5
Babel

Babel representa uma abordagem inovadora para a conversão de dados tradicionais para os formatos da Web Semântica, e apresenta uma série de melhorias em relação às técnicas existentes. Em primeiro lugar, a maioria das ferramentas RDB para RDF são baseadas em alguma estratégia de mapeamento. Mapeamento consiste no processo de redefinição do conteúdo, conceitos e relações em termos de descrições semanticamente mais ricas, por exemplo, vocabulários RDF [46][68]. A fim de fazê-lo, os usuários são obrigados a aprender alguma técnica ou estratégia de mapeamento [69], um processo cognitivamente exigente, que requer muitas vezes familiaridade com alguma linguagem de mapeamento empregada.

Atualmente existem inúmeras estratégias de mapeamento de dados relacionais para RDF [23]. Sahoo [70] fornece uma análise compreensível das abordagens existentes, na qual são analisadas mais de 15 estratégias, classificadas em 3 grandes classes (Projetos e Ferramentas, Projetos Prova de Conceito e Aplicações de domínio específico), das quais Babel pode ser classificada em Projetos e Ferramentas.

Mais importante do que o número de estratégias existentes, é o fato de que elas estão em constante evolução. Durante o período que este estudo estava sendo realizado, a W3C produziu três versões para seus padrões, tanto para o mapeamento direto, que define uma transformação direta em que o vocabulário RDF reflete diretamente os nomes de elementos do domínio [67], quanto para o R2RML, uma linguagem que permite realizar mapeamentos personalizados de bancos de dados relacionais em conjuntos de dados RDF [71-73].

5.1
Contribuições

Babel facilita o processo de aprendizagem, uma vez que tem um enfoque baseado em templates que guiam os usuários durante o processo de construção

dos mapeamentos [74]. Templates permitem capturar e encapsular o conhecimento do domínio e orientar os usuários na realização das tarefas necessárias [75].

Além disso, Babel é mais flexível, pois abrange outros formatos da Web Semântica que não RDF/XML, incluindo OWL/XML, RDFa e N3. A possibilidade de serialização de informações em mais de um formato da Web Semântica é muito importante no cenário atual, por possibilitar a utilização da mesma informação por diversas aplicações e a criação de Mashups [76][53].

Babel foi projetado para facilitar e promover a conversão de dados de formatos tradicionais, em particular aqueles armazenados em bancos de dados relacionais e planilhas, para formatos compatíveis com a Web Semântica, mas permite extensões cutomizadas, o que possibilita aos usuários escrever extensões para habilitar o uso de outras fontes de dados, através de Plugins. Um modelo de desenvolvimento preferido pela comunidade de engenharia de software, por encapsular o conhecimento sobre como resolver problemas, operar em diversos ambientes de desenvolvimento, e promover a integração com os sistemas existentes, programas e aplicações [77]. Arquiteturas de Plugins são extensíveis, permitindo aos usuários adicionar novas funcionalidades, personalizar e adaptar a ferramenta às suas necessidades específicas. Elas são tipicamente menos complexas do que as ferramentas monolíticas, modular, e mais portável [78].

A principal contribuição desta abordagem em relação a ferramentas de conversão de dados existentes [17], no entanto, é o apoio que fornece aos usuários. Babel suaviza a curva de aprendizagem, eliminando a necessidade de aprender uma nova estratégia de mapeamento, que é substituída pelo uso de templates. Os templates são utilizados para guiar o usuário na definição dos mapeamentos, na escolha dos vocabulários adequados, e na definição do formato de saída dos dados. Na Tabela 2, abaixo, temos uma classificação da ferramenta seguindo a tabela de classificação de Sahoo [70].

Criação de Mapeamento	Implementação e acessibilidade de Mapeamento		Implementação de mapeamento	Implementação de consulta	Domínio da aplicação	Integração da informação	
	Linguagem de representação	Mapeamento de Acesso	Estático (ETL) ou Dinâmico			Sim/Não	Número de

							datasets
Manual /Automática	TML + Arquivo de configuração Babel	Arquivo de Configuração Babel + Babel GUI	Ambos	Linked Data através do Babel Servlet	Genérico	Sim	Multiplos

Tabela 2. Classificação do Babel segundo a tabela de classificação de Sahoo[70].

5.2 Conceitos

Babel foi concebido sob dois conceitos principais o DataStore e a Collection. Um DataStore representa uma fonte de informação. Como o processo de conversão pode utilizar várias fontes de informação (banco de dados, arquivos proprietários e planilhas) a arquitetura proposta permite com que DataStores possam ser facilmente adicionados. Cada DataStore poderá possuir uma ou mais visões para a informação que armazena, esta abstração é chamada de Collection.

Uma Collection, representa um conjunto de informações, com as mesmas propriedades, que pode ser obtida através da aplicação de um filtro sob um DataStore. A forma de se definir o filtros para diferentes Collections é ditada pela implementação do DataStore. A Collection, ilustrada na Figura 16, é importante para restringir o acesso a informações sensíveis e.g. senhas e dados pessoais.

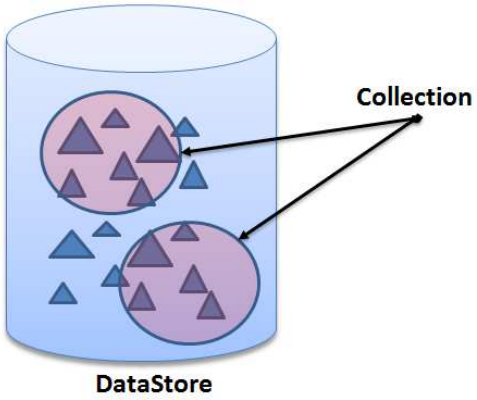

Figura 16. Ilustração de um DataStore contendo duas coleções de dados.

Babel já oferece suporte à leitura das principais fontes de dados, como planilhas, banco de dados relacionais e estruturas em memória que serão apresentados na seção 8.

5.3 Arquitetura

Nesta seção apresentaremos a arquitetura do Babel. Listamos, a seguir, os componentes, que serão detalhados nas seções subseqüentes. A Figura 17 ilustra os principais componentes, sua organização, e como estão relacionados.

- DataStore: abstração de fonte de dados;
- Collection: abstração de conjunto de dados;
- TML: define o conjunto de regras para a criação dos Templates;
- Template: permite criar, salvar e compartilhar mapeamentos;
- Máquina de processamento de Templates (Template Engine): responsável pelo processamento dos dados proveniêntes das Collections e inseri-las nos templates.
- Arquivo de Configuração (Config File): permite criar, salvar e compartilhar configurações de execução do Framework;
- Servlet: permite publicar dados na Web através de uma interface HTTP.
- Interface Gráfica de Usuário (GUI): permite definir coleções, criar mapeamentos automatizados, salvar conFigurações, e converter informações de uma forma gráfica.
- Inteface de linha de comando: permite a execução da Template Engine através de comandos digitados.

