UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Henrique Pamplona

Desenvolvimento de uma ferramenta para extração, comparação e validação de conformidades entre dados históricos armazenados em um PIMS

Florianópolis
2024

Henrique Pamplona

Desenvolvimento de uma ferramenta para extração, comparação e validação de conformidades entre dados históricos armazenados em um PIMS

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.
Orientador: Prof. Marcelo Ricardo Stemmer, Dr.
Supervisor: Lucas Messeder Corrêa, Eng.

Florianópolis
2024

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Pamplona, Henrique
Desenvolvimento de uma ferramenta para extração, comparação e validação de conformidades entre dados históricos armazenados em um PIMS / Henrique Pamplona ; orientador, Marcelo Stemmer, 2024.
68 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2024.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Automação Industrial. 3. Plant Information Management System (PIMS). 4. Extração de Dados. 5. Análise de Conformidade. I. Stemmer, Marcelo. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Henrique Pamplona

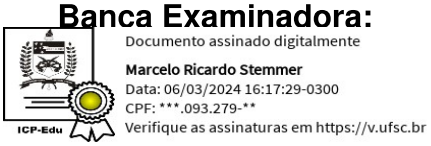
Desenvolvimento de uma ferramenta para extração, comparação e validação de conformidades entre dados históricos armazenados em um PIMS

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis, 16 de fevereiro de 2024.

Prof. Marcelo de Lellis Costa de oliveira, Dr.
Coordenador do Curso

Banca Examinadora:



Prof. Marcelo Ricardo Stemmer, Dr.
Orientador
UFSC/CTC/DAS

Lucas Messeder Corrêa, Eng.
Supervisor
Radix Engenharia e Desenvolvimento de Software

Prof. Max Hering de Queiroz, Dr.
Avaliador
UFSC/CTC/DAS

Prof. Hector Bessa Silveira, Dr.
Presidente da Banca
UFSC/CTC/DAS

Este trabalho é dedicado à minha família e a todos que
me acompanharam de alguma forma nessa trajetória.

AGRADECIMENTOS

Agradeço primeiramente a Deus.

Agradeço aos meus pais, Marcelo e Joselena, por toda a minha criação e por todo o apoio, amor e carinho que me deram.

Agradeço aos meus irmãos, Leonardo e Patrícia, por todo o companheirismo e preocupação durante a vida.

Agradeço ao restante da minha família, em especial à minha tia-avó Maria Celeste e à minha vó Maria Aparecida por todo o apoio e suporte dado ao longo dos anos.

Agradeço à Universidade Federal de Santa Catarina e aos professores e colaboradores do Departamento de Automação e Sistemas por toda a estrutura, ensinamentos e aprendizados que me proporcionaram, em especial ao Prof. Marcelo Stemmer por ter aceitado ser meu orientador nesse trabalho.

Agradeço à Radix pela oportunidade de trabalho e por toda a confiança e acolhimento que recebi pelos meus colegas desde que entrei, em especial ao Lucas, por todo o apoio e por ter me ensinado tanto.

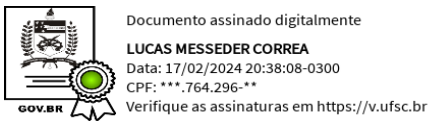
E, por fim, agradeço à todos que de alguma forma e em algum momento estiveram presentes nessa caminhada e que contribuíram para meu aprendizado e crescimento pessoal e profissional.

DECLARAÇÃO DE PUBLICIDADE

Florianópolis, 16 de fevereiro de 2024.

Na condição de representante da Radix Engenharia e Desenvolvimento de Software na qual o presente trabalho foi realizado, declaro não haver ressalvas quanto ao aspecto de sigilo ou propriedade intelectual sobre as informações contidas neste documento, que impeçam a sua publicação por parte da Universidade Federal de Santa Catarina (UFSC) para acesso pelo público em geral, incluindo a sua disponibilização *online* no Repositório Institucional da Biblioteca Universitária da UFSC. Além disso, declaro ciência de que o autor, na condição de estudante da UFSC, é obrigado a depositar este documento, por se tratar de um Trabalho de Conclusão de Curso, no referido Repositório Institucional, em atendimento à Resolução Normativa n° 126/2019/CUn.

Por estar de acordo com esses termos, subscrevo-me abaixo.



Lucas Messeder Corrêa
Radix Engenharia e Desenvolvimento de Software

RESUMO

O objetivo do projeto consistiu em desenvolver uma ferramenta que realiza a extração, comparação e validação de conformidade entre dados históricos armazenados em um *Plant Information Management System* (PIMS), de forma a assegurar a qualidade dos dados ou apontar discrepâncias entre pares e possíveis causas. Considerando que o problema abordado foi a necessidade de garantir a consistência e integridade dos dados históricos em ambientes industriais complexos, a solução proposta envolveu o desenvolvimento de uma ferramenta capaz de realizar de forma eficaz a extração, análise de dados e geração de relatórios com indicadores de conformidade. Para isso, a metodologia utilizada abordou o estudo de maneiras de extração de dados, de métodos de análise de conformidade e de possíveis causas das não conformidades, tendo como foco o software AVEVA PI System, amplamente utilizado como um PIMS. Por fim, a solução desenvolvida contempla a implementação de algoritmos e de uma interface que permite a utilização da ferramenta em ampla escala, de forma a facilitar a identificação de conformidades ou não conformidades e a padronização de relatórios. Além disso, a estrutura desenvolvida permite uma fácil adaptação para outros meios de extração ou de análise de dados, podendo servir de base para projetos futuros.

Palavras-chave: Automação Industrial. *Plant Information Management System* (PIMS). Dados.

ABSTRACT

The objective of the project consisted of developing a tool that performs extraction, comparison, and validation of compliance between historical data stored in a *Plant Information Management System* (PIMS), in order to ensure data quality or point out discrepancies between pairs and possible causes. Considering that the addressed problem was the need to ensure consistency and integrity of historical data in complex industrial environments, the proposed solution involved the development of a tool capable of effectively performing data extraction, analysis, and report generation with compliance indicators. To achieve this, the methodology used approached the study of data extraction methods, compliance analysis methods, and possible causes of non-compliance, focusing on the AVEVA PI System software, widely used as a PIMS. Finally, the developed solution includes the implementation of algorithms and an interface that allows the tool to be used on a large scale, facilitating the identification of compliance or non-compliance and the standardization of reports. Additionally, the developed structure allows for easy adaptation to other means of data extraction or analysis, serving as a basis for future projects.

Keywords: Industrial Automation. Plant Information Management System (PIMS). Data.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O uso de sistemas PIMS (Plant Information Management Systems) tem se tornado cada vez mais frequente e relevante na indústria nacional e internacional, visto que se trata de um sistema capaz historiar dados de diferentes fontes, sejam elas dados de equipamentos, de laboratório, ou até mesmo de cálculos gerados pelo próprio sistema em tempo contínuo. É necessário que os dados coletados por esses sistemas sejam de qualidade e consistentes, visto que eles podem ser usados para inúmeras iniciativas como, por exemplo, a confecção de análises para a detecção precoce de defeitos em máquinas ou de indicadores que serão utilizados na tomada de decisão da empresa.

É comum que empresas do ramo da indústria química, indústria têxtil, entre outros, possuam PIMS em sua linha de produção e que surja a necessidade de migrar de um software para outro, sendo necessário realizar um projeto para adaptar a estrutura de um sistema para o outro. Também ocorre de empresas possuírem mais de um servidor para armazenar os dados do PIMS, seja por questões de melhor otimização da rede ou por necessidade de restringir o acesso de determinados dados à determinados grupos de usuários.

Nos dois casos é necessário que os dados sejam consistentes entre seus pares, seja na comparação entre um dado lido diretamente do PIMS antigo para o novo correspondente gerado após a migração, com fins de validação, seja na comparação entre dados do mesmo PIMS mas de servidores diferentes, visto que a replicação ou a coleta dos dados pode apresentar falhas e, consequentemente, gerar inconsistências.

O projeto desenvolvido buscou desenvolver uma ferramenta para realizar a validação de conformidades entre dados extraídos de diferentes servidores que utilizem o software PI System da Aveva, sendo capaz de extrair os dados, analisá-los e gerar indicadores sobre a conformidade. A ferramenta facilitará a execução dessa atividade e poderá ser usada em diferentes projetos da Radix, que já atendeu demandas tanto de migração entre diferentes PIMS como também de suporte para clientes que possuem mais de um servidor para o sistema.

1.1 OBJETIVOS

O principal objetivo do projeto é o desenvolvimento de um software capaz de realizar de forma intuitiva e facilitada a extração, comparação e análise de conformidade entre dados históricos extraídos do PI System, tendo os seguintes objetivos específicos que o software deve atingir:

- Permitir a extração de dados de diferentes bases de dados do PI System de forma facilitada;

- Realizar a transformação dos dados para um padrão definido;
- Realizar a análise de conformidade entre pares de dados extraídos;
- Gerar indicadores sobre a conformidade de forma automática.

