Revista

Cadernos de Finanças Públicas

03 | 2024

TESOURONACIONAL

XBRL PROCESSOR: UMA FERRAMENTA PARA GERAÇÃO DE INST NCIAS XBRL

Henderson Acosta Bragança

Paulo Caetano da Silva

UNIFACS - Universidade Salvador - Brasil

Silas Pinho Ladislau

Daniel José Díaz

Universidade Nacional de Rosário - Argentina

RESUMO

É evidente a utilização da tecnologia *eXtensible Business Reporting Language* (XBRL) no contexto de reportes financeiros na internet, seja por suas vantagens e benefícios ou por imposições governamentais. No entanto, os dados a serem transportados por essa linguagem estão, em sua maioria, armazenados em estruturas definidas como banco de dados relacionais, arquivos JSON ou arquivos CSV. Destarte, é imprescindível para as organizações a integração da tecnologia XBRL com outras tecnologias de armazenamento de dados. Este artigo apresenta uma solução de *Extract, Transform and Load* (ETL), voltados para a extração de dados em fontes de diversos formatos de armazenamento e geração de instâncias XBRL, denominada XBRL Processor. Esta ferramenta contempla diferentes tipos de fonte de dados e gera a instância XBRL. Adicionalmente, foram implementadas parametrizações para atender a entrega da Matriz de Saldos Contábeis para o Sistema de Informações Contábeis e Financeiras da Secretaria do Tesouro Nacional do Brasil (SICONFI) e realizado um estudo de caso, de modo a validar a ferramenta XBRL Processor.

Palavras-Chave: XBRL Processor, XBRL, Integração XBRL, Integração de dados, Mapeamento de dados
JEL: C80, C88, H83, O30

SUMÁRIO

1. INTRODUÇÃO

Transportar dados financeiros e contábeis através da internet mantendo sua integridade estrutural e semântica tem sido alcançado utilizando a tecnologia *eXtensible Business Reporting Language* (XBRL[1]). A XBRL é uma tecnologia derivada da *eXtensible Markup Language* (XML), que é considerada pelo *World Wide Web Consortium* (W3C[2]) como um importante meio de intercâmbio de dados na internet, uma vez que possibilita a troca de dados independente de plataforma (JAYASHREE; PRIYA, 2020). O projeto da tecnologia XML favorece a simplicidade, generalidade e a usabilidade (ZHU *et al*., 2017), estes atributos propiciam o uso da tecnologia XML na internet e como meio de integração e intercâmbio de dados. Diversas alternativas são propostas na literatura para a integração e mapeamento de dados entre documentos XML, os quais são usados como meio de transporte de dados na internet, e diferentes bases de dados (SONG; HAW; CHUA, 2019).

Existem algumas soluções para mapeamento de dados XBRL, porém adicionam custo computacional, e.g. propostas da Fujitsu[3] e Oracle[4], pois é necessário criar bases de dados paralelas com os dados advindos dos documentos XML, além de ter que armazenar o documento XML para resguardar a integridade das informações. Ademais, essas soluções são proprietárias e com um custo muito elevado. Em outros casos é preciso desenvolver aplicações específicas para mediar o intercâmbio de dados entre os diferentes ambientes, i.e. XML, bases relacionais, CSV[5], NoSQL[6] e JSON[7].

A tecnologia XBRL, foi desenvolvida para o intercâmbio de informações financeiras na internet, sendo consolidada como o padrão a ser utilizado por órgãos governamentais em diversos países (DUNCE; SILVA; VIANA, 2013). Esta tecnologia possui a capacidade de transportar os dados financeiros juntamente com a sua semântica, algo imprescindível para uma análise correta da informação. Por portar os dados juntamente com a sua semântica, exclui-se a necessidade de descobrir o significado da informação fornecida, quando não é acompanhada do seu contexto, ou seja, sem a semântica (BRAGANÇA *et al*., 2019).

A XBRL é uma tecnologia que se tornou padrão internacional para o intercâmbio de da-

1 https://www.xbrl.org/
2 https://www.w3.org/
3 https://www.fujitsu.com/global/products/software/middleware/application-infrastructure/interstage/solutions/xbrl/
4 https://www.oracle.com/performance-management/#rc30p7
5 Comma Separated Values (valores separados por vírgula)
6 Not Only SQL
7 JSON é o acrônimo de Java Script Object Notation, é um padrão de troca de dados entre sistemas.

dos financeiros, e.g. US-SEC[8], Comitê Europeu de Supervisores Bancários (CEBS)[9], Banco da Espanha[10], Banco do Japão[11], são apenas alguns exemplos de órgão governamentais que adotaram a XBRL para o intercâmbio de dados com suas organizações supervisionadas[12]. No Brasil, diante das imposições normativas, como é o caso da lei 12.527/11 que dispõe sobre o acesso às informações dos Estados, Municípios e do Distrito Federal, popularmente chamada de Lei da Transparência, coadunado com a norma 896/17 da Secretaria do Tesouro Nacional[13], direciona as esferas governamentais para o uso de XBRL em suas publicações conexas às demonstrações contábeis junto aos órgãos de controle.

Tanto os governos em seus diferentes níveis (federal, estadual e municipal), quanto as organizações privadas, não possuem ferramentas gratuitas e *open source* que possibilitem a exportação das informações financeiras e contábeis para o formato XBRL, principalmente quando extraídas diretamente das bases de dados relacionais ou NoSQL, criando uma dificuldade para as organizações privadas e governos para representar seus dados financeiros e fiscais em XBRL.

Existem diferentes tecnologias para o armazenamento de dados, baseadas em diferentes modelos, o mais comum é o relacional e, mais recentemente, o NoSQL, comumente utilizado na internet. O modelo de dados relacional representa o banco de dados como uma coleção de relações (NAVATHE, 2013), que após a sua concepção na década de 60, passou a ser utilizado por grande parte das aplicações de software, sendo um modelo que deverá continuar a ser usado por muitos anos em razão da sua robustez (NAVATHE, 2013). Devido a importância e ampla utilização nas aplicações privadas e governamentais, é imprescindível considerar esse modelo como fonte de dados. Também a abordagem relacionada a arquivos CSV como fonte de dados é pertinente, uma vez que existem cenários[14] nos quais os dados são exportados apenas em arquivos CSV (BRAGANÇA *et al*., 2019).

Os bancos de dados NoSQL trabalham com dados desnormalizados (SOARES; BOSCARIOLI, 2013). Os bancos de dados NoSQL estão sendo adotados por grandes empresas, e.g. Google, o Facebook e o Twitter (SOARES; BOSCARIOLI, 2013), uma vez que este modelo possui arquitetura que facilita o tratamento de expressiva demanda por consultas, vinculada ao

8 https://www.sec.gov/
9 https://www.europeansources.info/record/committee-of-european-banking-supervisors-cebs/
10 https://www.bde.es/f/webbde/INF/MenuVertical/Supervision/Normativa_y_criterios/informacion/ficheros/IE_2008_02.pdf
11 https://www2.boj.or.jp/archive/en/announcements/release_2006/fsk0602a.htm
12 https://www.xbrl.org/the-standard/why/ten-countries-with-open-data/
13 https://www.gov.br/tesouronacional/pt-br
14 Os Sistemas Integradores da Administração Financeira (SIAF) dos Estados na sua maioria são incapazes de gerar a instância XBRL por utilizarem o SGBD Adabas/SoftwareAG, mas possuem a capacidade de gerar arquivos CSV.

elevado número de informações, com alta escalabilidade dos dados (SOARES; BOSCARIOLI, 2013). Por conseguinte, os bancos de dados NoSQL também são considerados como alternativa de fonte de dados na solução proposta neste trabalho.

A Secretaria do Tesouro Nacional (STN) através do Sistema de Informações Contábeis e Fiscais do Setor Público Brasileiro (Siconfi)[15] definiu a tecnologia XBRL como o principal formato de dados para a Matriz de Saldos Contábeis, de maneira a implementar melhoria da qualidade da informação no Setor Público Brasileiro (STN, 2017). A Matriz de Saldos Contábeis (MSC) é uma estrutura planejada para conter os detalhes da informação contábil do ente federativo[16] sendo recomendado pela STN a extração direta da fonte de dados para minimizar erros de preenchimento de planilhas ou formulários (STN, 2017).

O propósito deste trabalho, portanto, é desenvolver uma solução *open source* que atenda aos mais diversos cenários de fontes de dados, exportando as informações referentes à MSC para instâncias XBRL, atendendo a exigência da STN através do Siconfi. A partir desta introdução, na Seção 2 são discutidos os trabalhos correlacionados ao proposto. Na Seção 3 a solução proposta é motivada. A Seção 4 descreve a abordagem conceitual e o desenvolvimento da ferramenta para extração, transformação e carga de dados (ETL) para a tecnologia XBRL, nomeada como XBRL Processor. A Seção 6 expõe as conclusões deste trabalho, suas contribuições, limitações e propostas de trabalhos futuros.

2. TRABALHOS CORRELATOS

É salutar observar que as propostas encontradas na literatura, terem o seu foco em solucionar problemas específicos, como atender uma legislação ou identificar similaridade semântica em instâncias XBRL, isto posto, pode-se perceber que os trabalhos se limitam a tratar à questão de ler e exibir os dados das instâncias XBRL. Devido à escassez de trabalhos que tratam da geração de instâncias XBRL e ao consórcio responsável pela XBRL admitir as linguagens XML, JSON e arquivos CSV como tecnologias para o transporte de dados, optou-se por investigar mapeamentos relacionados a XML, JSON e CSV na busca de trabalhos que satisfizessem a investigação quanto a soluções de integração ou mapeamento de dados para XBRL e vice-versa. Nesta perspectiva identificamos os trabalhos (JAYASHREE; PRIYA, 2020), (ZHU

15 https://Siconf.tesouro.gov.br/Siconf/index.jsf

16 Ente federativo pode ser entendido também como um Estado, sendo uma unidade autônoma (autogoverno, autolegislação e autoarrecadação) dotada de governo próprio, constituição e que, com outros estados, forma uma federação.

et al., 2017), (SONG; HAW; CHUA, 2019), (NASSIRI; MACHKOUR; HACHIMI, 2018), (CHEN, 2018), (SALEM *et al*., 2017), (LYAMIN; CHEREPOVSKAYA, 2018), (ALAMI; BAHAJ, 2017), (SONG; HAW, 2020), (QTAISH; AHMAD, 2016), (NASSIRI; MACHKOUR; HACHIMI, 2017), (LIU; ETUDO; YOON, 2020), (BIKAKIS *et al*., 2015), (NIEWERTH; SCHWENTICK, 2018), (MAATUK; ALI; ALJAWARNEH, 2015), (ASIMADI *et al*., 2017), (BAI *et al*., 2015), (PETKOVIĆ, 2017a, 2017b), (YAGHMAZADEH; WANG; DILLIG, 2018) e (DOI; TOYAMA, 2019) que tratam do mapeamento dos dados de instâncias XML ou JSON e arquivos CSV para bancos de dados relacionais.

