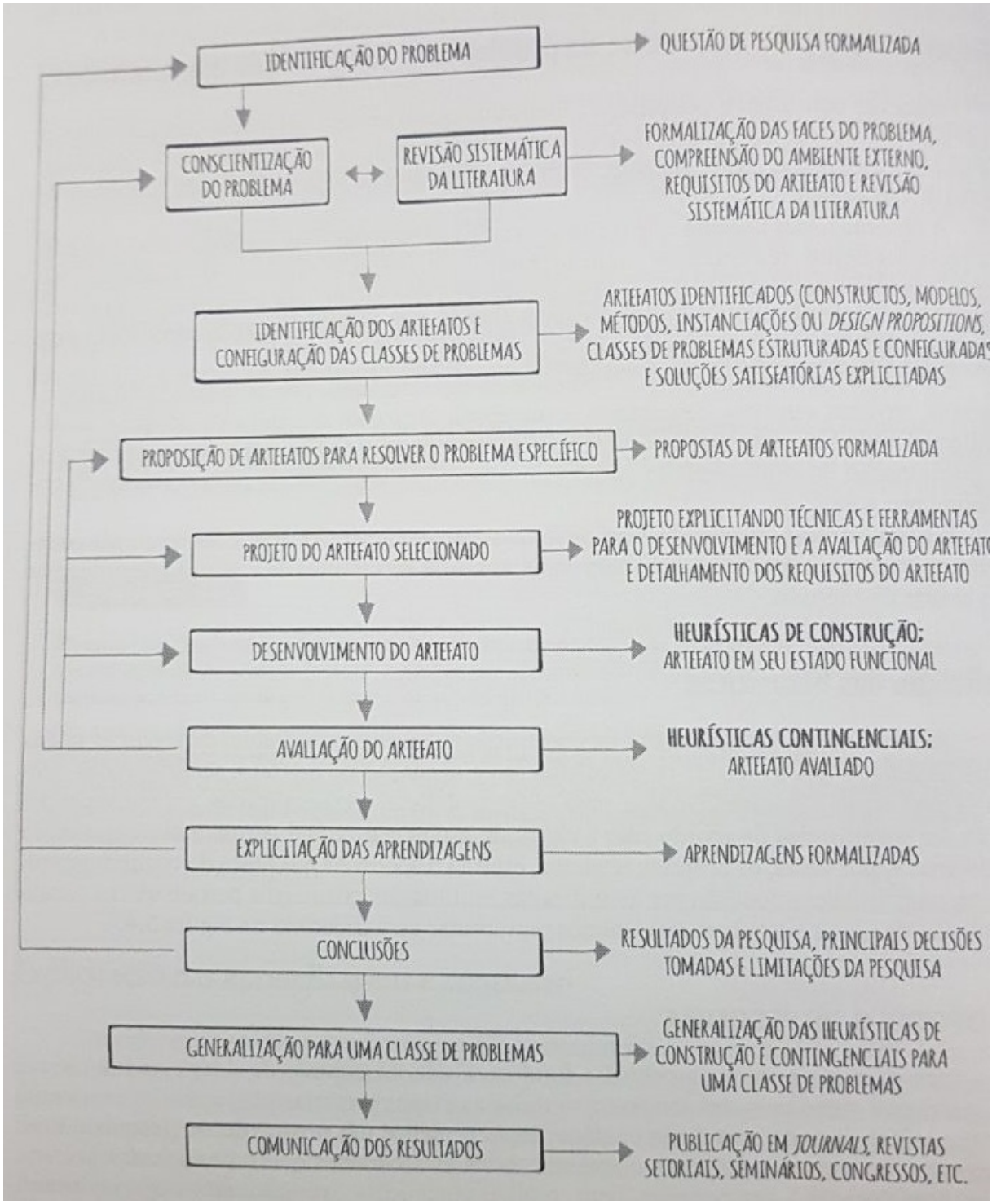

Figura 3.1: Ilustração da metodologia DSR. Imagem retirada de [5].

3.1 Pesquisa de Ferramentas

Nesta etapa foi realizado através de pesquisas quais ferramentas melhores se adequam para serem usadas no projeto. O critério das escolhas foram divididas de duas maneiras. A consideração de respeitar a arquitetura proposta em [1] como um módulo WMS e por critérios elaborados para guiar a escolha da linguagem de programação, framework e bibliotecas. **Os critérios para se adequar ao módulo WMS**:

1. Framework de workflow a ser desenvolvido deve ser implementado visando o comportamento do o módulo (WMS) do projeto;
2. O módulo WMS deverá dispor de interface para comunicar com o KMS e o CMS;

Os critérios da escolha da **linguagem de programação e framework para o desenvolvimento** envolvem os seguintes pontos:

1. Disponibilidade de bibliotecas ou suporte para gerenciamento de processos de negócios em BPMN;
2. Ferramenta de modelagem flexível para modificações;
3. Disponibilidade de bibliotecas para ferramentas semânticas;
4. Frameworks e bibliotecas frutos de projetos sotfware livre [10] ou open-source [11];
5. Comunidade do projeto ou biblioteca ativa e disponível para eventuais dúvidas no desenvolvimento;
6. Uma documentação clara e forte dos projetos ou bibliotecas;
7. Curva de aprendizado da linguagem e do framework;
8. Limitações da linguagem, framework ou biblioteca;

Esta seção se encontra dividida da seguinte forma. A Subseção 3.1.1 apresenta a metodologia de pesquisa usada para pesquisar ferramentas existêntes e escolher uma direção para o trabalho. As Subseções à seguir seguem a lista de etapas nela listada. Na Subseção 3.1.2 pesquisamos por ferramentas existentes em geral, na Subseção 3.1.3 é pesquisado por ferramentas de *workflow* em RoR e Django Framework, na Subseção 3.1.4 pesquisamos por bibliotecas que dêem suporte a anotação semântica nas linguagens encontradas nas pesquisas nas Subseções 3.1.2 e 3.1.3 e na Subseção 3.1.5 é avaliado o caminho do trabalho em cada linguagem de programação pesquisada nas Subseções 3.1.2 e 3.1.3 e tomada uma direção do trabalho.

3.1.1 Procedimento de pesquisa

Na lista à seguir é apresentado a maneira que foi realizada a pesquisa pelas ferramentas que compõem o artefato:

1. Pesquisa por ferramentas de *workflow engine* de modo geral;
2. Pesquisa por ferramentas de *workflow engine* em Django Framework e Ruby on Rails;
3. Ferramentas semânticas existentes nas linguagens de programação encontradas nas etapas anteriores;
4. Conclusão

A primeira etapa, item 1.a, é realizada considerando o critério 1 de 3.1, sobre modelagem e execução de processos de negócios em BPMN, independente da linguagem de programação ou framework utilizado. São pesquisados ferramentas existentes e por ferramentas existentes dentro dos frameworks listados em 1.b . Essas duas opções escolhidas são alvo da pesquisa devido essas linguagens e frameworks serem favoráveis aos critérios 6, 3 e 5 de desenvolvimento.

O item 2 consta de verificar as disponibilidades de ferramentas semânticas e de workflow existentes que dêem suporte às linguagens resultados da pesquisa no item 1.

No item 3 pesquisamos por ferramentas que auxiliem na modelagem de ontologias e realizar a etapa de modelagem de uma ontologia leve de domínio que satisfaça as necessidades da aplicação.

Ao final do levantamento é identificado o melhor candidato de cada linguagem de programação para desenvolver o artefato do módulo WMS. Com o levantamento de prós e contras entre as opções a escolher, decidir o caminho mais favorável e com menor risco ao êxito da etapa de desenvolvimento.

3.1.2 Pesquisa por ferramentas e bibliotecas existentes

Com a pesquisa direta por trabalhos e bibliotecas existentes, em consideração com os critérios listados, encontramos os seguintes resultados:

- bpmn-js para javascript;
- Camunda, Activity e JBPM para java;

O primeiro item na lista apresentada acima, é uma ferramenta de modelagem BPMN já utilizada e citada nos trabalhos [7, 30]. A biblioteca faz parte do **bpmn-io** [34], um projeto de código aberto*(open-source)*. O fato de ser na linguagem de programação Javascript é

interessante devido a sua capacidade de poder ser integrada independente do framework escolhido. Com a biblioteca **bpmn-js** é possível modelar intuitivamente diagramas na linguagem BPMN. Após a modelagem do BPD, o bpmn-js possibilita salvar a imagem do diagrama em formato .svg e o BPD gerado em formato XML, com a extensão .bpmn .

No segundo item listado. **Camunda**, **Activiti** e **JBPM** são projetos *open-source* de bibliotecas *workflow* em Java. Os ambientes de execução de processos de negócios são estudados no trabalho [30], para aplicação de semântica. Para avaliar melhor cada uma das três encontradas e possivelmente escolher uma para o projeto olhamos cada uma delas mais a fundo. O projeto jBPM foi criado em 2006. Quatro anos depois o Activiti é originado de um *fork* do jBPM e em 2012 o Camunda é originado de um *fork* do Activiti [44].

Para o Java, o Camunda, o jBPM e o Activiti suportam modos de implantação integrados e autônomos. Uma implantação integrada permite executar o BPM *Engine* como parte de um aplicativo existente (dentro da mesma JVM). Isso pode fornecer benefícios de desempenho à medida que os dados são passados na memória em vez de chamadas de rede. Uma implantação independente expõe várias funções da API que podem ser invocadas por um cliente via REST. Isso pode ser benéfico caso se deseja reutilizar uma única instância de *workflow* em diferentes aplicativos. Também pode ser uma maneira de integrar um aplicativo ao *workflow* engine que pode ser escrito em uma linguagem de programação diferente da API do produto BPM [44]. O Trabalho [30] e as fontes [45, 46] comparam os três projetos. Para o desenvolvimento deste artefato, optamos pelo Camunda como melhor candidato ao Java devido a documentação intuitiva apresentada no site do Camunda[1] e a quantidade de tutoriais disponíveis.

3.1.3 Pesquisa por ferramentas em Django Framework e Ruby on Rails

Django Framework

Para o Django FrameWork foram encontrados os pacotes listados na tabela 3.1.

Uma tabela com os status dos repositórios listados pode ser acessada em [47]. Dentre os pacotes exibidos, aparentemente apenas o *viewflow* ainda se mantém ativo e dando suporte ao Python 3. Os pacotes restantes foram descontinuados. Desta forma, aparentemente o *Viewflow* é o único pacote na categoria que está sendo mantido. O pacote é uma versão gratuita do *Viewflow Pro* e provê interface gratuita para rodar e instanciar processos no modelo BPMN. Em análise à documentação do pacote, a criação do diagrama BPMN se

[1] `https://docs.camunda.org/get-started/`

Tabela 3.1: Pacotes de workflow para Django Framework

Pacote
DJANGO-VIEWFLOW
ACTIVFLOW - WORKFLOW ENGINE
DJANGO-RIVER
WORKFLOW
DJANGO-XWORKFLOWS
DJANGO-FLOWS
DJANGO-WORKFLOWS
GOFLOW
DJANGO WORKFLOW

encontra *hardcoded*, e os recursos de importação e exportação de diagrama BPMN para a aplicação é disponibilizada apenas na versão Viewflow Pro.

A importação e exportação do diagrama modelado no framework são essenciais para a construção do projeto. Entretanto, o Viewflow Pro é desinteressante ao desenvolvimento do artefato devido ao critério 3. Outro fato foi a documentação fraca e incompleta do software na versão gratuita, o que fere o critério 5 para desenvolvimento listado.

Ruby on Rails (RoR)

Para Ruby on Rails, o projeto mais atual encontrado foi o rails_workflow[2]. Atualmente o projeto não aparenta estar em uma fase madura e fere os critérios 1 ,4 ,5 e 7. Neste framework, o mais próximo encontrado foram bibliotecas de *statemachine* que podem auxiliar em um desenvolvimento de *workflow engine*.

3.1.4 Suporte da linguagem à ferramentas semânticas

Para cada linguagem de programação encontrada nas pesquisas das Subseções anteriores 3.1.2 e 3.1.3, analisaremos o suporte a bibliotecas semânticas. Ao final das pesquisas, encontramos resultados para seguir o trabalho nas linguagens de programação à seguir:

- Java
- Ruby
- Python

Todas as três linguagens analisadas fornecem suporte de bibliotecas semânticas.