1.2 METODOLOGIA UTILIZADA

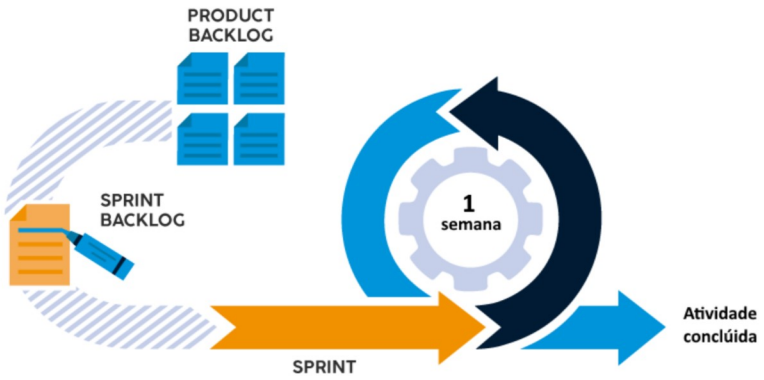
Inicialmente foram determinadas quatro diferentes fases para o desenvolvimento do projeto: "Planejamento", "Extração de dados", "Análise de dados"e "Geração de indicadores e interface com o usuário".

A primeira etapa abordou o levantamento de requisitos a para a realização do projeto, incluindo o estudo dos métodos de análise de conformidade e das ferramentas que foram utilizadas.

As atividades realizadas durante a etapa de “Extração de dados” englobaram o desenvolvimento de códigos buscando a conexão com o PIMS, a extração dos valores em si e a estrutura de dados utilizada para possibilitar a análise de conformidade. No terceiro tópico foram desenvolvidas as atividades de manipulação dos dados extraídos, utilizando análises estatísticas para gerar os indicadores de conformidade e outros indicadores pertinentes à ferramenta.

Por fim, o tópico da “Geração de indicadores e interface com o usuário” envolveu tanto a parte de geração de relatórios de acordo com as necessidades do usuário como também o desenvolvimento da interface gráfica do software.

Figura 1 – Conceitos da metodologia Scrum.

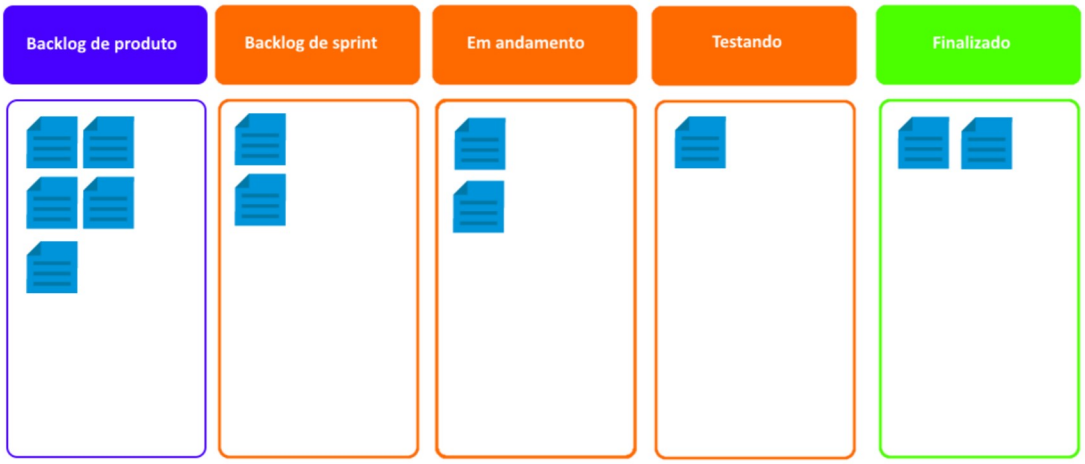


Fonte: Acervo pessoal.

Para o desenvolvimento da ferramenta foram utilizadas metodologias ágeis com aspectos das metodologias Scrum e Kanban. Da primeira metodologia, foi utilizado o conceito de *sprint*, que consiste em curtos ciclos de tempo em que determinadas atividades devem ser planejadas, executadas e entregues, e também o conceito de

backlog, que corresponde às atividades conhecidas que devem fazer parte da execução do projeto tendo elas complexidades e prioridades de execução mapeadas e sendo separadas em "*backlog* do produto" (atividades que ainda não devem ser desenvolvidas) e "*backlog* da *sprint*" (atividades que devem ser executadas durante a *sprint*). Ao final de cada sprint as atividades são revistas, validadas e é feito o planejamento do próximo ciclo utilizando informações do *backlog*. O uso dessa metodologia foi adaptado e usado inicialmente com *sprints* de uma semana, sem a necessidade de reuniões diárias como é comum no uso da metodologia, a Figura 1 apresenta de forma visual como é o funcionamento da metodologia.

Figura 2 – Conceitos da metodologia Kanban.



Fonte: Acervo pessoal.

Já a metodologia Kanban se trata de uma metodologia visual de divisão de atividades em *cards* posicionados em diferentes colunas de um quadro de controle dividido em diferentes etapas que englobam o trabalho desde o *backlog* até a conclusão da atividade, foi utilizada em conjunto com os aspectos da metodologia Scrum de forma que os cards das atividades que estarão no backlog do produto estarão no começo do quadro e os referentes à cada sprint estarão nas colunas de execução. A Figura 2 ilustra um exemplo de um quadro da metodologia Kanban.

1.3 ESTRUTURA DO DOCUMENTO

O presente documento está dividido em sete capítulos que tem como finalidade apresentar o contexto em que o projeto foi desenvolvido, quais atividades foram realizadas e os resultados obtidos.

No capítulo 2 é feita uma breve descrição sobre a Radix Engenharia e Desenvolvimento de Software, apresentando a empresa, seus segmentos de atuação e as

soluções que ela oferece.

No capítulo 3 é explicado brevemente sobre a estrutura da pirâmide da automação e a importância do PIMS na indústria, aspectos conceituais considerados no projeto e as principais tecnologias utilizadas.

No capítulo 4 são apresentados os requisitos gerais do projeto, casos de uso e fluxos determinados para a solução final.

No capítulo 5 é apresentado o desenvolvimento da ferramenta em si, detalhando a arquitetura do software desenvolvido, os padrões de dados, interfaces, modelos de conexão e a interface gráfica.

No capítulo 6 é feita uma breve apresentação de testes realizados utilizando a ferramenta para validar o atingimento dos objetivos e seus resultados obtidos.

Por fim, no capítulo final é apresentada a conclusão apontando o que foi atingido e indicando possíveis melhorias e atividades futuras.

2 RADIX ENGENHARIA E DESENVOLVIMENTO DE SOFTWARE

2.1 EMPRESA

A Radix é uma multinacional de origem brasileira do ramo de tecnologia e engenharia que está no mercado desde março de 2010. A empresa possui escritórios no Rio de Janeiro, São Paulo, Minas Gerais, Houston e Atlanta, possuindo atuação em todos os continentes do planeta e oferecendo serviços e soluções de tecnologia altamente qualificados e com independência tecnológica.

Para tal, a empresa conta com um time de gerentes e consultores multidisciplinares, com mais de 30 anos de experiência em projetos de Engenharia, Automação & TI Industrial e Desenvolvimento de Software. Um outro facilitador para a atuação em grande escala e constante crescimento da empresa é ela possuir um programa chamado *Radix Everywhere*, com o foco no trabalho remoto e permitindo que grande parte dos colaboradores desenvolvam suas atividades em *home office*.

Desde 2010 a empresa está na lista de melhores empresas para se trabalhar segundo o *Great Place to Work Institute*, recentemente sendo eleita em 2021 o 1º lugar do Rio de Janeiro e do Brasil (na pesquisa Melhor Empresa para se Trabalhar, categoria médias empresas) e o 2º lugar no ranking específico de TI no Brasil, atrás somente da Microsoft, gigante tecnológica e parceira da Radix desde 2016. Além da Microsoft, a organização possui outras parcerias estratégicas com empresas de renome como, por exemplo, a Amazon Web Services, OSIsoft, AVEVA, Hexagon PPM, Schneider Eletric, etc.

Como base da empresa, a Radix possui bem definidos sua Missão (”Transformar conhecimento técnico-científico em resultados”), Visão (”Ser o provedor *Top of Mind* de soluções tecnológicas para problemas complexos”) e Valores (”Relação de longo prazo, agilidade, ética, desafio, inovação, foco no ser humano e comprometimento”), prezando por esses indicadores e inclusive já sendo reconhecida pelo valor ”ética”, recebendo no ano de 2021, pela quarta vez consecutiva, o prêmio de Empresa Pró-Ética, iniciativa promovida pela Controladoria-Geral da União (CGU), que busca identificar, nos setores público e privado, empresas comprometidas com a ética, o combate à corrupção e a prática de negócios íntegros. A Radix foi uma das 67 empresas reconhecidas, reflexo do alto nível de maturidade do Programa de Compliance interno da empresa.