Somente uma proposta apresentou solução para gerar instâncias XBRL, isto para atender especificamente uma imposição legal (BRAGANÇA *et al*., 2019), este trabalho utiliza apenas arquivos CSV como fonte de dados e o sistema de ETL proprietário Hitachi Pentaho[17]. Quando relacionado a linguagem JSON, juntamente com arquivos CSV, não foram localizadas soluções que tivessem como objetivo instanciar os dados nos moldes definidos pelas especificações XBRL-JSON[18] ou XBRL-CSV[19] do consórcio XBRL. Desta forma, a proposta de uma ferramenta capaz de conectar as mais diversas fontes de dados, vincule os dados à taxonomia escolhida e gere a instância XBRL, guiou o desenvolvimento deste trabalho.

3. MOTIVAÇÃO

Para que uma informação seja contextualizada em determinado domínio, ela deve ser interligada com outras informações, sejam elas contextuais, informações do passado, experiências ou informações futuras (CERQUEIRA; SILVA, 2021) e (BEELITZ, 2017). No domínio financeiro e contábil a tecnologia que atende a esses requisitos é a XBRL.

Em 2008 Silva afirmou que a tecnologia XBRL estava se tornando um padrão tecnológico para a troca, armazenamento e divulgação de informações financeiras na internet (SILVA *et al*., 2008). De fato, essa previsão se confirmou, a adoção da tecnologia por relevantes instituições tem impulsionado a utilização da XBRL, como é o caso da US-SEC[20] que apontou a tecnologia XBRL como a chave para a sua modernização (GRAY; MILLER, 2009).

A tecnologia XBRL por ser robusta e representar completamente o domínio financeiro e contábil, traz consigo a complexidade inerente a soluções destinadas a problemas complexos,

17 https://www.hitachivantara.com/en-us/products/pentaho-platform/data-integration-analytics.html
18 https://www.xbrl.org/guidance/xbrl-json-tutorial/
19 https://www.xbrl.org/guidance/xbrl-csv-tutorial/
20 https://www.sec.gov/

como é o caso do transporte de informações financeiras e contábeis com todo o seu arcabouço contextual.

A geração de instâncias XBRL utilizando como fonte os repositórios de dados financeiros e contábeis não é trivial (ASIMADI *et al*., 2017), apenas ferramentas proprietárias (e.g. Amelkis, aSISt, Vizor)[21] disponíveis no mercado são capazes de extrair de forma automática os dados armazenados em diferentes repositórios que usam diversas tecnologias de armazenamento (e.g. relacional, NoSQL, CSV). Ainda assim, normalmente essas ferramentas manipulam dados de um modelo específico, i.e., elas não possuem generalidade suficiente para manipularem diversos tipos de formatos de dados e XBRL. Ademais, a ausência de ferramentas *open source* pode dificultar a adoção desta tecnologia de forma massiva, principalmente no território brasileiro, cujo único projeto em operação é o da Secretaria do Tesouro Nacional (STN), denominado Siconfi. O Siconfi exige dos entes federados a entrega das informações contábeis e financeiras baseada na tecnologia XBRL, como não existem soluções *open source* e gratuitas, a alternativa para os entes da federação brasileira é o uso de sistemas proprietários. Atender as exigências da STN no que se refere à entrega da MSC em instância XBRL permite ao ente cumprir a Lei de Responsabilidade Fiscal a qual todo o setor público brasileiro está sujeito.

Para atender a legislação, as unidades da federação precisam inserir os dados contábeis e financeiros em sistemas proprietários, em muitos casos de forma manual, para gerar o documento XBRL, uma vez que a conexão com a fonte de dados, e.g. bancos de dados relacionais, não é simples. Isto gera custos de recursos humanos e computacionais, oriundos da baixa interoperabilidade entre os ambientes XBRL e as fontes de dados, o que implica em custos financeiros adicionais. É possível enviar os dados da MSC no formato CSV para o Siconfi, no entanto, mesmo o formato CSV adiciona custos adicionais de recursos humanos e computacionais. O formato CSV continua sendo aceito pelo Siconfi devido existir entes federados brasileiros que não possuem ferramentas que possibilitem a entrega dos dados da MSC na linguagem XBRL.

Pode-se perceber no trabalho de Bragança (BRAGANÇA *et al*., 2019) que existe uma carência nos poderes executivos do Brasil, seja na esfera federal, estadual ou municipal, por soluções que atendam a demanda na geração das instâncias XBRL a serem entregues ao Siconfi. Para atender essa demanda, busca-se neste trabalho desenvolver uma solução ETL *open source* dividida em dois estágios de execução de modo a facilitar o desenvolvimento de novas funcionalidades, o primeiro estágio destina-se a conexão com a fonte de dados e o segundo a

21 https://www.xbrl.org/the-standard/how/tools-and-services/

vincular os dados com a taxonomia[22] do Siconfi, possibilitando a geração da instância XBRL. Neste artigo é apresentado uma ferramenta que busca atender aos mais diversos cenários de intercâmbio de dados financeiros, com configurações reutilizáveis e com redução da complexidade do ambiente. A solução foi planejada de forma a facilitar a manutenção, a extensão e a integração a softwares contábeis e financeiros existentes, para gerar instâncias das mais variadas taxonomias.

Desta forma, o problema de pesquisa central que norteia o desenvolvimento deste trabalho é:

• Questão 1: Existem soluções que permitam o mapeamento de dados de diferentes formatos para XBRL?

A partir desta questão de pesquisa principal (Questão 1) pôde-se inferir outras questões de pesquisa:

• Questão 2: As soluções existentes são *open source*, ou seja, são de código aberto e gratuitas?

• Questão 3: as soluções propostas possuem generalização suficiente de maneira que permitam o mapeamento de diferentes modelos de dados (e.g. relacional, NoSQL, CSV, JSON) para XBRL?

• Questão 4: as soluções propostas atendem aos requisitos do Siconfi e Matriz de Saldos Contábeis (MSC) da STN?

• Questão 5: É possível construir uma ferramenta com a qual qualquer taxonomia possa ser utilizada para gerar instâncias XBRL a partir de qualquer fonte de dados?

Para o problema de pesquisa investigado, formulou-se a seguinte hipótese:

• Hipótese 1: É possível construir uma ferramenta de extração, transformação e carga de dados a partir de qualquer modelo de dados, seja de SGBD's relacionais, NoSQL ou arquivos CSV, e exportá-los como instâncias XBRL, capaz de adaptar-se às necessidades conexas à fonte de dados e taxonomias dos utilizadores.

4. FERRAMENTA XBRL PROCESSOR

Iniciativas de código aberto e gratuitas para suprir a ausência de soluções que tenham a capacidade de gerar instâncias XBRL podem ser observadas no trabalho de Bragança (BRA-

22 A taxonomia XBRL é um conjunto de documentos baseados nas tecnologias XML Schema e Xlink que permite a definição de conceitos financeiros e contábeis e seus relacionamentos.

GANÇA *et al*., 2019), propondo uma ferramenta ETL, no entanto, a solução proposta coleta apenas dados de arquivos CSV, restringindo a capacidade de adoção da solução. Pode-se perceber ainda que existe uma carência nos poderes executivos do Brasil, seja na esfera federal, estadual ou municipal, por soluções que atendam a demanda de geração das instâncias XBRL a serem entregues ao Siconfi (BRAGANÇA *et al*., 2019).

A revisão sistemática da literatura apresentada em Bragança (BRAGANCA; CAETANO; BERNADINO, 2022), identificou que existe uma preocupação primária em desenvolver softwares que sejam capazes de ler e exibir os relatórios contidos nas instâncias XBRL, no entanto, não existe essa mesma preocupação em gerar instâncias XBRL (BRAGANCA; CAETANO; BERNADINO, 2022). Partindo dessa constatação, desenvolveu-se neste trabalho uma ferramenta voltada para a geração das instâncias XBRL de forma a preencher essa lacuna.

Esta seção descreve o desenvolvimento da ferramenta para leitura, transformação e carga de dados (ETL) para a tecnologia XBRL, intitulada como XBRL Processor. O principal objetivo do desenvolvimento desta solução é permitir a extração de dados de diferentes sistemas de armazenamento (SGBD's relacionais, NoSQL, CSV), e carregá-los em instâncias XBRL, definidas por uma taxonomia. Para realizar o desenvolvimento da ferramenta utilizou-se alguns critérios:

• A utilização da MSC (Matriz de Saldos Contábeis) no formato e taxonomia XBRL definidos pelo Siconfi (Sistema de Informações Contábeis e Fiscais do Setor Público Brasileiro) para que a aplicação possa ser validada e esteja pronta para solucionar o problema da entrega da MSC no formato XBRL ao Siconfi/STN, uma exigência feita aos entes federados do Brasil;

• O desenvolvimento da ferramenta proposta foi baseado na linguagem TypeScript (BOGNER; MERKEL, 2022) em razão da facilidade de compreensão do código, de possuir performance de execução, disponibilidade de biblioteca gratuita e editável para leitura e geração de instâncias XML;

• Por fim, utilizou-se como fonte de dados os SGBD's MySQL (CHRISTUDAS, 2019), Sqlite3 (BIN *et al*., 2022) e MongoDB (GYŐRÖDI *et al*., 2022), com os quais foi possível aferir a conexão da ferramenta com SGBD's relacionais e NoSQL, seguindo o princípio de utilização de fontes de dados de diferentes formatos.