O Java fornece suporte à ferramentas semânticas através do *framework* livre e *open-source* Jena[3]. O *framework* auxilia para a construção de aplicações de Web Semântica e

[2] `https://github.com/madzhuga/rails_workflow`
[3] `https://jena.apache.org/`

Dados Conectados. O projeto se encontra em uma fase madura e fornece serviço de API via REST, uma maneira de integrar aplicativos modularizados. Desta forma, o Jena é favorável aos critérios de modularização WMS.

O Python fornece a biblioteca Ontospy para leitura de RDF e ontologia e para suporte a ferramentas semânticas de leitura e busca em ontologia.

o Ruby contém as bibliotecas semânticas rdf, sparql e linkeddata. Ambas as implementações com as bibliotecas semânticas do ruby ou python podem ser facilmente implementadas na aplicação.

3.1.5 Considerações sobre a arquitetura do projeto

Nesta subseção analisaremos a melhor direção a se seguir o trabalho entre as três linguagens de programações apontadas. com a etapa de desenvolvimento do artefato.

Para a opção de desenvolvimento em Django Framework, é mais interessante o *workflow engine* ser desenvolvido do que usar o Viewflow. Assim, o desenvolvimento constará da integração com bpmn-js e o *workflow engine* desenvolvido dentro da plataforma implementado com um banco de dados relacional, como MySQL ou PostgreSQL. O Python fornece a biblioteca ElementTree para leituras automáticas nos formatos XML, que pode ser utilizado para a leitura dos arquivos em extensão .bpmn gerados pelo bpmn-js.

Para a opção de seguir com o framework RoR, têm-se a biblioteca webpacker[4] para gerenciar pacotes javascript utilizando o Webpack[5] e Yarn [48] como gerenciador de pacotes javascript. Neste framework também é mais interessante o desenvolvimento da workflow engine dentro da aplicação. Assim como no Django Framework, a *workflow engine* pode ser facilmente simulado com o uso de um banco de dados relacional junto a uma integração com o bpmn-js. Para a integração dos arquivos gerados pelo bpmn-js, o RoR fornece a biblioteca Nikogiri para leitura de arquivo em formato XML.

A opção em Java é a que mais se aproxima em complexidade e funcionalidades para a elaboração do artefato. O fator limitante da escolha do Camunda é devido a linguagem de programação Java pra web exigir uma maior curva de aprendizado e uma complexidade maior que as linguagens Python ou Ruby.

Seguiremos o desenvolvimento no RoR devido a recente adição de pacotes que fornecem suporte à integração de módulos Javascript.

Integração bpmn-io e Ruby on Rails

De acordo com [34] atualmente existem duas formas para adicionar a biblioteca bpmn-js à uma aplicação. Uma versão pré-empacotada da biblioteca, tudo em um, permite adicionar

[4] `https://github.com/rails/webpacker`
[5] `https://webpack.js.org/`

o BPMN rapidamente a qualquer site. E a versão npm cuja configuração é mais complexa, entretanto dá acesso a componentes de biblioteca individuais e permite uma extensibilidade mais fácil. Faremos uma análise das duas aproximações para realizar a escolha de qual pacote melhor se adequa à proposta do artefato. Começamos com um repositório separado para implementar a versão simplificada da biblioteca BPMN para melhor compreender seu funcionamento. O repositório da implementação do módulo *bpmn-js modeler* pode ser encontrada no github[6]. A versão simplificada é mantida pelo gerenciador de pacotes bower[7]. O gerenciador de pacotes Bower é uma ferramenta descontinuada e motivado a sua não utilização. A sua utlização não é motivada devido a situação atual do pacote. No entanto, a biblioteca bpmn-js no pacote é mantida atualizada pelo bpmn-io.

A versão via npm é exemplificada com a utilização do **browserify**[8], para modulazrização de pacotes. O Ruby on Rails 5.1 , diferente do apontado até então pelo [34], por padrão adota um gerenciador de pacotes Javascript e um *bundler*(agrupador de pacotes) diferentes. Usa-se O gerenciamento de pacotes **Yarn** e o *bundler* **webpack**.

O pacote bpmn-js encontra-se também disponível pelo Yarn, o que possibilita o seu uso junto ao webpack de forma alternativa ao npm. O Yarn foi lançado pelo facebook em 2016, um gerenciador de pacotes de JavaScript de código aberto que prometeu instalações mais rápidas e confiáveis do que o existente npm. Ambos são semelhantes e possibilitam migração simples de um para o outro [9]. Para entender a diferença entre os pacotes, uma comparação entre os dois pode ser encontrada em [49].

Integração da modelagem de processos

Realizaremos a integração do bpmn-js com a utilização do gerenciador de pacotes Yarn. A escolha se deve pelo suporte do Ruby on Rails à dupla Yarn e Webpack embutida no framework. A migração para esta abordagem para o desenvolvimento de aplicações Javascript está se tornando uma tendência. Da mesma forma, enquanto o gerenciador de pacotes Bower ainda é mantido, é motivado pela sua comunidade a migração para o uso de Yarn e Webpack para projetos em Javascript. Desta maneira, essa escolha nos assegura a questão de manutenabilidade da ferramenta e suporte do framework RoR.

3.2 Pesquisa dos papéis na redação jornalística

Nesta Seção será definida os papéis e os processos necessários de uma redação jornalística para ser seguida como referência para o trabalho. O levantamento destas informações

[6] `https://github.com/marcelobbfonseca/bpmnio-simple-bower-html`
[7] `https://bower.io/`
[8] `http://browserify.org/`
[9] https://yarnpkg.com/en/docs/migrating-from-npm

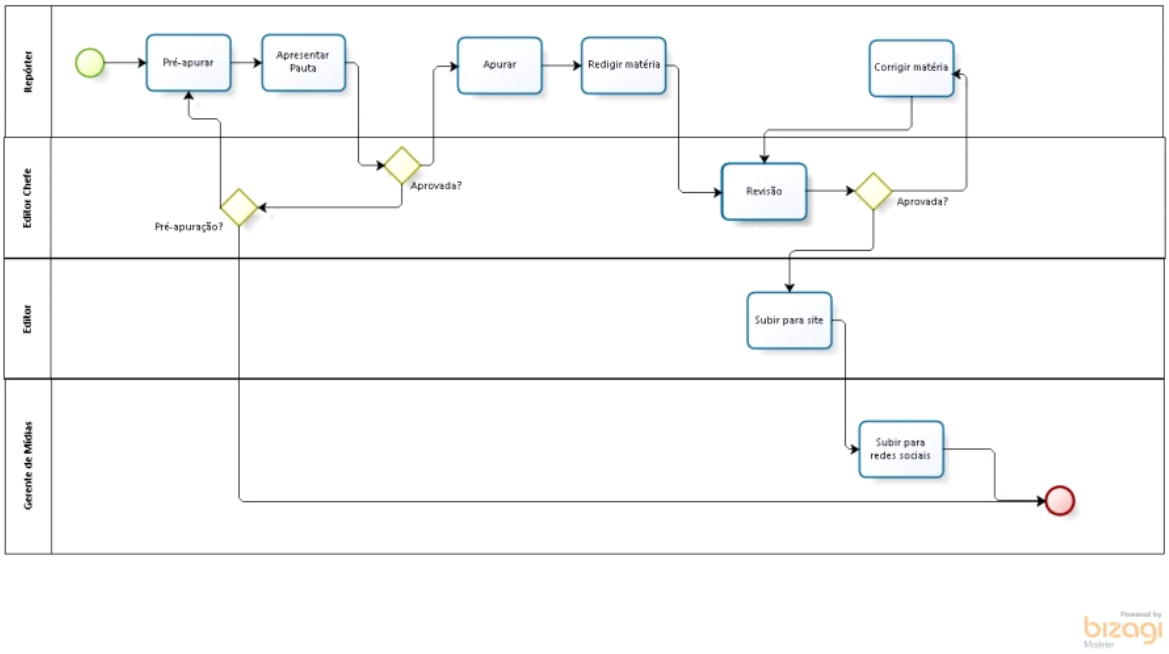

Figura 3.2: Processo da Campus Multimídia mapeado em [6]

partem do ponto de visto técnico. na Subseção 3.2.1 selecionamos o processo da redação jornalística a ser usado para este trabalho e na Subseção 3.2.2 mapeamos as tarefas de cada papel baseado em sua existência no processo da Subseção 3.2.2 anterior.

Os papéis e suas funções podem variar bastante de uma redação jornalística para a outra. Portanto, definir os papéis e suas funções para uma redação jornalística genérica não é uma tarefa fácil. Portanto para este trabalho usaremos como referência a redação jornalística **Campus Multimídia**.

O Campus Multimídia é composto por canais no Instagram, Facebook, Twitter e Site próprio. A redação jornalística trabalha com materiais jornalístico exclusivo para cada plataformas, utilizando as potencialidades de cada meio [50].

A preferência por essa redação ocorre devido a já existência de um mapeamento de processo da publicação da notícia que possamos usar para referenciar neste trabalho.

3.2.1 Processo

O processo da Figura 3.2 foi mapeado utilizando o bizagi[10]. Adotaremos o processo da Figura 3.2 da produção da notícia para seguir com o trabalho. Com o processo podemos definir os papéis nele encontrados e as suas respectivas tarefas. A seguir, como utilizaremos

[10]`https://www.bizagi.com/pt`

o pacote **bpmn-js**, o processo é refeito utilizando a sua modelagem. O novo BPD sofre poucas alterações e é exibido na Figura 3.4.

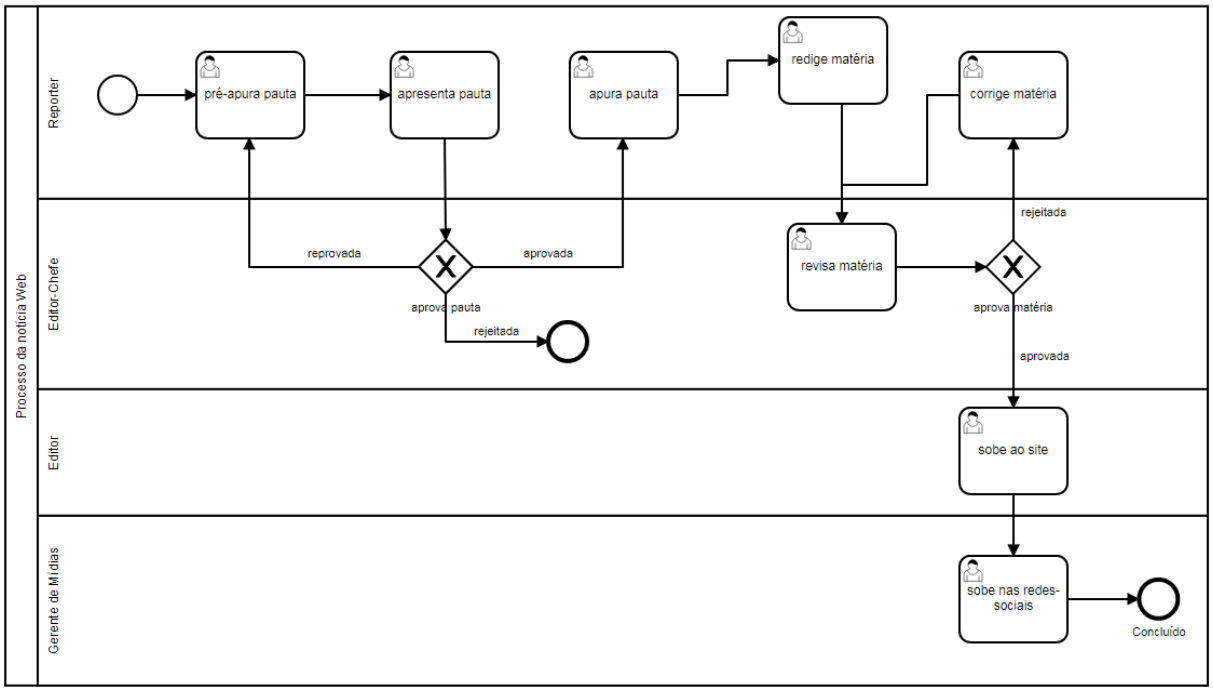

Figura 3.3: Processo da Campus Multimídia refeito no bpmn-js

Na nova modelagem, todas as atividades realizadas por uma entidade são convertidas para *user task*. Essa mudança será necessária pois o algoritmo de sugestão automática que fará a busca SPARQL vai percorrer o arquivo BPMN do processo em busca dos nodos de tipo *user task*.

