Revista

Cadernos de Finanças Públicas

03 | 2024

TESOURO NACIONAL

XBRL PROCESSOR: A TOOL FOR GENERATING XBRL INSTANCES

Henderson Acosta Bragança

Paulo Caetano da Silva

UNIFACS - Universidade Salvador - Brasil

Silas Pinho Ladislau

Daniel José Díaz

Universidade Nacional de Rosário - Argentina

ABSTRACT

The use of *eXtensible Business Reporting Language* (XBRL) technology in the context of financial reporting on the internet is evident, whether due to its advantages and benefits or governmental impositions. However, the data to be transported by this language is mostly stored in defined structures such as relational databases, JSON files or CSV files. It is therefore essential for organizations to integrate XBRL technology with other data storage technologies. This article presents an *Extract, Transform and Load* (ETL) solution for extracting data from different storage formats and generating XBRL instances, called XBRL Processor. This tool includes different types of data source and generates the XBRL instance. In addition, parameterizations were implemented to meet the delivery of the Accounting Balances Matrix to the Accounting and Financial Information System of Brazil's National Treasury Secretariat (SICONFI) and a case study was carried out in order to validate the XBRL Processor tool.

Keywords: XBRL Processor, XBRL, XBRL integration, Data integration, Data mapping
JEL: C80, C88, H83, O30

SUMMARY

1. INTRODUCTION

Transporting financial and accounting data over the internet while maintaining its structural and semantic integrity has been achieved using *eXtensible Business Reporting Language* (XBRL[1]) technology. XBRL is a technology derived from the eXtensible Markup Language (XML), which is considered by the *World Wide Web Consortium* (W3C[2]) to be an important means of exchanging data on the Internet, since it enables platform-independent data exchange (JAYASHREE; PRIYA, 2020). The design of XML technology favors simplicity, generality and usability (ZHU *et al*., 2017), these attributes favor the use of XML technology on the internet and as a means of data integration and exchange. Several alternatives are proposed in the literature for integrating and mapping data between XML documents, which are used as a means of transporting data on the internet, and different databases (SONG; HAW; CHUA, 2019).

There are some solutions for mapping XBRL data, but they add computational costs, e.g. proposals from Fujitsu[3] and Oracle[4] , because it is necessary to create parallel databases with the data from the XML documents, as well as having to store the XML document to safeguard the integrity of the information. Furthermore, these solutions are proprietary and very expensive. In other cases, specific applications have to be developed to mediate data exchange between different environments, i.e. XML, relational databases, CSV[5] , NoSQL[6] and JSON[7] .

XBRL technology was developed for the exchange of financial information on the internet, and has been consolidated as the standard to be used by government bodies in various countries (DUNCE; SILVA; VIANA, 2013). This technology has the capacity to transport financial data along with its semantics, which is essential for a correct analysis of the information. By transporting the data along with its semantics, there is no need to find out what it is. Meaning of the information provided, when it is not accompanied by its context, i.e. without semantics (BRAGANÇA *et al*., 2019).

XBRL is a technology that has become an international standard for the exchange of financial data, e.g. US-SEC[8] , European Committee of Banking Supervisors (CEBS)[9] , Bank

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
6 Not Only SQL
7 JSON stands for Java Script Object Notation and is a standard for exchanging data between systems.
8 https://www.sec.gov/
9 https://www.europeansources.info/record/committee-of-european-banking-supervisors-cebs/

of Spain[10] , Bank of Japan[11], are just a few examples of government bodies that have adopted XBRL for the exchange of data with their supervised organizations[12]. In Brazil, in the face of regulatory impositions, such as Law 12.527/11, which provides for access to information from States, Municipalities and the Federal District, popularly known as the Transparency Law, in conjunction with Standard 896/17 of the National Treasury Secretariat[13], government bodies are being directed to use XBRL in their publications related to accounting statements for control bodies.

Both governments at their different levels (federal, state and municipal) and private organizations do not have free and *open source* tools that make it possible to export financial and accounting information into XBRL format, especially when extracted directly from relational or NoSQL databases, creating a difficulty for private organizations and governments to represent their financial and fiscal data in XBRL.

There are different technologies for storing data, based on different models, the most common being relational and, more recently, NoSQL, which is commonly used on the internet. The relational data model represents the database as a collection of relationships (NAVATHE, 2013), which after its conception in the 1960s, came to be used by most software applications, and is a model that should continue to be used for many years due to its robustness (NAVATHE, 2013). Due to its importance and widespread use in private and government applications, it is essential to consider this model as a data source. Also the approach related to CSV files as a source data is pertinent, since there are scenarios[14] in which data is only exported in CSV files (BRAGANÇA *et al*., 2019).

NoSQL databases work with denormalized data (SOARES; BOSCARIOLI, 2013). NoSQL databases are being adopted by large companies, e.g. Google, Facebook and Twitter (SOARES; BOSCARIOLI, 2013), since this model has an architecture that makes it easier to handle the significant demand for queries, linked to the large amount of information, with high data scalability (SOARES; BOSCARIOLI, 2013). Consequently, NoSQL databases are also considered as an alternative data source in the solution proposed in this work.

The National Treasury Secretariat (STN), through the Brazilian Public Sector Accounting

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 Most of the states' Financial Administration Integration Systems (SIAF) are unable to generate the XBRL instance because they use the Adabas/SoftwareAG DBMS, but they do have the capacity to generate CSV files.

and Fiscal Information System (Siconfi)[15], has defined XBRL technology as the main data format for the Accounting Balance Matrix, in order to improve the quality of information in the Brazilian Public Sector (STN, 2017). The Matrix of Accounting Balances (MSC) is a structure designed to contain the details of the federative entity's accounting information[16] . The STN recommends direct extraction from the data source to minimize errors in filling out spreadsheets or forms (STN, 2017).

The purpose of this work, therefore, is to develop an *open source* solution that meets the most diverse data source scenarios, exporting the information relating to the MSC to XBRL instances, meeting the STN's requirement through Siconfi. Following on from this introduction, Section 2 discusses related work. Section 3 motivates the proposed solution. Section 4 describes the conceptual approach and the development of the data extraction, transformation and loading (ETL) tool for XBRL technology, called XBRL Processor. Section 6 sets out the conclusions of this work, its contributions, limitations and proposals for future work.

2. RELATED WORK

It is worth noting that the proposals found in the literature focus on solving specific problems, such as complying with legislation or identifying semantic similarity in XBRL instances. That said, it can be seen that the studies are limited to dealing with the issue of reading and displaying data from XBRL instances. Due to the scarcity of papers dealing with the generation of XBRL instances and the fact that the consortium responsible for XBRL accepts XML, JSON and CSV files as technologies for transporting data, we decided to investigate mappings related to XML, JSON and CSV in the search for papers that would satisfy the investigation into solutions for integrating or mapping data to XBRL and vice versa. From this perspective, we identified the works (JAYASHREE; PRIYA, 2020), (ZHU *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,

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

16 A federative entity can also be understood as a state, being an autonomous unit (self-government, self--legislation and self-taxation) endowed with its own government, constitution and which, with other states, forms a federation.

2017b), (YAGHMAZADEH; WANG; DILLIG, 2018) and (DOI; TOYAMA, 2019) which deal with the mapping data from XML or JSON instances and CSV files to relational databases.

Only one proposal presented a solution to generate XBRL instances, specifically to meet a legal requirement (BRAGANÇA *et al*., 2019), this work uses only CSV files as a data source and the Hitachi Pentaho proprietary ETL system[17]. When it comes to the JSON language, along with CSV files, no solutions were found that aimed to instantiate the data along the lines defined by the XBRL-JSON[18] or XBRL-CSV[19] specifications of the XBRL consortium. As such, the proposal for a tool capable of connecting the most diverse data sources, linking the data to the chosen taxonomy and generating the XBRL instance, guided the development of this work.

3. MOTIVATION

For information to be contextualized in a given domain, it must be interconnected with other information, whether contextual, past information, experiences or future information (CERQUEIRA; SILVA, 2021) and (BEELITZ, 2017). In the financial and accounting domain, the technology that meets these requirements is XBRL.

In 2008 Silva stated that XBRL technology was becoming a technological standard for the exchange, storage and disclosure of financial information on the Internet (SILVA *et al*., 2008). In fact, this prediction has been confirmed, and the adoption of the technology by relevant institutions has boosted the use of XBRL, as is the case with the US-SEC[20], which pointed to XBRL technology as the key to its modernization (GRAY; MILLER, 2009).

Because XBRL technology is robust and fully represents the financial and accounting domain, it brings with it the complexity inherent in solutions aimed at complex problems, such as the transportation of financial and accounting information with all its contextual framework.

Generating XBRL instances using financial and accounting data repositories as a source is not trivial (ASIMADI *et al*., 2017), only proprietary tools (e.g. Amelkis, aSISt, Vizor)[21] available on the market are capable of automatically extracting data stored in different repositories using various storage technologies (e.g. relational, NoSQL, CSV). Even so, these tools usually handle data from a specific model, i.e. they don't have enough generality to handle different types of data formats and XBRL. In addition, the lack of *open source* tools may hinder the mass

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

adoption of this technology, especially in Brazil, where the only project in operation is that of the National Treasury Secretariat (STN), called Siconfi. Siconfi requires federated entities to submit accounting and financial information based on XBRL technology. As there are no free *open source* solutions, the alternative for Brazilian federation entities is to use proprietary systems. Meeting the STN's requirements with regard to delivery of the MSC in XBRL instance allows the entity to comply with the Fiscal Responsibility Law to which the entire Brazilian public sector is subject.

In order to comply with the legislation, federal units have to enter accounting and financial data into proprietary systems, often manually, in order to generate the XBRL document, since the connection to the data source, e.g. relational databases, is not simple. This generates human and computer resource costs due to the low interoperability between XBRL environments and data sources, which implies additional financial costs. It is possible to send the MSC data in CSV format to Siconfi, but even the CSV format adds additional human and computer resource costs. The CSV format continues to be accepted by Siconfi because there are Brazilian federated entities that do not have tools that make it possible to deliver MSC data in the XBRL language.

Bragança's work (BRAGANÇA *et al*., 2019) shows that Brazil's executive powers, whether at the federal, state or municipal level, lack solutions that meet the demand for generating XBRL instances to be delivered to Siconfi. In order to meet this demand, the aim of this work is to develop an *open source* ETL solution divided into two execution stages to facilitate the development of new functionalities. The first stage is designed to connect to the data source and the second to link the data to Siconfi's[22] taxonomy, making it possible to generate the XBRL instance. This article presents a tool that seeks to meet the most diverse financial data exchange scenarios, with reusable configurations and a reduction in the complexity of the environment. The solution was designed to facilitate maintenance, extension and integration with existing accounting and financial software, to generate instances of the most varied taxonomies.