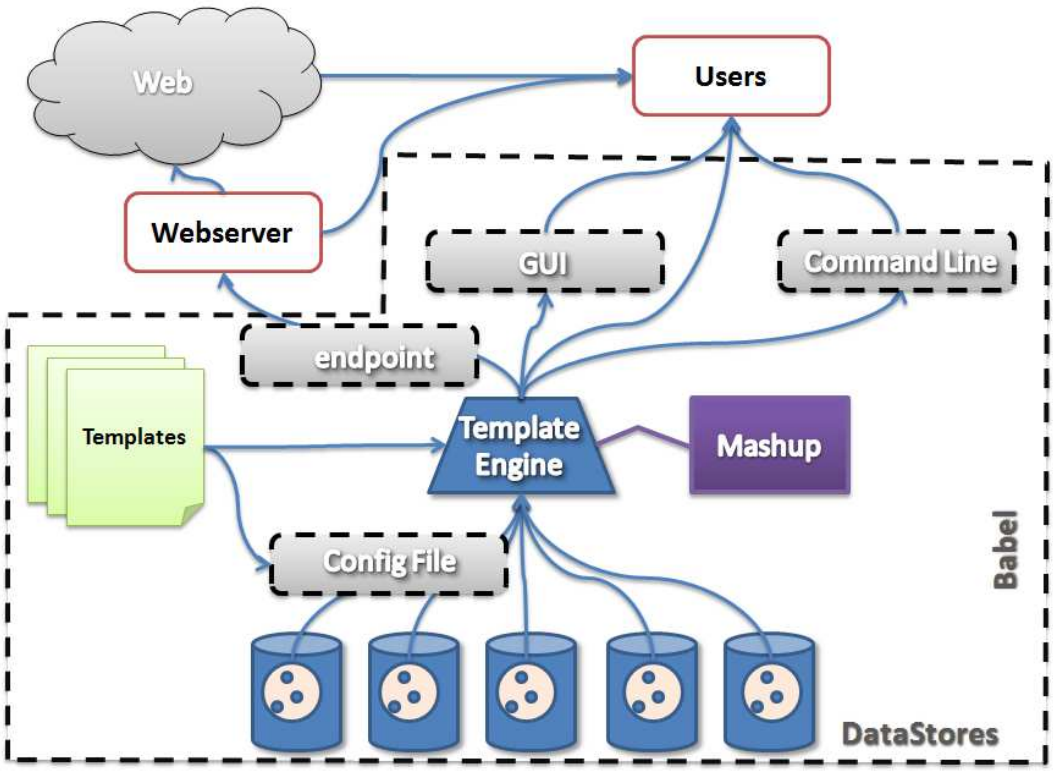

Figura 17. Diagrama que ilustra os componentes principais da arquitetura do Babel e suas relações.

5.3.1
O template

O template baseia-se na linguagem TML, apresentada no capítulo 5, e, portanto, é o mecanismo pelo qual a informação é mapeada. A utilização do template é importante porque fornece tanto a estrutura quanto o vocabulário que será utilizado. TML é flexível, pois não define uma estrutura fixa para a informação, possibilitando a portabilidade da informação original para qualquer tipo de estrutura. Entretanto, Babel suporta apenas a serialização de estruturas de dados no formato XML, triplas e quádruplas (N4) [35].

A criação e o mapeamento do template utilizando TML são mais fáceis porque pessoas que são familiarizadas com estruturas da Web Semântica, por exemplo, RDF, se tornam aptas a editar e fazer o mapeamento manualmente, utilizar qualquer ferramenta da Web Semântica, que possibilite criar a estrutura da informação em um dos formatos possíveis ou utilizar estruturas e ontologias existentes. Como um dos objetivos da Web Semântica é a publicação e interligação de informações, é cada vez mais fácil encontrar estruturas que

poderão ser utilizadas como templates. Elas podem ser encontradas utilizando ferramentas de busca de ontologias [10], ou acessadas através de *endpoints* SPARQL.

Após a definição da estrutura, o próximo passo passa a ser o mapeamento. Para fazê-lo o usuário deverá obedecer ao conjunto de regras estabelecidas pela TML, isto é, a informação deverá ser endereçada na estrutura do template através de marcações, onde o nome da coleção da estrutura de origem deverá ser precedido de tralha (“#”), seguido de um ponto (“.”) e o nome do atributo da coleção, eg: “#” + coleção + “.” + atributo, como demonstrado na seção 6.

No Babel, o endereçamento deverá ser feito utilizando o *id* da Collection. Este mecanismo permite maior flexibilidade uma vez que os detalhes referentes tanto a conexão quanto ao meio de obtenção da informação ficam reservados à aplicação.

5.3.2
O Arquivo de Configuração

O arquivo de configuração é uma interface que permite definir, salvar ou alterar parâmetros de configurações utilizados pela ferramenta, sem que seja necessário fazer alterações no código. O template, o conjunto de DataStores e as Collections são exemplos de configurações que podem ser inseridas no Arquivo de Configuração.

O formato do arquivo de configuração é XML. A criação do arquivo é regulamentada por um conjunto de regras definidas pelo esquema (*XML Schema* Definition) do Babel na Figura 18, que serão detalhadas a seguir.

O arquivo deve ser iniciado e ser terminado com a marcação *config*. A marcação *config* pode possuir de uma a várias tags do tipo DataStore, e uma tag do tipo template, onde serão colocados os templates que serão utilizados no processo de conversão. Atualmente a ferramenta suporta três tipos de formatos de templates: N3, N4 e XML.

A tag DataStore possui um Descriptor e um conjunto de Collections. Na tag Descriptor serão colocados os parâmetros de conexão correspondentes à implementação do DataStore que será utilizado.

O Descriptor possui tags do tipo Param, que servem para adicionar parâmetros específicos de cada implementação. A tag Param possui dois atributos: *name* e *value*. O atributo *name* corresponde ao nome do parâmetro e *value* ao valor. Essa tag é especialmente importante para garantir a versatilidade do modelo, pois possibilita que parâmetros possam ser adicionados, removidos ou alterados de acordo com a necessidade de cada implementação utilizada. Uma tag Param não contém restrições de tipo ou quantidade de tags que pode encapsular.

A Collection possui um identificador definido pelo atributo *id*. Esse identificador será útil durante o processo de conversão da informação, pois é utilizado no mapeamento das informações no template. A Collection também possui um atributo que permite localizar o identificador único da Collection, e um conjunto te tags do tipo Param. O indentificador único possibilita correlacionar informações de diferentes Collections e as tags do tipo Param e, dentre outras coisas, a restringir o conjunto de informações que farão parte da coleção.

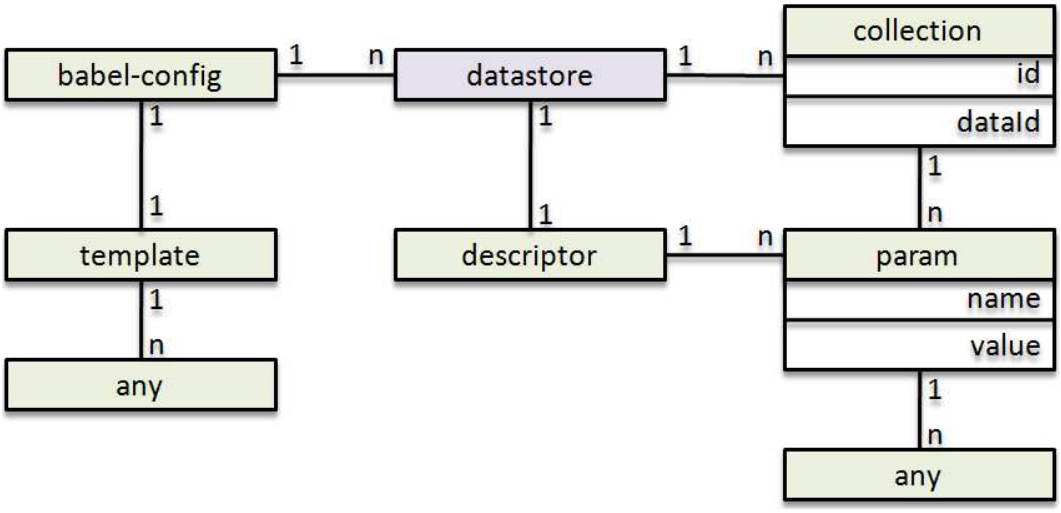

Figura 18. Diagrama de modelo do esquema do arquivo de configuração.