Para validar o processo e os papéis nele contido. Foram realizados duas entrevistas com dois ex-membros, um do campus multimídia e outro do campus online. Do tempo da modelagem do processo em [6], o processo na Campus Multimídia havia sofrido algumas mudanças por consequência de sua alta rotatividade. A cada semestre acontecem mudanças dentro da redação o que causa alterações nos processos. Porém, as funcionalidades de cada papel não sofreram fortes mudanças e as etapas do processo da produção da notícia sofreram poucas alterações.

As mudanças que afetam o processo mapeado:

- O papel **editor** executava as mesmas funções do **editor-chefe** e foi retirada devido a ausência de tarefas;
- O papel **gerencia de midias** não existe mais, sendo o papel dela passada para o editor;

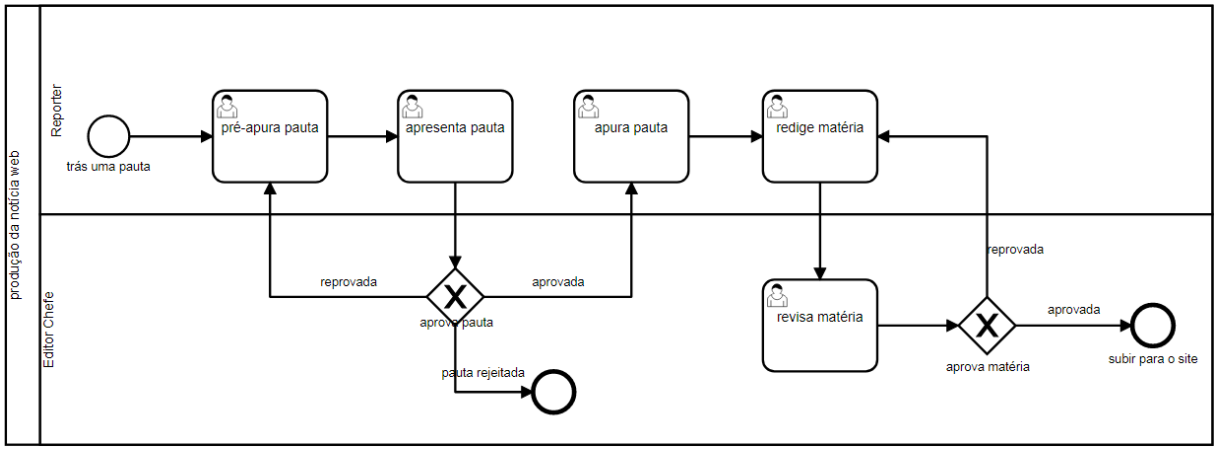

Figura 3.4: BPD da produção da notícia atualizado

- Qualquer papel pode sugerir uma pauta;
- Editor-chefe fica acima do editor e pode desempenhar o mesmo papel do editor;

Com base nas mudanças listadas, alteramos o processo modelado para o representado na Figura 3.4. Apesar de qualquer papel na redação poder trazer uma pauta. Representaremos como papel do repórter para simplificar a representação do processo e a modelagem da OWL.

3.2.2 Tarefas

Para determinar quais papeis utilizaremos e as tarefas responsáveis por cada papel em nossa ontologia, usamos o novo processo da Figura 3.4 como referência. Desta forma, temos os papeis **Reporter** e **Editor**. Com as tarefas presentes no processo podemos definir a tarefa que se encontra na raia de um papel, a tarefa definitiva daquele determinado papel. Desta forma podemos anotar o domínio do processo de forma que possamos organizar triplas RDF como apresentado na tabela 3.2.

Tabela 3.2: Mapeando as tarefas de cada papel em tripla RDF

Sujeito	Predicado	Objeto
Repórter	pré-apura	pauta
Repórter	apura	pauta
Repórter	apresenta	pauta
Repórter	redige	matéria
Editor-chefe	revisa	matéria
Editor-chefe	aprova	pauta
Editor-chefe	aprova	matéria

O próximo Capítulo detalha as etapas de desenvolvimento conduzidos para validar a metodologia e os princípios de modelagem descritas neste capítulo.

Capítulo 4

Desenvolvimento do projeto

Este Capítulo descreve quais foram os procedimentos necessários para implementar o *framework* web proposto no objetivo geral, a sua documentação e a arquitetura. A descrição detalha a implementação e integração dos módulos para a construção do artefato. Desta forma, ao final deste Capítulo ficarão mais claras as razões para as decisões de implementação, arquitetura e o funcionamento do artefato. O repositório do trabalho pode ser encontrado no github[1].

4.1 Framework Semântico

Nesta seção explicaremos o Framework Semântico proposto em [1] mais detalhadamente e como ela afeta no desenvolvimento da aplicação.

O Framework semântico é composto principalmente por três módulos:

- Content Management System (CMS)
- Workflow Management System (WMS)
- Knowledge Management System (KMS)

Como explicado brevemente na introdução pela Figura 1.2, este trabalho representa O módulo WMS do Framework Semântico. O WMS não aparece diretamente para o usuário, uma vez que ele normalmente é algo interno ao sistema de informação[1].

O Diagrama de casos de uso do WMS é representado na Figura 4.1 à seguir:

[1] `https://github.com/marcelobbfonseca/workflow-api`

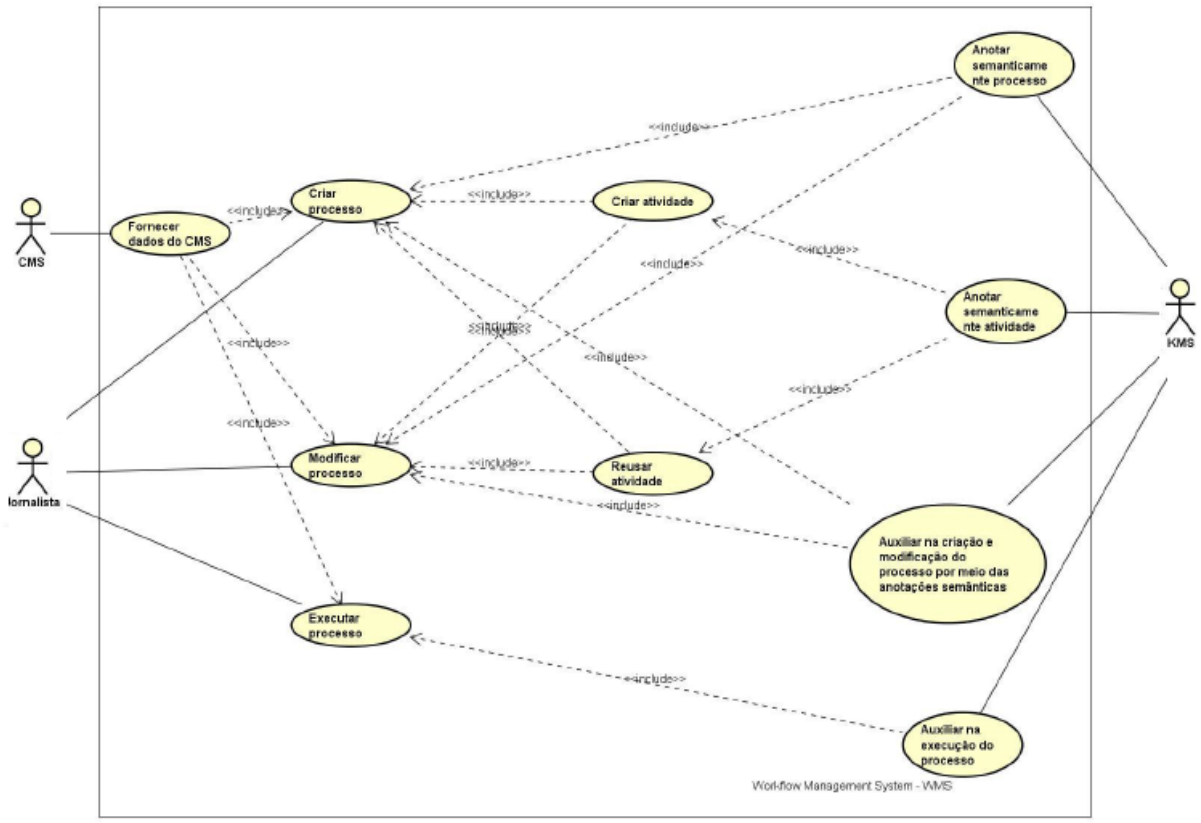

Figura 4.1: Diagrama de caso de uso do WMS em [1]

A principal influência na decisão de desenvolvimento que isso causa, é direcionar o desenvolvimento do módulo WMS neste trabalho para uma Application Programming Interface (API). As funcionalidades dependentes dos módulos KMS e CMS também serão desenvolvidos no artefato para alcançar o objetivo proposto.

4.2 Arquitetura

No *framework* WMS proposto, as funcionalidades de *workflow* devem ser comunicáveis com o módulo CMS. Os dados dos processos armazenados no WMS serão exibidos e modificados através de UI no módulo CMS. Após as alterações dos dados no CMS, eles são retornados ao WMS e suas alterações salvas no WMS. A realização desta comunicação, foi desenvolvida no artefato com o uso do padrão Representational State Transfer (REST), uma abstração de arquitetura da W3C como protocolo. A troca de informações neste padrão pode ser representada pela Figura 4.2 à seguir:

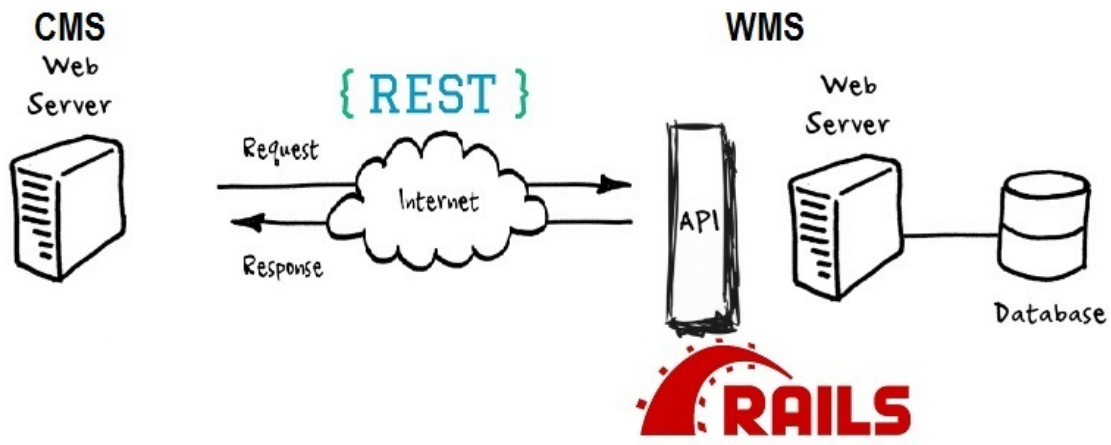

Figura 4.2: Comunicação entre módulos

O protocolo possibilita a identificação dos dados no formato XML, HTML ou JSON. Para simplificar, utilizaremos o protocolo em conjunto com o formato JavaScript Object Notation (JSON) [51]. O formato de notação irá prover a troca de dados entre a aplicação WMS e o CMS. Desta forma o módulo WMS funciona como uma web API REST service. O RoR fornece bibliotecas e geradores para dar suporte ao desenvolvimento de API utilizando JSON, apresentamos em detalhes as bibliotecas e seu funcionamento na seção de configurações de ambiente do artefato e no apêndice A.1.

Utilizamos a ferramenta **Postman** para auxiliar nas requisições HTTP e para mapear todas as rotas utilizadas. A lista das rotas utilizadas para o trabalho estão exibidas na Figura 4.3. Uma documentação de seu uso é disponibilizado no apêndice A.4.

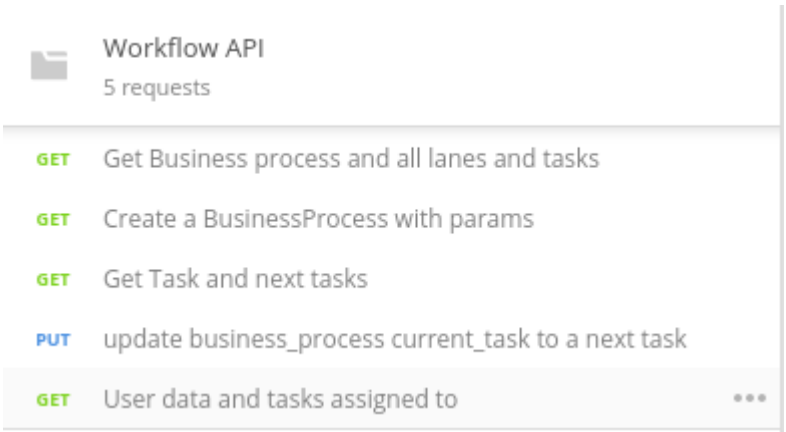

Figura 4.3: Lista de rotas do servidor

O acesso aos dados do WMS pelos usuários da aplicação serão através do CMS por meio da interface REST. Desta forma, a autenticação no módulo e manipulação direta dos dados é encarregada apenas ao administrador. Suas funções são de modelagem e gerência dos processos. As funcionalidades para ele contidas são manipular os dados através do painel de administrador e modelar processos.