Thus, the central research problem guiding the development of this work is:

• Question 1: Are there solutions that allow data from different formats to be mapped to XBRL?

From this main research question (Question 1), other research questions could be inferred:

• Question 2: Are the existing solutions open source, i.e. free and open source?

22 The XBRL taxonomy is a set of documents based on XML Schema and Xlink technologies that allows the definition of financial and accounting concepts and their relationships.

• Question 3: Do the proposed solutions have sufficient generalization to allow the mapping of different data models (e.g. relational, NoSQL, CSV, JSON) to XBRL?

• Question 4: Do the proposed solutions meet the requirements of Siconfi and the STN's Matrix of Accounting Balances (MSC)?

• Question 5: Is it possible to build a tool with which any taxonomy can be used to generate XBRL instances from any data source?

For the research problem investigated, the following hypothesis was formulated:

• Hypothesis 1: It is possible to build a tool for extracting, transforming and loading data from any data model, whether relational DBMS, NoSQL or CSV files, and export them as XBRL instances, capable of adapting to the needs related to the data source and user taxonomies.

4. XBRL PROCESSOR TOOL

Open source and free initiatives to make up for the lack of solutions capable of generating XBRL instances can be seen in Bragança's work (BRAGANÇA *et al*., 2019), proposing an ETL tool, however, the proposed solution only collects data from CSV files, restricting the solution's adoption capacity. It can also be seen that there is a lack of solutions in Brazil's executive powers, whether at federal, state or municipal level, that meet the demand for generating XBRL instances to be delivered to Siconfi (BRAGANÇA *et al*., 2019).

The systematic review of the literature presented in Bragança (BRAGANCA; CAETANO; BERNADINO, 2022) identified that there is a primary concern with developing software that is capable of reading and displaying the reports contained in XBRL instances, however, there is not the same concern in generating XBRL instances (BRAGANCA; CAETANO; BERNADINO, 2022). With this in mind, this paper developed a tool to generate XBRL instances in order to fill this gap.

This section describes the development of the tool for reading, transforming and loading data (ETL) for XBRL technology, called XBRL Processor. The main objective of developing this solution is to enable data to be extracted from different storage systems (relational DBMS, NoSQL, CSV) and loaded into XBRL instances, defined by a taxonomy. A number of criteria were used to develop the tool:

• The use of the MSC (Matrix of Accounting Balances) in the XBRL format and taxonomy defined by Siconfi (Brazilian Public Sector Accounting and Tax Information System) so that the application can be validated and ready to solve the problem of delivering the MSC in XBRL format to Siconfi/STN, a requirement made of Brazil's federated entities;

• The development of the proposed tool was based on the TypeScript language (BOGNER; MERKEL, 2022) due to the ease of understanding the code, its execution performance, the availability of a free and editable library for reading and generating XML instances;

• Finally, the MySQL DBMS was used as a data source (CHRISTUDAS, 2019), Sqlite3 (BIN *et al*., 2022) and MongoDB (GYŐRÖDI *et al*., 2022), with which it was possible to check the tool's connection to relational and NoSQL DBMSs, following the principle of using data sources in different formats.

The development of the XBRL Processor tool consisted of three stages:

• In the first stage, the ETL tool for XBRL was built based on the XBRL taxonomy, version 2022, defined for the Siconfi Accounting Balance Matrix. The data source used was the Sqlite3 DBMS with fictitious and anonymized, yet coherent, data provided by the Rondônia State Finance Department (SEFIN - RO);

• In the second stage, other DBMSs were incorporated into the tool's code design. In this way, it was possible to check the adaptability of the code to different formats and data sources. Connectors were added for MySQL and MongoDB DBMSs;

• In the third stage, the XBRL instance generated by the XBRL Processor was compared with the one provided by the Rondônia State Finance Department. Finally, the XBRL instance generated with the fictitious data was validated by the STN in Siconfi and by the Rondônia State Finance Department in the test and production environments respectively.

Section 4.1 then presents the architecture designed for the proposed tool so that it can be adapted to include new databases and data formats. Section 4.2 shows the development of the data extraction, transformation and loading (ETL) tool for XBRL technology, called XBRL Processor.

4.1 XBRL PROCESSOR ARCHITECTURE

The proposed architecture, shown in Figure 1, aims to connect different databases, e.g. relational, NoSQL, and make it possible to join the data extracted from these databases with the taxonomy to be used, generating the XBRL instance.

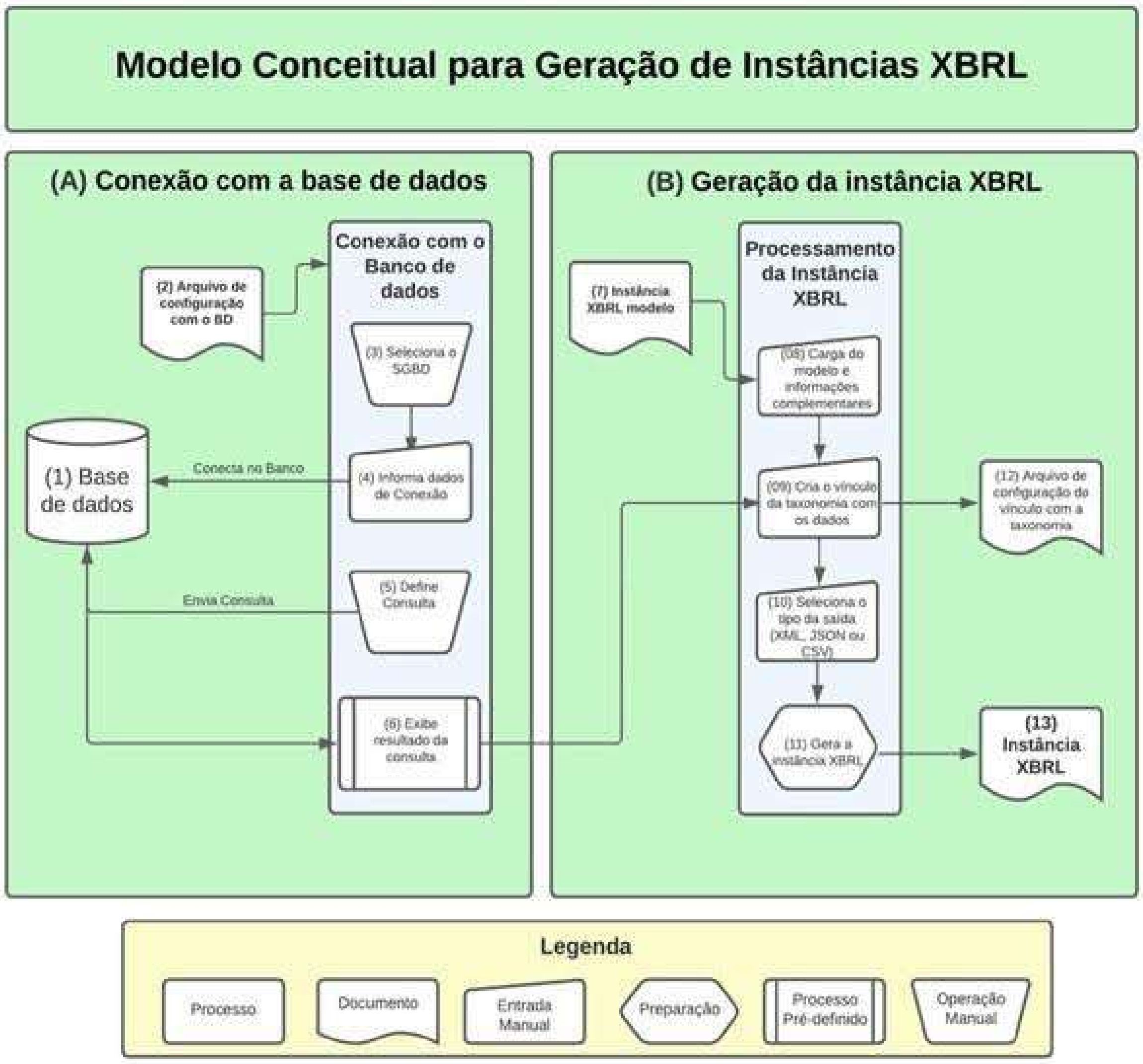
The proposed process for generating XBRL instances was divided into two stages: (A) connecting to the database and (B) generating the XBRL instance from the taxonomy provided.

Dividing the solution into two stages will allow for better maintenance of the code and the development of possible improvements and new functionalities, so that each stage has its own functional context, increasing cohesion and reducing coupling. In software, modularization allows a change in one component to have minimal impact on others (LAZZARI; FARIAS, 2022).

The process of connecting to the database starts with choosing the DBMS, as we can see in Figure 1(A-3), the list of supported DBMSs will be limited only by the available *drivers*, however, any *driver* can be added, e.g. driver for PostgreSQL, Oracle, MySQL, SQL Server, MongoDB, or any other driver required for extracting data to generate the XBRL instance.

Following Figure 1(A-4), the connection data is requested in order to allow connection to the DBMS. Data such as TCP port, login and password are essential information for the connection. When the connection data is entered, a connection test is run.

Figure 1 - Architecture for generating XBRL instances


Modelo Conceitual para Geração de Instâncias XBRL
(A) Conexão com a base de dados
Conexão com o Banco de dados
(2) Arquivo de configuração com o BD
(3) Seleciona o SGBD
(1) Base de dados
Conecta no Banco
(4) Informa dados de Conexão
Envia Consulta
(5) Define Consulta
(6) Exibe resultado da consulta
(B) Geração da instância XBRL
Processamento da Instância XBRL
(7) Instância XBRL modelo
(08) Carga do modelo e informações complementares
(09) Cria o vínculo da taxonomia com os dados
(12) Arquivo de configuração do vínculo com a taxonomia
(10) Seleciona o tipo da saída (XML, JSON ou CSV)
(11) Gera a instância XBRL
(13) Instância XBRL
Legenda
Processo
Documento
Entrada Manual
Preparação
Processo Pré-definido
Operação Manual

Once the connection process is complete, insertion of the query code Figure 1(A-5) begins, e.g. SQL or the code pertinent to the DBMS selected in the previous activities. The possibility of inserting the code directly into the application allows for flexibility in querying the data in order to facilitate adjustments and the addition of columns with static data, e.g. the information related to financial values being in Brazilian currency (BRL). After executing the code, you can see in Figure 1(A-6) that the next activity is to display the result of the query for checking.