O desenvolvimento da ferramenta XBRL Processor consistiu em três etapas:

• Na primeira etapa, construiu-se a ferramenta ETL para XBRL baseada na taxonomia XBRL, versão 2022, definida para a Matriz de Saldos Contábeis do Siconfi. A fonte de dados utilizada foi o SGBD Sqlite3 com dados fictícios e anonimizados, no entanto coerentes, forne-

cidos pela Secretaria de Finanças do Estado de Rondônia (SEFIN - RO);

• Na segunda etapa outros SGBD's foram incorporados no projeto do código da ferramenta. Desta forma, foi possível verificar a adaptabilidade do código a diferentes formatos e fontes de dados. Foram adicionados conectores para os SGBD's MySQL e MongoDB;

• Na terceira etapa, comparou-se a instância XBRL gerada pelo XBRL Processor com a fornecida pela Secretaria de Finanças do Estado de Rondônia. Por fim, a instância XBRL gerada com os dados fictícios foram validadas pela STN no Siconfi e pela Secretaria de Finanças do Estado de Rondônia nos ambientes de testes e de produção respectivamente.

A seguir na Seção 4.1 é apresentado a arquitetura concebida para a ferramenta proposta de forma que ela seja adaptável para inclusão de novas bases e formatos de dados. A Seção 4.2 mostra o desenvolvimento da ferramenta para extração, transformação e carga de dados (ETL) para a tecnologia XBRL, nomeada como XBRL Processor.

4.1 ARQUITETURA DA XBRL PROCESSOR

A arquitetura proposta, representado na Figura 1, tem a finalidade de conectar diferentes bases de dados, e.g. relacional, NoSQL, e possibilitar a junção dos dados extraídos dessas bases de dados com a taxonomia a ser utilizada gerando a instância XBRL.

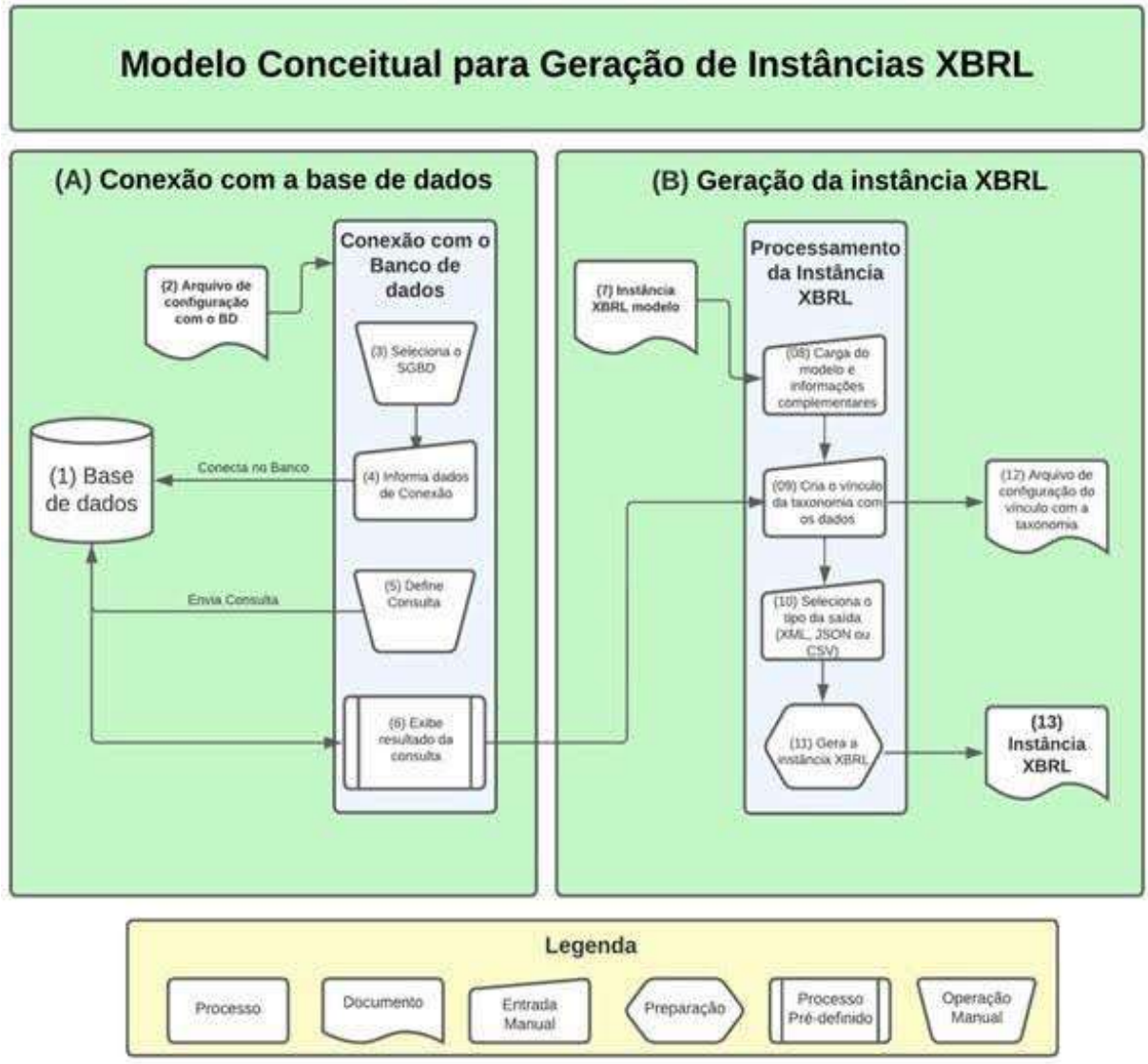
A proposta do processo de geração de instâncias XBRL foi dividida em duas etapas: (A) conexão com a base de dados e (B) geração da instância XBRL a partir da taxonomia informada.

Dividir a solução em duas etapas permitirá uma melhor manutenção no código e desenvolvimento de possíveis melhorias e novas funcionalidades, com isso cada etapa possui o seu contexto funcional, aumentando a coesão e diminuindo o acoplamento. Em softwares a modularização permite que uma mudança em um componente tenha impacto mínimo em outros (LAZZARI; FARIAS, 2022).

O processo de conexão com o banco de dados inicia com a escolha do SGBD, como podemos observar na Figura 1(A-3), a lista dos SGBD's suportados será limitada apenas pelos *drivers* disponíveis, no entanto, pode ser adicionado quaisquer *driver's*, e.g. driver para PostgreSQL, Oracle, MySQL, SQL Server, MongoDB, ou outro que seja necessário para a extração de dados para a geração da instância XBRL.

Na sequência Figura 1(A-4), os dados de conexão são solicitados de modo a permitir a conexão com o SGBD. Dados como porta TCP, login e senha são informações indispensáveis para a conexão. Quando os dados de conexão são informados, um teste de conexão é executado.

Figura 1 – Arquitetura para a geração de instâncias XBRL



Após completado o processo de conexão, se inicia a execução da inserção do código da consulta Figura 1(A-5), e.g. SQL ou o código pertinente ao SGBD selecionado nas atividades anteriores. A possibilidade de inserir o código direto na aplicação, permite uma flexibilidade na consulta dos dados de modo a facilitar ajustes e adição de colunas com dados estáticos, e.g. a informação relacionada aos valores financeiros serem em moeda brasileira (BRL). Após a execução do código, pode-se verificar na Figura 1(A-6), que a atividade seguinte é a exibição do resultado da consulta para conferência.

A conexão com o banco de dados foi planejada de modo que todos os dados e parametrizações possam ser reaproveitados em uma posterior execução, para isso o arquivo com as configurações é reaproveitado nas execuções subsequentes Figura 1(A-2).

O processo de vincular a taxonomia aos dados não é um processo ordinário, (ASIMADI *et al*., 2017) apontou em seu trabalho que o processo de integração da XBRL permanece com-

plexo, portanto, o início do processamento da instância XBRL, Figura 1(B-7), traz a possibilidade de usar uma instância XBRL modelo para recuperar o maior número possível de vínculos entre a taxonomia e os dados. A instância modelo pode ser fornecida pelo criador da taxonomia para o qual os dados serão enviados.

No passo seguinte, dados complementares podem ser adicionados Figura 1(B-8), entende-se como informações complementares àquelas que são exclusivas da organização que está gerando a instância XBRL, e.g. código de cadastro da organização junto ao órgão que receberá os dados.

Após a inserção da taxonomia modelo e informações complementares, cria-se o vínculo dos dados recebidos do banco de dados com a taxonomia, Figura 1(B-9), informações que são utilizadas repetidas vezes podem ser preservadas, se necessário para posterior recuperação, em arquivo de configuração pertinente, Figura 1(B-12) (DIMOU *et al*., 2014).

O consórcio XBRL, no ano de 2021, definiu novas formas de utilizar a tecnologia, definindo como recomendação além da XBRL estendida da XML, a XBRL estendida do JSON e de arquivos CSV (CERQUEIRA; SILVA, 2016). A XBRL em JSON e CSV são limitadas à instância não sendo admitido nesses formatos o arquivo contendo a taxonomia. Também, não foi esclarecido pelo consórcio responsável pela XBRL como serão as instâncias XBRL-JSON e XBRL-CSV, quando geradas a partir da taxonomia GL[23]. No entanto, para estar alinhado com as recomendações do consórcio XBRL é preciso permitir a possibilidade de gerar a instância nos três formatos admitidos pelo consorcio XBRL. Essa previsão existe nesta etapa do processo, como pode ser observado na Figura 1(B-10).

Por fim, é executado o processamento dos dados para gerar a instância XBRL, Figura 1(B-11), tendo como produto a instância com todos os dados e seus vínculos com a taxonomia, bem como os dados complementares de unidade e referências através dos links, Figura 1(B-13).

4.2. IMPLEMENTAÇÃO

O desenvolvimento da ferramenta XBRL Processor utilizou a arquitetura proposta na Seção 4.1, a qual propõe uma solução para o problema latente de geração de instâncias XBRL a partir de diferentes SGBD's. Desse modo, a codificação foi executada para permitir a validação externa das instâncias geradas pela aplicação.