O WMS requer algumas dependências dos outros módulos. Entretanto como os outros módulos ainda não se encontram desenvolvidos, as funcionalidades do KMS ou CMS que apresentam alguma dependência para o andamento do projeto serão desenvolvidas neste trabalho. A funcionalidade de instanciar um usuário à uma tarefa, o armazenamento da ontologia e a busca semântica serão implementados no artefato.

4.2.1 Arquitetura do Ruby on Rails

O RoR utiliza a arquitetura como base no padrão de projeto Model View Controller (MVC). O MVC possibilita a divisão do projeto em camadas muito bem definidas. Cada uma delas, o *Model*, o *Controller* e a *View*, executa o que lhe é definido e nada mais do que isso [52].

O RoR fornece diversos geradores e componentes para auxiliar no desenvolvimento de aplicações. Para elaborar o trabalho usaremos os seguintes componentes:

Active Record É o M no MVC. O Active Record facilita a criação e uso de objetos de negócios cujos dados exigem armazenamento persistente em um banco de dados. É uma implementação do padrão Active Record, que em si é uma descrição de um sistema *Object Mapping Relacional*[2].

Active Record Associations. "No *Rails*, uma associação é uma conexão entre dois modelos *Active Record*" [53]. Eles fazem operações comuns mais simples e fáceis em seu código. As associações podem enriquecer as conexões do modelo relacional e serão importantes para resgatar informações na aplicação(por exemplo, para as tarefas, podemos associar a ela as tarefas seguintes no processo ou suas tarefas anteriores.

Active Record Callbacks "são métodos que são chamados em determinados momentos do ciclo de vida de um objeto. Com chamadas de retorno, é possível escrever o código que será executado sempre que um objeto Active Record for criado, salvo, atualizado, excluído, validado ou carregado a partir do banco de dados" [53]. Isso auxilia para efetuar as transições de estados de um processo. Após mover para uma próxima tarefa no processo, podemos automaticamente atualizar o status da nova tarefa para algo como "inicializada"ou "ativa", e a tarefa anterior para "terminada"ou "completa".

[2] `http://guides.rubyonrails.org/active_record_basics.html`

Filters. "Os *filters* são métodos que são executados antes, depois ou em torno de uma ação de uma controladora " [53] e serão usadas para melhor modularizar o código nas controladoras com o tratamento de parâmetros.

4.3 Desenvolvimento do artefato

A elaboração do artefato foi divida em seis etapas apontadas a seguir:

1. Módulos e Bibliotecas;
2. Modelagem dos banco de dados e modelos relacionais;
3. Modelagem de processos bpmn;
4. Criação automática de processos;
5. Adicionar funcionalidades de workflow;
6. Modelar a ontologia leve de domínio para uma redação jornalística;
7. Busca SPARQL;

4.3.1 Módulos e Bibliotecas

Nesta Seção consta um visão geral dos principais módulos e todas as bibliotecas necessárias para o desenvolvimento do artefato. Para entendermos a arquitetura do artefato desenvolvido, passaremos por cada um dos módulos e bibliotecas necessárias que compõem a aplicação. À seguir a integração das principais ferramentas para desenvolver a busca semântica podem ser apresentados na Figura 4.4 a seguir:

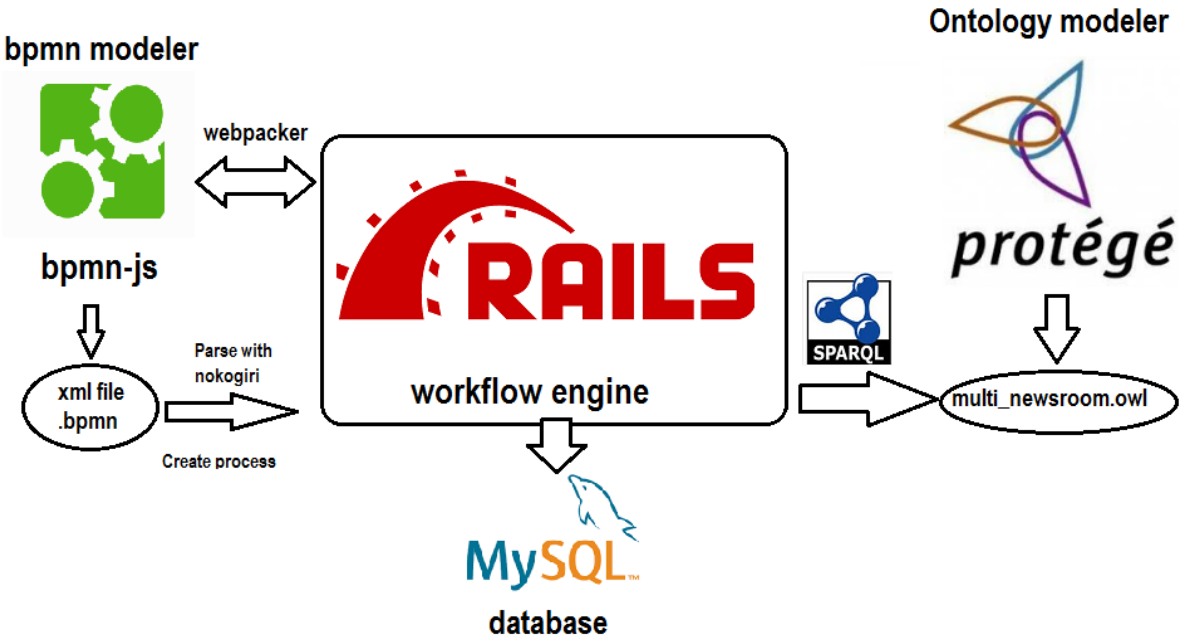

Figura 4.4: Arquitetura do artefato

O arquivo multi_newsroom.owl representado na Figura 4.4 é a nossa ontologia modelada em OWL. Todas as ferramentas apresentadas na imagem com exceção da ferramenta de modelagem *Protégé* se encontram embutidas no *framework* RoR.

A tabela 4.1 à seguir apresenta as bibliotecas da linguagem ruby que foram necessárias para desenvolver o projeto. Entretanto, outras linguagems de programações terão seus respectivos pacotes encarregados pelos mesmos serviços ou por serviços semelhantes.

A preparação do ambiente de trabalho pode ser encontrado no apêndice A.1

A lista de gems se encontram na tabela a seguir:

Tabela 4.1: Bibliotecas ruby utilizadas

Gem	Descrição
Nokogiri	É um Rubygem que permite manusear HTML, XML, SAX e Parsers de leitura com XPath e suporte de seletor CSS.
rdf	Uma biblioteca de Ruby para trabalhar com dados de Resource Description Framework (RDF). Dependente para gem sparql.
linkedata	Uma meta-distribuição de RDF.rb.
sparql	Biblioteca SPARQL ruby.
webpacker	Para gerenciar módulos JavaScript no Rails.
cancancan	Biblioteca de autorização para Ruby on Rails.
devise	Solução de autenticação flexível para Rails com Warden.
rails_admin	RailsAdmin é um mecanismo Rails que fornece uma interface fácil de usar para gerenciar dados.

4.3.2 Modelagem do banco de dados e modelos relacionais

Para a modelagem dos dados neste trabalho, levamos em consideração o *workflow engine* e a armazenagem do diagrama e a ontologia. O *workflow engine* exige a modelagem das entidades de forma a poder simular de maneira genérica qualquer processo. Podemos nos beneficiar do modelo relacional para simular a transição pelos seus estados de um processo e instanciar usuários para as tarefas. A utilização de um banco de dados relacional assegura a persistência dos dados e permite facilmente a navegação pelo processo através de ligações entre entidades através de chaves estrangeiras. Desta forma, modelamos no MySQL Workbench um processo genérico o suficiente para atender sobre qualquer diagrama modelado. A modelagem segue a linguagem BPMN e é apresentada o modelo composto de 7 tabelas explicadas a seguir na Figura 4.5:

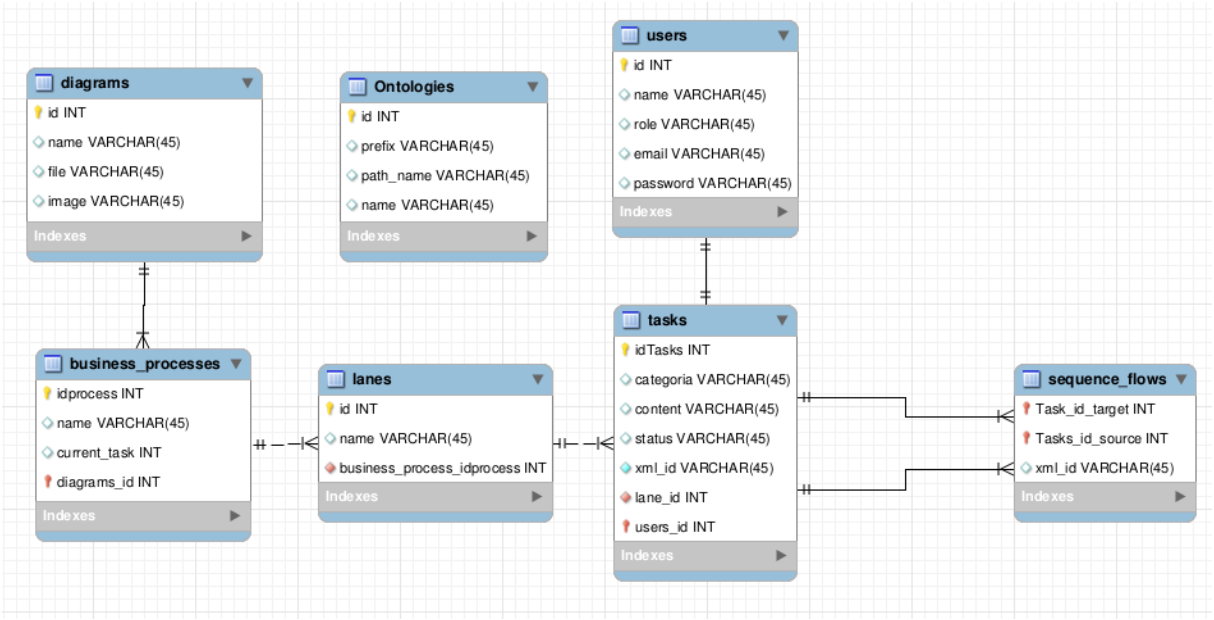

Figura 4.5: Modelagem no mysql workbench.

diagrams

Um diagrama é composto de seu arquivo xml e sua foto em .svg, como a ilustrado na Figura 3.4. Ilustra-se essa característica na figura 4.5 Ambos arquivos gerados pelo modelador de processos do bpmn-js. Um diagrama tem uma relação 1 pra N com processos, desta forma vários processos podem atuar sobre um diagrama.

business_processes

Representa a entidade do processo. Um processo tem uma chave estrangeira para a tarefa em execução, current_task, para fácil acesso em qual ponto do processo, o processo se encontra. Um processo possui relacionamento 1 para N com raias, o que permite um processo ter mais de um papel atuando no processo.

lanes

Representa uma raia na linguagem BPMN. Composta de seu nome e uma chave estrangeira para seu o processo. Uma raia é composta de várias tarefas.

tasks

Representa o nodo da linguagem gráfica *BPMN*. Portanto no projeto, um *gateway* pode ser considerado uma *task*. Uma *task* contém uma enumeração, *category*, para identificar

qual o tipo do nodo(ex: *user_task*, *task* ou *exclusive_gateway*), *xml_id* age como um identificador de si no arquivo XML durante o *parser*. *Content* é referente ao conteúdo dentro da tarefa(ex:Apurar pauta), status é um enumerador que classifica a tarefa como feita, em andamento ou não inicializada. A chave estrangeira para usuário instância para quem aquela tarefa é designada.

sequence_flows

Representa as setas de direcionamento da linguagem gráfica BPMN. Assim, identificamos a tarefa anterior e a tarefa seguinte. Uma tarefa tem uma relação 1 pra N com um direcionamento o que permite direcionamento para N tarefas seguintes e anteriores presentes no diagrama. A identificação precisa da tarefa anterior se dá pelo status da tarefa anterior que apresenta o status concluído.

users

representa os usuários que irão consumir a aplicação de alguma maneira. um usuário é constituído de nome, email e *role*. *Role* identifica qual o papel do usuário na aplicação(ex:administrador). A relação 1 para N com tarefas permite um usuário a ser instanciado a mais de uma tarefa em processos existentes.

ontologies

Armazena as ontologias no projeto. Para o projeto apenas uma ontologia foi utilizada e é armazenada localmente pelo seu arquivo de extensão **.owl**. *path_name* representa o seu local de armazenamento para acesso ao arquivo, seguidos de seu nome e prefixo para acesso.