A instituição também possui outros selos e certificações como ISO 9001, ISO 14001, OHSAS 18001 e certificado de maturidade e desenvolvimento CMMI Nível 5, além de ser reconhecida como uma Empresa Estratégica de Defesa pelo Ministério de Defesa. Tais reconhecimentos são um dos motivos da empresa possuir clientes e atuar em todos os segmentos de negócios, como óleo e gás, energia, metais e mineração, papel e celulose, transportes, finanças, varejo, etc.

2.2 SEGMENTOS

Além de ser uma empresa global e com atuação em mais de 30 países, a Radix desenvolve soluções personalizadas de transformação digital em diferentes segmentos do mercado, sendo os principais exemplificados na Figura 3.

Figura 3 – Segmentos de atuação da Radix.

Fonte: Radix.

2.3 SOLUÇÕES

Visando atender a necessidade dos seus clientes e buscar atingir resultados concretos, eficazes e de longa duração, a Radix possui um portfólio de serviços completo para apoiar empresas dos diversos segmentos indicados na seção anterior de forma a atuar em cada etapa de transformação digital. Para isso, a empresa possui expertise em soluções divididas em quatro grandes áreas mencionadas nas próximas subseções.

2.3.1 Consultoria

Os serviços de consultoria da Radix buscam garantir o desenvolvimento de soluções técnicas com impacto relevante para os objetivos dos clientes, atuando diretamente na solução dos desafios de negócio a partir dos campos apresentados na Figura 4.

2.3.2 Engenharia

A área de engenharia da empresa engloba todo o ciclo de vida de projetos industriais, desde a análise de viabilidade técnica e econômica, passando pela execução e o comissionamento, além de oferecer apoio à operação.

A Figura 5 apresenta o portfólio de serviços de engenharia da Radix.

Figura 4 – Soluções da Radix - Consultoria.

Fonte: Radix.

Figura 5 – Soluções da Radix - Engenharia.

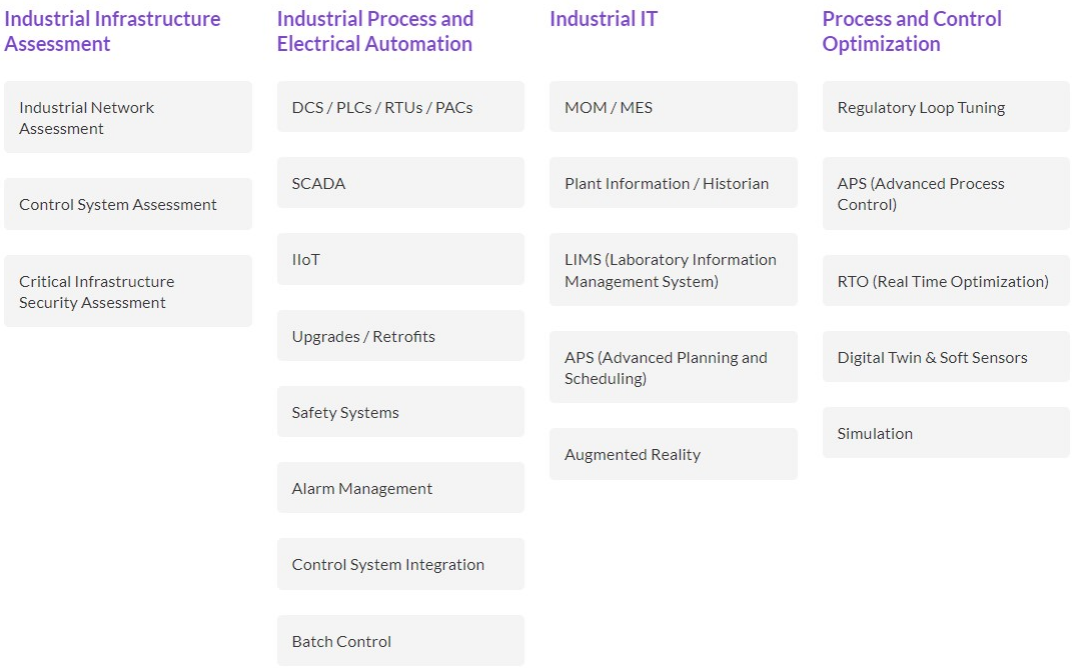
Fonte: Radix.

2.3.3 Automação Industrial

A área de automação industrial da empresa se baseia na introdução de camadas de inteligência operacional a partir da TI Industrial em tecnologias de automação de fábrica, buscando gerir grandes volumes de dados e otimizar operações de forma a aumentar a eficiência, eficácia e gestão de riscos. Para isso, são abordados os tópicos

apresentados na Figura 6.

Figura 6 – Soluções da Radix - Automação Industrial.



Fonte: Radix.

2.3.4 Software & TI

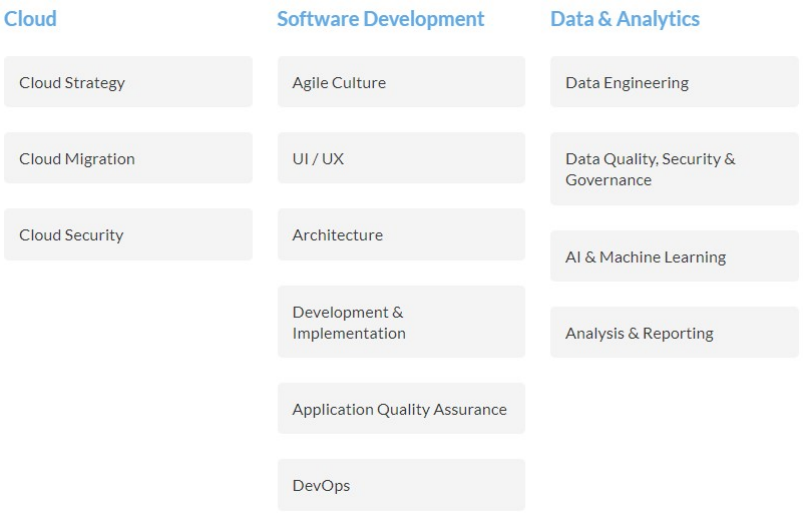
Por fim, a Radix atua no desenvolvimento de sistemas e aplicações, abordando todo o ciclo da criação de soluções personalizadas e também oferecendo expertise em serviços de inteligência de dados, incluindo soluções de inteligência de dados e *analytics* e considerando os principais tópicos apresentados na Figura 7.

2.4 DIVISÃO SOCIETÁRIA

No ínicio de 2015 a Radix recebeu aporte de capital financeiro do Grupo Sotreq, grupo 100% brasileiro e um dos maiores distribuidores da Caterpillar no mundo, buscando continuar o processo acelerado de expansão. Atualmente a divisão societária da empresa está dividida entre o Grupo Sotreq e os fundadores da Radix, conforme Figura 8.

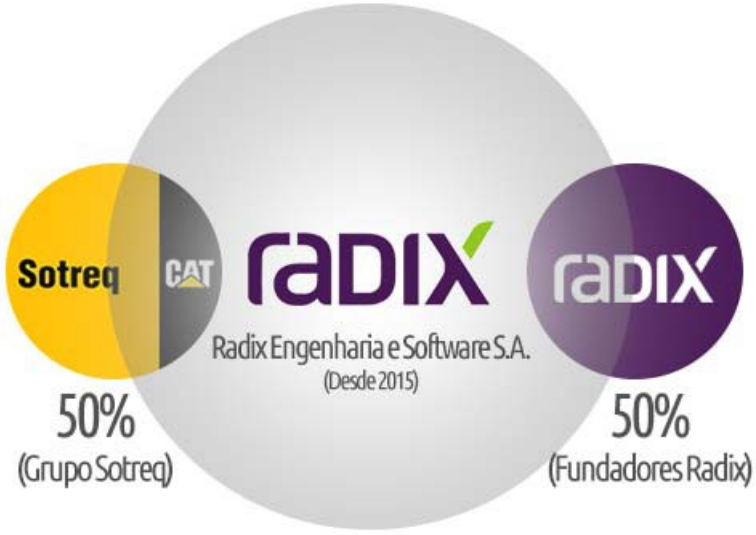
Além de soluções tecnológicas, a empresa investe também no suporte estratégico à tomada de decisão do cliente. As tecnologias desenvolvidas pela Radix aumentam a eficiência e agregam valor ao negócio do cliente. Algumas destas tecnologias surgiram de soluções, frameworks e metodologias elaboradas de forma customizada

Figura 7 – Soluções da Radix - Software & TI.



Fonte: Radix.

Figura 8 – Divisão Societária Radix.