Tanto o elemento *template* quanto o elemento *param* podem encapsular elementos de qualquer tipo, e em qualquer quantidade. Essa característica garante a versatilidade do modelo uma vez que, caso haja necessidade, elementos de diferentes naturezas podem ser adicionados ao modelo.

5.3.2.1 Exemplo

Imaginemos agora que desejamos publicar informações provenientes de uma base de dados utilizando o vocabulário FOAF. Esta base de dados possui uma tabela cujo nome é *person*, contendo informações sobre pessoas como: nome, endereço, idade, email, CPF e RG.

Como podemos perceber esta tabela possui um conjunto de atributos sensíveis que, dependendo do objetivo e da plataforma de veiculação dos dados, não deverão ser publicados e um conjunto de atributos passível de publicação (nome, endereço, email, idade). Chamamos de dados sensíveis, aqueles que possibilitam a identificação do indivíduo ou que contém informações pessoais como senhas. Portanto iremos publicar apenas os dados pertinentes através da criação da Collection cujo nome é o mesmo da tabela, isto é, *person.*

Para publicar o conteúdo da tabela person é preciso, antes de tudo, definir o DataStore. Para definir o datastore é preciso criar um Descriptor que contenha os parâmetros de conexão. No nosso exemplo iremos nos conectar a um banco de dados hipotético cuja conexão é realizada por um DataStore que deverá receber três parâmetros: URL_CONNECTION, que contém uma URL de conexão baseada no padrão JDBC (“jdbc:mysql://host:port/dbName/”); o User, que contém o nome do usuário que se conectará no banco de dados; e o PASSWORD, onde é inserida a senha do usuário. Ambos vazios no exemplo.

Após configurar os parâmetros do DataStore, é necessário configurar a Collection. Cada Collection contém um conjunto de parâmetros que serão interpretados por seu DataStore. No nosso exemplo o parâmetro que restringe o conjunto de dados da Collection é *query* e ele possui como valor uma cláusula *select* do banco de dados (“Select name from person”).

```
<config>
  <datastore>
   <descriptor>
      <param name="URL_CONNECTION"
        value="jdbc:mysql://host:port/dbName/”>
     <param name="USER" value=""/>
     <param name="PASSWORD" value=""/>
   </descriptor>
   <collection id=”person”>
              <param name=”query” value=”Select name from person”>
   </collection>
  </datastore>
   <tenplate>
   <foaf:Person>
            <foaf:name>#{person.name}</foaf:name>
   </foaf:Person>
  </tenplate>
</config>
```

Listagem 11. Exemplo hipotético de um arquivo de configuração do Babel.

Após definir o conjunto de DataStores e Collections, será preciso criar o template seguindo a orientação da linguagem TML no Capítulo 4 que no nosso exemplo, na Listagem 11, é uma estrutura RDF/XML do tipo FOAF.

5.3.3
Máquina de processamento de Templates (Template Engine)

A máquina de processamento de templates é responsável por processar o template, e mapear a informação necessária extraídas das Collections. Cada template possui uma estrutura particular, e por isso é necessário uma máquina específica que permita seu processamento.

A máquina de processamento de templates do framework proposto é composta por duas máquinas de processamento de estruturas, que possibilitam o processamento de templates em dois formatos: XML e Triplas (N3 e N4). Estas máquinas de processamento permitem a geração de conteúdo nos seguintes formatos da Web Semântica: RDF, RDFa, OWL, triplas, bem como em outros formatos RSS e HTML.

A validação da estrutura do template fica a cargo do usuário, o que significa que, caso seja inserida um template com uma estrutura inválida, teremos um dos seguintes resultados:

- O template não processado;
- Uma exceção causada pela má-formação da estrutura do template;
- Geração de conteúdo em um formato inválido.

5.3.4
Interface por Comando Texto (Textual Command User Interface)

A interface por comando foi criada para usuários mais avançados, e permite o acesso os serviços básicos oferecidos pela Máquina de processamento, através

do arquivo de configuração, que pode ser criado manualmente ou através da interface gráfica.

Esta interface permite com que as informações provenientes dos DataStores declarados no arquivo de configuração possam ser convertidas para a estrutura declarada como template.

Para utilizar a interface, o usuário deve utilizar o comando texto *java* seguido de *org.babel.Babel,* como ilustrado pelo fragmento de código a seguir:

```
"java org.babel.Babel"
```

O comando *java* é utilizado para acionar o interpretador e executar o programa escrito e compilado em Java. Em seguida o usuário poderá passar os seguintes parâmetros:

- *"-c configpath"*: o parâmetro c serve para indicar o caminho onde está o arquivo de configuração, a exemplo:
    - "-c myConfigFile.xml" – indica que o arquivo myConfigFile.xml que se encontra no diretório corrente deverá ser utilizado.
- *"-t templatepath"*: o parâmetro *t* serve para indicar o caminho onde está o template, se o arquivo não for especificado o programa utilizará o primeiro template encontrado no diretório corrente, se ainda assim o arquivo não existir a lista de comandos será exibida, a exemplo:
    - "-t c:\mytemplate.xml" – indica que o arquivo mytemplate.xml que se encontra na raiz do diretório 'c' deverá ser usando como template.
- *"-out outputfile"*: o parâmetro out serve para indicar onde o resultado será escrito, se o arquivo de saída não for especificado o programa escreverá o resultado no arquivo result.xml, a exemplo:
    - "-out c:\myrdf.rdf" – indica que o resultado da conversão deverá ser escrito no arquivo *myrdf.rdf* na raiz do diretório '*c'*.
- *-h*: o comando *h* solicita a exibição da lista de comandos.

5.3.5
Interface Web

A interface Web provê um mecanismo para publicar a informação na Web. Deve ser instalada em um Servidor de Aplicações Java (Tomcat[38], JBOSS[39]). As informações publicadas através da interface Web estarão acessíveis através de requisições HTTP, onde será possível, especificar o formato de serialização da informação. Assim como na Interface por Comando Texto, é necessária a especificação dos arquivos de configuração e do template, sem os quais não será possível realizar a serialização da informação no formato desejado.

A interface padrão não provê um mecanismo de filtrar a informação desejada nem implementa o protocolo SPARQL. Para cada conjunto de meta-informação que o usuário desejar publicar, será necessário a instalação de um novo *endpoint* no servidor.