4.3.3 Modelagem dos modelos relacionais

Ao total temos sete modelos relacionais, cada uma referente à sua tabela gerada do banco de dados relacional.

São adicionados relacionamentos associativos do *active record*, mencionados em 4.2.1, aos modelos relacionais. Detalhes referentes a implementação dos modelos relacionais estão disponíveis no Apêndice A.2. Desta forma, a partir de uma tarefa podemos acessar dados como as suas próximas tarefas e tarefas anteriores, a partir de um processo, conseguir acessar facilmente todas as suas tarefas e qual a sua tarefa atual.

Nos modelos relacionais *diagrams* e *ontologies*, é configurado o uso da *gem carrierwave* para permitir o armazenamento de arquivos. Desta forma, estas entidades permitem o armazenamento de arquivos.

4.3.4 Modelagem de processos bpmn

Esta seção consta de adicionar a modelagem de processos BPMN ao framework RoR. A inclusão do modelador de processos se dá com a integração do pacote **bpmn-js** ao artefato no *framework* RoR. Com essa integração adicionamos ao projeto a interface de modelagem. Por ser uma interface web, uma demonstração da interface bpmn-js para modelagem é disponibilizada na pagina do projeto[3]. Os detalhes referente a integração estão descritas no Apêndice A.3.

4.3.5 Criação automática de processos

Com a conclusão da etapa anterior podemos modelar e salvar processos de acordo com o padrão BPMN. Nesta etapa, geramos a criação automática de processos de acordo com algum BPD modelado. Como próxima etapa para atingir a funcionalidade de *workflow* será necessário iniciar um processo baseado no BPD gerado pelo modelador de processos. Através de uma requisição POST, recebemos um arquivo XML de diagrama como parâmetro. Desta forma, é feito uma leitura de sua estrutura e criado um processo no MySQL dinamicamente. Todo processo novo criado aponta sua chave estrangeira *current_task* para o evento de *StartEvent*. Os detalhes da implementação da criação automática de processos estão disponíveis no apêndice A.5.

4.3.6 Adicionar funcionalidades de workflow

Com o fim da etapa anterior, temos a modelagem de processos com interface e a sua criação automatizada. Nesta etapa adicionamos ao processo as seguintes funcionalidades de *workflow*:

- Um processo deve ser inteiramente exibido e onde ele se encontra. Sua tarefa atual, raias e tarefas existentes;
- Ter início e fim. Um processo deve percorrer suas tarefas até um evento de término;
- Uma tarefa deve conseguir instanciar um usuário para sua execução;
- Um usuário instanciado deve poder visualizar as tarefas de sua responsabilidade;

Através da requisição GET /business_processes/1.json na Figura 4.3 acessamos os dados da entidade de processo de negócios. Para satisfazer o primeiro item, adicionamos os dados de tarefa correspondente e uma lista de todas as suas raias do processo com suas respectivas tarefas. Para o item 2 utilizaremos a transição de estados em conjunto com o

[3] `https://demo.bpmn.io/` acessado em 23/11/2017

campo status. Assim que um novo processo é criado, todas as tarefas são iniciadas por padrão com o status **inactive**. A primeira tarefa do processo de categoria start_event, é iniciada como started. A transição de estados se dá pela requisição de atualização de um processo. Pela requisição REST, atualizamos o **current_task**. Com atualização é feita a transição de status da tarefa anterior para **completed** e a tarefa nova como **started**. Com a chave estrangeira da tabela de usuários em **tasks**, podemos designar uma tarefa para algum usuário no painel de administrador e relacionar um usuário da redação jornalística a suas tarefas ativas pelo qual foi designado e satisfazemos aos dois últimos itens.

4.3.7 Modelagem da ontologia de domínio

Após a determinação dos papéis dentro da redação jornalística escolhida, elaboramos a modelagem da ontologia no *prótégé* em cima dos papéis e tarefas de acordo com o novo processo de produção da notícia web usada como apresentado ao final do Capítulo 3. Com as classes previamente determinadas, a criação da ontologia se dá pelos quatro passos seguintes:

1. Hierarquia de classes
2. Hierarquia de objetos(predicados)
3. Atributos da classe.
4. Indivíduos das classes.

Ao final da primeira etapa, a hierarquia de classes da ontologia se encontra de acordo com a Figura 4.6 a seguir:

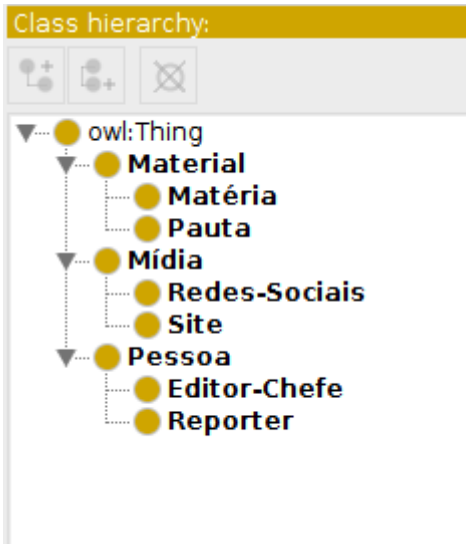

Figura 4.6: Modelagem das classes no *prótégé*

Classes que não se associam são modeladas disjuntas(ex:pessoa e material). A nomeação das classes segue o padrão de iniciar com letra maiúscula e underscore para caso de nome composto(ex: redes_sociais). Na etapa seguinte modelamos os objetos de acordo com as tarefas apresentadas no processo:

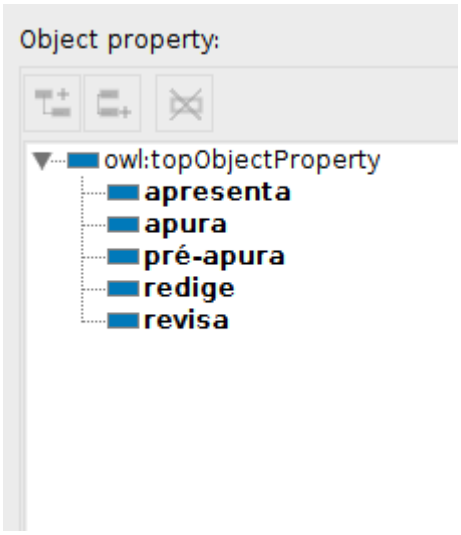

Figura 4.7: Modelagem da árvore de propriedas de objeto no *prótégé*

Toda **objectProperty** criada herda de **topObjectProperty**. A próxima etapa consta da criação dos indivíduos. Para atender a busca semântica de forma mais generalizada, modelamos os indivíduos reporter, instância da classe Reporter, editor, indivíduo da classe Editor. O mesmo se segue para matéria e pauta. Com essa modelagem construímos triplas -rdf mais genéricas para atender nosso objetivo. As formações das Triplas-RDF

geradas pela ontologia podem ser verificadas se estão de acordo com nossa necessidade através do validador rdf do W3C. O validador é disponibilizado e pode ser acessado no site [4].

4.3.8 Busca SPARQL

Nesta etapa da implementação mostraremos a implementação para realizar a busca SPARQL em cada tarefa de um processo para em seguida fazer a sugestão automática de papéis para cada uma delas. Com a modelagem da ontologia, são elaborados as triplas que desejamos consumir para nosso artefato. Queremos formar triplas de forma a conseguir identificar qual papel é responsável por qual tarefa. No validador do W3C, podemos verificar as triplas RDF formadas com nossos indivíduos criados como o mostrado na Figura 4.8 à seguir:

56	http://www.semanticweb.org/2017/multi-newsroom#reporter	http://www.semanticweb.org/2017/multi-newsroom#apresenta	http://www.semanticweb.org/2017/multi-newsroom#pauta
57	http://www.semanticweb.org/2017/multi-newsroom#reporter	http://www.semanticweb.org/2017/multi-newsroom#apura	http://www.semanticweb.org/2017/multi-newsroom#pauta
58	http://www.semanticweb.org/2017/multi-newsroom#reporter	http://www.semanticweb.org/2017/multi-newsroom#pré-apura	http://www.semanticweb.org/2017/multi-newsroom#pauta
59	http://www.semanticweb.org/2017/multi-newsroom#reporter	http://www.semanticweb.org/2017/multi-newsroom#redige	http://www.semanticweb.org/2017/multi-newsroom#matéria

Figura 4.8: W3C RDF Validation Service

Com a modelagem realizada, para cada tarefa presente no processo em análise fazemos uma busca com o predicado e objeto da tarefa. Entretanto, a estrutura das triplas para realizar a busca desejada é muito sensível. Para conseguir atender esta estrutura rdf, o processo precisará seguir uma regra de implementação. o conteúdo de cada tarefa deve ser escrita de forma a estrutura ser separada no formato “predicado”, “objeto”(ex: revisa pauta) para encontrar o sujeito desta tripla:

[4] urlhttps://www.w3.org/RDF/Validator/

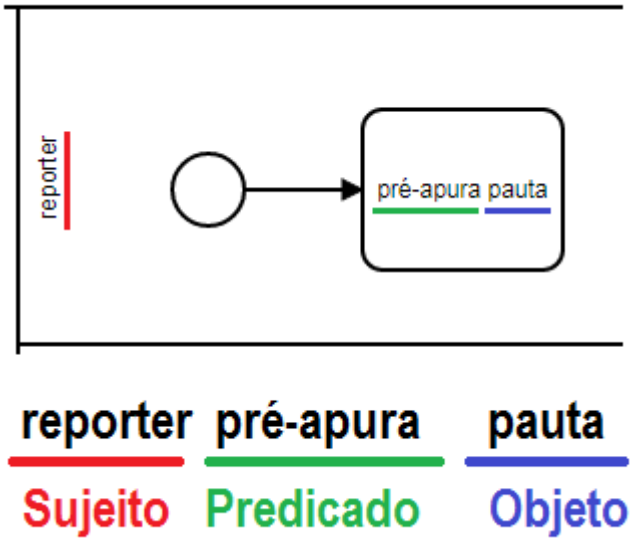

Figura 4.9: Tripla RDF no processo

Desta forma a tripla se encontra organizada como o papel, nome da raia no processo, sendo classificado como o sujeito na tripla e o par predicado e objeto presentes no conteúdo de cada tarefa como ilustrado na Figura 4.9. Desta forma a busca sparql vai retornar o sujeito da tarefa.

Salvamos e adicionamos o arquivo .owl da ontologia modelada ao artefato através do painel administrador. Para realizarmos a busca SPARQL Devemos percorrer todas as tarefas de um processo modelado e fazer a busca para cada uma delas e verificar os papéis. Para isso, lemos uma lista de tarefas de um BPD modelado e efetuamos a busca SPARQL para cada tarefa. A interface para esta verificação é representada pela Figura 4.10.

O formulário recebe o nome do BPD como parâmetro e realiza a pesquisa SPARQL. Internamente implementamos um módulo que faz a leitura do arquivo bpmn e retorna uma lista das tarefas existentes no diagrama. Assim, para cada elemento do vetor fazemos a busca SPARQL(para cada tarefa presente). Após a busca SPARQL, os resultados são armazenados como sugestões dos papéis em um novo vetor e são exibidas em tela. Uma desta tela da aplicação é ilustrada pela Figura 4.11.