23 http://www.xbrl.org/int/gl/2016-12-01/gl-framework-2017-PWD-2016-12-01.html

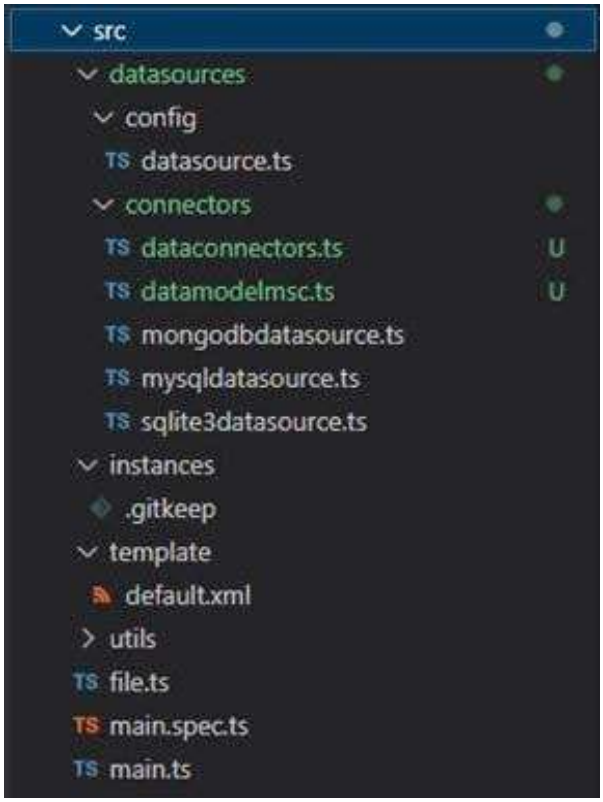
4.2.1. ASPECTOS DE IMPLEMENTAÇÃO

A organização do código fonte se deu com foco na arquitetura limpa, de forma a facilitar o entendimento do código, recursos e funções (SOMMERVILLE, 2011). A arquitetura limpa possibilita a evolução da aplicação sem o consumo excessivo de recursos humanos ou de tempo. O fato de separar o projeto em camadas independentes, propicia que alterações em uma camada não interfiram nas demais (SOMMERVILLE, 2011). Dessa maneira, espera-se que o código possa ser estendido ou incorporado a aplicações financeiras e contábeis. O código fonte deste trabalho pode ser acessado no repositório github[24]. Seguindo o conceito definido na Seção 4.1, o código está dividido em dois componentes, no qual o primeiro componente definido na Figura 1(A), que trata da conexão com a fonte de dados, pode-se relacionar com a pasta src/*datasource* do código, sendo subdividida em src/datasource/config e src/datasource/connectors. O segundo componente exibido na Figura 1(B), que trata da geração da instância XBRL, foi integrado ao arquivo src/main.ts. A organização do código pode ser observada na Figura 2, que expõe a estrutura de pastas utilizada para organizar o código.

A fonte de dados foi separada em: os códigos relacionados à conexão com o SGBD, configurações da fonte de dados (SGBD) e a estrutura da MSC esperada na consulta. Os arquivos contendo o código relacionado a conexão, tais como, portas TCP, login, podem ser acessados no arquivo src/datasource/config/datasource.ts. Esse arquivo é representado conceitualmente na Figura 1(A-4) e teve seu código adaptado para o SGBD MySQL, como observado na Figura 3.

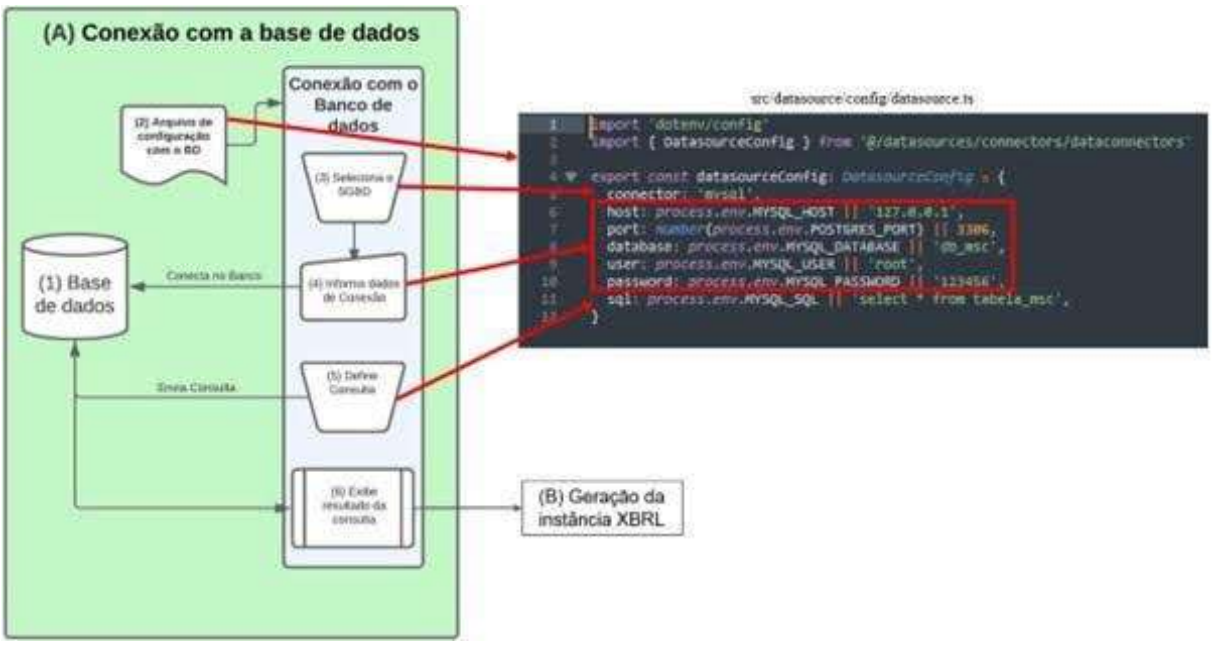
24 Endereço ocultado devido a determinação do Edital STN N°10 de 29 de Maio de 2023, 28° PRÊMIO TESOURO NACIONAL 2023, não identificando o(s) autor(es) em nenhum trecho, inclusive nas propriedades do arquivo.

Figura 2 - Estrutura de pastas do código



O arquivo definido na Figura 1 (A-2), referente aos dados de conexão com a base de dados, não foi gerado separadamente dos arquivos do código, sendo mantido no arquivo src/datasource/config/datasource.ts, de modo a atender a restrição de reaproveitar as parametrizações pela ferramenta, a Figura 3, ilustra esta relação do código com a arquitetura.

Figura 3 - Dados de conexão com o SGBD MySQL



Cada fonte de dados possui um código para a leitura dos dados armazenados em sua estru-

tura. Para acomodar esse código foi criada a pasta src/datasources/connectors/, onde cada fonte de dados possui, separadamente, o seu respectivo código. As configurações para as fontes de dados SQLite3 (*sqlite3datasource.ts*), MySQL (*mysqldatasource.ts*) e MongoDB (*mongodb-datasource.ts*) estão disponíveis para serem utilizados, para outras fontes de dados e.g. SGBD PostgreSQL, novos arquivos com os códigos referente a configuração, serão necessários. Os códigos referentes aos SGBD's disponíveis podem ser observados nas Figuras 4, 5 e 6.

Figura 4 - Implementação do SGBD MongoDB

```
import { DataModelMSC, Datasource } from './datamodelmsc'
import { MongoClient } from 'mongodb'
import { datasourceConfig } from '../config/datasource'
import { assertConnectorType } from '@/utils/functions'
import { MongodbConnector } from './dataconnectors'

/**
 * Area destinada as implementações gerais da fonte de dados MongoDB.
 */

class MongoDbDatasource implements Datasource {
  private client: MongoClient | null = null

  private async connect() {
    assertConnectorType<MongodbConnector>(datasourceConfig, 'mongodb')
    this.client = new MongoClient(datasourceConfig.connection_string)
    console.log('Conectado ao MongoDb')
  }

  async getData(): Promise<DataModelMSC[]> {
    try {
      assertConnectorType<MongodbConnector>(datasourceConfig, 'mongodb')

      await this.connect()

      const database = this.client?.db(datasourceConfig.database)
      const collection = database?.collection(datasourceConfig.collection)

      const rows = (await collection
        ?.find({})
        .toArray()) as unknown as DataModelMSC[]

      console.log('Lido os dados do Mongodb')

      return rows
    } catch (error) {
      console.error(`Erro ao ler dados no Mongodb ${error}`)
    }
    const data = {} as DataModelMSC
    return [data]
  }

  async close() {
    await this.client?.close()
    console.log('Conexão com Mongodb encerrada')
  }
}
export { MongoDbDatasource }
```

Figura 5 - Implementação do SGBD SQLite3

```
import sqlite3, { Database } from 'sqlite3'
import path from 'path'
import { DataModelMSC, Datasource } from './datamodelmsc'
import { datasourceConfig } from '../config/datasource'
import { assertConnectorType } from '@/utils/functions'
import { SqliteConnector } from './dataconnectors'
/**
 * Area destinada as implementações gerais da fonte de dados SQLITE.
 */
class Sqlite3Datasource implements Datasource {
  private database: Database | null = null
  private async connect() {
    return new Promise((resolve, reject) => {
      assertConnectorType<SqliteConnector>(datasourceConfig, 'sqlite')
      const absolutePath = path.resolve(
        __dirname,
        '..',
        datasourceConfig.databaseFileName,
      )
      this.database = new sqlite3.Database(
        absolutePath,
        sqlite3.OPEN_READWRITE,
        (err) => {
          if (err) reject(err)
          resolve(
            console.log(`Conectou com a fonte de dados sqlite ${absolutePath}`),
          ),
        },
      )
    })
  }
  async getData(sql?: string): Promise<DataModelMSC[]> {
    await this.connect()
    return new Promise((resolve, reject) => {
      if (!sql) reject(new Error('SQL não definido'))
      else {
        this.database?.all(
          sql,
          [],
          (error: Error | null, rows: DataModelMSC[]) => {
            if (error) reject(error)
            resolve(rows)
          },
        )
      }
    })
  }
  close() {
    console.log('Encerrou a conexão com a fonte de dados sqlite')
    this.database?.close()
  }
}
export { Sqlite3Datasource }
```