Fonte: Radix.

para empresas, sempre considerando o que já existe no mercado, atentando à relação custo e benefício e seguindo os melhores conceitos de engenharia de projetos

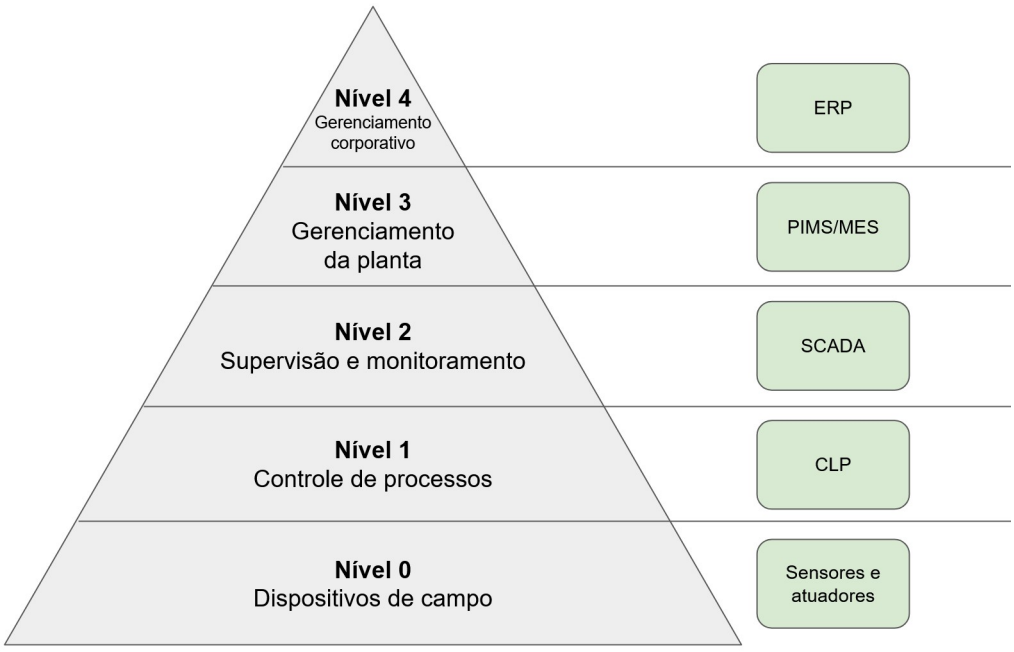
Internamente a Radix é dividida em diferentes áreas chamadas de Unidades de Negócios (UNs), sendo elas: UN de Serviços, UN de Óleo e Gás e UN de Metais e Mineração. As atividades descritas neste relatório foram realizadas dentro da UN de Serviços, mais especificamente na área de Automação dessa unidade.

3 ASPECTOS CONCEITUAIS E FERRAMENTAS UTILIZADAS

Neste capítulo serão brevemente explicados conceitos de algumas das principais tecnologias relacionadas ao trabalho, abordando a pirâmide da automação industrial, o que é um PIMS e seu papel na pirâmide da automação, descrevendo o PI System, suas funcionalidades utilizadas e quais as tecnologias utilizadas no desenvolvimento da ferramenta.

3.1 PIRÂMIDE DA AUTOMAÇÃO INDUSTRIAL

Figura 9 – Pirâmide da automação segundo norma ANSI/ISA-95.



Fonte: Adaptação da norma ANSI/ISA-955.

O modelo da pirâmide da automação segundo a norma ANSI/ISA-95 foi pensado inicialmente para a automação em nível industrial, sendo separado em cinco diferentes níveis independentes entre si e com suas respectivas funções. O nível 0, conhecido popularmente como ”chão de fábrica”, é o nível onde estão localizados os dispositivos de campo, bem como os sensores e atuadores. Já o nível 1 é responsável pelo controle dos processos que ocorrem no nível inferior, exemplificado através da figura dos Controladores Lógico-Programáveis (CLPs). No nível 2 ocorre a supervisão e monitoramento dos processos através dos sistemas SCADA, responsáveis pela coleta de dados em tempo real de uma grande variedade de fontes distintas, tais dados podem ser repassados, por exemplo, para um sistema *Plant Information Management*

System (PIMS) que se encontra no nível 3 da pirâmide, encarregado pelo planejamento e coordenação geral das plantas. Por fim, o nível 4 é responsável pela tomada de decisões e gerenciamento de toda a estrutura em um contexto de gerenciamento corporativo, simbolizado pelos sistemas ERP (*Enterprise Resource Planning*).

Quadro 1 – Níveis da Pirâmide da Automação segundo a norma ANSI/ISA-95

Nível	**Descrição**
Nível 0: Equipamentos	Sensores, atuadores e dispositivos de campo que monitoram e controlam processos físicos.
Nível 1: Controle	Controladores de processo, como PLCs (Controladores Lógicos Programáveis) e PACs (Controladores de Automação Programáveis), responsáveis pelo controle e monitoramento dos equipamentos.
Nível 2: Controle de Processo	Sistemas de Controle de Processo, responsáveis pelo controle e supervisão de unidades de produção e processos específicos.
Nível 3: Gerenciamento da Produção	Sistemas de Gerenciamento da Produção (MES - Manufacturing Execution Systems), que coordenam e controlam atividades de produção dentro da planta.
Nível 4: Gerenciamento Corporativo	Sistemas de Gerenciamento Corporativo (ERP - Enterprise Resource Planning) que integram atividades de negócio em toda a empresa.

Fonte: Adaptado de Norma ANSI/ISA-95.

Embora a norma não aborde diretamente a nomenclatura de *Plant Information Management System*, é possível traçar um paralelo com os sistemas *Manufacturing Execution Systems*, visto que ambos são sistemas de informações utilizados em ambientes industriais para melhorar a eficiência operacional e a gestão da produção.

3.2 PLANT INFORMATION MANAGEMENT SYSTEM (PIMS)

O *Plant Information Management System* (PIMS) é um sistema responsável por coletar, armazenar, processar e distribuir informações relevantes sobre processos industriais em tempo real. Considerando os avanços da automação industrial e da Indústria 4.0 em si, esse sistema desempenha um papel crucial na tratativa de dados provenientes de diversas fontes dentro de uma planta industrial, proporcionando uma visão holística das operações. O PIMS é projetado para lidar com uma variedade de dados de forma contínua, incluindo informações de sensores, dados de produção, dados de laboratório e de qualidade, entre outros.

O gerenciamento eficiente e a confiabilidade de dados é fundamental para o sucesso da automação industrial, sendo que os sistemas PIMS tem um importante

papel nesse contexto, servindo, por exemplo, como um intermediário entre os níveis 2 e 4 da pirâmide da automação (apresentada na Figura 9), ou seja, fornecendo informações dos níveis mais baixos para o nível de gerenciamento corporativo e auxiliando na tomada de decisões estratégicas ou corporativas.

3.2.1 Benefícios

De forma geral, o PIMS é caracterizado por oferecer os seguintes benefícios dentro do contexto da indústria:

- **Visão centralizada da planta:** O PIMS fornece uma visão unificada de toda a planta, integrando dados de produção, qualidade, manutenção, segurança e outros departamentos;
- **Melhoria da tomada de decisões:** Devido ao acesso a dados históricos e em tempo real identificação de problemas, otimização de processos e tomada de decisões mais inteligentes é facilitada;
- **Aumento da eficiência operacional:** Com o sistema é possível identificar gargalos e áreas de ineficiência, possibilitando a otimização da produção e a redução de custos;
- **Melhoria da qualidade do produto:** O PIMS serve para realizar o monitoramento e a análise dados de qualidade em tempo real, permitindo a identificação e correção de problemas de qualidade de forma eficaz;
- **Maior segurança e confiabilidade:** O PIMS fornece informações valiosas para a análise de riscos e a implementação de medidas de segurança e confiabilidade.

3.2.2 Principais Componentes

Um *Plant Information Management System* é caracterizado por se relacionar com diferentes componentes que trabalham juntos para coletar, armazenar, processar e distribuir informações relevantes sobre processos industriais. Sendo eles descritos nos próximos itens.

- **Interfaces de Comunicação**

  As interfaces de comunicação permitem a conexão do PIMS com diferentes dispositivos e sistemas dentro da planta industrial, como sistemas de controle de processos, sensores, bancos de dados e sistemas de gestão empresarial (ERP). Elas garantem a interoperabilidade com diversos protocolos de comunicação, como OPC UA, Modbus, APIs e interfaces wireless (Wi-Fi, Bluetooth), assegurando a comunicação confiável e segura entre os sistemas.

- **Servidores de Dados**

  Os servidores de dados são responsáveis por armazenar grandes volumes de informações coletadas pelo PIMS, garantindo a segurança, integridade e acessibilidade dos dados. Eles podem ser servidores de arquivos, bancos de dados relacionais (MySQL, SQL Server, por exemplo) ou bancos de dados NoSQL (MongoDB, por exemplo), escolhidos de acordo com as necessidades específicas do cliente.

- **Sistemas de Coleta de Dados**

  Os sistemas de coleta de dados são responsáveis por coletar dados brutos de diversas fontes, como sensores, sistemas de controle de processos e dispositivos IoT (Internet das Coisas). Eles pré-processam e filtram os dados antes de transmiti-los para os servidores de dados, garantindo a qualidade e confiabilidade das informações coletadas.