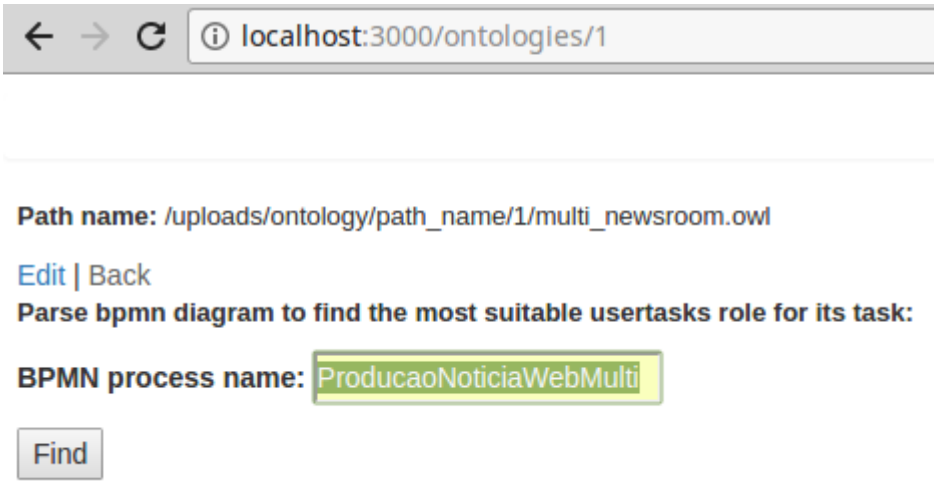

Figura 4.10: Captura de tela na aplicação: Consulta SPARQL na ontologia criada

Figura 4.11: Captura de tela da aplicação: Sugestões de papéis da consulta SPARQL.

4.4 Funcionamento

Uma documentação da configuração e um passo-a-passo da utilização do artefato desenvolvido pode ser encontrada neste link[5]. O artefato é composto da modelagem de processos, busca semântica e workflow engine. A autenticação no WMS é apenas para o responsável em modelar e alterar os processos da redação(administrador).

[5] `https://github.com/marcelobbfonseca/workflow-api`

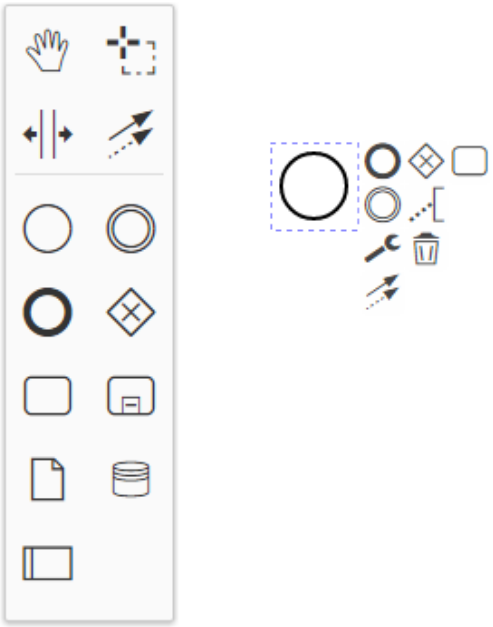

Figura 4.12: Captura de tela: Modelando um novo processo.

Ao autenticar no sistema, o administrador pode acessar a interface de modelagem para modelar um novo processo da geração da notícia web. A tela para modelagem de processos é representada pela Figura 4.12. A seguir, salvamos o diagrama em arquivo e em imagem.

Supomos que o administrador modele o processo da produção da notícia com um erro. A tarefa "apura pauta"fique designada para um Editor-Chefe, em vez de ao Reporter. Assim, o processo final a ser salvo é representado pela Figura 4.13

Ao acessar o painel de Administrador, criamos um novo diagrama para representar o diagrama recém-criado e entramos com o arquivo do diagrama e a sua imagem, ambos salvos anteriormente(O uso da imagem é opcional). Após salvarmos o diagrama, o painel do administrador se encontrará da seguinte forma representada na Figura 4.14

Temos agora um diagrama criado na aplicação. O administrador pode verificar se a modelagem das tarefas foi realizada corretamente de acordo com os papéis da organização. Através de interface ele pode realizar uma consulta que irá retornar as sugestões de papel mais indicado para cada tarefa identificada no diagrama. A interface de consulta é representada pela Figura 4.10. O resultado da Consulta no nosso contexto será representada pela Figura 4.15.

Desta forma, em caso de o modelador precisar desenvolver um novo processo de produção da notícia, após modelar e salvar o processo novo, ele pode consultar a sugestão automática das tarefas no processo. Caso exista a necessidade de se efetuar alguma mu-

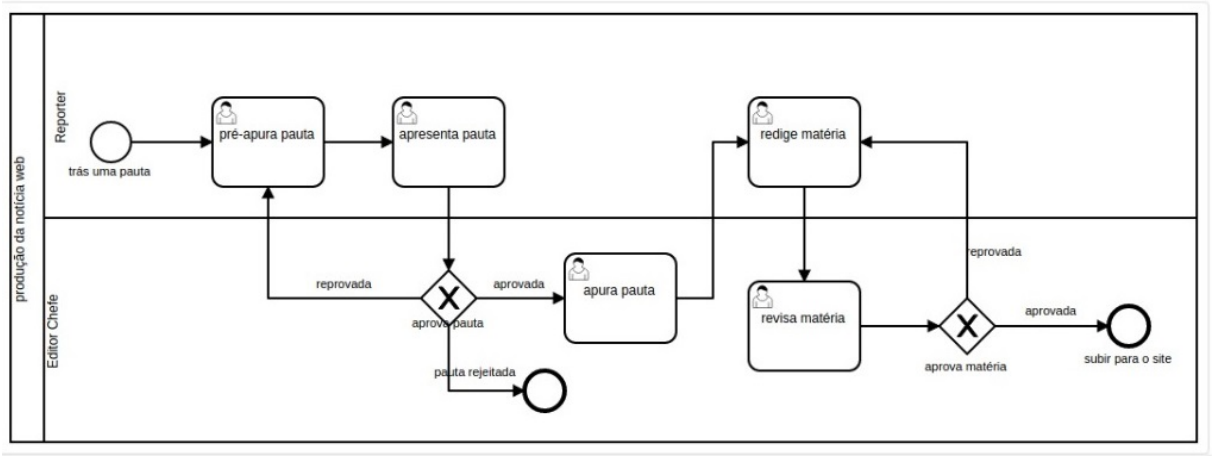

Figura 4.13: Captura de tela: Processo modelado com erro.

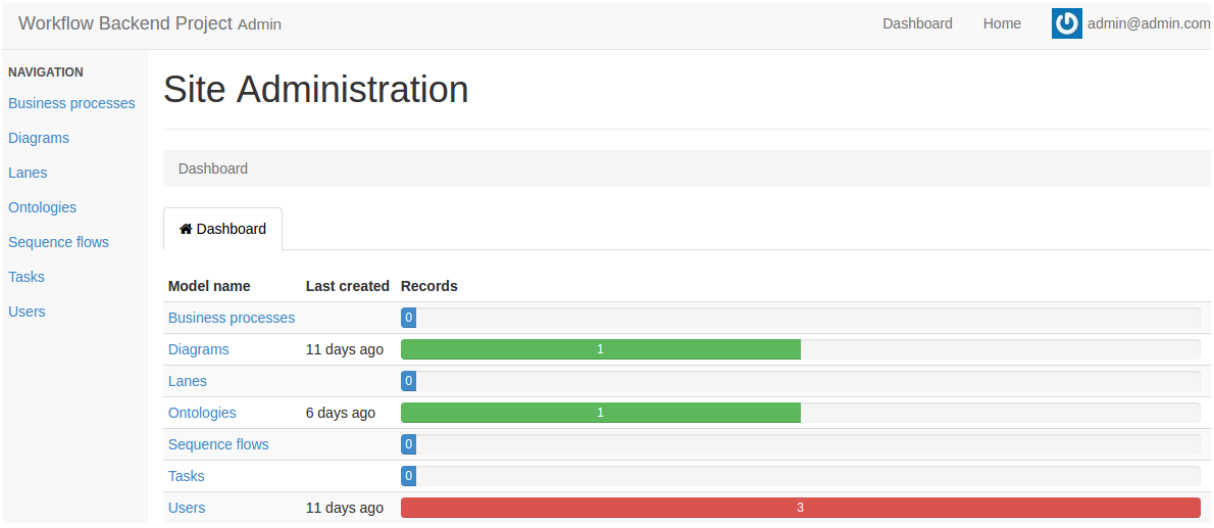

Figura 4.14: Captura de tela: Painel administrador após criar diagrama.

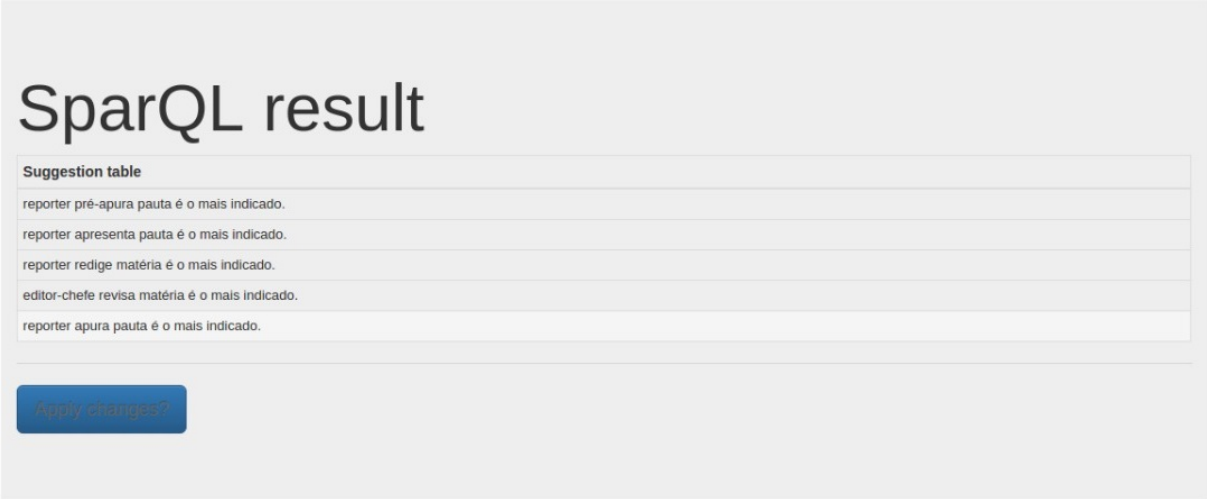

Figura 4.15: Captura de tela: Resultado da consulta SPARQL.

dança devido a um erro de modelagem apontada pela sugestão automática, a interface de modelagem é flexível o suficiente para editar o processo previamente modelado e salvar as alterações. Edição de um processo existênte pode ser ilustrado na Figura 4.16

Com o processo corretamente modelado, um novo processo pode ser criado em cima do diagrama construído. Passamos o nome do diagrama como parâmetro pelo *endpoint*:

```
1 GET /bpmn/create.json?name="bpmn_file_name"
```

Tudo ocorrendo bem, uma mensagem de sucesso será exibida. Ao retornar para o painel do administrador, ela deverá aparentar como na Figura 4.17.

O Administrador pode navegar pela lista de tarefas e designar uma tarefas para um usuários como representado na Figura 4.18. As informações de um usuário junto com uma lista de suas tarefas designadas, podem ser acessadas através do *endpoint*:

```
1 GET /users/1.json
```

Todas as outras funcionalidades de workflow desenvolvidas na Subseção 4.3.6 também se tornam disponíveis a partir da criação de um processo.

Com a implementação da busca SPARQL e o seu funcionamento, o desenvolvimento do artefato é finalizado. Na próxima seção apresentamos os resultados e conclusões sobre o artefato desenvolvido e sua contribuição.