Cada *endpoint* permite a publicação da informação em três formatos: triplas, RDF/XML e HTML, que deverão ser fornecidos como parâmetros na requisição HTTP. Caso o parâmetro seja omitido a informação será serializada em HTML, que é o formato padrão para a serialização de dados na Web.

É possível declarar os DataStores e as Collections programaticamente. A interface atual também pode ser modificada conforme os requisitos do usuário. Nada impede, por exemplo, a adição de novos parâmetros, que possibilitem a definição do conjunto de meta-informações a ser visualizado, filtros.

A interface Web é baseada em Servlets. Um Servlet é um mecanismo que permite chamar dinamicamente uma classe em Java, que é acessada através de um modelo de requisições e resposta. Embora em teoria os servlets possam responder a qualquer tipo de requisição, eles são comumente utilizados em aplicações hospedados em servidores Web para gerar conteúdo HTML e XML. A Figura 19, a seguir ilustra o modelo de requisição e resposta HTTP utilizado pelo endpoint do framework.

[38] http://tomcat.apache.org

[39] http://www.jboss.org

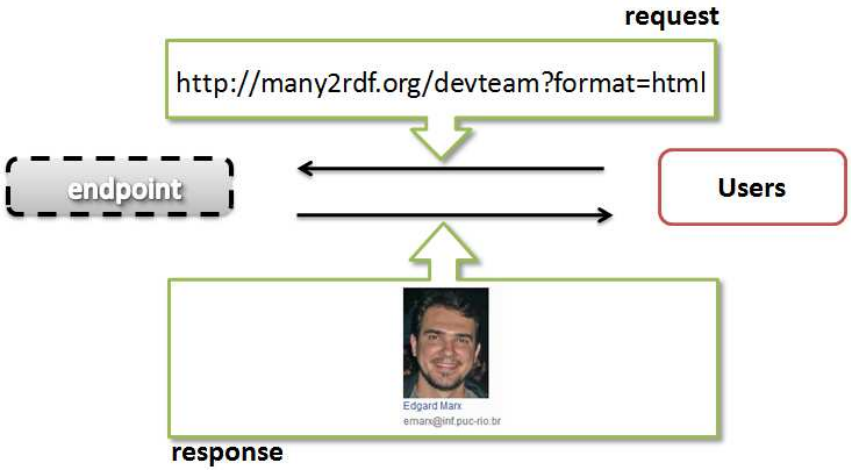

Figura 19. Ilustração do modelo requisição e resposta HTTP utilizado pelo *endpoint* do framework Babel.

5.3.6
Inteface gráfica

A fim de promover a adoção dos Padrões Linked Data e o uso da ferramenta por não-especialistas, desenvolvemos uma interface gráfica que permite definir DataStores, novos tempaltes e criar Collections. Após a definição destes parâmetros, o usuário poderá serializar essa configuração em um arquivo de configuração do Babel, converter a informação em RDF (ou qualquer outro formato de serialização permitido), bem como publicá-la.

A interface gráfica do Babel, Figuras 20 e 21, também permite a geração automática de templates seguindo heurísticas de mapeamento direto, e a sincronização de um repositório RDF local ou remoto, e.g., repositório RDF Virtuoso, com a base de dados de origem - útil para aqueles que possuem repositórios de dados independentes e desejam mantê-los atualizados.

Apesar de possibilitar a conversão dos DataStores suportados pela Máquina de processamento de templates, a interface gráfica atual só permite manipular e selecionar graficamente coleções do tipo de banco de dados relacionais, que é particularmente interessante por permitir que usuários não familiarizados com SQL e banco de dados possam selecionar a informação ou criar o mapeamento.

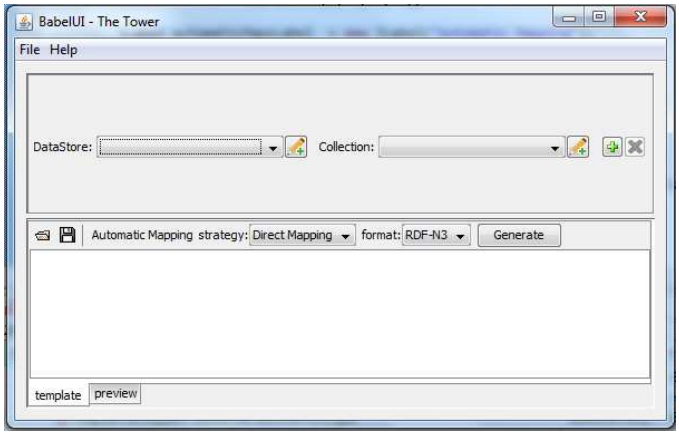

Figura 20. Interface gráfica do framework proposto.

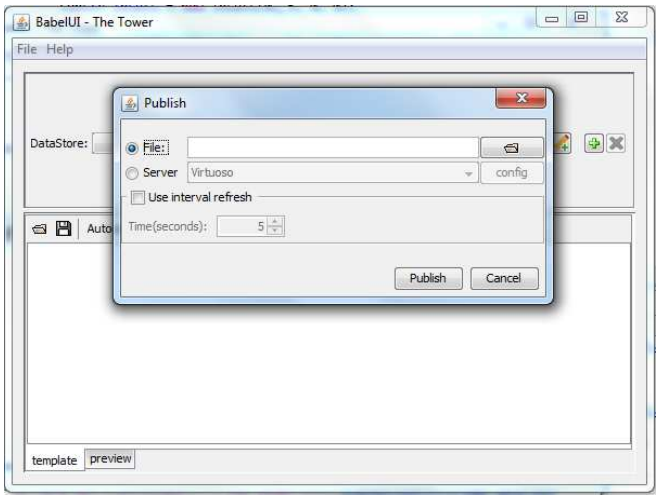

Figura 21. Publicando dados através da interface gráfica.

5.4 Implementação

Nesta seção, serão apresentadas as implementações da Máquina de Porcessamento de Tempaltes e dos DataStores que possibilitaram o desenvolvimento dos vários componentes abordados na seção anterior. Essas implementações podem ser acessadas e obtidas gratuitamente através do endereço do framework, www.babeltool.org.

O framework foi desenvolvido levando em consideração a natureza das necessidades dos usuários. Parte da solução requeria um componente capaz de extrair a informação de uma fonte de dados qualquer e realizar a conversão, com base em um template pré-estabelecido. Os requisitos para este componente estão

bem entendidos e bastante estáveis, a parte restante, um componente flexível para lidar com a implementação de uma variedade de fonte de informações que são muito voláteis, como argumentado nas seções anteriores.

A solução encontrada foi separar a implementação em dois módulos distintos, Figura 22. O primeiro módulo converte informações para o formato base do template e é responsável por: (1) realizar a leitura do template contendo o mapeamento, (2) extrair as informações das Collections nos respectivos DataStores, e (3) inserir as informações extraídas das Collections no template e serializá-la. O segundo módulo trata-se de uma ponte, que implementa o suporte a uma grande variedade de fonte de dados e provê um mecanismo que permite a criação de coleções (seleção dos dados que serão inseridos no template) encapsuladas nas Collections. Esta arquitetura separa as duas principais preocupações da implementação proposta, mas ainda mais importante, promove o isolamento do componente responsável pelo acesso à informação que tem uma maior probabilidade de sofrer alteração. A intenção é oferecer uma maneira de atualizar ou possibilitar a adição de outras fontes de dados sem que haja necessidade de alteração no código principal.