The connection to the database was planned so that all the data and parameterizations can be reused in a subsequent run, so the file with the configurations is reused in subsequent runs Figure 1(A-2).

The process of linking the taxonomy to the data is not an ordinary process, (ASIMADI *et al*., 2017) pointed out in his work that the XBRL integration process remains complex, so the start of XBRL instance processing, Figure 1(B-7), brings the possibility of using a template XBRL instance to retrieve as many links as possible between the taxonomy and the data. The template instance can be provided by the creator of the taxonomy to which the data will be sent.

In the next step, complementary data can be added Figure 1(B-8), complementary information is understood to be information that is exclusive to the organization that is generating the XBRL instance, e.g. the organization's registration code with the body that will receive the data.

After inserting the model taxonomy and complementary information, the data received from the database is linked to the taxonomy, Figure 1(B-9), information that is used repeatedly can be preserved, if necessary for later retrieval, in the relevant configuration file, Figure 1(B-12) (DIMOU *et al*., 2014).

In 2021, the XBRL consortium defined new ways of using the technology, defining as a recommendation, in addition to XBRL extended from XML, XBRL extended from JSON and CSV files (CERQUEIRA; SILVA, 2016). XBRL in JSON and CSV are limited to the instance and the file containing the taxonomy is not allowed in these formats. Also, the consortium responsible for XBRL has not clarified what the XBRL-JSON and XBRL-CSV instances will look like when generated from the GL taxonomy[23]. However, in order to be in line with the recommendations of the XBRL consortium, it is necessary to allow the possibility of generating the instance in the three formats accepted by the XBRL consortium. This provision exists at this stage of the process, as can be seen in Figure 1(B-10).

Finally, the data is processed to generate the XBRL instance, Figure 1(B-11), with the

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

product being the instance with all the data and its links to the taxonomy, as well as the complementary unit data and references via the links, Figure 1(B-13).

4.2. IMPLEMENTATION

The XBRL Processor tool was developed using the architecture proposed in Section 4.1, which proposes a solution to the latent problem of generating XBRL instances from different DBMSs. In this way, coding was carried out to allow external validation of the instances generated by the application.

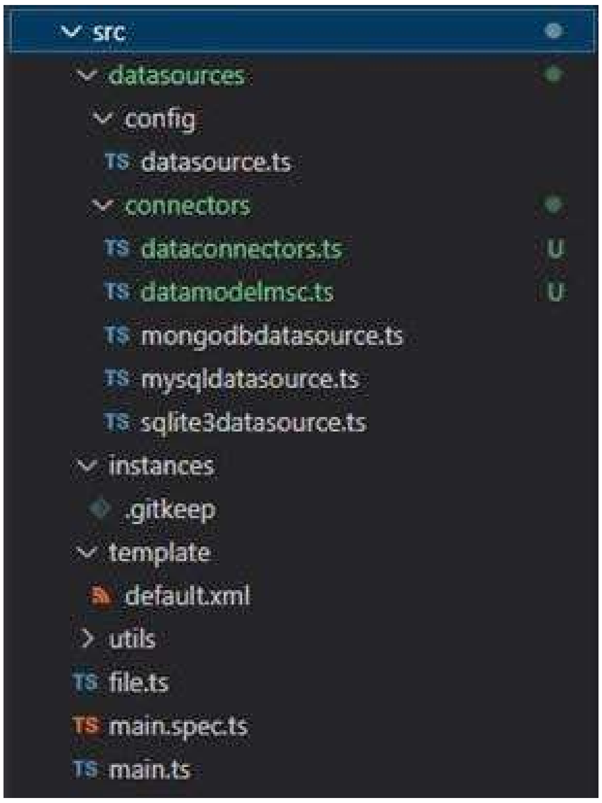
4.2.1. IMPLEMENTATION ASPECTS

The source code was organized with a focus on clean architecture, in order to facilitate understanding of the code, resources and functions (SOMMERVILLE, 2011). Clean architecture allows the application to evolve without consuming excessive human resources or time. The fact that the project is separated into independent layers means that changes to one layer do not interfere with the others (SOMMERVILLE, 2011). In this way, it is hoped that the code can be extended or incorporated into financial and accounting applications. The source code for this work can be accessed on the github repository[24]. Following the concept defined in Section 4.1, the code is divided into two components, in which the first component defined in Figure 1(A), which deals with the connection to the data source, can be related to the src/*datasource* folder of the code, being subdivided into src/datasource/config and src/datasource/connectors. The second component shown in Figure 1(B), which deals with generating the XBRL instance, has been integrated into the src/main.ts file. The organization of the code can be seen in Figure 2, which shows the folder structure used to organize the code.

The data source was separated into: the code related to the connection to the DBMS, the data source (DBMS) settings and the MSC structure expected in the query. The files containing the connection-related code, such as TCP ports, login, can be accessed in the src/datasource/config/datasource.ts file. This file is represented conceptually in Figure 1(A-4) and its code wasadapted for the MySQL DBMS, as can be seen in Figure 3.

24 Address hidden due to the determination of STN Notice No. 10 of May 29, 2023, 28th NATIONAL TREASURY AWARD 2023, not identifying the author(s) in any way, including in the file properties.

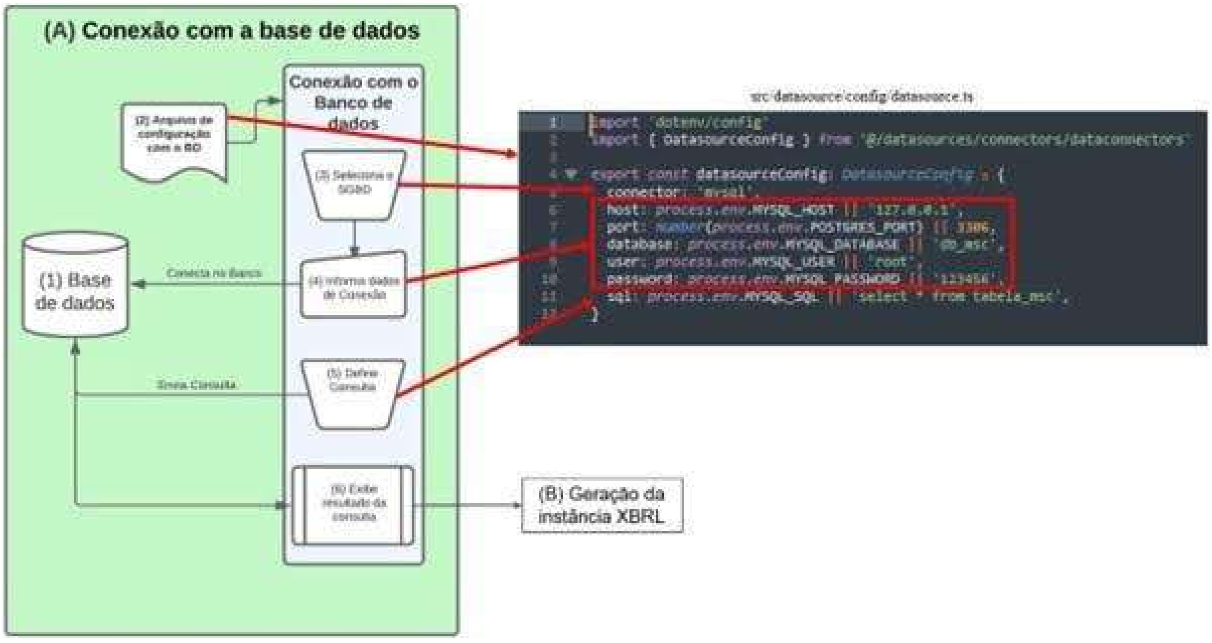
Figure 2 - Folder structure of the code



```
src
  datasources
    config
      datasource.ts
    connectors
      dataconnectors.ts        U
      datamodelmsc.ts          U
      mongodbdatasource.ts
      mysqldatasource.ts
      sqlite3datasource.ts
  instances
    .gitkeep
  template
    default.xml
  utils
  file.ts
  main.spec.ts
  main.ts
```

The file defined in Figure 1 (A-2), referring to the connection data with the database, was not generated separately from the code files, but was kept in the src/datasource/config/datasource.ts file, in order to meet the restriction of reusing the parameterizations by the tool; Figure 3 illustrates this relationship between the code and the architecture.

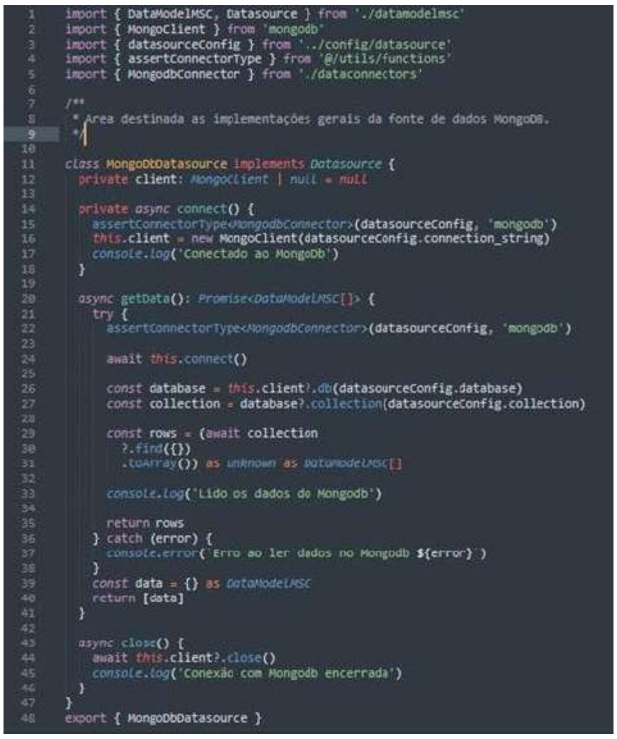
Figure 3 - MySQL DBMS connection data


(A) Conexão com a base de dados
Conexão com o Banco de dados
(2) Arquivos de configuração com o BD
(3) Seleciona o SGBD
(1) Base de dados
Conecta no Banco
(4) Informa dados de Conexão
Envia Consulta
(5) Define Consulta
(6) Exibe resultado da consulta
(B) Geração da instância XBRL
src/datasource/config/datasource.ts

```
import 'dotenv/config'
import { DatasourceConfig } from '@/datasources/connectors/dataconnectors'

export const datasourceConfig: DatasourceConfig = {
  connector: 'mysql',
  host: process.env.MYSQL_HOST || '127.0.0.1',
  port: number(process.env.POSTGRES_PORT) || 3306,
  database: process.env.MYSQL_DATABASE || 'db_msc',
  user: process.env.MYSQL_USER || 'root',
  password: process.env.MYSQL_PASSWORD || '123456',
  sql: process.env.MYSQL_SQL || 'select * from tabela_msc',
}
```