  Nessa etapa podem existir parâmetros de compressão e exceção para otimizar o armazenamento dos dados visando reduzir a quantidade de espaço utilizado pelo historiador, mas de forma a ainda ter informações relevantes de acordo com a necessidade.

- **Módulos de Processamento de Dados**

  Os módulos de processamento de dados são responsáveis por processar e analisar os dados coletados pelo PIMS. Eles realizam cálculos, agregações e correlações de dados, além de extrair informações significativas dos dados brutos através de técnicas como limpeza de dados, transformação de dados, análise estatística, mineração de dados e machine learning.

- **Interfaces de Visualização e Análise**

  As interfaces de visualização e análise fornecem ferramentas para visualizar os dados processados pelo PIMS. Geralmente apresentam dashboards customizáveis com diferentes tipos de gráficos.

  Podem ser usado para a visualização de tendências de qualidade do produto para identificar problemas ou para monitorar o desempenho da produção em tempo real de forma à identificar gargalos e possibilidades de otimização, por exemplo.

- **Ferramentas de Integração**

  As ferramentas de integração desempenham um papel fundamental no sistema PIMS ao possibilitar a comunicação eficiente com outros sistemas presentes

na planta industrial, como os *Manufacturing Execution Systems* (MES), *Computerized Maintenance Management System* (CMMS) e sistemas de *Enterprise Resource Planning* (ERP).

Essas ferramentas permitem o compartilhamento transparente de dados entre os diferentes sistemas, o que resulta em uma visão integrada e abrangente das operações industriais, promovendo uma visão geral para o gerenciamento e a tomada de decisões na planta

- **Módulos de Segurança e Controle de Acesso**

  Esses módulos são responsáveis por garantir a segurança e a integridade dos dados armazenados e processados pelo PIMS. Eles incluem recursos de autenticação de usuários, controle de acesso, criptografia de dados e auditoria de atividades para proteger os dados contra acessos não autorizados e ataques cibernéticos.

- **Sistema de Backup e Recuperação de Dados**

  Para garantir a confiabilidade e disponibilidade dos dados, os PIMS contam com sistemas de backup e recuperação de dados, geralmente também contando com sistemas de redundância de forma a garantir a recuperação rápida em caso de falhas ou desastres.

3.3 AVEVA PI SYSTEM

Segundo (AVEVA, 2024b) O PI System é uma plataforma de software desenvolvida pela OSI Soft/AVEVA que se destaca como uma das principais soluções para gerenciamento de dados em tempo real e análise de operações industriais. Essa plataforma é projetada para coletar, armazenar, processar e distribuir uma ampla gama de dados provenientes de diversos dispositivos e sistemas dentro de uma planta industrial. Com suas capacidades avançadas de análise de dados e visualização, o PI System permite aos usuários monitorar o desempenho da planta, identificar tendências, detectar anomalias e tomar decisões informadas para otimizar processos e melhorar a eficiência operacional.

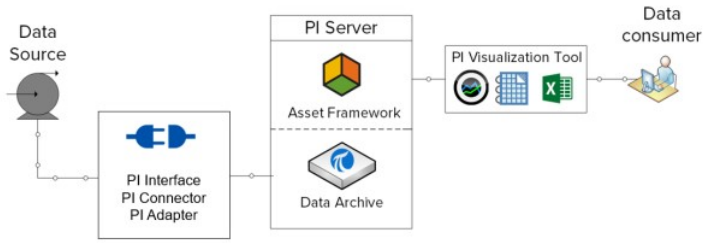
De forma geral, o PI System apresenta um ou mais servidores, chamados PI Server, e que seguem a estrutura geral apresentada na Figura 10.

Nesta seção serão abordados os principais componentes do PI System relacionados com o trabalho desenvolvido.

3.3.1 PI Data Archive

Servindo como componente central do PI System, o PI Data Archive (PI DA) é responsável pelo armazenamento e gerenciamento de grandes volumes de dados

Figura 10 – Estrutura genérica de um PI Server.



Fonte: (AVEVA, 2024a).

historiados. Ele é uma plataforma robusta e escalável para coleta, armazenamento e recuperação eficiente de dados históricos e em tempo real, coletados de diferentes fontes como, por exemplo, sistemas de controle de processos, sensores e dispositivos IoT.

Esse componente utiliza uma arquitetura de banco de dados de séries temporais altamente otimizada, que permite a captura e armazenamento eficiente de grandes volumes de dados de processo com alta taxa de amostragem. Esta arquitetura oferece flexibilidade com configurações single-node, multi-node e distribuída, adaptando-se a diferentes necessidades de implementação. Além disso, oferece recursos avançados de compressão e agregação de dados para minimizar o armazenamento e melhorar o desempenho de consultas.

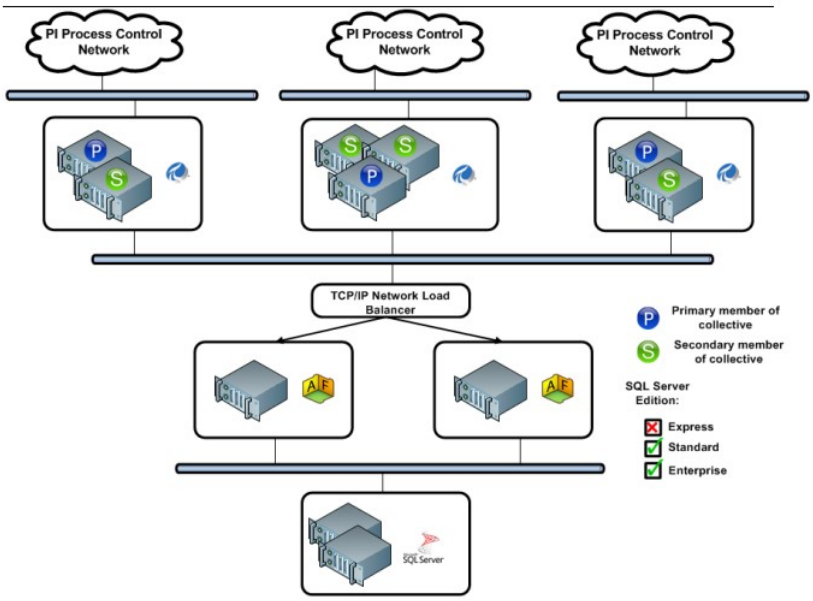
Este componente é projetado para garantir a integridade, confiabilidade e disponibilidade dos dados, com recursos robustos de redundância, replicação e backup. Ele também oferece uma ampla gama de interfaces de programação e ferramentas de análise para facilitar o acesso aos dados e a integração com outras aplicações e sistemas.

A Figura 11 apresenta uma configuração de instalação do PI System contendo diferentes servidores com o PI Data Archive, na topografia apresentada as fontes de dados de cada servidor são diferentes, mas pode existir uma configuração em que uma mesma fonte esteja relacionada com diferentes servidores.

3.3.2 PI Asset Framework

O PI AF é uma estrutura hierárquica que organiza e gerencia informações de ativos em sistemas de automação industrial. Ele fornece uma maneira estruturada de representar ativos, como equipamentos, sistemas e áreas, bem como suas inter-relações e hierarquias. O PI AF permite a criação de modelos de ativos abrangentes, incluindo metadados, hierarquias, relações e atributos, para facilitar a análise e a tomada de decisões em toda a organização.

Figura 11 – Estrutura genérica de um PI Server contendo diferentes Data Archives.



Fonte: (AVEVA, 2024a).

3.3.3 PI System Management Tools

O PI SMT é uma coleção de ferramentas de gerenciamento do PI System que permite aos usuários administrar e configurar diferentes aspectos do sistema. Ele oferece uma interface gráfica para tarefas como gerenciamento de servidores, configuração de pontos de dados, monitoramento de desempenho e diagnóstico de problemas. O PI SMT simplifica as operações de administração do PI System, fornecendo ferramentas intuitivas e eficientes para os usuários.

3.3.4 PI AF SDK

O PI AF SDK é um conjunto de bibliotecas e ferramentas de desenvolvimento que permite aos desenvolvedores criar aplicativos personalizados que interagem com o PI AF. Ele fornece acesso programático aos recursos e funcionalidades do PI AF, permitindo a criação de aplicativos que automatizam tarefas, realizam análises avançadas e integram o PI AF com outros sistemas e tecnologias. O PI AF SDK oferece suporte para várias linguagens de programação, como C, VB.NET e Java.

3.3.5 PI Web API

O PI Web API é uma interface de programação de aplicativos baseada na web que permite acessar e interagir com dados e serviços do PI System por meio de chamadas HTTP. Ele fornece uma maneira simples e flexível de integrar o PI System com

aplicativos da web, dispositivos móveis e outras plataformas de software. O PI Web API permite a consulta de dados de processo, a execução de análises e o gerenciamento de ativos de forma eficiente e segura pela internet.