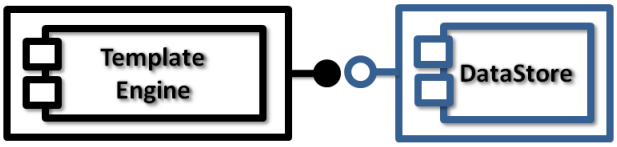

Figura 22. Módulos principais do framework.

Como as fontes de dados estão em constante evolução, qualquer aplicação que implemente o acesso deve ser flexível a ponto de permitir tais mudanças. Dessa forma, as implementações de acesso foram isoladas em um componente separado, acessível através de uma interface, o que permitiu separar as partes da aplicação que são mais estáveis, daquelas que tem mais chances de sofrer modificações. Ao fazermos isso, nós promovemos a reutilização e a evolução do framework.

A manutenção e a adição de novas fontes de acesso é possível através da interface DataStoreFactory que é construída em cima do Service Provider Interface (SPI). A SPI é uma API bem conhecida, amplamente adotada no

desenvolvimento de componentes reutilizáveis em diversas tecnologias e.g.: Java Database Connectivity, Java Cryptography Extension, Java Naming and Directory Interface, Java Application Program Interface for XML Processing, e GIS (Geotools), entre outros. A descrição e especificações para SPI são encontrados em [79][40], respectivamente. O uso de SPI torna a implementação do DataStore resceptivo à mudanças, pois permite a atualização e adição de novos algoritmos através da implementação de uma interface simples na qual os usuários podem substituir (ou modificar) o algoritmo de acesso, sem causar alterações na Máquina de processamento. O Diagramas 1, 2, 3, 4 e 5 mostram a implementação proposta, com destaque para a interface DataStoreFactory.

A interface DataStoreFactory é responsável por selecionar a implementação adequada do DataStore com base nos parâmetros fornecidos pelo usuário, através de dois métodos:

- *canProcess* – Método que recebe como parâmetro as informações configuradas na tag do *DataStore* do arquivo de configuração e é responsável por determinar se o DataStore poderá processar o conjunto de parâmetros fornecidos, retornando *sim*, caso seja, e *não*, caso contrário.
- *createDataStore* – Esse método retorna a implementação adequada do DataStore, com base nos parâmetros fornecidos, que gerarão exceção caso sejam inválidos.

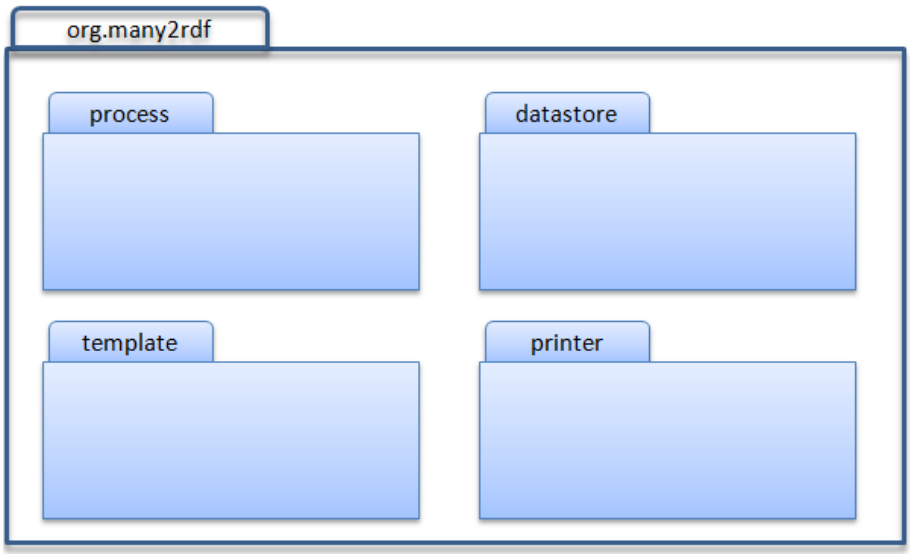

Diagrama 1. Diagrama de pacotes.

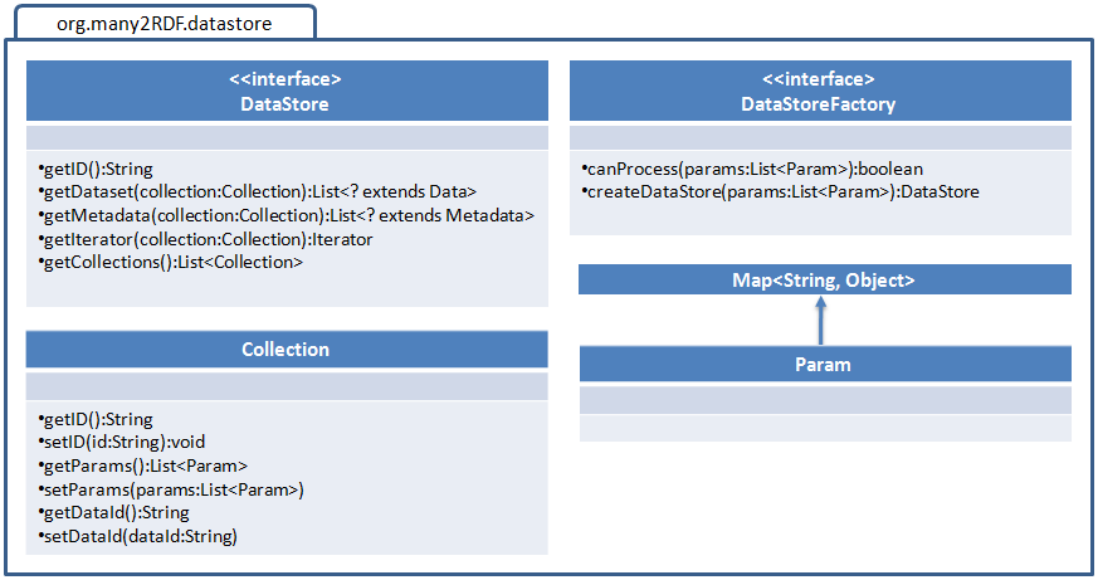

Diagrama 2. Diagrama de classes do pacote *datastore*, do Diagrama 1.

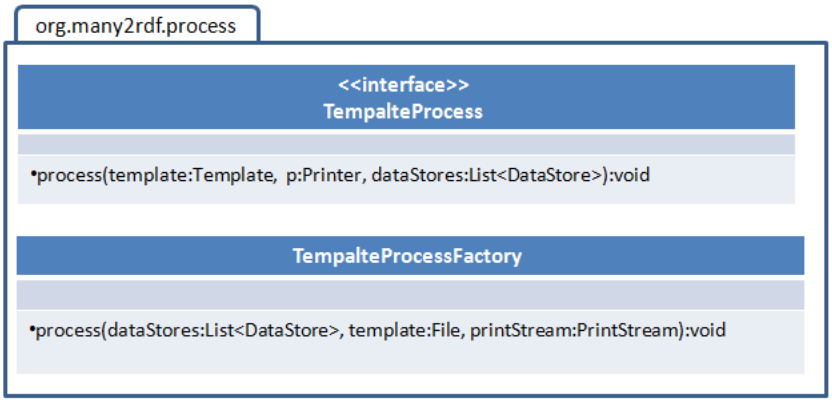

Diagrama 3. Diagrama de classes do pacote *process*, do Diagrama 1.