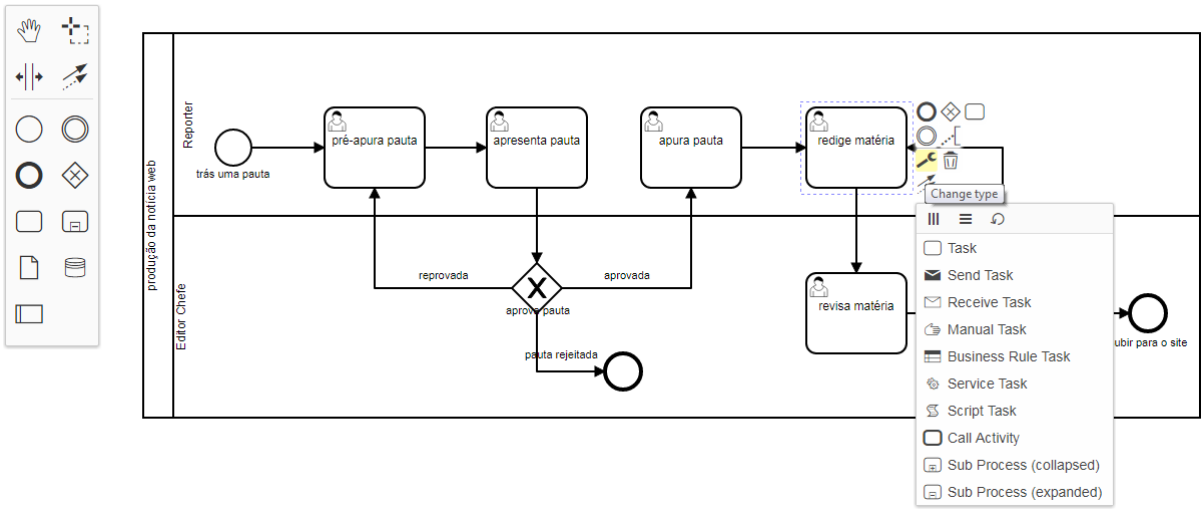

Figura 4.16: Captura de tela: Editando um processo existênte

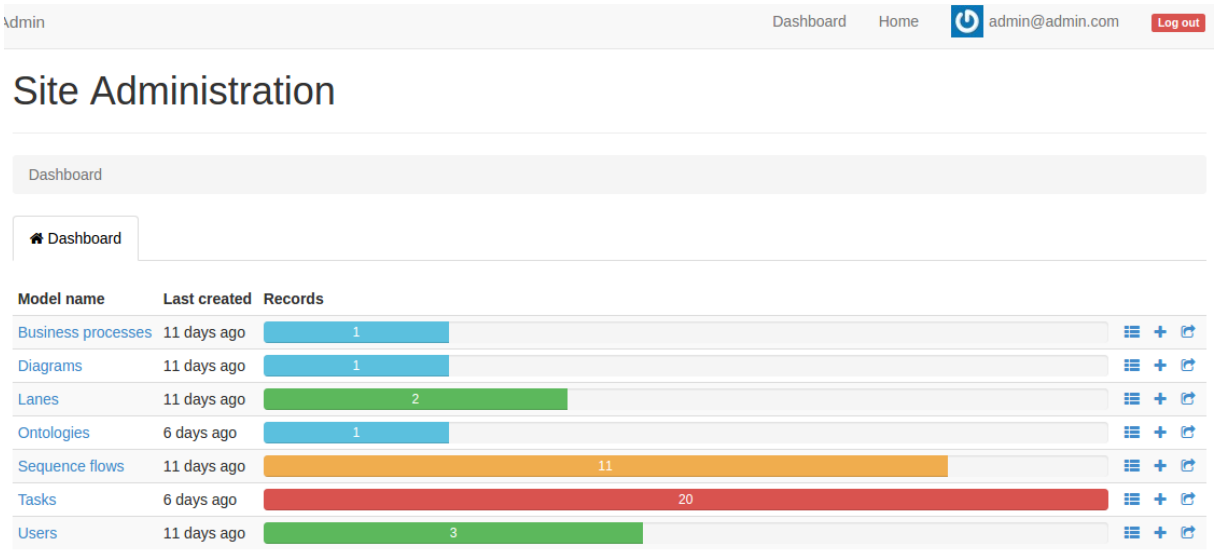

Figura 4.17: Captura de tela: Painel Administrador após criar um processo

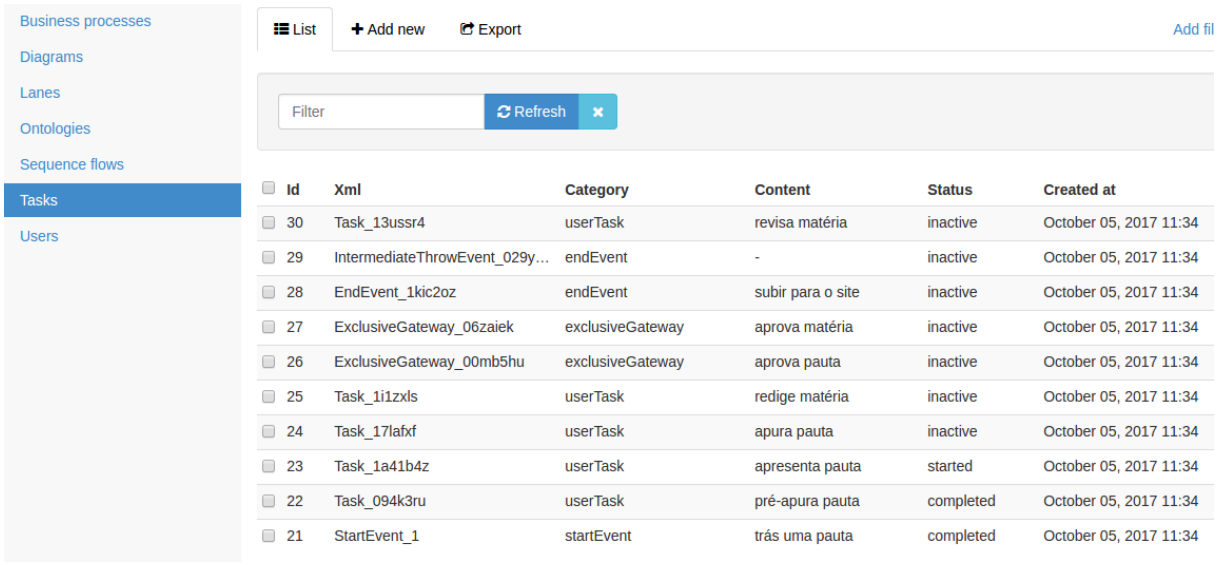

Figura 4.18: Captura de tela: Lista de tarefas

Capítulo 5

Conclusões

Os objetivos de implementação foram alcançados. Apenas com a utilização de ferramentas de código aberto concluímos o desenvolvimento deste artefato e foi possível cumprir com a especificação pelo objetivo geral.

Uma questão importante na pesquisa foi trabalharmos com a Campus Multimídia no intuito de viabilizar a proposta da solução com uma organização real. O protótipo implementa as funcionalidades básicas de um WMS com a capacidade de elaborar processos flexíveis para a produção de notícias. A flexibilidade se dá por meio da modificação de processos com o auxílio de ferramentas semânticas. A sugestão automática de papéis ,desenvolvida por meio de ferramentas semânticas, auxilia o modelador a corrigir possíveis erros de modelagem no momento de designar as tarefas em um processo.

A validação do funcionamento da sugestão automática foi feita com três processos. Os processos da Campus Multi exibidos nas Figuras 3.4 e 3.3 e o processo 4.13 intensionalmente errado para testar a sugestão automática.

5.1 Contribuição

O desenvolvimento do framework conseguiu demonstrar o auxílio da anotação semântica em processos de negócios com o uso da Ontologia desenvolvida. Para mais, a aplicação desenvolvida só utilizou de softwares de licença livre e está disponível[1] para quem tiver o interesse de utilizá-la para seus interesses.

5.2 Avaliação

Os objetivos de implementação foram alcançados. No entanto o artefato desenvolvido pode não apresentar uma interface trivial para a utilização um profissional na área de

[1] `https://github.com/marcelobbfonseca/workflow-api`

jornalismo. A aplicação abstrai grande parte das tarefas através da interface de usuário. Entretanto a configuração e usabilidade podem resultar em dificuldades. Para a modelagem de processos e manipulação de processos em geral, é exigido do modelador conhecimento na linguagem BPMN.

O funcionamento, teste e demonstração do protótipo ,no escopo deste trabalho , demonstra a sua viabilidade.

A avaliação pode ser conduzida em um ambiente experimental ou em um contexto real, de diferentes maneiras. Para isso haverá a necessidade de interação entre o pesquisador, usuários e as pessoas da organização na qual o artefato está sendo instanciado. A saída resultante da etapa de avaliação será o artefato devidamente avaliado, por meio das quais podemos explicar os limites do artefato e suas condições de utilização, ou seja, a relação do artefato com o ambiente externo em que irá atuar, qual foi especificado durante a conscientização do problema [5]. Desta forma, essa etapa entra para Trabalhos Futuros, a seção seguinte.

5.3 Trabalhos futuros

Este trabalho é uma parte de implementação do *framework* para redação jornalística [1] com *workflow* flexível. Os resultados alcançados são concretos, como todo trabalho que envolve uma implementação. Por ser um trabalho inicial, muito há que ser feito ainda para melhorar o poder semântico sobre a modelagem dos processos no artefato.

5.3.1 Anotação Semântica

Existem diversas formas de se fazer uma anotação semântica de processos. Elas irão depender do objetivo a ser atingido com a anotação. No caso para a demonstração deste trabalho, a anotação permite a sugestão automática do ator mais apropriado para realizar uma determinada tarefa. O artefato desenvolvido explora uma das funcionalidades aplicáveis com o uso de ontologia de domínio apresentada em [27]. O poder semântico com a anotação semântica pode ser enriquecida com as outras funcionalidades também listadas.

Neste trabalho foi implementado o item 1 listado. Como trabalho futuro pode ser viabilizado a implementação de um segundo item ou um aprimoramento do item feito.

Modelagem da Ontologia

A especificação dos indivíduos na ontologia são elaboradas de maneira genérica. Os indivíduos são papéis da redação jornalística e, no entanto, a modelagem de indivíduos como

pessoas físicas que trabalham na redação jornalística diminui a abstração e possibilita maior flexibilidade ao trabalho. A mudança aumenta a complexidade da busca SPARQL e a anotação semântica atende o *workflow* de forma mais específica no momento de instanciar tarefas. Desta forma possibilitamos a sugestão automática dos funcionários melhores indicados para uma determinada tarefa e não se limitar apenas aos papéis.

Anotação do processo

Algoritmos de *fuzzy string match* podem contribuir para deixar mais flexível as palavras escritas em uma *user task* durante a modelagem.

5.3.2 Workflow

Outras funcionalidades do [34] podem ser exploradas para incrementar ao *workflow*. Além da modelagem de processos, o bpmn-js também apresenta os módulos *Embed and Annotate* e *Extend*. *Embed and Annotate* estende a aplicação para exibir e escrever anotações que enriquecem as informações de um processo. *Extend* integra um mecanismo de processo no navegador com simulação de processo, estilo ou interatividade aprimorada[2].

[2] `https://bpmn.io/toolkit/bpmn-js/`

Referências

[1] Dr. Benedito Medeiros Neto, Edison Ishikawa e: *Jour 3.0: A newsroom framework for journalists*. Relatório Técnico, Departamento de Ciência de computação, 2017. ix, 1, 2, 19, 29, 30, 50

[2] https://camunda.org/bpmn/tutorial/. acessado em 20/11/2017. ix, 8

[3] *Free tools, information and resources*. http://linkeddatatools.com. ix, 9, 10, 11, 13

[4] Mijung Kim, Jake Cobb, M.J. Harrold Jake Cobb: *Efficient regression testing of ontology-driven systems*. ix, 12

[5] Aline Dresch, José Antonio Valle Antunes Júnior, Daniel Pacheco Lacerda: *Design science research: método de pesquisa para avanço da ciência e tecnologia*. bookman, Porto Alegre, BR, 2015. ix, 17, 18, 50

[6] Silva, Rafael Montenegro da: *Fluxo de produção de notícias dos campus online e multi*, 2016. Trabalhos Finnal da Disciplina: Jornalismo em Ambientes Digitais, Faculdade de Comunicação – FAC/UnB, 2016. ix, 25, 26

[7] Tobias Weller, Maria Maleshkova: *Adaptive semantic process modeling tool*. 2016. xi, 8, 14, 16, 20

[8] Journalists. ICFJ, International Center for: *A study of technology in newsroom*. 1

[9] Neto, Dr. Benedito Medeiros: *Proposta de pós-doutoramento. título do projeto: Uma proposta de modelo para um framework semântico de ambiente colaborativo para gestão da informação em redação jornalística.* 2016. Projeto de Pesquisa Vinculado: Edital MEC/MCTI/CAPES/CNPq N0 09/2014. CIC/UnB. 1

[10] FSF. https://www.gnu.org/philosophy/free-sw.html. acessado em 20/10/2017. 2, 19

[11] Initiative, Open Source. https://opensource.org/osd-annotated. acessado em 20/10/2017. 2, 19

[12] Dr. Benedito Medeiros, Edison Ishikawa e: *Jour 3.0: A newsroom framework for journalists*. Relatório Técnico, Universidade de Brasília, Departamento de Ciência de computação, 2017. Apresentação. 2

[13] Benedito Medeiros Neto, Edison Ishakawa, George Ghinea: *A framework model for contemporary newsrooms*. Projeto Técnico, Projeto Mídia digital multimodal em redações jornalísticas: um modelo computacional semântico numa estrutura digital

convergente. Estudo dos sistemas de informação no Brasil, Costa Rica, Inglaterra e Estados Unidos. Edital MEC/MCTI/CAPES/CNPq N0 09/2014. CIC/UnB, 2017. 2