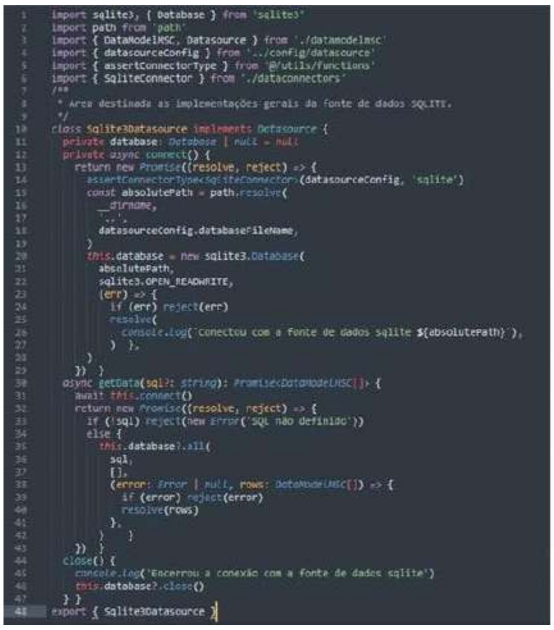
Each data source has a code for reading the data stored in its structure. To accommodate this code, the src/datasources/connectors/ folder was created, where each data source has its own code. The configurations for the SQLite3 (*sqlite3datasource.ts*), MySQL (*mysqldatasource.ts*) and MongoDB (*mongodbdatasource.ts*) data sources are available for use; for other data sources, e.g. PostgreSQL DBMS, new files with the configuration codes will be required. The codes for the available DBMSs can be seen in Figures 4, 5 and 6.

Figure 4 - MongoDB DBMS implementation



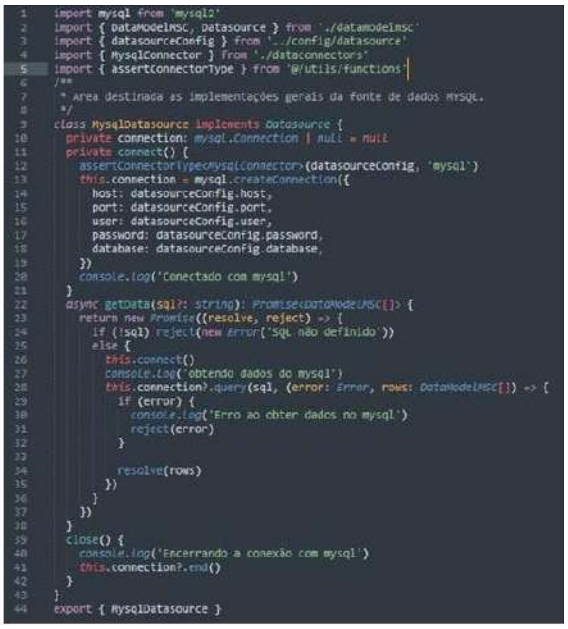
```
import { DataModelMSC, Datasource } from './datamodelmsc'
import { MongoClient } from 'mongodb'
import { datasourceConfig } from '../config/datasource'
import { assertConnectorType } from '@/utils/functions'
import { MongodbConnector } from './dataconnectors'

/**
 * Área 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'
/**
 * Área 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 }
```

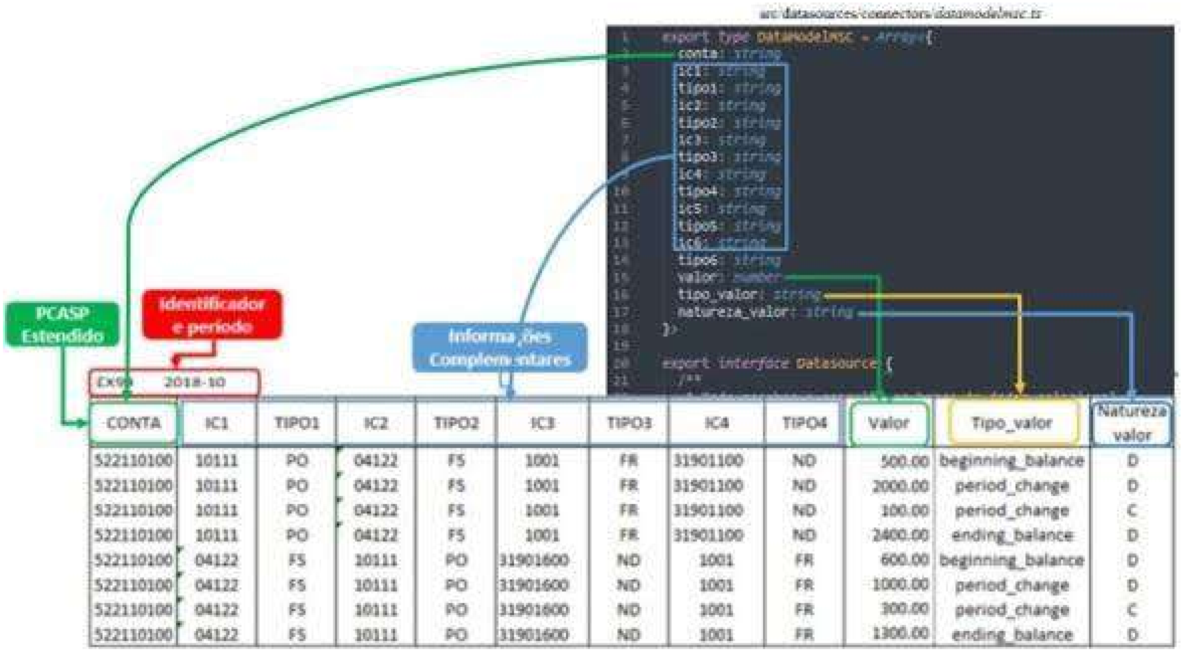
Figura 6 - Implementação do SGBD MySQL



```
import mysql from 'mysql2'
import { DataModelMSC, Datasource } from './datamodelmsc'
import { datasourceConfig } from '../config/datasource'
import { MysqlConnector } from './dataconnectors'
import { assertConnectorType } from '@/utils/functions'
/**
 * Área 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 }
```

The structure proposed by Siconfi for the MSC was maintained, in which the fields follow the definitions of the template[25] defined in the Siconfi manual. The MSC layout was designed to represent the information based on the XBRL standard (STN, 2017). Although the tool does not validate the data against the taxonomy, the query must return the data from the data repository, following the structure determined by Siconfi. You can see the coding of the structure defined by Siconfi in lines 1 to 18 of Figure 7.

Figure 7 - Structure of the MSC defined by Siconfi described in the XBRL Processor


src/datasources/connectors/datamodelmsc.ts
PCASP Estendido
Identificador e período
Informa ções Complem entares
CX99 2018-10

```
1   export type DataModelMSC = Array<{
2       conta: string
3       ic1: string
4       tipo1: string
5       ic2: string
6       tipo2: string
7       ic3: string
8       tipo3: string
9       ic4: string
10      tipo4: string
11      ic5: string
12      tipo5: string
13      ic6: string
14      tipo6: string
15      valor: number
16      tipo_valor: string
17      natureza_valor: string
18  }>
19
20  export interface Datasource {
21      /**
```

CONTA	IC1	TIPO1	IC2	TIPO2	IC3	TIPO3	IC4	TIPO4	Valor	Tipo_valor	Natureza valor
522110100	10111	PO	04122	FS	1001	FR	31901100	ND	500.00	beginning_balance	D
522110100	10111	PO	04122	FS	1001	FR	31901100	ND	2000.00	period_change	D
522110100	10111	PO	04122	FS	1001	FR	31901100	ND	100.00	period_change	C
522110100	10111	PO	04122	FS	1001	FR	31901100	ND	2400.00	ending_balance	D
522110100	04122	FS	10111	PO	31901600	ND	1001	FR	600.00	beginning_balance	D
522110100	04122	FS	10111	PO	31901600	ND	1001	FR	1000.00	period_change	D
522110100	04122	FS	10111	PO	31901600	ND	1001	FR	300.00	period_change	C
522110100	04122	FS	10111	PO	31901600	ND	1001	FR	1300.00	ending_balance	D

The diversity of DBMSs used in the application will make it easier to adapt to any environment. Figure 8 shows the sequence diagram describing the connection to the DBMS in the first module of the XBRL Processor tool. It is hoped that, when applied in a production environment, the DBMS used by the organization will be parameterized in XBRL Processor, eliminating the need to modify the code.

The second module, as described in Figure 1(B), loads the data, loads the instance template, the taxonomy and proceeds to generate the XBRL instance. The instance template contained in the src/template folder contains the header, the unit code provided by Siconfi, the reference to the currency and the periods to which the instance to be generated refers, e.g. in this case to MSC 2023- 01-31.