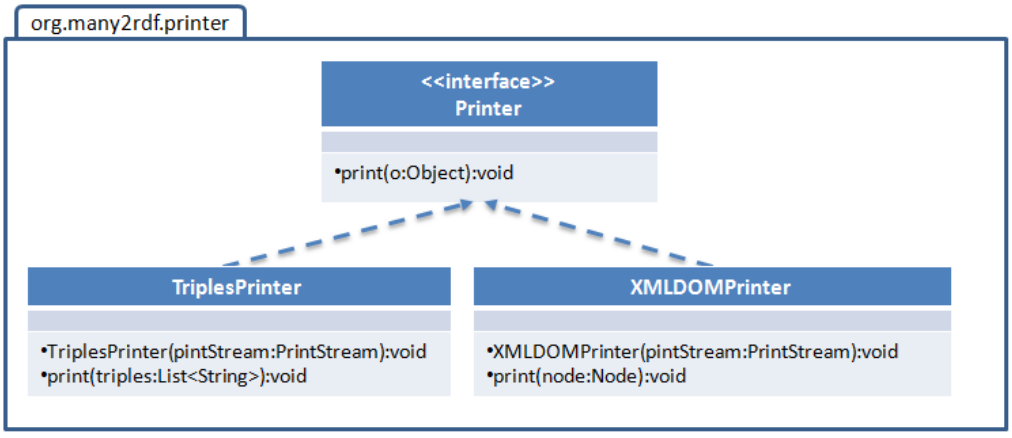

Diagrama 4. Diagrama de classes do pacote *printer*, do Diagrama 1.

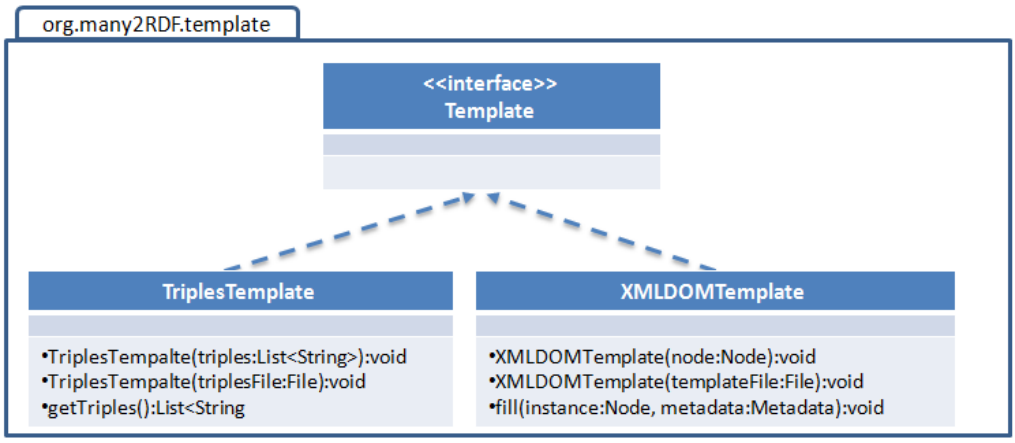

Diagrama 5. Diagrama de classes do pacote *tempalte,* do Diagrama 1.

5.4.1
Máquina de Processamento de Templates (Template Engine)

A máquina de processamento de templates é responsável por inserir os dados extraídos das coleções no template e atualmente ela permite processar tanto templates em XML quanto em triplas. O método de processamento do template pode ser sintetizado em dois passos:

- Coleta dos parâmetros: Nesse passo são colhidas as informações como o DataStores, Collections e o Template, que poderão ser fornecidos através do arquivo de configuração ou declarados programaticamente;
- Processamento: No processamento, os parâmetros obtidos durante a fase de coleta são verificados. Exceções podem ser lançadas caso seja encontrado algum erro nos parâmetros fornecidos que vão desde a estrutura do documento (template, arquivo de configuração) até existência de algum parâmetro inválido. Caso não seja detectado nenhum erro, inicia-se o processamento dos parâmetros através de uma chamada à API, onde é selecionada a Máquina de Processamento adequada com base no formato do template fornecido.

Cada Máquina de Processamento recebe três parâmetros: o template, uma impressora (Printer) e um conjunto de DataStores. Durante o processamento, os dados são extraídos dos DataStores com base na restrição das Collections,

indivíduo a indivíduo, são incorporados ao template, e ao final do processo, o resultado é impresso pela impressora fornecida. Não há uma ordem fixa para a execução das coleções ou impressão do resultado. Cada Máquina de Processamento é livre para implementar o algoritmo de processamento da forma que melhor lhe convir, dessa forma, a ferramenta pode ser facilmente integrada com ferramentas ou sistemas existentes, possibilitando a geração dinâmica de conteúdo para sites ou web services. A execução do processamento do template pode ser sintetizada pelo Diagrama 6.