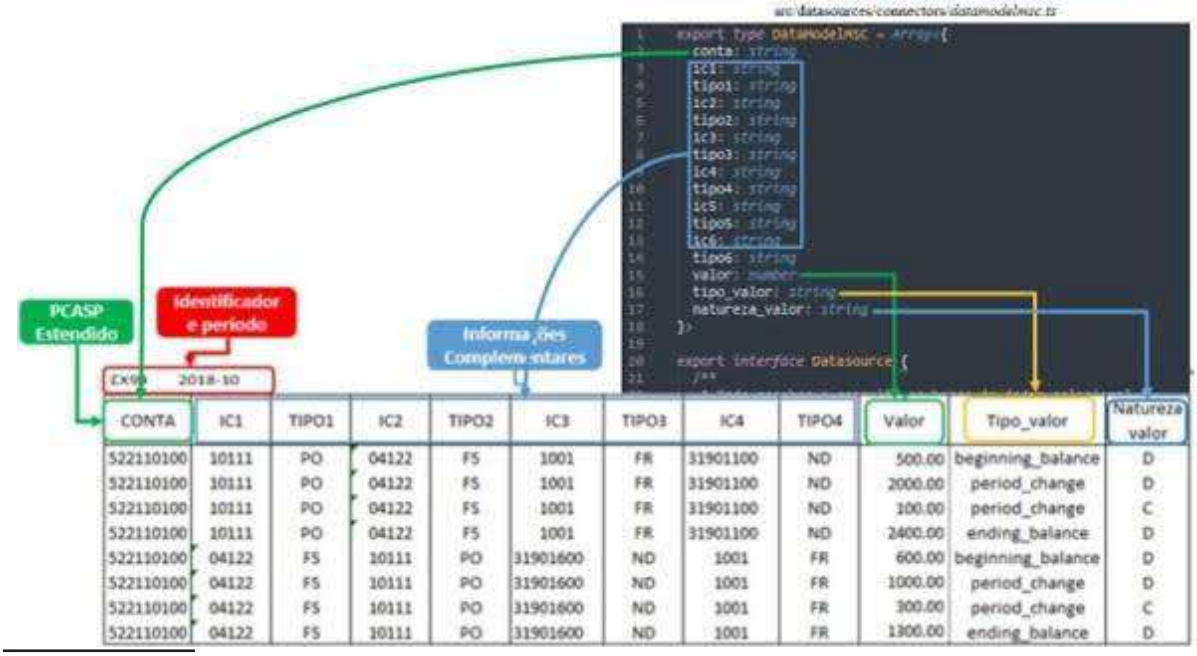
Figura 6 - Implementação do SGBD MySQL

```
import mysql from 'mysql2'
import { DataModelMSC, Datasource } from './datamodelmsc'
import { datasourceConfig } from '../config/datasource'
import { MysqlConnector } from './dataconnectors'
import { assertConnectorType } from '@/utils/functions'
/**
 * Area destinada as implementações gerais da fonte de dados MYSQL.
 */
class MysqlDatasource implements Datasource {
  private connection: mysql.Connection | null = null
  private connect() {
    assertConnectorType<MysqlConnector>(datasourceConfig, 'mysql')
    this.connection = mysql.createConnection({
      host: datasourceConfig.host,
      port: datasourceConfig.port,
      user: datasourceConfig.user,
      password: datasourceConfig.password,
      database: datasourceConfig.database,
    })
    console.log('Conectado com mysql')
  }
  async getData(sql?: string): Promise<DataModelMSC[]> {
    return new Promise((resolve, reject) => {
      if (!sql) reject(new Error('SQL não definido'))
      else {
        this.connect()
        console.log('obtendo dados do mysql')
        this.connection?.query(sql, (error: Error, rows: DataModelMSC[]) => {
          if (error) {
            console.log('Erro ao obter dados no mysql')
            reject(error)
          }

          resolve(rows)
        })
      }
    })
  }
  close() {
    console.log('Encerrando a conexão com mysql')
    this.connection?.end()
  }
}
export { MysqlDatasource }
```

Manteve-se a estrutura proposta pelo Siconfi para a MSC, na qual os campos seguem as definições do template[25] definido no manual do Siconfi. O layout da MSC foi planejado para representar as informações com base no padrão XBRL (STN, 2017). Embora a ferramenta não faça a validação dos dados frente a taxonomia, a consulta precisa retornar os dados do repositório de dados, seguindo a estrutura determinada pelo Siconfi. Pode-se observar a codificação da estrutura definida pelo Siconfi nas linhas 1 a 18 da Figura 7.

Figura 7 - Estrutura da MSC definida pelo Siconfi descrita no XBRL Processor



25 https://Siconf.tesouro.gov.br/Siconf/pages/public/conteudo/conteudo.jsf?id=12503

A diversidade de SGBD's utilizados na aplicação, facilitará a adequação a qualquer ambiente. Na Figura 8 é ilustrado o diagrama de sequência que descreve a conexão com o SGBD no primeiro módulo da ferramenta XBRL Processor. Espera-se que, aplicado em um ambiente de produção, o SGBD utilizado pela organização seja parametrizado no XBRL Processor, eliminando de modificação do código.

O segundo módulo, conforme descrito na Figura 1(B), executa o carregamento dos dados, carrega o modelo da instância, a taxonomia e procede com a geração da instância XBRL. O modelo da instância contido na pasta src/template, contém o cabeçalho, o código da unidade fornecido pelo Siconfi, a referência à moeda e aos períodos para o qual a instância a ser gerada se refere, e.g. neste caso à MSC 2023-01-31.

Figura 8 - Diagrama de sequência para carga de dados

Quando necessário, as atualizações desses dados devem ser feitas no arquivo modelo. Os demais dados (e.g. contexto, conta contábil etc.) contidos na instância modelo, são substituídos pelos dados recuperados do repositório de dados. Na Figura 9 pode-se verificar os dados carregados pela aplicação provenientes do arquivo modelo.

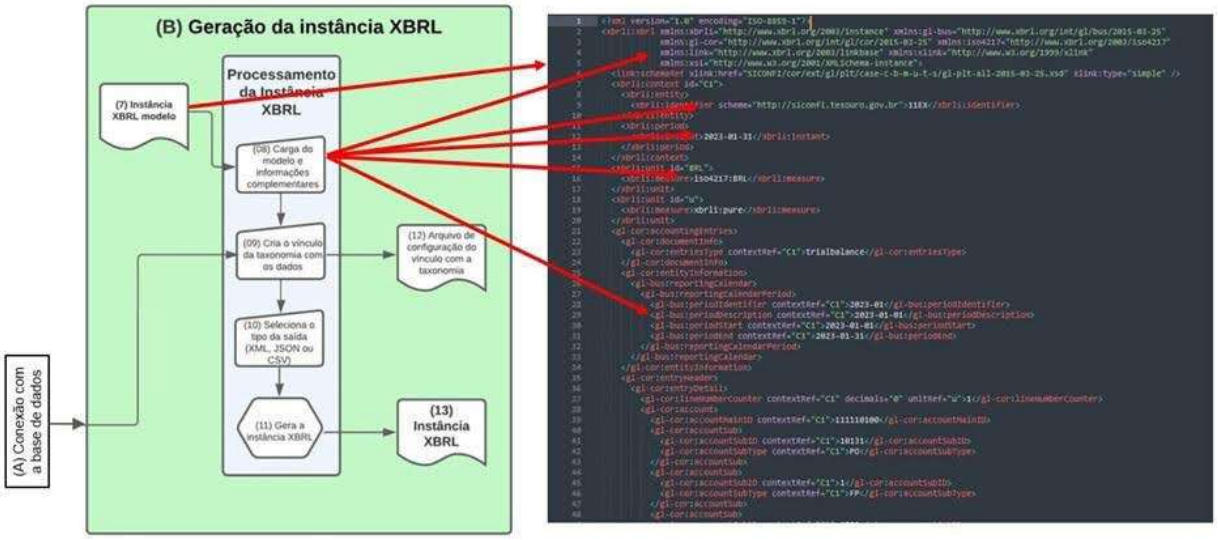
O segundo módulo da aplicação está concentrado no arquivo src/main.ts. Podemos observar as dependências dos componentes a e suas referências ao código na Figura 10.

A ferramenta utiliza a biblioteca fast-xml-parser[26] para a leitura e escrita de documentos XML. Devido as restrições da biblioteca fast-xml-parser, parte da taxonomia foi adicionada ao código para possibilitar a vinculação dos dados à taxonomia. A biblioteca fast-xml-parser é aberta e permite a evolução do código, no entanto, a edição da biblioteca para atender a leitura da taxonomia, sem que elementos da taxonomia estejam presentes no código, não foi contemplada nesta versão da ferramenta, portanto o arquivo definido na Figura 1(B-12) será gerado em trabalhos futuros. Um fragmento do código no qual é tratada a taxonomia utilizando a biblioteca fast-xml-parser presente no código, pode ser observado na Figura 11.

As últimas linhas do arquivo *main*, possui o código que gera a instância XBRL da MSC com os dados provenientes da fonte de dados escolhida. A instância é gerada e carregada na pasta src/instances com o nome instance.xml.

Para auxiliar na configuração e utilização da aplicação, um arquivo *README* foi adicionado aos arquivos da ferramenta com o resumo das atividades a serem seguidas e configurações necessárias para o funcionamento do XBRL Processor.

Figura 9 - Instância XBRL modelo da MSC



26 https://github.com/NaturalIntelligence/fast-xml-parser

Figura 10 - Diagrama de componentes da ferramenta XBRL Processor.