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

Figure 8 - Sequence diagram for loading data

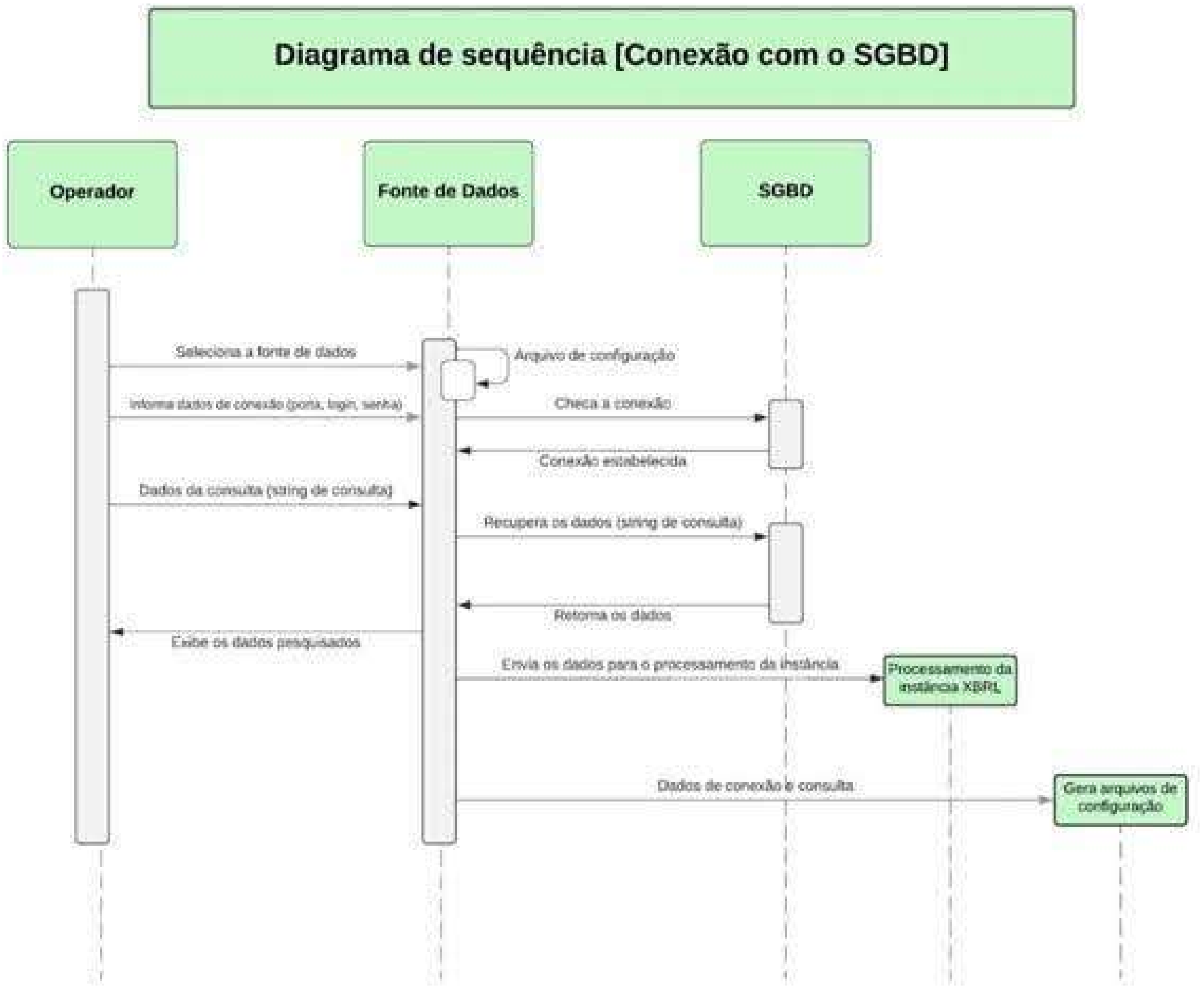


Diagrama de sequência [Conexão com o SGBD]
Operador
Fonte de Dados
SGBD
Seleciona a fonte de dados
Arquivo de configuração
Informa dados de conexão (porta, login, senha)
Checa a conexão
Conexão estabelecida
Dados da consulta (string de consulta)
Recupera os dados (string de consulta)
Retorna os dados
Exibe os dados pesquisados
Envia os dados para o processamento da instância
Processamento da instância XBRL
Dados de conexão e consulta
Gera arquivos de configuração

When necessary, these data must be updated in the template file. The other data (e.g. context, G/L account, etc.) contained in the template instance is replaced by the data retrieved from the data repository. In Figure 9 you can check the data loaded by the application from the template file.

The application's second module is concentrated in the src/main.ts file. We can see the dependencies of the components and their references to the code in Figure 10.

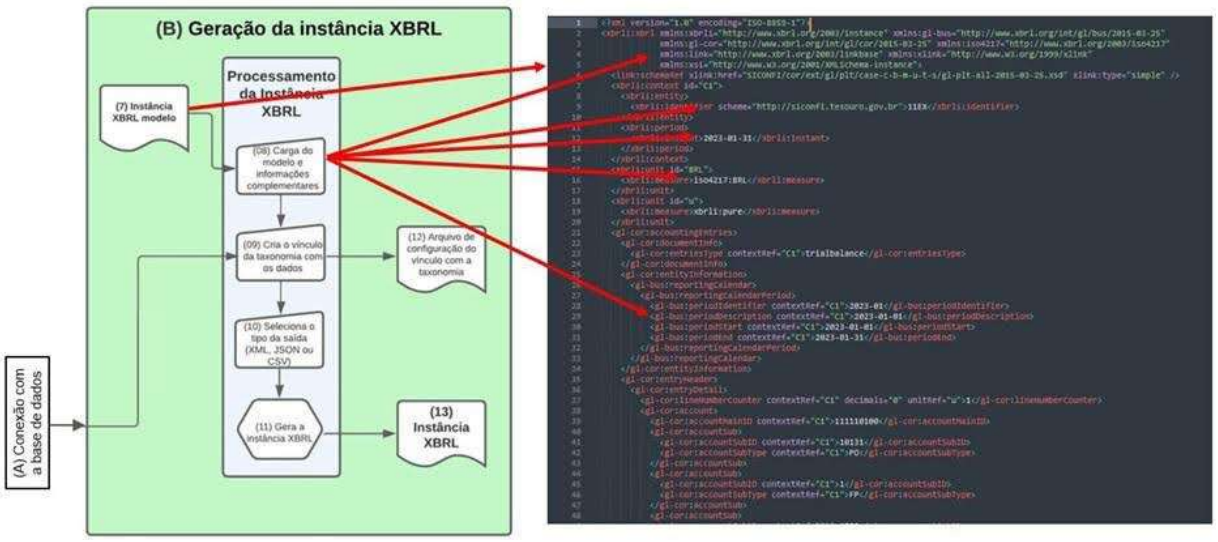
The tool uses the fast-xml-parser library[26] to read and write XML documents. Due to the restrictions of the fast-xml-parser library, part of the taxonomy was added to the code to make it possible to link the data to the taxonomy. The fast-xml-parser library is open and allows the code to evolve, however, editing the library to read the taxonomy without the fact that the elements of the taxonomy are present in the code has not been included in this version of the tool, so the file defined in Figure 1(B-12) will be generated in future work. A fragment of the code in which the taxonomy is handled using the fast-xml-parser library present in the code can be seen in Figure 11.

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

The last lines of the *main* file contain the code that generates the MSC XBRL instance with the data from the chosen data source. The instance is generated and loaded into the src/instances folder with the name instance.xml.

To help with the configuration and use of the application, a *README* file was added to the tool's files summarizing the activities to be followed and the settings required for the XBRL Processor to work.

Figure 9 - MSC model XBRL instance



(B) Geração da instância XBRL
Processamento da Instância XBRL
(7) Instância XBRL modelo
(08) Carga do modelo e informações complementares
(09) Cria o vínculo da taxonomia com os dados
(10) Seleciona o tipo da saída (XML, JSON ou CSV)
(11) Gera a instância XBRL
(12) Arquivo de configuração do vínculo com a taxonomia
(13) Instância XBRL
(A) Conexão com a base de dados