3.4 PYTHON

A linguagem Python é uma linguagem de programação interpretada, de alto nível e com uma sintaxe simples se comparada com outtras linguagens de programação. Criada por Guido van Rossum e lançada pela primeira vez em 1991, Python ganhou imensa popularidade devido à sua facilidade de uso, versatilidade e vasta gama de bibliotecas e ferramentas disponíveis. (PYTHON. . . , 2024)

Na análise de dados, Python se tornou uma ferramenta indispensável. Sua crescente adoção se deve à sua capacidade de lidar com todas as etapas do processo de análise de dados, desde a coleta até a visualização dos resultados.

A linguagem Python é compatível com a programação orientada à objetos, o que a torna atrativa para diferentes tipos de desenvolvimento de software.

Os próximos itens apresentam de forma resumida as bibliotecas utilizadas no projeto.

- **Pandas**

  A biblioteca Pandas é uma ferramenta poderosa e versátil para análise de dados em Python. Ela fornece estruturas de dados de alto desempenho, chamadas de dataframes, e ferramentas intuitivas para manipular, limpar, transformar e visualizar dados em diversos formatos. (TEAM, 2024)

  Possui como funcionalidades essenciais:

  - **Leitura e Gravação de Dados:** Leitura e escrita de dados de diferentes fontes e formatos como, por exemplo, arquivos CSV, JSON;
  - **Manipulação de Dados:** Proporciona funcionalidades de limpeza, organização, ordenação, filtragem e agregação dos dados contidos nos dataframes com facilidade;
  - **Análise Estatística:** Possui de forma nativa a funcionalidade de realizar cálculos de estatísticas descritivas, como média, mediana, desvio padrão, correlação e regressão linear;
  - **Visualização de Dados:** A partir da biblioteca é possível criar gráficos e visualizações interativas e personalizáveis, podendo também utilizar a estrutura de dataframes com outras bibliotecas;
  - **Integração com outras Bibliotecas:** Integração com outras bibliotecas populares de Python para análise de dados, como NumPy, Matplotlib e Scikit-learn.

- **Dataframes**

  O dataframe é a principal estrutura de dados do Pandas. Ele tem formato bidimensional, similar à uma tabela. Para armazenar dados de forma organizada e eficiente, cada coluna do dataframe possui um tipo de dado específico, como texto, números, datas ou valores booleanos.

  As principais características dos dataframes são:

  - **Colunas:** Nomeadas e com tipos de dados específicos.
  - **Linhas:** Índices que identificam cada linha de forma única.
  - **Valores:** Intersecção entre linhas e colunas, armazenando os dados propriamente dito e em diferentes formatos.
  - **Seleção e Filtragem:** Seleção de subconjuntos de dados por meio de índices, colunas e valores específicos.
  - **Agregação:** Cálculo de estatísticas descritivas e outras operações matemáticas para resumir os dados.
  - **Manipulação:** Adição, remoção e modificação de colunas, linhas e valores de forma eficiente.

- **Python.NET**

  A biblioteca Python.NET é uma ferramenta de integração entre as linguagens Python e .NET, permitindo que desenvolvedores utilizem recursos e bibliotecas de ambas as plataformas de forma conjunta e transparente.

- **Requests**

  A biblioteca Requests é uma ferramenta essencial para lidar com requisições HTTP de forma simples e eficiente em Python. Desenvolvida para facilitar o processo de comunicação com APIs da web e outros serviços online, ela oferece uma interface amigável e intuitiva para enviar solicitações, manipular respostas e trabalhar com dados da web. (REITZ, 2024)

- **Matplotlib**

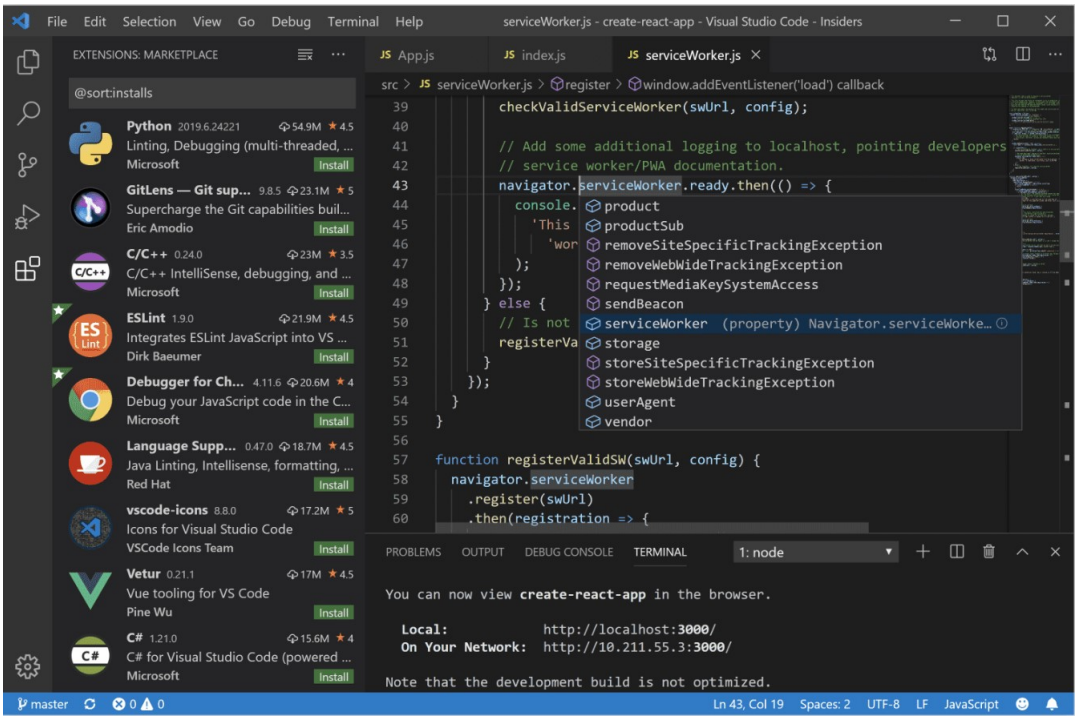
  A biblioteca Matplotlib é uma ferramenta poderosa e de código aberto para visualização de dados em Python (HUNTER, 2007). Criada por John D. Hunter em 2003, ela oferece uma ampla variedade de funcionalidades para criação de gráficos estáticos, interativos e personalizados, permitindo aos desenvolvedores e cientistas de dados explorar e comunicar eficientemente informações contidas em conjuntos de dados.

- **Custom TKinter**

  Segundo (SCHIMANSKY, 2024), a biblioteca CustomTkinter é uma extensão da biblioteca Tkinter padrão do Python, que oferece novos widgets modernos e totalmente personalizáveis para a criação de interfaces de usuário. Esses widgets podem ser utilizados como os widgets normais do Tkinter e também podem ser combinados com elementos padrão do Tkinter.

- **Visual Studio Code**

Figura 12 – Visual Studio Code.



Fonte: (MICROSOFT, 2024).

O Visual Studio Code (VSCode) é um editor de código multiplataforma de código aberto desenvolvido pela Microsoft. É uma ferramenta versátil para desenvolvedores de software, web designers e qualquer usuário que trabalhe com código.

Tem como suas principais funcionalidades:

- **Edição de código:** Suporte para diversas linguagens de programação, com destaque para JavaScript, TypeScript e Python;
- **Depuração:** Ferramentas integradas para depuração de código, incluindo pontos de interrupção, rastreamento de pilha e variáveis de escopo;
- **Controle de versão:** Integração com Git e outros sistemas de controle de versão;

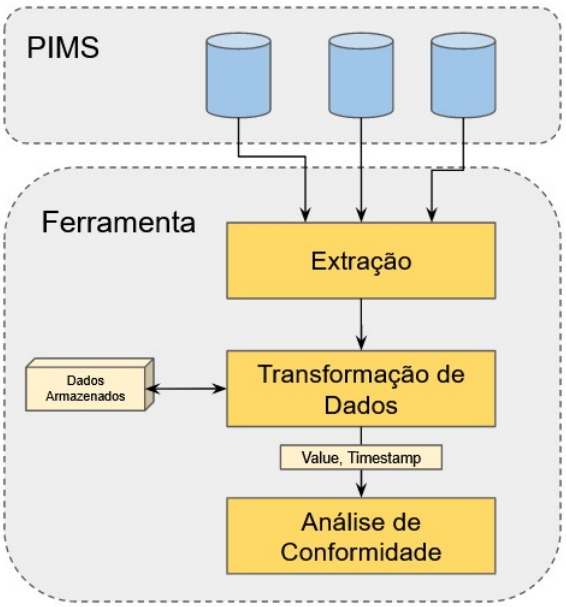
- **Extensões:** Ampla variedade de extensões para adicionar funcionalidades ao editor, como suporte para novas linguagens de programação e formatação de código;
- **Terminal integrado:** Terminal integrado para executar comandos e scripts;
- **Multiplataforma:** Disponível para Windows, macOS e Linux.

Extensões: Ampla variedade de extensões para adicionar funcionalidades ao editor, como suporte para novas linguagens de programação, ferramentas de linting e formatação de código.