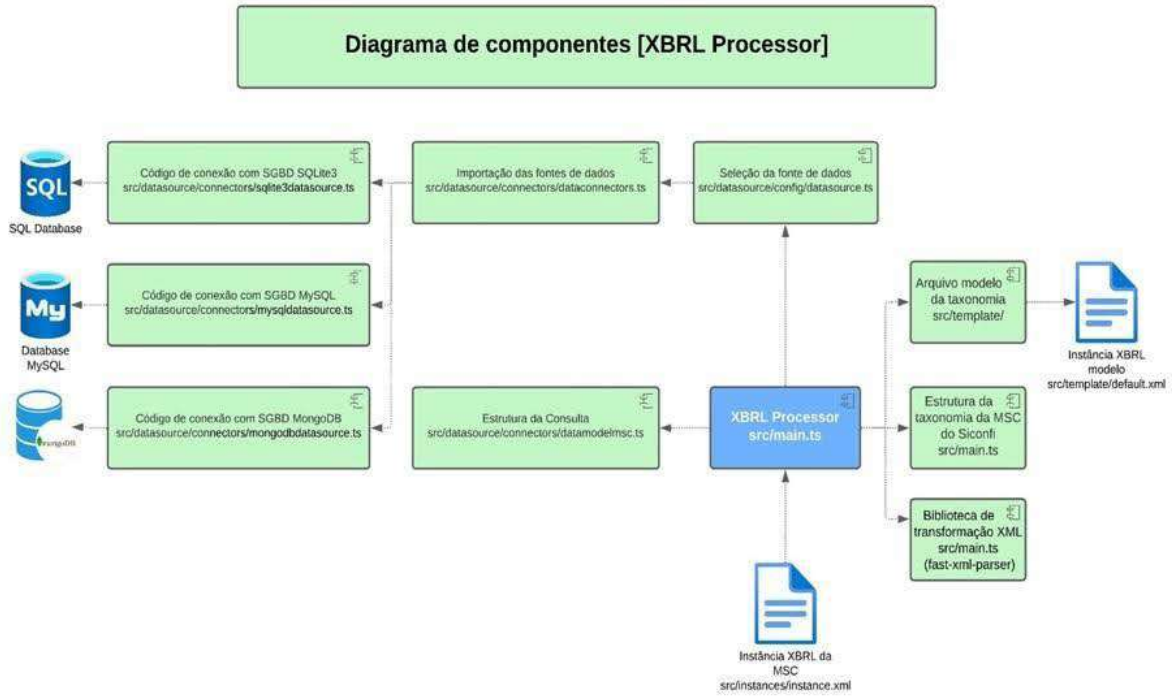


Figura 11 - Trecho do código do arquivo main contendo a taxonomia da MSC

```
    const accountSubs = state.map(({ subTypeText, subId }) => {
      return {
        'gl-cor:accountSubID': {
          '#text': subId,
          '@_contextRef': 'C1',
        },
        'gl-cor:accountSubType': {
          '#text': subTypeText,
          '@_contextRef': 'C1',
        },
      }
    })
    const result = {
      'gl-cor:lineNumberCounter': {
        '#text': index + 1,
        '@_contextRef': 'C1',
        '@_decimals': '0',
        '@_unitRef': 'u',
      },
      'gl-cor:account': {
        'gl-cor:accountMainID': {
          '#text': item.conta,
          '@_contextRef': 'C1',
        },
        'gl-cor:accountSub': accountSubs,
      },
      'gl-cor:amount': {
        '#text': item.valor,
        '@_contextRef': 'C1',
        '@_decimals': '2',
        '@_unitRef': 'BRL',
      },
      'gl-cor:debitCreditCode': {
        '#text': item.natureza_valor,
        '@_contextRef': 'C1',
      },
      'gl-cor:xbrlInfo': {
        'gl-cor:xbrlInclude': {
          '#text': item.tipo_valor,
          '@_contextRef': 'C1',
        },
      },
    }
    return result
  })
```

5. ESTUDO DE CASO PARA GERAÇÃO DA INST NCIA XBRL E VALIDAÇÃO DA SUA ESTRUTURA

O desenvolvimento deste trabalho e a ferramenta XBRL Processor foram planejados de modo que fosse possível aferir o resultado do processamento dos dados e resolvesse o problema de conversão de formato de dados para XBRL. A validação do resultado da geração da instância produzida pela ferramenta foi realizada por um ente da federação e pela ferramenta de validação do Siconfi (STN), organizações externas a este trabalho.

Portanto, desenvolver a ferramenta XBRL Processor parametrizada para a taxonomia e padrão de dados da MSC para o Siconfi, conforme observado nas Figuras 7 e 11, teve alguns benefícios: (i) resolver o problema de geração de instância XBRL; (ii) resolver o problema da entrega da MSC dos Estados e Municípios para a STN; (iii) ter a estrutura da instância para a MSC verificada pelo órgão regulador do Brasil.

Para compreender as validações apresentadas nesta seção, é importante saber que a instância foi submetida ao Siconfi de duas formas distintas, a primeira utilizando o ambiente de teste do Siconfi, através da central de atendimento, e a segunda submetendo a instância XBRL ao ambiente de produção do Siconfi através da Secretaria de Finanças do Estado de Rondônia, seguindo a entrega padrão da instância XBRL da MSC dos Estados, conforme ilustrado na Figura 12.

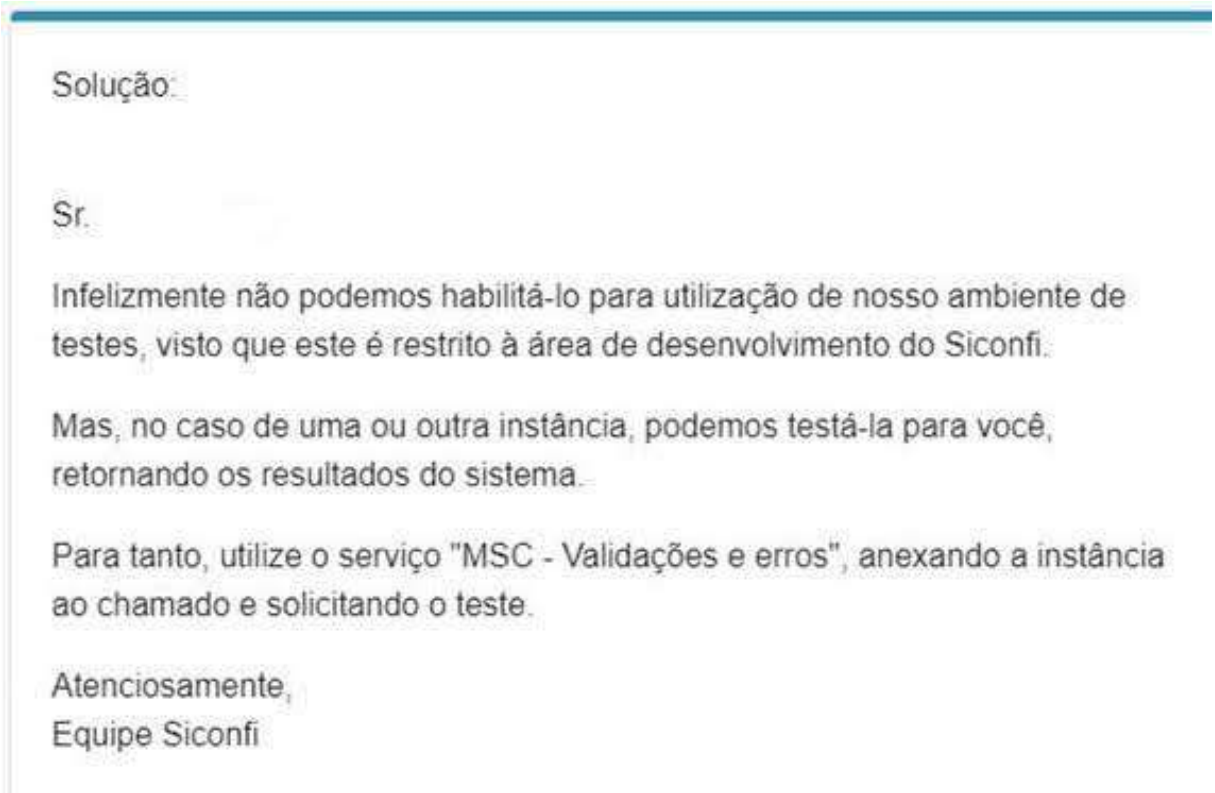
Figura 12 – Sequência da submissão da instância XBRL para validação

5.1 RESULTADOS

Inicialmente, para efeito de comparação, a Secretaria de Finanças do Estado de Rondônia forneceu dados de meses anteriores, já validados no Siconfi, para facilitar a comparação da instância XBRL gerada pela XBRL Processor e a instância entregue ao Siconfi pela Secretaria de Finanças do Estado de Rondônia, e validada sem erros ou inconsistências.

Planejou-se que a validação da estrutura da instância XBRL seria efetuada pelo Siconfi, uma vez que a comparação da instância gerada pela ferramenta XBRL Processor com as instâncias fornecidas pela Secretaria de Finanças do Estado de Rondônia se mostraram idênticas. Portanto, a Secretaria de Finanças do Estado de Rondônia forneceu dados fictícios e anônimos, ou seja, dados utilizados na base de dados de desenvolvimento, os quais foram carregados nos SGBD's SQLite3, MySQL e MongoDB, para que a instância pudesse ser gerada e entregue ao Siconfi para validação. Após contato com a equipe de suporte do Siconfi, foi orientado solicitar através do sistema do Siconfi[27] a validação da instância utilizando a opção "MSC - Validações e erros". A Figura 13[28] contém a resposta a solicitação de acesso ao ambiente de testes do Siconfi. Seguiu-se a orientação da equipe de suporte do Siconfi e através do sistema de chamados foi requerida a opção "MSC - Validações e erros" e fornecida a instância XBRL.

Figura 13 - Resposta da equipe de atendimento do Siconfi



Solução:

Sr.

Infelizmente não podemos habilitá-lo para utilização de nosso ambiente de testes, visto que este é restrito à área de desenvolvimento do Siconfi.

Mas, no caso de uma ou outra instância, podemos testá-la para você, retornando os resultados do sistema.

Para tanto, utilize o serviço "MSC - Validações e erros", anexando a instância ao chamado e solicitando o teste.

Atenciosamente,
Equipe Siconfi

A instância gerada pelo XBRL Processor com dados fictícios fornecidos pela Secretaria de Finanças do Estado de Rondônia é ilustrada na Figura 14.

27 https://e-servicos.tesouro.gov.br/#/

28 O cabeçalho do e-mail foi ocultado devido a determinação do Edital STN N°10 de 29 de maio de 2023, 28° PRÊMIO TESOURO NACIONAL 2023, não identificando o(s) autor(es) em nenhum trecho, inclusive nas propriedades do arquivo.

Figura 14 - Trecho da Instância XBRL gerada pela ferramenta XBRL Processor

A resposta fornecida validou a estrutura da instância XBRL gerada pela XBRL Processor, ou seja, a estrutura da instância está em conformidade com a instância XBRL da MSC requerida pelo Siconfi. Na Figura 15 é mostrada a resposta oficial do Siconfi sobre a instância fornecida.