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<xbrli:xbrl xmlns:xbrli="http://www.xbrl.org/2003/instance" xmlns:gl-bus="http://www.xbrl.org/int/gl/bus/2015-03-25"
            xmlns:gl-cor="http://www.xbrl.org/int/gl/cor/2015-03-25" xmlns:iso4217="http://www.xbrl.org/2003/iso4217"
            xmlns:link="http://www.xbrl.org/2003/linkbase" xmlns:xlink="http://www.w3.org/1999/xlink"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <link:schemaRef xlink:href="SICONFI/cor/ext/gl/plt/case-c-b-m-u-t-s/gl-plt-all-2015-03-25.xsd" xlink:type="simple" />
  <xbrli:context id="C1">
    <xbrli:entity>
      <xbrli:identifier scheme="http://siconfi.tesouro.gov.br">11EX</xbrli:identifier>
    </xbrli:entity>
    <xbrli:period>
      <xbrli:instant>2023-01-31</xbrli:instant>
    </xbrli:period>
  </xbrli:context>
  <xbrli:unit id="BRL">
    <xbrli:measure>iso4217:BRL</xbrli:measure>
  </xbrli:unit>
  <xbrli:unit id="u">
    <xbrli:measure>xbrli:pure</xbrli:measure>
  </xbrli:unit>
  <gl-cor:accountingEntries>
    <gl-cor:documentInfo>
      <gl-cor:entriesType contextRef="C1">trialbalance</gl-cor:entriesType>
    </gl-cor:documentInfo>
    <gl-cor:entityInformation>
      <gl-bus:reportingCalendar>
        <gl-bus:reportingCalendarPeriod>
          <gl-bus:periodIdentifier contextRef="C1">2023-01</gl-bus:periodIdentifier>
          <gl-bus:periodDescription contextRef="C1">2023-01-01</gl-bus:periodDescription>
          <gl-bus:periodStart contextRef="C1">2023-01-01</gl-bus:periodStart>
          <gl-bus:periodEnd contextRef="C1">2023-01-31</gl-bus:periodEnd>
        </gl-bus:reportingCalendarPeriod>
      </gl-bus:reportingCalendar>
    </gl-cor:entityInformation>
    <gl-cor:entryHeader>
      <gl-cor:entryDetail>
        <gl-cor:lineNumberCounter contextRef="C1" decimals="0" unitRef="u">1</gl-cor:lineNumberCounter>
        <gl-cor:account>
          <gl-cor:accountMainID contextRef="C1">111110100</gl-cor:accountMainID>
          <gl-cor:accountSub>
            <gl-cor:accountSubID contextRef="C1">10131</gl-cor:accountSubID>
            <gl-cor:accountSubType contextRef="C1">PO</gl-cor:accountSubType>
          </gl-cor:accountSub>
          <gl-cor:accountSub>
            <gl-cor:accountSubID contextRef="C1">1</gl-cor:accountSubID>
            <gl-cor:accountSubType contextRef="C1">FP</gl-cor:accountSubType>
          </gl-cor:accountSub>
          <gl-cor:accountSub>
```

Figure 10 - XBRL Processor tool component diagram.

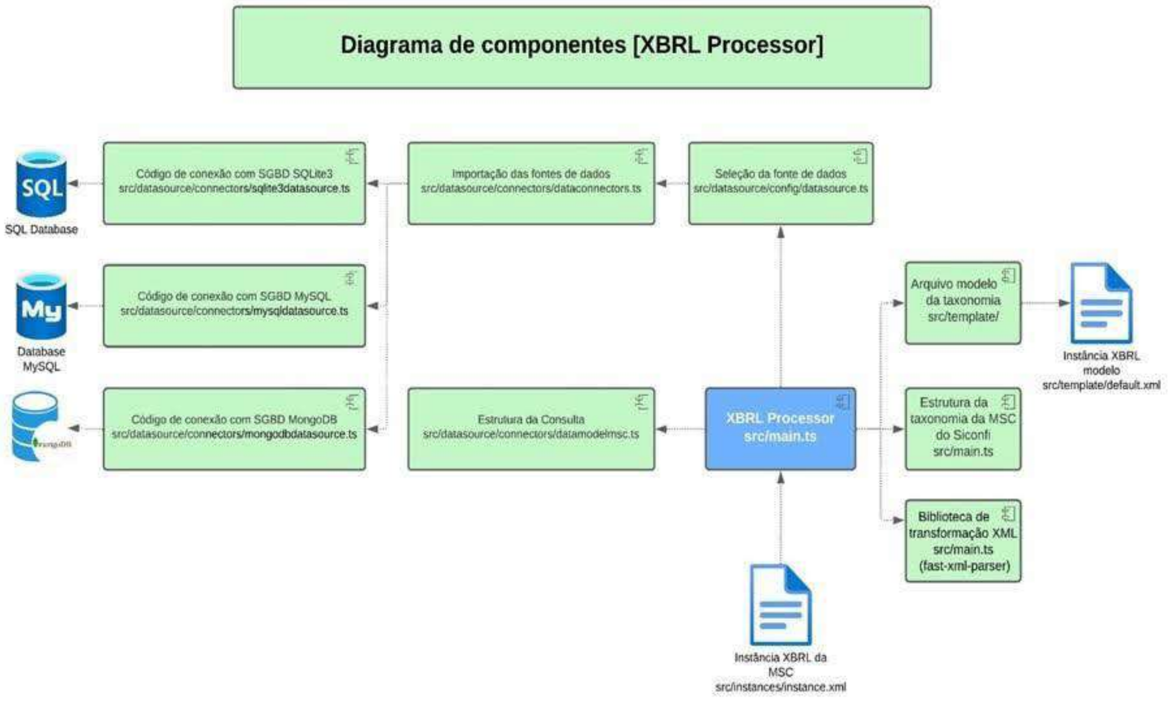
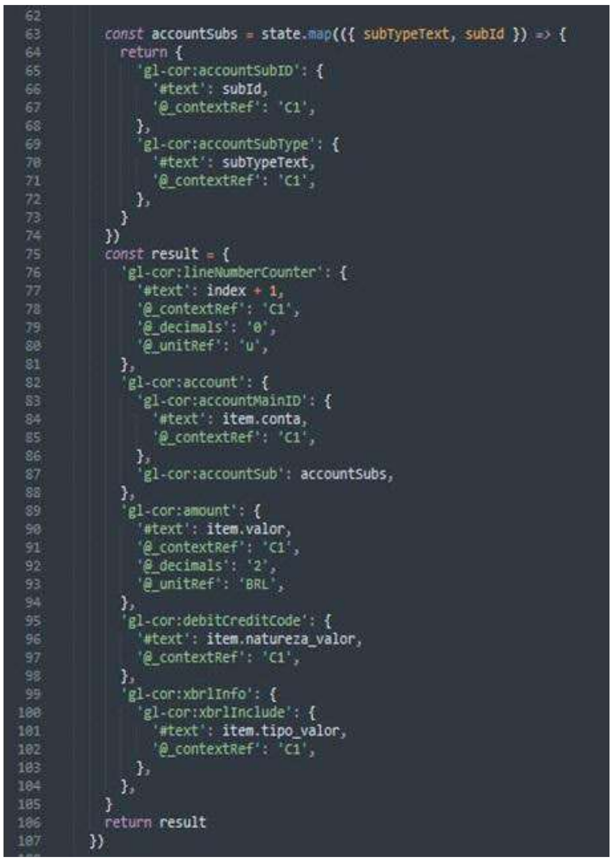


Diagrama de componentes [XBRL Processor]
SQL
SQL Database
Database MySQL
Código de conexão com SGBD SQLite3 src/datasource/connectors/sqlite3datasource.ts
Código de conexão com SGBD MySQL src/datasource/connectors/mysqldatasource.ts
Código de conexão com SGBD MongoDB src/datasource/connectors/mongodbdatasource.ts
Importação das fontes de dados src/datasource/connectors/dataconnectors.ts
Estrutura da Consulta src/datasource/connectors/datamodelmsc.ts
Seleção da fonte de dados src/datasource/config/datasource.ts
XBRL Processor src/main.ts
Arquivo modelo da taxonomia src/template/
Instância XBRL modelo src/template/default.xml
Estrutura da taxonomia da MSC do Siconfi src/main.ts
Biblioteca de transformação XML src/main.ts (fast-xml-parser)
Instância XBRL da MSC src/instances/instance.xml

Figure 11 - Code excerpt from the main file containing the MSC taxonomy