4 REQUISITOS

O objetivo deste projeto é desenvolver uma ferramenta capaz de seguir o fluxo de extração de dados do Plant Information Management System (PIMS), transformação de dados e análise de conformidade, conforme apresentado na Figura 13. Essa ferramenta será fundamental para garantir a integridade, qualidade e conformidade dos dados de processo em ambientes industriais, permitindo aos usuários extrair insights valiosos e tomar decisões informadas com base nas informações coletadas.

Figura 13 – Fluxo geral da ferramenta.



Fonte: Projeto.

A norma IEEE Std 830-1998 define requisitos de software como ”condições que o software deve atender ou possuir”. Esses requisitos são classificados em duas classes principais: requisitos funcionais e requisitos não funcionais (IEEE. . . , 1998)

Nas próximas seções serão expostos os requisitos gerais e funcionais que foram considerados para o desenvolvimento da solução. Além disso, será apresentado um diagrama de caso de uso exemplificando o fluxo de uso do sistema.

4.1 REQUISITOS FUNCIONAIS

Os requisitos funcionais (RFs) delineiam as funcionalidades específicas do software e as operações que ele deve realizar. Eles detalham as entradas, saídas, processamentos e comportamentos esperados do sistema.

os requisitos funcionais propostos para se atingir o objetivo do projeto foram discriminados de acordo com as principais etapas de execução que contemplam a

solução proposta, apresentados nas seções seguintes.

4.1.1 Extração de Dados

A ferramenta deve permitir a extração de dados de diferentes PIMS de forma específica, facilitada e permitindo diferentes tipos de conexões, permitindo o gerenciamento de conexões com múltiplas bases de dados, incluindo a capacidade de especificar critérios de seleção dos dados, como, por exemplo, o intervalo de tempo e o tag específico.

4.1.2 Transformação de Dados

Após a extração dos dados, a solução deve ser capaz de transformá-los para um padrão definido. Isso envolve a realização de operações de limpeza, normalização e padronização dos dados, garantindo consistência nos dados extraídos a partir de diferentes métodos ou fontes.

Além disso, os dados transformados devem ser armazenados seguindo uma estrutura definida, de forma a permitir o uso posterior para novas validações ou outros usos específicos.

A solução deve permitir a visualização dos dados transformados tanto após a etapa de extração como após a análise de conformidade, possibilitando ao usuário visualizar informações específicas dos tags de forma individual ou de forma comparativa.

4.1.3 Análise de Conformidade

O sistema deve realizar análises de conformidade entre pares de dados extraídos, conforme definido pelo usuário. Isso inclui a comparação de dados históricos de diferentes fontes, identificando discrepâncias, tendências e padrões relevantes para a análise de conformidade.

Além disso, a ferramenta deve permitir gerar indicadores de forma automática e exportar os insights gerados.

4.2 REQUISITOS NÃO FUNCIONAIS

os requisitos não funcionais (RNFs) estabelecem as características e restrições do software, como desempenho, segurança e usabilidade. Eles definem critérios de qualidade essenciais para o software e foram separados de forma a indicar os parâmetros de desempenho, segurança, usabilidade .

4.2.1 Desempenho

Os requisitos de desempenho estabelecem os padrões de tempo de resposta e escalabilidade esperados pela ferramenta, sendo eles:

- **Tempo de Resposta:** O sistema deve ser capaz de realizar a extração e análise de dados em tempo hábil, garantindo que os resultados estejam disponíveis para os usuários dentro de prazos aceitáveis;
- **Escalabilidade:** O sistema deve ter suporte para lidar com grandes volumes de dados, garantindo que sua performance não seja comprometida mesmo em situações de alta demanda.

4.2.2 Segurança

Os requisitos de segurança definem as medidas necessárias para proteger os dados sensíveis e garantir a integridade do sistema, nesse caso:

- **Autenticação e Autorização:** Deve haver uma forma de solicitar ao usuário credenciais de acesso para realizar a extração dos dados garantindo que apenas usuários autorizados tenham permissão para acessar os dados históricos contidos no PIMS.

4.2.3 Usabilidade

Os requisitos de usabilidade garantem que o sistema seja intuitivo e fácil de usar para os usuários, independentemente de seu nível de conhecimento técnico, incluindo:

- **Interface Intuitiva:** A interface do usuário deve ser projetada de forma a facilitar a interação e a execução das funcionalidades do sistema, proporcionando uma experiência sem a necessidade de conhecimentos específicos sobre o sistema PIMS em si;
- **Suporte Multilíngue:** Visto que a Radix é uma empresa global que possui colaboradores de diversas nacionalidades, a interface do usuário deve estar disponíveis em diferentes idiomas, permitindo que usuários de diferentes origens culturais e linguísticas possam utilizar o sistema de forma eficaz.

4.2.4 Manutenabilidade

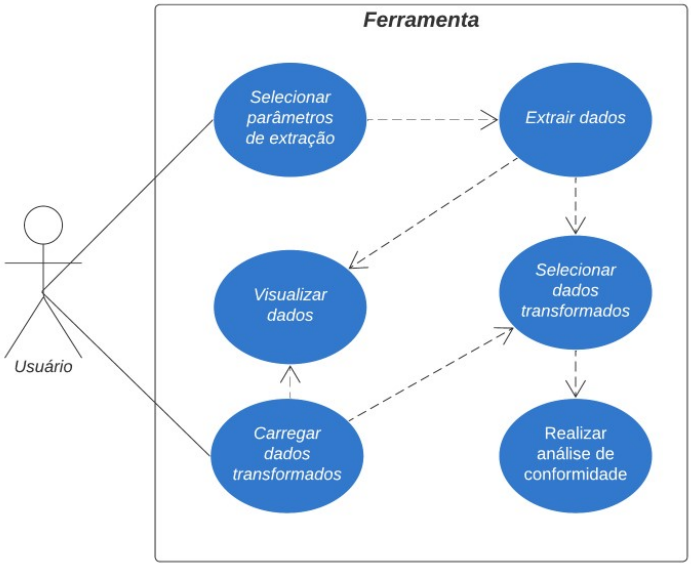
Os requisitos de manutenabilidade visam garantir que o sistema seja fácil de manter, atualizar e testar. Isso inclui:

- **Código Modular e Reutilizável:** O código da ferramenta desenvolvida deve ser modular e reutilizável, facilitando a manutenção, implementação de novas funcionalidades e a atualização do software.

4.3 CASO DE USO

Nesta seção, será descrito o principal caso de uso que representa as interações do usuário com a ferramenta desenvolvida para o projeto. Este caso de uso é essencial para compreender como os usuários interagem com a ferramenta para obter dados do PIMS, visualizar esses dados e realizar análises de conformidade.

Figura 14 – Diagrama de caso de uso geral da ferramenta.



Fonte: Projeto.

4.3.1 Descrição do Caso de Uso

O caso de uso ”Extrair Dados, Visualizar e Analisar” permite que o usuário realize as seguintes etapas:

1. **Selecionar Parâmetros de Extração:** O usuário define os parâmetros necessários para a extração de dados do PIMS, selecionando informações como o modelo de conexão, o intervalo de tempo, as variáveis a serem consideradas e outros critérios relevantes.
2. **Extrair Dados:** Com base nos parâmetros selecionados, a ferramenta extrai os dados relevantes do PIMS.

3. **Visualizar Dados:** Os dados extraídos são apresentados ao usuário na interface de visualização, permitindo que ele explore e analise os dados de forma eficaz.

4. **Selecionar Dados Transformados e Realizar Análise de Conformidade:** O usuário pode optar por realizar uma análise de conformidade selecionando os dados transformados conforme necessário e executando a análise de acordo com a seleção.

4.3.2 Fluxo Alternativo

Além do fluxo principal, o usuário também pode optar por carregar dados já transformados previamente salvos para visualização e análise, eliminando as etapas de seleção de parâmetros de extração e a extração em si.

4.3.3 Pré-condições e Pós-condições

Para utilizar a solução de forma eficaz, o usuário deve ter acesso à ferramenta e permissão para extrair dados do PIMS. Após a conclusão do caso de uso, o usuário poderá visualizar os dados extraídos, realizar análises de conformidade e tomar decisões informadas com base nas informações apresentadas.

5 IMPLEMENTAÇÃO DA FERRAMENTA

Embora o objetivo principal da solução desenvolvida seja realizar a extração de dados de PIMS de uma maneira geral, a ferramenta elaborada teve seu foco no PI System, decisão tomada devido ao fato de este ser uma plataforma amplamente utilizada em nível mundial e devido à vasta experiência da Radix em projetos com esse PIMS.

Neste capítulo serão abordados as etapas de desenvolvimento do projeto em si, apresentando a implementação do software de forma detalhada. Serão discutidas as definições da estrutura do programa, incluindo sua arquitetura, as interfaces e módulos desenvolvidos, além dos padrões de dados estabelecidos. Também serão explorados aspectos relacionados ao código-fonte e à interface do usuário, proporcionando uma visão abrangente do processo de implementação da ferramenta.