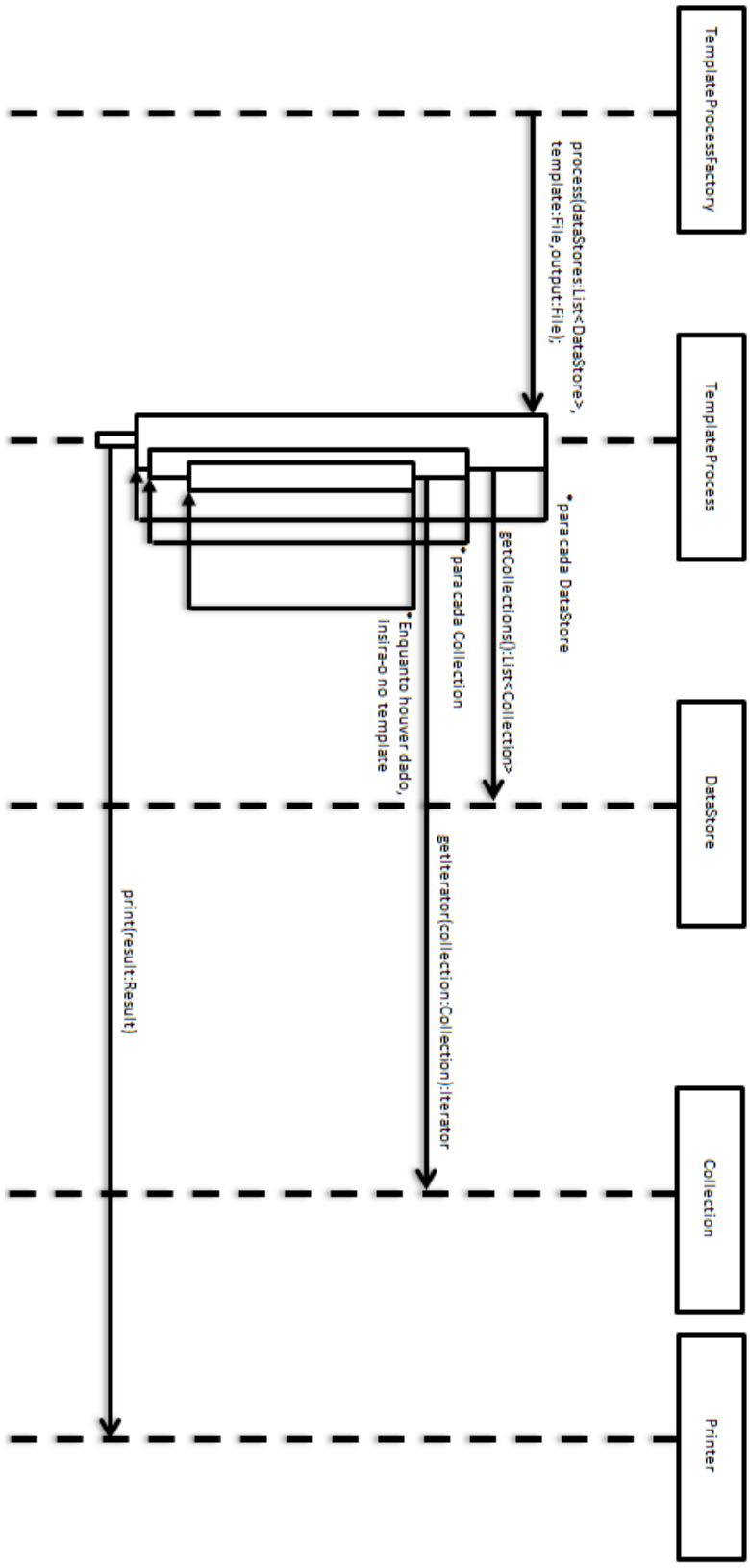

Diagrama 6. Exemplo de um diagrama de execução hipotético de um template pela Máquina de Processamento de Tempaltes.

A máquina de processamento de templates XML funciona através da API DOM, e por isso, possui restrições quanto ao volume de informações que pode manipular. Além disso, o resultado só é impresso após o processamento de todas as Collections. Já a máquina de processamento triplas, não encontra a mesma limitação, uma vez que cada template de tripla é processado e impresso separadamente do conjunto.

5.4.1.1 Desambiguação

Para remover à ambiguidade já discutida na seção 4.1 a máquina de processamento utiliza o parâmetro collectionID da coleção no arquivo de configuração. Assim, se o parâmetro não for especificado, significa que a coleção será inteiramente mapeada para a estrutura de destino não observando o identificador, o primeiro exemplo da seção 4.1.

De outra forma, havendo duas coleções distintas sendo elas relacionadas por um identificador único, ele poderá ser especificado. Assim, a máquina de processamento saberá quais elementos serão pertencentes a uma determinada sub-estrutura, e apenas estes elementos serão mapeados.

Supondo que um determinado elemento tenha uma propriedade P’ pertencente a uma coleção C’ e outra P’’ pertencente à coleção C’’, com o uso do identificador, apenas as propriedades P’ e P’’ que tiverem a mesma chave serão mapeadas para o elemento, como o resultado da direita da Figura 11.

Algumas fontes de dados podem conter chaves primárias compostas, relacionando um determinado objeto de um conjunto com outros objetos de outros conjuntos (ex.: cidade, estado, país). Como visto, nossa solução apenas possibilita realizar relacionamento de uma coleção através de um único identificador.

5.4.1.2 Máquina de processamento de template em XML

XML está presente em vários padrões da Web e pode ser interpretado por máquinas. É uma linguagem muito flexível e é composto basicamente por três marcações: elementos complexos; elementos simples; e atributos. Para criar um

template em XML o usuário deverá mapear a coleção desejada em uma das duas últimas marcações.

Antes de inserir os dados das coleções no template, a máquina realiza um pré-processamento extraíndo informações como:

- Os elementos que serão afetados por cada coleção: no tempalte pordem existir várias marcações, desta forma apenas as marcações que tiverem mapeamento serão processadas.
- Os nós mais razos afetados pelas coleções: Esta informação será útil para realizar a replicação do nó mais razo quando houver um idêntificador (CollectionID) especificado na coleção. Neste caso, cada mudança no valor do idêntificador irá refletir em uma replicação do nó mais razo.
- A ordem de execução das Coleções: cada nó do template que possui um mapeamento é executado pela máquina separadamente, do nó mais profundo ao mais razo. Em outras palavras, os nós mais profundos serão os primeiros a serem processados.

Após a coleta destas informações, cada coleção é inserida na ordem estabelecida no pré-processamento. Um a um cada elemento da coleção é mapeado durante um processo conhecido como XMLizer. Neste processo, cada novo elemento da coleção irá refletir em um novo elemento mapeado do template (simples ou composto).

5.4.1.3 Máquina de processamento de template em Triplas ou Quadruplas

No modelo RDF cada tripla ou quádrupla constitui uma afirmação (*satement*). Na máquina de processamento de template cada afirmação pode conter diferentes combinações de coleções e é processada separadamente, uma a uma.

As informações provenientes das coleções são inseridas na ordem: objeto, predicado e posteriormente o sujeito. Quando a máquina não encontrar nenhum mapeamento em um dos 3 níveis, ela irá automaticamente passar para o nível seguinte.

Ao processar uma coleções mapeada que contendo identificadores (CollectionID), apenas os dados que possuírem o mesmo identificador serão

inseridos na afirmação, caso não haja esse mapeamento, todas as informações das coleções serão inseridas, resultando portanto, em um produto cartesiano de ambas as coleções contidas no template da afirmação.

5.4.2 DataStore

A interface DataStore é o ponto de acesso da Máquina de Processamento com as diversas fontes de dados. Com ela é possível verificar se o DataStore pode processar determinada coleção, bem como, adquirir sua lista de indivíduos. Essas funcionalidades são possíveis através da implementação de cinco métodos:

- *getID* – retorna o rótulo da fonte de dados;
- *canProcess* – recebe como parâmetro uma coleção e retorna verdadeiro caso possa processá-la e falso, caso contrário;
- *getCollections* – retorna o conjunto de coleções do DataStore;
- *getMetadata* – recebe como parâmetro uma coleção e retorna seu conjunto de meta-informações, tais como: atributos, chaves primárias e estrangeiras, etc.;
- *getDataset* – recebe como parâmetro uma Collection e retorna seu conjunto de indivíduos;
- *getIterator* – recebe como parâmetro uma Collection e retorna seu interador.

Como é observado através do Diagrama 6, e da interface do DataStore, os dados podem ser carregados sob demanda por meio do interador (Iterator), o que possibilita manipular grandes volumes de informações.

5.4.2.1 JDBCDataStore, DataStore de Banco de dados relacionais

O DataStore para Banco de dados permite a extração de dados de uma variedade de implementações de Banco de Dados através do driver JDBC (SQLServer, Postgres, MySQL e Oracle). É importante notar que para utilizar determinado *driver* é preciso incluí-lo no *classpath* do framework e, além disso, fornecer os parâmetros necessários para a conexão, como: nome do driver,

endereço do servidor (host), porta de conexão, senha, usuário e nome do banco, através da interface desejada.

Os filtros das Collections de um DataStore de Banco de Dados são determinados pelo parâmetro *query*, como na Listagem 12. O valor do parâmetro *query* será uma consulta compatível com a versão do driver utilizado pelo DataStore. Dessa forma, o usuário poderá utilizar todas as facilidades que uma consulta possui para selecionar ou até mesmo criar as informações desejadas.