```
    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
  })
```

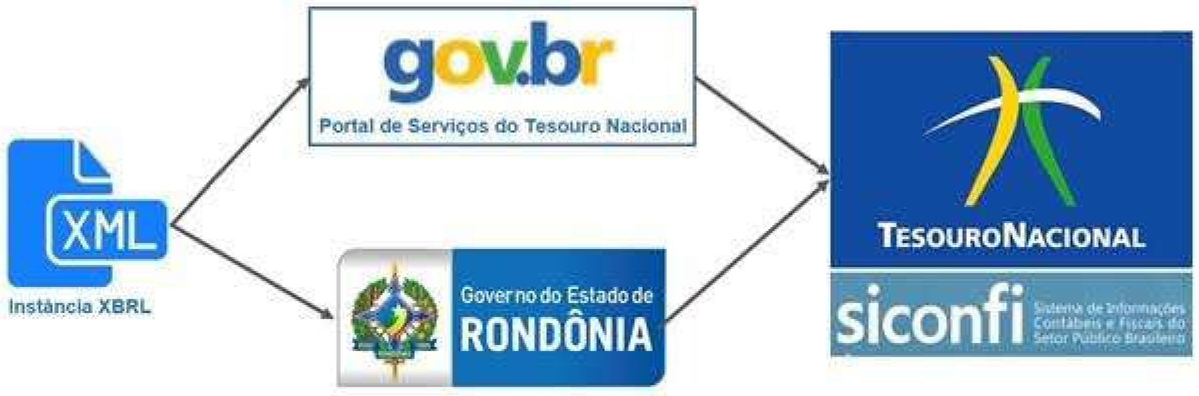
5. CASE STUDY FOR GENERATING THE XBRL INSTANCE AND VALIDATING ITS STRUCTURE

The development of this work and the XBRL Processor tool were planned so that it would be possible to check the result of the data processing and solve the problem of converting the data format to XBRL. Validation of the result of generating the instance produced by the tool was carried out by a federation entity and by the Siconfi validation tool (STN), organizations external to this work.

Therefore, developing the XBRL Processor tool parameterized for the MSC taxonomy and data standard for Siconfi, as seen in Figures 7 and 11, had some benefits: (i) solving the XBRL instance generation problem; (ii) solve the problem of handing over the MSC from the states and municipalities to the STN; (iii) have the structure of the MSC instance verified by the Brazilian regulatory body.

In order to understand the validations presented in this section, it is important to know that the instance was submitted to Siconfi in two different ways, the first using Siconfi's test environment, through the call center, and the second submitting the XBRL instance to Siconfi's production environment through the Rondônia State Finance Department, following the standard delivery of the MSC's XBRL instance of the States, as illustrated in Figure 12.

Figure 12 - Sequence for submitting the XBRL instance for validation


XML
Instância XBRL
gov.br
Portal de Serviços do Tesouro Nacional
Governo do Estado de
RONDÔNIA
TESOURONACIONAL
siconfi
Sistema de Informações Contábeis e Fiscais do Setor Público Brasileiro

5.1 *RESULTS*

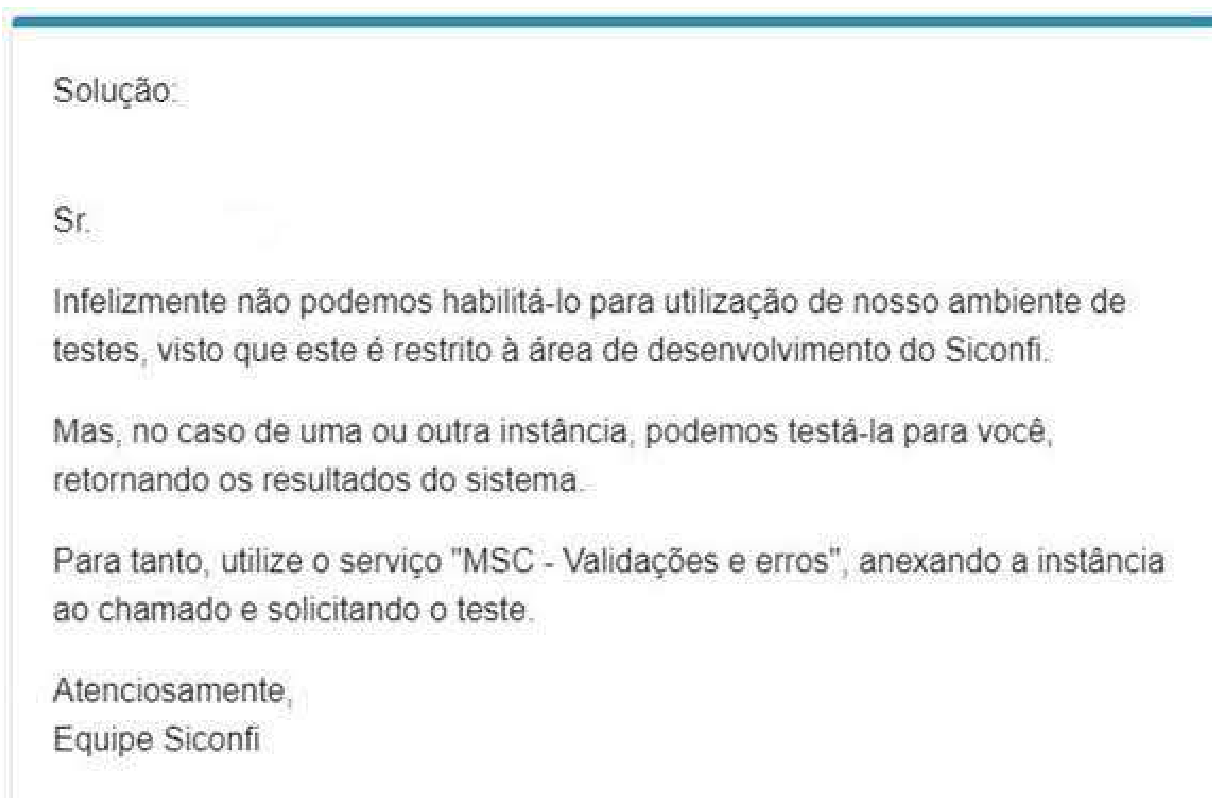
Initially, for comparison purposes, the Rondônia State Finance Department provided data from previous months, already validated in Siconfi, to facilitate comparison of the XBRL instance generated by XBRL Processor and the instance delivered to Siconfi by the Rondônia State Finance Department, and validated without errors or inconsistencies.

It was planned that the structure of the XBRL instance would be validated by Siconfi, since the comparison of the instance generated by the XBRL Processor tool with the instances provided by the Rondônia State Finance Department proved to be identical. Therefore, the Rondônia State Finance Department provided fictitious and anonymous data, i.e. data used in the development database, which was loaded into the SQLite3, MySQL and MongoDB DBMSs, so that the instance could be generated and delivered to Siconfi for validation. After contacting the Siconfi support team, they were instructed to request validation of the instance via the Siconfi system[27] using the "MSC - Validations and errors" option. Figure 13[28] shows the response to the request for access to the Siconfi test environment. As instructed by the Siconfi support team, the "MSC - Validations and errors" option was requested through the ticketing system and the XBRL instance was provided.

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

28 The header of the e-mail has been hidden due to the determination of STN Notice No. 10 of May 29, 2023, 28th NATIONAL TREASURY AWARD 2023, not identifying the author(s) in any part, including in the properties of the file.

Figure 13 - Response from the Siconfi service team



Solução:

Sr. [redacted]

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

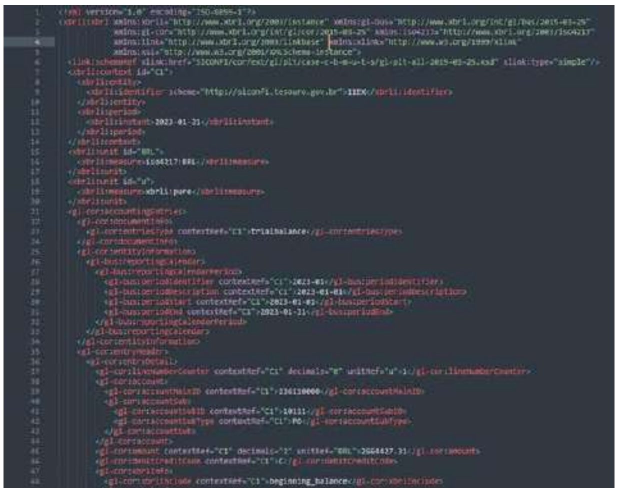
Mas, no caso de uma ou outra instância, podemos testá-la para você, retornando os resultados do sistema.

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

Atenciosamente,
Equipe Siconfi

The instance generated by the XBRL Processor with fictitious data provided by the Finance Department of the State of Rondônia is illustrated in Figure 14.

Figure 14 - Excerpt from the XBRL Instance generated by the XBRL Processor tool

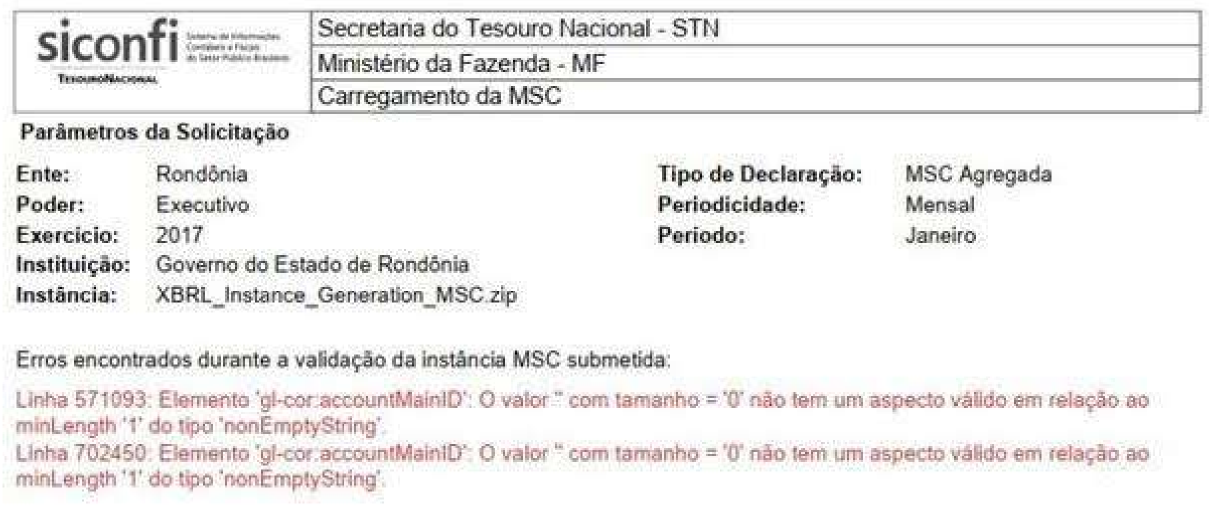


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xbrli:xbrl xmlns:xbrli="http://www.xbrl.org/2003/instance" xmlns:gl-bus="http://www.xbrl.org/int/gl/bus/2015-03-25"
            xmlns:gl-cor="http://www.xbrl.org/int/gl/cor/2015-03-25" xmlns:iso4217="http://www.xbrl.org/2003/iso4217"
            xmlns:link="http://www.xbrl.org/2003/linkbase" xmlns:xlink="http://www.w3.org/1999/xlink"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <link:schemaRef xlink:href="SICONFI/cor/ext/gl/plt/case-c-b-m-u-t-s/gl-plt-all-2015-03-25.xsd" xlink:type="simple"/>
  <xbrli:context id="C1">
    <xbrli:entity>
      <xbrli:identifier scheme="http://siconfi.tesouro.gov.br">11EX</xbrli:identifier>
    </xbrli:entity>
    <xbrli:period>
      <xbrli:instant>2023-01-31</xbrli:instant>
    </xbrli:period>
  </xbrli:context>
  <xbrli:unit id="BRL">
    <xbrli:measure>iso4217:BRL</xbrli:measure>
  </xbrli:unit>
  <xbrli:unit id="u">
    <xbrli:measure>xbrli:pure</xbrli:measure>
  </xbrli:unit>
  <gl-cor:accountingEntries>
    <gl-cor:documentInfo>
      <gl-cor:entriesType contextRef="C1">trialbalance</gl-cor:entriesType>
    </gl-cor:documentInfo>
    <gl-cor:entityInformation>
      <gl-bus:reportingCalendar>
        <gl-bus:reportingCalendarPeriod>
          <gl-bus:periodIdentifier contextRef="C1">2023-01</gl-bus:periodIdentifier>
          <gl-bus:periodDescription contextRef="C1">2023-01-01</gl-bus:periodDescription>
          <gl-bus:periodStart contextRef="C1">2023-01-01</gl-bus:periodStart>
          <gl-bus:periodEnd contextRef="C1">2023-01-31</gl-bus:periodEnd>
        </gl-bus:reportingCalendarPeriod>
      </gl-bus:reportingCalendar>
    </gl-cor:entityInformation>
    <gl-cor:entryHeader>
      <gl-cor:entryDetail>
        <gl-cor:lineNumberCounter contextRef="C1" decimals="0" unitRef="u">1</gl-cor:lineNumberCounter>
        <gl-cor:account>
          <gl-cor:accountMainID contextRef="C1">236110000</gl-cor:accountMainID>
          <gl-cor:accountSub>
            <gl-cor:accountSubID contextRef="C1">10111</gl-cor:accountSubID>
            <gl-cor:accountSubType contextRef="C1">PO</gl-cor:accountSubType>
          </gl-cor:accountSub>
        </gl-cor:account>
        <gl-cor:amount contextRef="C1" decimals="2" unitRef="BRL">2664427.31</gl-cor:amount>
        <gl-cor:debitCreditCode contextRef="C1">C</gl-cor:debitCreditCode>
        <gl-cor:xbrlInfo>
          <gl-cor:xbrlInclude contextRef="C1">beginning_balance</gl-cor:xbrlInclude>
```

The response provided validated the structure of the XBRL instance generated by XBRL Processor, i.e. the structure of the instance complies with the MSC XBRL instance required by Siconfi. Figure 15 shows Siconfi's official response to the instance provided.

Siconfi requests that the date of the fiscal year, which appears in the document, be disregarded, as it is the environment where the 2023 taxonomy is installed in Siconfi. The response provided by Siconfi's validation indicated two errors, in lines 571093 and 702450. However, these errors refer to the data and not the structure of the instance, as can be seen from the e-mail sent by the Siconfi team and shown in Figure 16[29], since the test data is fictitious, errors like this were expected. Following the planned strategy, the same instance that was delivered to Siconfi The XBRL instance generated was also delivered to the Rondônia State Finance Department so that it could be submitted to Siconfi following the routine MSC delivery procedures. In this way, it was possible to submit the instance generated by the XBRL Processor tool to the production environment, i.e. the real validation environment. As the data was fictitious, the Rondônia State Finance Department did not finish sending the instance to Siconfi, but only submitted it to the production environment until it had been validated.

Figure 15 - Validation of the XBRL instance by the Siconfi system



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

Parâmetros da Solicitação

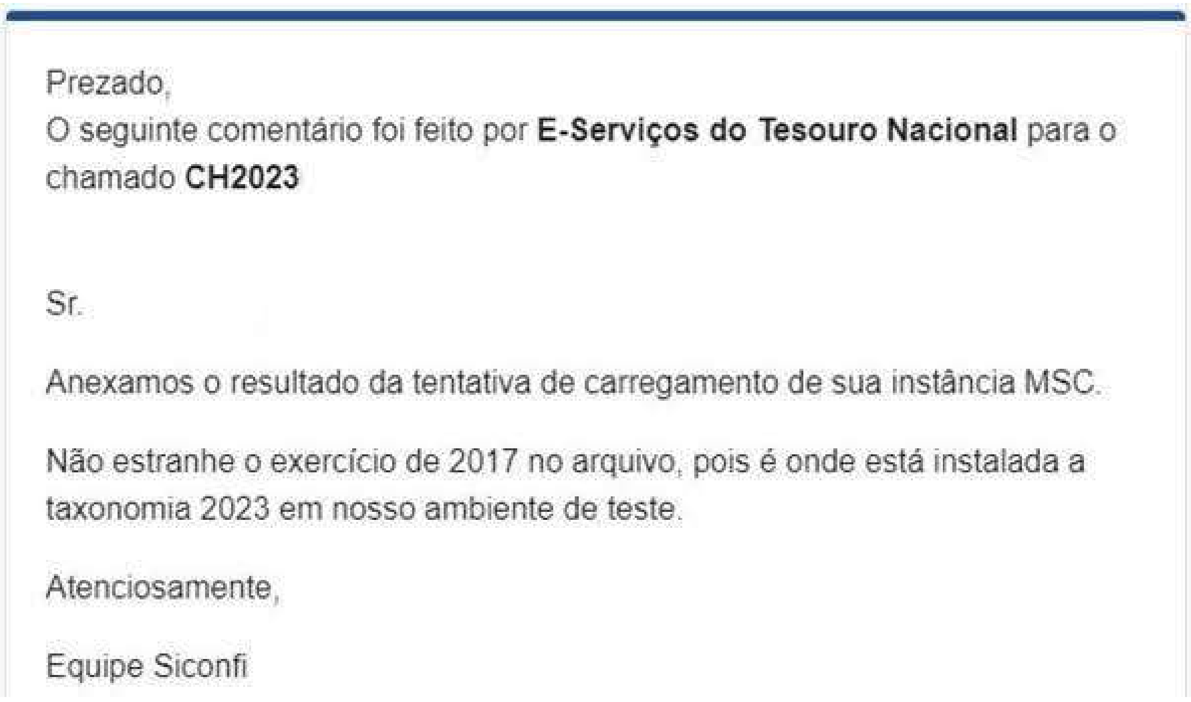
Ente:	Rondônia	**Tipo de Declaração:**	MSC Agregada
Poder:	Executivo	**Periodicidade:**	Mensal
Exercicio:	2017	**Periodo:**	Janeiro
Instituição:	Governo do Estado de Rondônia		
Instância:	XBRL_Instance_Generation_MSC.zip		

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'.