O Siconfi solicita que seja desconsiderado a data do exercício, que consta no documento, por ser o ambiente onde a taxonomia 2023 está instalada no Siconfi. Na resposta fornecida pela validação feita pelo Siconfi foram indicados dois erros, nas linhas 571093 e 702450, no entanto esses erros referem-se aos dados e não à estrutura da instância, como pode ser aferido pelo e-mail enviado pela equipe do Siconfi e mostrado na Figura 16[29], uma vez que os dados de teste são fictícios, erros como este eram esperados. Seguindo a estratégia planejada, a mesma instância que foi entregue ao Siconfi A instância XBRL gerada também foi entregue à Secretaria de Finanças do Estado de Rondônia para que fosse submetida ao Siconfi seguindo os procedimentos rotineiros de entrega da MSC. Dessa forma, foi possível submeter a instância gerada pela ferramenta XBRL Processor ao ambiente de produção, ou seja, o ambiente real de validação. Por se tratar de dados fictícios, a Secretaria de Finanças do Estado de Rondônia não concluiu o envio da instância para o Siconfi, submeteu-a no ambiente de produção apenas até o momento da validação.

29 O cabeçalho do e-mail foi ocultado devido a determinação do Edital STN N°10 de 29 de Maio de 2023, 28° PRÊMIO TESOURO NACIONAL 2023, não identificando o(s) autor(es) em nenhum trecho, inclusive nas propriedades do arquivo.

Figura 15 - Validação da instância XBRL feita pelo sistema do Siconfi

siconfi

Secretaria do Tesouro Nacional - STN
Ministério da Fazenda - MF
Carregamento da MSC

Parâmetros da Solicitação

Ente:	Rondônia	**Tipo de Declaração:**	MSC Agregada
Poder:	Executivo	**Periodicidade:**	Mensal
Exercício:	2017	**Periodo:**	Janeiro
Instituição:	Governo do Estado de Rondônia		
Instância:	XBRL_Instance_Generation_MSC.zip		

Erros encontrados durante a validação da instância MSC submetida:

Linha 571093: Elemento 'gl-cor:accountMainID': O valor '' com tamanho = '0' não tem um aspecto válido em relação ao minLength '1' do tipo 'nonEmptyString'.
Linha 702450: Elemento 'gl-cor:accountMainID': O valor '' com tamanho = '0' não tem um aspecto válido em relação ao minLength '1' do tipo 'nonEmptyString'.

Figura 16 - Pedido de desconsideração do exercício 2017

Prezado,
O seguinte comentário foi feito por **E-Serviços do Tesouro Nacional** para o chamado **CH2023**

Sr.

Anexamos o resultado da tentativa de carregamento de sua instância MSC.

Não estranhe o exercício de 2017 no arquivo, pois é onde está instalada a taxonomia 2023 em nosso ambiente de teste.

Atenciosamente,

Equipe Siconfi

Figura 17 - Validação da instância XBRL em ambiente de produção

Nota-se que os erros nas linhas 571093 e 702450 são os mesmos da validação do ambiente de testes do Siconfi. Portanto a ferramenta XBRL Processor gerou a instância XBRL da Matriz de Saldos Contábeis do Estado de Rondônia para o Sistema de Informações Contábeis e Fiscais do Setor Público Brasileiro da Secretaria do Tesouro Nacional com a sua estrutura sem erros ou inconsistências.

Portanto, a aplicação está em funcionamento para conectar com fontes de dados relacionais ou NoSQL e gerar a instância XBRL da Matriz de Saldos Contábeis de acordo com a taxonomia definida pelo Siconfi ou ser incorporada a aplicações financeiras e contábeis existentes.

6. CONCLUSÃO

De acordo com o que foi apresentado, gerar instâncias XBRL não é um processo simples, e a ausência de ferramentas *open source* que tratam da problemática pode ser um fator contribuinte para a não adesão da tecnologia XBRL de forma expressiva por diversos órgãos reguladores e instituições privadas, principalmente nas organizações no Brasil. Este cenário é agravado pela escassez de ferramentas XBRL que utilizam os SGBD's relacionais ou NoSQL como fonte de dados, normalmente o processo de criação da instância não é automatizado, feito por inserção de dados manualmente nas ferramentas, é o caso da ferramenta da Fujitsu[30] e Amelkis[31].

Conforme exposto neste trabalho, existe carência de soluções que atendam a geração de instâncias XBRL com dados provenientes de bases de dados relacionais ou NoSQL, mesmo com a crescente necessidade de gerar instâncias XBRL de dados nos formatos avaliados, em

30 https://www.fujitsu.com/global/products/software/middleware/application-infrastructure/interstage/solutions/xbrl/

31 https://www.amelkis-solutions.com/

função da evolução e expansão da internet e de imposições legais.

A partir dos resultados apresentados neste artigo, pôde-se responder às questões de pesquisa apresentadas inicialmente, as quais são:

• Questão 1: Existem soluções que permitam o mapeamento de dados de diferentes formatos para XBRL?

Existem soluções, no entanto, são soluções proprietárias que se limitam a atender seguimentos específicos, como o bancário. Outra limitação importante está relacionada a geração de instâncias XBRL apenas a partir de dados contidos em bases de dados específicas de determinada organização, não permitindo a utilização de outras fontes ou SGBD's.

• Questão 2: As soluções existentes são *open source*, ou seja, são de código aberto e gratuitas?

Apenas o software Arelle[32] é *open source*, no entanto, tem como finalidade a leitura de instâncias XBRL e validação na taxonomia, já a geração de instâncias XBRL não é contemplada, assim como o carregamento dos dados oriundos de diferentes fontes e formatos de dados.

• Questão 3: as soluções propostas possuem generalização suficiente de maneira que permitem o mapeamento de qualquer modelo de dados (e.g. relacional, NoSQL, CSV, JSON) para XBRL?

Nenhum software investigado apresentou a capacidade de ler dados de diversas fontes.

• Questão 4: as soluções propostas atendem aos requisitos do Siconfi e Matriz de Saldos Contábeis (MSC) da STN?

Um dos trabalhos investigados apresentou solução para a entrega da MSC da STN, o trabalho de (BRAGANÇA *et al*., 2019), porém limitado a leitura dos dados apenas no formato CSV e utilização de softwares ETL de terceiros para gerar a instância XBRL.

• Questão 5: É possível construir uma ferramenta com a qual qualquer taxonomia possa ser utilizada para gerar instâncias XBRL de qualquer fonte de dados?

Este artigo mostrou que é possível construir uma ferramenta para gerar instâncias XBRL a partir de qualquer taxonomia com dados provenientes de variadas fontes de dados.

Também, a hipótese formulada neste artigo é válida, a qual verificou-se que:

• Hipótese 1: É possível construir uma ferramenta de extração, transformação e carga de dados no formato XBRL, desacoplada de qualquer taxonomia e de diferentes modelos de dados e capaz de adaptar-se à taxonomia e fonte de dados necessárias.

Foi apresentado a ferramenta XBRL Processor para a geração de instâncias XBRL com

32 https://arelle.org/arelle/

dados de fontes relacionais ou denormalizadas. A ferramenta possui código fonte aberto e está parametrizada com a taxonomia do Siconfi para gerar a MSC, no entanto, a parametrização pode ser atualizada ou configurada para outra taxonomia XBRL.

Por fim, foi exposto por meio de um estudo de caso que a ferramenta está apta a atender a necessidade dos estados e municípios brasileiros na entrega da MSC para o Siconfi, atendendo assim a determinação legal do governo federal.

6.1. CONTRIBUIÇÕES

A principal contribuição deste trabalho é a ferramenta proposta para a extração, transformação e carga de dados provenientes de diferentes fontes e formatos de dados para gerar instâncias XBRL. Adicionalmente, teve-se como contribuição o desenvolvimento da ferramenta adaptável a diferentes taxonomias XBRL. Esta ferramenta foi desenvolvida com o foco na adaptabilidade do código aos diferentes ambientes tecnológicos, sendo capaz de extrair dados de variados modelos distintos de tecnologias de armazenamento de dados, simplificando a geração de instâncias XBRL.

Desta forma a ferramenta XBRL Processor, com seu código fonte aberto, consegue atender a necessidade de gerar instâncias XBRL da MSC para o Siconfi e permite ser estendida para solucionar outras necessidades de geração de instâncias XBRL, até mesmo ser incorporado a aplicações em desenvolvimento ou nortear o desenvolvimento de soluções próprias para ambientes legados.

6.2. LIMITAÇÕES E TRABALHOS FUTUROS

Apesar dos benefícios alcançados por este trabalho, algumas limitações foram identificadas e que precisam ser superadas em trabalhos futuros:

• Embora a ferramenta facilite a adição de novas fontes de dados, não está disponível o carregamento de arquivos CSV como fonte de dados, no entanto, já está em fase de pesquisa a adaptação da ferramenta Motor XBRL (BRAGANÇA *et al*., 2019) para ser incorporada na XBRL Processor;

• Validar os dados utilizados para gerar a instância XBRL confrontando com as especificações contidas na taxonomia, extrapola o escopo abordado neste trabalho. Contudo, é possível que a ferramenta XBRL Processor evolua para superar essa limitação;

• Outra limitação identificada é a presença de elementos da taxonomia diretamente no código, sendo necessário, para usufruir de outras taxonomias ou atualizações, a edição do código fonte;

• Outra limitação refere-se à geração das instâncias XBRL apenas no formato XML, o consórcio responsável pela tecnologia XBRL admite também instâncias nos formatos JSON e CSV.