[14] OMG: *Business process model and notation (bpmn)*, 2011. Version 2.0 specification. 3, 7, 8

[15] Evellin Cardoso, Katsiaryna Labunets, Fabiano Dalpiaz John Mylopoulos Paolo Giorgini: *Modeling structured and unstructured processes: An empirical evaluation.* 5

[16] Aalst, Wil M. P. van der: *Business process management: A comprehensive survey.* 2013. 5, 6

[17] 6, 7

[18] https://www.w3.org/. acessado em 21/11/2017. 9, 10, 11, 12

[19] Dairon, Jaison: *Introdução a web semântica.* 2012. 9

[20] Breitman, Karin Koogan: *WEB SEMÂNTICA A INTERNET DO FUTURO.* LTC, 2005. 9, 11

[21] VICTORINO, MARCIO DE CARVALHO: *Organização da informação para dar suporte à arquitetura orientada a serviços: Reuso da informação nas organizações.* 11

[22] T., Gruber: *What is an ontology?* www.ksl.stanford.edulkst/what-is-an-ontology.html, 2004. acessado em 22/11/2017. 11

[23] Silva, Michele Andréia Borges, Maria Cristina Pfeiffer Fernandes Viviane Sartori Fernando José Spanhol Andreza Regina Lopes da: *Ontologia como representação do conhecimento: aplicação no curso de formação continuada em tecnologias educacionais na web.* 11

[24] Guarino, Nicola: *Formal ontology and information systems.* Proceedings of FOIS'98, junho 1998. 11

[25] http://protege.stanford.edu/. acesso em 09/08/2017. 12

[26] Chiara Di Francescomarino, Chiara Ghidini, Marco Rospocher Luciano Serafini e Paolo Tonella: *Reasoning on semantically annotated processes.* 13

[27] Di Martino, A. Esposito, S. Nacchia S.A. Masito: *Semantic annotation of bpmn: current approaches and new methodologies.* iiWAS2015, página 95–99, 2015. 13, 14, 15, 16, 50

[28] Beniamino Di Martino, Antonio Esposito, Salvatore Augusto Maisto Stefania Nacchia: *A methodology and implementing tool for semantic business process annotation.* 14

[29] *Super project.* https://www.sti-innsbruck.at/results/movies/sbp-execution-developed-super. acessado em 23/11/2017. 14

[30] Krzysztof Kluza, Krzystof Kaczor, Grzegorz J. Nalepa Mateusz Slazynski: *Opportunities for business process semantization in open-source process execution environments.* 14, 16, 20, 21

[31] Marco ROSPOCHER, Chiara GHIDINI, Luciano SERAFINI: *An ontology for the business process modelling notation.* 2014. 14

[32] Chiara Di Francescomarino, Chiara Ghindini, Marco Rospocher Luciano Serafini Paolo Tonella: *Semantically-aided business process modeling.* 14, 16

[33] *Semantic mediawiki.* https://www.semantic-mediawiki.org/wiki/Semantic_MediaWiki. acessado em 23/11/2017. 14

[34] *bpmn.io. a bpmn 2.0 rendering toolkit and web modeler.* https://bpmn.io/. acessado em 20/10/2017. 14, 20, 23, 24, 51

[35] Weller, Tobias: *A semantic approach for process annotation and similarity analysis.* 14

[36] Niels Lohmann, Eric Verbeek e Remco Dijkman: *Petri net transformations for business processes – a survey.* 14

[37] Remco M. Dijkman a, Marlon Dumas b, c Chun Ouyang c.: *Semantics and analysis of business process models in bpmn.* 15

[38] Remco M. Dijkman, Marlon Dumas, Chun Ouyang: *Semantics and analysis of business process models in bpmn.* 2008. 15

[39] Hoare, C.A.R.: *A model for communicating sequential processes.* July 2007. 15

[40] Peter Y.H. Wong, Jeremy Gibbons: *A process semantics for bpmn.* July 2007. 15

[41] Abdelhamid Malki, Sidi Mohammed Benslimane: *Building semantic mashup.* 15

[42] Weber, Matthias BornChristian BrelageIvan MarkovicDaniel PfeifferIngo: *Autocompletion for executable business process models.* 2009. BPM 2008. Lecture Notes in Business Information Processing, vol 17. Springer, Berlin, Heidelberg. 16

[43] Matthias Born, Florian Dorr, Ingo Weber: *User-friendly semantic annotation in business process modeling.* 16

[44] *Comparing and contrasting open source bpm projects.* https://medium.com/capital-one-developers/comparing-and-contrasting-open-source-bpm-projects-196833f23391, 2016. acessado em 07/11/2017. 21

[45] *5 reasons to switch from activiti to camunda.* http://www.bpm-guide.de/2016/10/19/5-reasons-to-switch-from-activiti-to-camunda/, 2016. acessado em 18/11/2017. 21

[46] *Camunda forks activiti.* https://www.infoq.com/news/2013/03/Camunda-Forks-Activiti, 2013. acessado em 18/11/2017. 21

[47] https://djangopackages.org/grids/g/workflow/. acessado em 20/10/2017. 21

[48] *Gerenciador de pacotes yarn.* https://yarnpkg.com/pt-BR/. acessado em 18/11/2017. 23

[49] https://www.sitepoint.com/yarn-vs-npm/. acessado em 22/08/2017. 24

[50] Zanei Barcellos, Vivien Doherty, Danielle Ferreira Assis, Isabela Alves Graton. Relato de criação e produção de veículo jornalistico digital multiplataforma inovador. IX Encuentro Internacional de Investigadores y Estudiosos de la Información y la Comunicación. 13-17 de novembro de 2017, Palacio de Convencion.La Havan, Cuba. 25

[51] PPGI/UnB: *Json. javascript object notation.* http://json.org/, 2013. Accessado: 04/11/2017. 31

[52] *Introdução ao padrão mvc.* https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308. acessado em 28/11/2017. 32

[53] *Rails guide.* http://guides.rubyonrails.org/. acessado em 28/11/2017. 32, 33

Apêndice A

Implementação

A.1 Configuração de ambiente

A lista de todas as bibliotecas ruby utilizadas se encontram no arquivo *gemfile.rb*. Esse arquivo permite que se especifique quais dependências de gem são necessárias para uma aplicação Rails[1].

A preparação do ambiente de trabalho pode ser dividida pelas três etapas à seguir:

1. Configuração do repositório;
2. Configuração do banco de dados;
3. Instalar e configurar as bibliotecas;

Com a aplicação e o repositório configurados, e as bibliotecas instaladas e configuradas. O ambiente está pronto para seguir com a etapa de desenvolvimento na seção 4.3.2.

1.Configuração do repositório

O primeiro item consta de criar um repositório no GitHub e criar uma aplicação Ruby on Rails localmente e configurar o repositório para a aplicação. A criação de um repositório pode ser adquirido pelo passo-a-passo disponibilizado pelo *github* neste endereço[2]. Em seguida, a aplicação *RoR* é criada localmente com o comando:

```
1 $ rails new semantic_workflow
```

No diretório raiz da aplicação, iniciamos um repositório vazio git e em seguida adicionamos a referencia ao repositório criado com os comandos:

[1] `http://guides.rubyonrails.org/getting_started.html`
[2] `https://help.github.com/articles/create-a-repo/`

```
1 $ git init
2 $ git remote add origin
```

Com as alterações anteriores, temos a aplicação RoR e o repositório online gerados e configurados. Com isto concluimos o item 1 listado e seguimos adiante.

A.1.1 2.Configuração do Banco de Dados

A configuração do banco de dados consta da instalação do MySQL 5.7 localmente e instalação e configuração da biblioteca mysql2 na aplicação. Uma nova aplicação, por padrão, vem com SQLite configurado através da biblioteca sqlite3. Para configurar o uso do mysql2, realizamos duas pequenas mudanças para atender o MySQL. Em gemfile.rb removemos sqlite3 e adicionamos:

```
1 gem 'mysql2'
```

Após substituir e instalar, alteramos no arquivo *config/database.yml* as configurações do sqlite3 para mysql2:

Listing A.1: config/database.yml

```
1  default: &default
2    adapter: mysql2
3    encoding: utf8
4    pool: 5
5    timeout: 5000
6
7
8  development:
9    <<: *default
10   username: root
11   password: 12344321
12   database: workflow_development
13
14 test:
15   <<: *default
16   username: root
17   password: 12344321
18   database: workflow_test
```

Com essas mudanças o ambiente ja está configurado para criar um banco de dados mysql e criamos nosso banco através do comando:

```
1 $ rails db:create
```

e o item 2 está concluído.

3.Instalar e configurar as bibliotecas

A próxima etapa consta de criarmos um painel admin para gerenciar as entidades relacionais à serem geradas. Nesta etapa instalamos e configuramos as bibliotecas devise, rails _admin e cancancan. As três bibliotecas juntas irão adicionar tabela de usuário, autenticação, painel administrativo e autorização de acesso. Com todas estas etapas prontas, concluímos a configuração do ambiente de desenvolvimento. a próxima etapa consta da geração das tabelas no banco de dados e das entidades relacionais. Esta etapa é descrita na subseção 4.3.2 seguinte.

A.2 Modelos Relacionais

Após a modelagem dos dados e criação das tabelas, adicionamos associações de relacionamento oferecidas pelo ActiveRecord, que facilitam e simplificam as buscas no banco de dados.

Ao total teremos 8 modelos relacionais(contando com *ability.rb* gerado pela gem cancancan). Em *app/models/business_process.rb*, adicionamos no modelo relacional a associação:

```
1 has_many :process, through: :lanes.
```

Desta forma simplificamos a busca por todos os processos de um determinado processo. O mesmo é feito em tasks e adicionamos associações para simplificar a busca por tarefas seguintes e tarefas anteriores.

A.3 Integração da Modelagem de processos BPMN

Integração

Seguiremos com a instalação do webpacker no RoR para possibilitar o uso do webpack na aplicação. Como pré-requisitos é exigido Node 6.4.0+ e Yarn instalados localmente. Um passo-a-passo para instalação pode ser encontrada aqui[3]. instalamos o pacote webpacker adicionando:

[3] `https://yarnpkg.com/lang/en/docs/install/#linux-tab`

```
1 gem 'webpacker'
```

ao arquivo *bundle.rb* e em seguida roda-se os comandos:

```
1 bundle install
2 rails webpacker:install
```

Assim são instaladas todas as dependências incluindo o gerenciador de pacotes *yarn* e adicionamos os seguintes pacotes javascript:

bpmn-js

bpmn-modeler

Para não misturar as funcionalidades da biblioteca com as do projeto, criamos a controladora app/controllers/bpmn_controller.rb , a controladora terá os métodos para renderizar o modelador de processos e na etapa seguinte, fazer a leitura do do arquivo XML.

A.4 Uso das Rotas da aplicação

Todas as rotas disponíveis da aplicação são apesentadas na Figura 4.3. No arquivo *readme.md* do projeto é incluso uma documentação completa de como realizar a requisição de cada uma das rotas que pode é disponibilizada no repositório da aplicação[4].

A.5 Criação automática de processos

Nesta seção apresentamos como funciona a implementação da criação automática de processos.

Após salvar um diagrama em XML pelo modelador de processos, podemos fazer uso do xml gerado para criar um processo. Para este feito, foi criado o método *create* em *app/controllers/bpmn_controller.rb*. O método da controladora fica responsável por fazer a leitura do arquivo XML e criar as tabelas do processo no MySQL. O método recebe como parâmetro o identificador de *diagram* e com a utilização da biblioteca **nokogiri**, fazemos a leitura do arquivo XML do BPD. Cria-se um processo e inicialmente é lido o seu nome e em seguida é feito a leitura e captura de seus nodos e setas de fluxos pela função emphidentify_and_create no módulo emphapp/helper/BpmnHelper.rb. A função emphdoc.xpath transforma todas as tags do xml no arquivo em nós, e a partir da raiz faz a leitura de todos os nós filhos para identificar emphlanes, emphsequence_flows e emphtask. Para essa leitura, O método é chamado através da rota:

[4] `https://github.com/marcelobbfonseca/workflow-api`

```
1 GET http://localhost:3000/bpmn/create.json?name="bpmn_file_name"
```

bpmn_file_name corresponde ao nome do arquivo bpmn a ser mandado como parâmetro. Ao final da leitura, todas as tabelas que compõem o processo é criado e a mensagem "Business process was successfully created" é exibida.