29 The header of the e-mail has been hidden due to the determination of STN Notice No. 10 of May 29, 2023, 28th NATIONAL TREASURY AWARD 2023, not identifying the author(s) in any part, including in the properties of the file

Figure 16 - Request for disregard for the 2017 financial year


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.

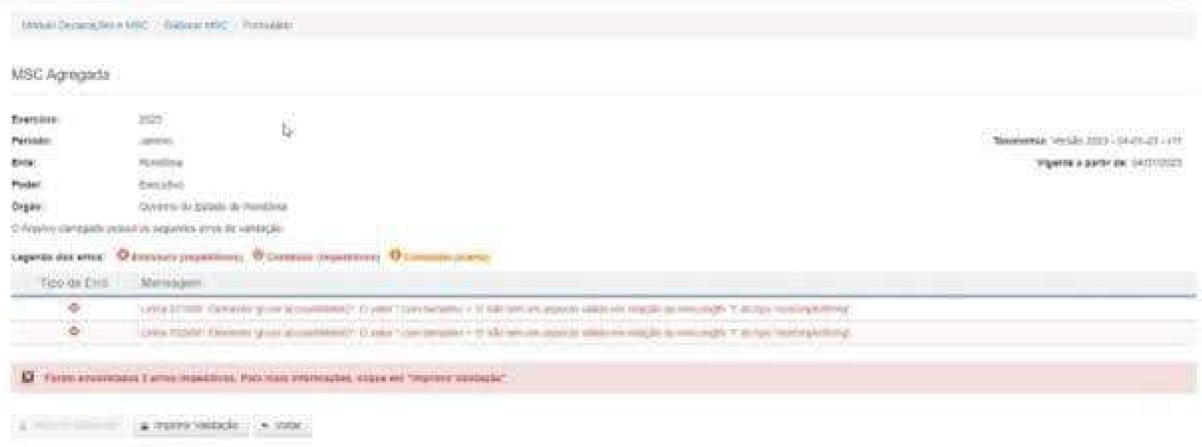
Não estranhe o exercício de 2017 no arquivo, pois é onde está instalada a taxonomia 2023 em nosso ambiente de teste.

Atenciosamente,

Equipe Siconfi

After submitting the instance to Siconfi via the Rondônia State Finance Department, the response obtained was exactly the same as that received from Siconfi via its customer service system. Figure 17 shows the validation of the instance by the Siconfi production environment.

Figure 17 - Validation of the XBRL instance in a production environment


[illegible]

MSC Agregada

Exercício: 2023
Período: Janeiro
Ente: Rondônia
Poder: Executivo
Órgão: Governo do Estado de Rondônia

Taxonomia: [illegible]
Vigente a partir de: [illegible]

[illegible]

Tipo de Erro	Mensagem
●	[illegible]
●	[illegible]

[illegible]

Note that the errors in lines 571093 and 702450 are the same as those in the validation of the Siconfi test environment. Therefore, the XBRL Processor tool generated the XBRL instance of the State of Rondônia's Matrix of Accounting Balances for the Brazilian Public Sector Accounting and Fiscal Information System of the National Treasury Secretariat with its structure without errors or inconsistencies.

Therefore, the application is up and running to connect to relational or NoSQL data sources and generate the XBRL instance of the Accounting Balance Matrix according to the taxonomy defined by Siconfi or be incorporated into existing financial and accounting applications.

6. CONCLUSION

According to what has been presented, generating XBRL instances is not a simple process, and the lack of *open source* tools that deal with the problem may be a contributing factor to the failure of various regulatory bodies and private institutions, especially organizations in Brazil, to adopt XBRL technology in a significant way. This scenario is aggravated by the scarcity of XBRL tools that use relational or NoSQL DBMSs as their data source. manually entering data into the tools, such as the Fujitsu tool[30] and Amelkis[31].

As explained in this paper, there is a lack of solutions for generating XBRL instances with data from relational or NoSQL databases, even with the growing need to generate XBRL instances of data in the formats evaluated, due to the evolution and expansion of the internet and legal requirements.

Based on the results presented in this article, it was possible to answer the initial research questions:

• Question 1: Are there solutions that allow data from different formats to be mapped to XBRL?

However, there are proprietary solutions that are limited to specific segments, such as banking. Another important limitation is that XBRL instances are only generated from data contained in databases specific to a given organization, not allowing the use of other sources or DBMSs.

• Question 2: Are the existing solutions *open source*, i.e. free and *open source*?

Only the Arelle software[32] is open source, but its purpose is to read XBRL instances and validate the taxonomy, while the generation of XBRL instances is not covered, nor is the loading of data from different sources and data formats.

• Question 3: Do the proposed solutions have sufficient generalization to allow the mapping of any data model (e.g. relational, NoSQL, CSV, JSON) to XBRL?

None of the software investigated was able to read data from different sources.

30 https://www.fujitsu.com/global/products/software/middleware/application-infrastructure/interstage/solutions/xbrl/
31 https://www.amelkis-solutions.com/
32 https://arelle.org/arelle/

• Question 4: Do the proposed solutions meet the requirements of Siconfi and the STN's Matrix of Accounting Balances (MSC)?

One of the studies investigated presented a solution for the delivery of the STN's MSC, the work of (BRAGANÇA *et al*., 2019), but limited to reading the data only in CSV format and using third-party ETL software to generate the XBRL instance.

• Question 5: Is it possible to build a tool with which any taxonomy can be used to generate XBRL instances from any data source? This article has shown that it is possible to build a tool to generate XBRL instances from any taxonomy with data from a variety of data sources.

Also, the hypothesis formulated in this article is valid, which found that:

• Hypothesis 1: It is possible to build a tool for extracting, transforming and loading data in XBRL format, decoupled from any taxonomy and different data models and capable of adapting to the taxonomy and data source required.

The XBRL Processor tool for generating XBRL instances with data from relational or denormalized sources was presented. The tool has open source code and is parameterized with the Siconfi taxonomy to generate the MSC, however, the parameterization can be updated or configured for another XBRL taxonomy.

Finally, a case study showed that the tool is able to meet the needs of Brazilian states and municipalities in delivering the MSC to Siconfi, thus meeting the federal government's legal requirement.

6.1. CONTRIBUTIONS

The main contribution of this work is the tool proposed for extracting, transforming and loading data from different sources and data formats to generate XBRL instances. In addition, the tool was developed to be adaptable to different XBRL taxonomies. This tool was developed with a focus on the adaptability of the code to different technological environments, and is capable of extracting data from a variety of different models of data storage technologies, simplifying the generation of XBRL instances.

In this way, the XBRL Processor tool, with its open source code, is able to meet the need to generate XBRL instances of the MSC for Siconfi and can be extended to solve other needs for generating XBRL instances, even being incorporated into applications under development or guiding the development of its own solutions for legacy environments.

6.2. LIMITATIONS AND FUTURE WORK

Despite the benefits achieved by this work, some limitations were identified that need to be overcome in future work:

• Although the tool makes it easy to add new data sources, it is not available to load CSV files as a data source; however, research is already underway to adapt the XBRL Engine tool (BRAGANÇA *et al*., 2019) to be incorporated into the XBRL Processor;

• Validating the data used to generate the XBRL instance against the specifications contained in the taxonomy is beyond the scope of this work. However, it is possible that the XBRL Processor tool will evolve to overcome this limitation;

• Another limitation identified is the presence of elements of the taxonomy directly in the code, making it necessary to edit the source code in order to take advantage of other taxonomies or updates;

• Another limitation is that XBRL instances are only generated in XML format; the consortium responsible for XBRL technology also allows instances in JSON and CSV formats.

As future research, we suggest adding other functionalities to the tool:

• Develop a graphical user interface (GUI) along with usability tests (AKRAM; AQEEL; HAMID, 2023) and *User Experience* to allow the inclusion of users with no knowledge of software development;

• Develop a module to check and validate the data loaded from the data sources against the selected taxonomy;

• Generate instances in the XBRL-JSON and XBRL-CSV formats accepted by the consortium responsible for XBRL technology, in addition to the XBRL-XML format;

• Adapt the fast-xml-parser library to load the taxonomy through its files, eliminating the presence of taxonomy elements in the source code, as shown in Figure 11, enabling easy exchange and updating, with the aim of covering a greater number of organizations capable of taking advantage of XBRL technology in the exchange of accounting and financial information;

BIBLIOGRAPHICAL REFERENCES

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, Dec. 1, 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, Aug. 6, 2022. Available at: <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. In: 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. Available at: <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: A tool for data transformation based on the XBRL-SICONFI taxonomy. 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. In: 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. In: 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. In: Proceedings of the 21st International Conference on Information Integration and Web-Based Applications &amp; 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, aug. 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. In: 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. &. Database Systems. [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. The Xbrl Framework. International Conference on Information Systems and Technology Management 5th, p. 4343-4365, 2008.

SOARES, B. E.; BOSCARIOLI, C. Columnar Database Model: Characteristics, Applications and Examples of Systems. Regional School of Database-Brazilian Computer Society (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. In: 2018 IEEE Conference on Open Systems, ICOS 2018, 2019b, [...]. 2019. p. 65-70.

STN, S. do T. N. Matrix of Accounting Balances. National Treasury Secretariat, v. 1, 2017. Available at: <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. In: Proceedings - 16th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2017, [...]. 2017. p. 839-843.