Como investigação futura, sugere-se adicionar a ferramenta outras funcionalidades:

• Desenvolver interface gráfica do utilizador (GUI) juntamente com testes de usabilidade (AKRAM; AQEEL; HAMID, 2023) e *User Experience* de modo a permitir a inclusão de utilizadores sem conhecimento de desenvolvimento de software;

• Desenvolver módulo de verificação e validação dos dados carregados das fontes de dados, frente a taxonomia selecionada;

• Gerar instâncias nos formatos XBRL-JSON e XBRL-CSV, admitidos pelo consórcio responsável pela tecnologia XBRL além do formato XBRL-XML;

• Adequar a biblioteca fast-xml-parser para o carregamento da taxonomia através de seus arquivos, eliminando a presença de elementos da taxonomia no código fonte, conforme Figura 11, possibilitando a troca e atualização facilitada, com a finalidade de abranger um maior número de organizações capazes de usufruir da tecnologia XBRL no intercâmbio de informações contábeis e financeiras;

REFERÊNCIAS BIBLIOGRÁFICAS

AKRAM, S.; AQEEL, M.; HAMID, K. Enhancing Software Quality Through Usability Experience and Hci Enhancing Software Quality Through Usability Experience and Hci Design Principles. n. February, p. 45–75, 2023.

ALAMI, A. El; BAHAJ, M. Framework for a complete migration of relational databases to other types of databases(object oriented OO, object-relational OR, XML). Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA, 2017.

ASIMADI, E.; REIFF-MARGANIEC, S.; DONNELLY, B.; BAKER, J.; FANG, D. Semantic approach to financial data integration for enabling new insights. CEUR Workshop Proceedings, v. 1890, p. 1–15, 2017.

BAI, L.; YAN, L.; MA, Z. M.; XU, C. Incorporating fuzziness in spatiotemporal XML and transforming fuzzy spatiotemporal data from XML to relational databases. Applied Intelligence, v. 43, n. 4, p. 707–721, 1 dez. 2015.

BEELITZ, C. The dilemma of XBRL-XML versus XBRL-JSON regarding linkage of financial information. CEUR Workshop Proceedings, v. 1890, p. 1–11, 2017.

BIKAKIS, N.; TSINARAKI, C.; STAVRAKANTONAKIS, I.; GIOLDASIS, N.; CHRISTODOULAKIS, S. The SPARQL2XQuery interoperability framework: Utilizing Schema Mapping, Schema Transformation and Query Translation to Integrate XML and the Semantic Web. World Wide Web, v. 18, n. 2, p. 403–490, 1 mar. 2015.

BIN, Y. U.; XU, L. U.; CONG, T.; ZHEN-HUA, D.; NAN, Z. Parallel Runtime Verification for Calling Sequences of SQLite3 Database APIs. Journal of Software, v. 33, n. 8, p. 2755–2768, 6 ago. 2022. Disponível em: <http://josen/article/abstract/6596>.

BOGNER, J.; MERKEL, M. To Type or Not to Type? A Systematic Comparison of the Software Quality of JavaScript and Typescript Applications on GitHub. Em: Proceedings of the 19th International Conference on Mining Software Repositories, 2022, New York, NY, USA. [...].

New York, NY, USA: Association for Computing Machinery, 2022. p. 658–669.

BRAGANCA, H. A.; CAETANO, P.; BERNADINO, N. Data Mapping for XBRL : A Systematic Literature Review. American Academic Scientific Research Journal for Engineering, Technology, and Sciences, v. 90, p. 124–143, 2022. Disponível em: <http://asrjetsjournal.org/>.

BRAGANÇA, H. A.; LADISLAU, S. P.; DA SILVA, M. A. P.; DA SILVA, P. C. XBRL-ETL ENGINE: A DATA TRANSFORMATION TOOL FOR XBRL-SICONFI TAXONOMY Motor XBRL-ETL: Uma ferramenta para transformação de dados baseada na taxonomia XBRL-SICONFI. n. 1, p. 1–19, 2019.

CERQUEIRA, M. G. De; SILVA, P. C. Da. A survey of XBRL adoption impact on financial software development processes and software quality. International Journal of Business Information Systems, v. 37, n. 2, p. 263–286, 2021.

CERQUEIRA, M. G.; SILVA, P. C. da. Coming Impacts of Xbrl Adoption in Financial Software Development Processes and Software Quality Factors: a Systematic Mapping. Proceedings of the 13th CONTECSI International Conference on Information Systems and Technology Management, v. 13, p. 3185–3209, 2016.

CHEN, Y. Worst case optimal joins on relational and XML data. Em: Proceedings of the ACM SIGMOD International Conference on Management of Data, 2018, New York, NY, USA. [...]. New York, NY, USA: Association for Computing Machinery, 2018. p. 1833–1835.

CHRISTUDAS, B. MySQL. Em: Practical Microservices Architectural Patterns: Event-Based Java Microservices with Spring Boot and Spring Cloud. Berkeley, CA: Apress, 2019. p. 877–884.

DIMOU, A.; SANDE, M. Vander; COLPAERT, P.; VERBORGH, R.; MANNENS, E.; VAN DE WALLE, R. RML: A generic language for integrated RDF mappings of heterogeneous data. CEUR Workshop Proceedings, v. 1184, 2014.

DOI, Y.; TOYAMA, M. ToT for CSV: Accessing Open Data CSV Files through SQL. Em:

Proceedings of the 21st International Conference on Information Integration and Web-Based Applications & Services, 2019, New York, NY, USA. [...]. New York, NY, USA: Association for Computing Machinery, 2019. p. 423–429.

DUNCE, M. M. M.; SILVA, P. C. da; VIANA, S. Similarity Evaluation Between Concepts Represented By Xbrl. p. 3933–3963, 2013.

GRAY, G. L.; MILLER, D. W. XBRL: Solving real-world problems. International Journal of Disclosure and Governance, v. 6, n. 3, p. 207–223, ago. 2009.

GYŐRÖDI, C. A.; DUMŞE-BURESCU, D. V.; ZMARANDA, D. R.; GYŐRÖDI, R. A Comparative Study of MongoDB and Document-Based MySQL for Big Data Application Data Management. Big Data and Cognitive Computing, v. 6, n. 2, 2022.

JAYASHREE, G.; PRIYA, C. Data Integration with XML ETL Processing. 2020 International Conference on Computer Science, Engineering and Applications, ICCSEA 2020, n. March, 2020.

LAZZARI, L.; FARIAS, K. An exploratory study on the effects of pair programming. [s.l.] Association for Computing Machinery, 2022. v. 121–28 p.

LIU, D.; ETUDO, U.; YOON, V. X-IM framework to overcome semantic heterogeneity across XBRL filings. Journal of the Association for Information Systems, v. 21, n. 4, p. 971–1000, 2020.

LYAMIN, A. V.; CHEREPOVSKAYA, E. N. XML-Relational mapping using production rule system. 2017 Intelligent Systems Conference, IntelliSys 2017, v. 2018-Janua, n. September, p. 422–429, 2018.

MAATUK, A. M.; ALI, M. A.; ALJAWARNEH, S. An algorithm for constructing XML Schema documents from relational databases. Em: ACM International Conference Proceeding Series, 2015, New York, NY, USA. [...]. New York, NY, USA: Association for Computing Machinery, 2015. v. 24-26- Sept

NASSIRI, H.; MACHKOUR, M.; HACHIMI, M. Integrating XML and Relational Data. Procedia Computer Science, v. 110, p. 422–427, 2017.

NASSIRI, H.; MACHKOUR, M.; HACHIMI, M. One query to retrieve XML and Relational Data. Procedia Computer Science, v. 134, p. 340–345, 2018.

NAVATHE, Elmasri. &. Sistemas de Banco de Dados. [s.l: s.n.]v. 6ed1689–1699 p.

NIEWERTH, M.; SCHWENTICK, T. Reasoning About XML Constraints Based on XML-to--Relational Mappings. Theory of Computing Systems, v. 62, n. 8, p. 1826–1879, 1 nov. 2018.

PETKOVIĆ, D. JSON Integration in Relational Database Systems. International Journal of Computer Applications, v. 168, n. 5, p. 14–19, 2017a.

PETKOVIĆ, D. SQL/JSON Standard: Properties and Deficiencies. Datenbank-Spektrum, v. 17, n. 3, p. 277–287, 2017b.

QTAISH, A.; AHMAD, K. XAncestor: An efficient mapping approach for storing and querying XML documents in relational database using path-based technique. Knowledge-Based Systems, v. 114, p. 167–192, 2016.

SALEM, R.; DARMONT, J.; BOUSSAID, O.; SALEM, R.; DARMONT, J.; BOUSSAID, O.; WEB, A. X.; SALEM, R.; BOUSSA, O. Active XML-based Web data integration To cite this version : HAL Id : hal-01433718 Active XML-based Web Data Integration. v. 15, n. 3, 2017.

SILVA, P. C.; SILVA, L.; SANTOS, A.; CRUZ, M. O Framework Xbrl. International Conference on Information Systems and Technology Management 5th, p. 4343–4365, 2008.

SOARES, B. E.; BOSCARIOLI, C. Modelo de Banco de Dados Colunar: Características, Aplicações e Exemplos de Sistemas. Escola Regional de Banco de Dados–Sociedade Brasileira de Computação (IX ERBD–SBC), 2013.

SOMMERVILLE, I. Software engineering. 9th. ed. Boston: Addison-Wesley, 2011. 773 p.

SONG, E.; HAW, S. C. XML-REG: Transforming xml into relational using hybrid-based mapping approach. IEEE Access, v. 8, p. 177623–177639, 2020.

SONG, E.; HAW, S. C.; CHUA, F. F. Handling XML to relational database transformation using model-based mapping approaches. 2018 IEEE Conference on Open Systems, ICOS 2018, p. 65–70, 2019a.

SONG, E.; HAW, S. C.; CHUA, F. F. Handling XML to relational database transformation using model-based mapping approaches. Em: 2018 IEEE Conference on Open Systems, ICOS 2018, 2019b, [...]. 2019. p. 65–70.

STN, S. do T. N. Matriz de Saldos Contábeis. Secretaria Do Tesouro Nacional, v. 1, 2017. Disponível em: <https://siconfi.tesouro.gov.br/siconfi/pages/public/arquivo/conteudo/Cartilha_Matriz_de_Saldos_Contabeis.pdf>.

YAGHMAZADEH, N.; WANG, X.; DILLIG, I. Automated Migration of Hierarchical Data to Relational Tables Using Programming-by-Example. Proc. VLDB Endow., v. 11, n. 5, p. 580–593, jan. 2018.

ZHU, H.; YU, H.; FAN, G.; SUN, H. Mini-XML: An efficient mapping approach between XML and relational database. Em: Proceedings - 16th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2017, 2017, [...]. 2017. p. 839–843.