A fim de validar a implementação, foi realizada uma comparação com algumas ferramentas do mercado. Os testes foram realizados sob um banco MySQL em um computador Intel Q6600, contendo 3,23 GB, e consistiu na conversão dos dados de uma tabela em indivíduos *foaf:Person*. Foram gerados 65 mil RDFs foaf:Person contendo apenas nome e email, ilustrados no Gráfico 4. Todas as ferramentas analisadas foram executadas em linha de comando, com exceção do Triplify que foi executado em um servidor PHP. Dentre as ferramentas analisadas, Babel apresentou um desempenho um pouco superior ao segundo colocado, Triplify, evidência que pode ser justificada pela diferença do ambiente de execução e pelo fato de Java ser uma linguagem compilada, ao passo que Triplify é um *script* que é interpretado em tempo de execução. Com esse resultado, Babel demonstrou ser uma ferramenta adequada para a geração de conteúdo RDF para dados provenientes de banco de dados relacionais.

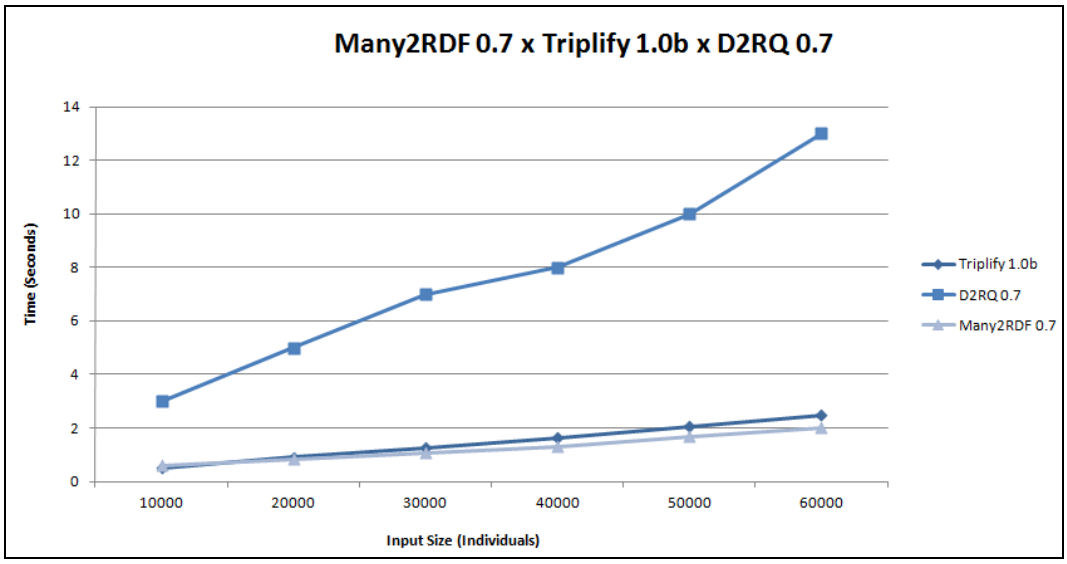

Gráfico 4. Comparação de desempenho entre Babel, Triplify e D2R.

```
<config>
   <dataStore>
        <descriptor>
             <param name="DRIVER_CLASS"
value="sun.jdbc.odbc.JdbcOdbcDriver"/>
             <param name="URL_CONNECTION" value="jdbc:odbc:exemplo"/>
             <param name="PASSWORD" value=""/>
             <param name="USER_NAME" value=""/>
       </descriptor>
       <collection id="table">
          <param name="SQL" value="SELECT * from table "/>
       </collection>
   </dataStore>
</config>
```

Listagem 12. Exemplo de arquivo de configuração declarando um DataStore de banco de dados relacional.

5.4.2.2 POIDataStore, DataStore de planilhas

O POIDataStore permite a extração de dados de planilhas através da API para documentos POI da Apache[40]. Embora a versão atual dê suporte apenas a planilhas, ela também permite extrair conteúdo de outros documentos (Word, Outlook, Powerpoint, Visio, dentre outros).

Para utilizar um DataStore de planilhas em um arquivo de configuração, basta incluir o parâmetro URI, contendo o caminho absoluto para o arquivo da planilha no descritor do DataStore.

Os filtros das Collections de um POIDataStore são determinados pelos parâmetros *ROWSTART*, *ROWEND*, *COLUMNNSTART*, *COLUMNEND*, e *SHEET*, como ilustrado na Figura 23 e Listagem 13. Os parâmetros ROWSTART e ROWEND, definem a linha de início e final respectivamente. Os parâmetros COLUMNSTART e COLUMNEND, as colunas de início e final respectivamente. Como um documento pode ter várias planilhas, o parâmetro SHEET, é utilizado para determinar a planilha que será utilizada.

Como pôde ser observado, o POIDataStore permite a seleção de uma ou mais colunas, bem como uma ou mais linhas, ao contrário de algumas implementações onde só é possível a seleção da coluna [48]. Essa propriedade é particularmente interessante porque a informação pode estar organizada de

[40] http://poi.apache.org/ - API de manipulação de documentos da Apache

diferentes formas. Por exemplo, em uma planilha que contenha informações subdivididas em tempo e lugar, os usuários poderão selecionar tanto informações de um lugar quanto de uma data específica.

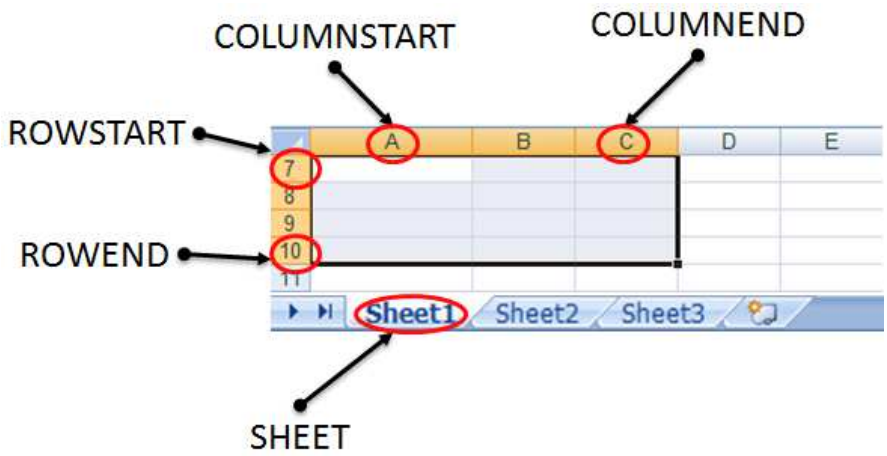

Figura 23. Parâmetros da planilha.

```
<config>
      <dataStore>
            <descriptor>
                  <param name="URI" value="C:\...\exemplo.xls"/>
            </descriptor>
            <collection id="empty">
                  <param name="SHEET" value="Sheet1"/>
                  <param name="ROWSTART" value="7"/>
                  <param name="ROWEND" value="10"/>
                  <param name="COLUMNSTART" value="0"/>
                  <param name="COLUMNEND" value="3"/>
            </collection>
      </dataStore>
<config>
```

Listagem 13. Exemplo de um arquivo de configuração declarando um DataStore de planilha.