As tecnologias utilizadas durante a implementação estão descritas no Capítulo 3.

5.1 ARQUITETURA DO SOFTWARE

O sistema foi desenvolvido seguindo o padrão arquitetural *Model-View-Controller* (MVC), segundo (FOWLER, 2024) um padrão arquitetural amplamente utilizado no desenvolvimento de software que em três componentes principais.

- **Modelo (*Model*):** Responsável pela representação dos dados da aplicação e pela lógica de negócios. Ele manipula os dados e notifica a visão sobre qualquer mudança nesses dados.
- **Visão (*View*):** É responsável pela apresentação dos dados ao usuário e pela interação com ele. Ela exibe as informações do modelo e encaminha as ações do usuário para o controle.
- **Controle (*Controller*):** Atua como intermediário entre o modelo e a visão. Ele interpreta as interações do usuário e as transforma em comandos para o modelo ou para a visão. Ele também pode atualizar o modelo com base nas entradas do usuário e modificar a visão conforme necessário.

Para facilitar o desenvolvimento e a manutenção dos códigos, eles foram divididos em diferentes pastas dentro do VSCode, conforme apresentado na Figura 15. Nas seções seguintes estão apresentados as classes dentro de suas respectivas pastas e uma breve descrição sobre suas funcionalidades.

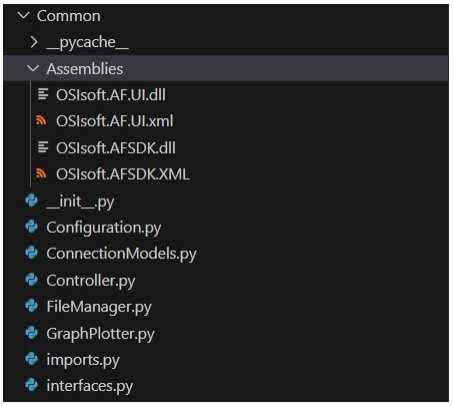
Figura 15 – Separação de pastas dentro do projeto.

Fonte: Projeto.

5.1.1 Pasta Common

Na pasta ”Common” estão localizadas as classes essenciais para o funcionamento do sistema, além dos *assemblies* necessários para o uso de bibliotecas ou funcionalidades não existentes no python.

Figura 16 – Arquivos presentes na pasta ”Common”.



Fonte: Projeto.

5.1.1.1 Configuration.py

Esta classe é responsável por ler um arquivo JSON chamado ”Configuration.json”, descrito posteriormente na seção 5.2.1, que contém informações gerais do software, como fuso horário, formato de data, modelos de conexão disponíveis e idioma. Os modelos de conexão são representados por um dicionário, contendo o nome, metadados de criação, nome da classe de conexão e extração de dados, parâmetros utilizados por essa classe e o nome de uma outra classe responsável por transformar os dados. Seu código fonte está apresentado na Figura 17.

Figura 17 – Código-fonte da classe Configuration.py.

Common > Configuration.py > ...

```python
from imports import os, json

class Configuration():
    def __init__(self, path=os.getcwd()):
        self.path = path
        config = self.get_json_info("Configuration.json")
        self.time_zone = config["Timezone"]
        self.date_format = config["Date Format"]
        self.available_connections_models = config["Available Connections Models"]
        self.language = config["Language"]

    def get_json_info(self, file_name):
        try:
            file_path = os.path.join(self.path, file_name)
            with open(file_path, 'r') as file:
                json_content = json.load(file)
            return json_content
        except FileNotFoundError:
            print(f"The file {file_path} was not found.")
            return
        except Exception as e:
            print(f"An error occurred while loading the JSON file: {e}")
            return
```

Fonte: Projeto.

5.1.1.2 ConnectionModels.py

Figura 18 – Código-fonte da classe ConnectionModels.py.

Common > ConnectionModels.py > ConnectionModels

```python

class ConnectionModels():
    def __init__(self, config):
        self.available_connections_models = config.available_connections_models
        self.connection_model_names = self.get_connection_model_names()

    def get_connection_model_info(self, name, info=[]):
        for model in self.available_connections_models:
            if model['Name'] == name:
                return model if not info else model[info]
        return None

    def get_connection_model_names(self):
        connection_model_names = []
        for model in self.available_connections_models:
            connection_model_names.append(model['Name'])
        return connection_model_names
```

Fonte: Projeto.

Apresentada na Figura 18, é a classe responsável por gerar os modelos de conexão com base nas informações obtidas pelo Configuration.

5.1.1.3 Controller.py

Com parte de seu código representado na figura 19, esta classe desempenha o papel de controle, integrando as classes do modelo com a view. Possui atributos como

Figura 19 – Código-fonte da classe Controller.py.

Fonte: Projeto.

conexão_models (instância de ConnectionModels), active_connections (lista de conexões ativas), active_connections_name (atualização dos nomes das conexões ativas), active_connections_info (informações das conexões ativas), extracted_data (dados extraídos), extracted_data_ids (IDs dos dados extraídos), file_manager (instância de FileManager), analysis (instância de análise) e message (mensagem de comunicação).

5.1.1.4 FileManager.py

Figura 20 – Código-fonte da classe FileManager.py.

Fonte: Projeto.

Classe responsável por ler e criar arquivos, tanto referentes aos dados extraídos e transformados quanto aos modelos de conexões criados. Possui diversas funções (conforme Figura 20) para tratar diferentes tipos de dados e gerar arquivos nos padrões definidos no projeto.

5.1.1.5 interfaces.py

Figura 21 – Código-fonte da classe interfaces.py.

```
Common > interfaces.py > ...
1   from imports import ABC, abstractmethod, final, datetime, timezone
2
3   class ConnectionInterface(ABC): ···
25
26
27  class DataTransformInterface(ABC): ···
46
47
48  class Analysis(ABC): ···
```

Fonte: Projeto.

Define os padrões a serem seguidos pelas classes de análise, conexão e transformação. Os padrões estão descritos na seção 5.3

5.1.1.6 GraphPlotter.py

Figura 22 – Código-fonte da classe GraphPlotter.py.

```
Common > GraphPlotter.py > ...
1   import matplotlib.pyplot as plt
2   from matplotlib.dates import DateFormatter
3   import pandas as pd
4
5   class GraphPlotter:
6       @staticmethod
7       def plot_trend(*dataframes):
8           fig, ax = plt.subplots(figsize=(10,6))
9
10          for _df in dataframes:
11              df = _df.copy()
12              df["Value"] = pd.to_numeric(df["Value"], errors='coerce').dropna()
13              df['Timestamp'] = pd.to_datetime(df['Timestamp'], format='%d/%m/%Y %H:%M:%S')
14              df.set_index('Timestamp', inplace=True)
15              plt.plot(df.index, df['Value'], marker='o', linestyle='-', label=_df.name)
16
17          plt.xlabel('Timestamp')
18          plt.ylabel('Value')
19          plt.grid(True)
20          plt.xticks(rotation=45)
21          date_format = DateFormatter("%d/%m/%y %H:%M")
22          ax.xaxis.set_major_formatter(date_format)
23          plt.legend()
24          plt.tight_layout()
25          plt.show()
26
27          return fig
```

Fonte: Projeto.

Esta classe é responsável por plotar gráficos dos dados transformados, utiliza as bibliotecas pandas e matplotlib.

5.1.1.7 imports.py

Classe responsável por importar todas as bibliotecas utilizadas pela ferramenta, de forma a centralizar todas as informações em um só arquivo.

5.1.2 Pasta View

Nesta pasta estão armazenados os arquivos referentes a parte de *View* da arquitetura, ou seja, da interface gráfica que o usuário tem acesso

Figura 23 – Arquivos presentes na pasta ”View”.

Fonte: Projeto.

5.1.2.1 View.py

Classe responsável pela geração da *graphical user interface* (GUI), utiliza a biblioteca Custom Tkinter, descrita na seção 3.4, e permite que o usuário interaja com os fluxos de extração e análise de dados.

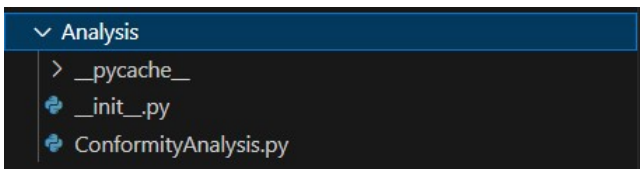
5.1.2.2 Auxiliar.py

Classe auxiliar que possui alguns elementos gráficos utilizados pela classe View.py, a separação por classe é uma boa prática para reduzir o tamanho da classe principal e permitir uma manuntenção facilitada do código. Também utiliza a biblioteca Custom Tkinter.

5.1.3 Pasta Analysis

Visando atender o requisito de manutenabilidade e facilitar a implementação de novas análises, a pasta ”Analysis” contém as classes responsáveis pelas análises executadas pela ferramenta, no caso do presente projeto foi criada somente uma classe referente à análise de Conformidade.

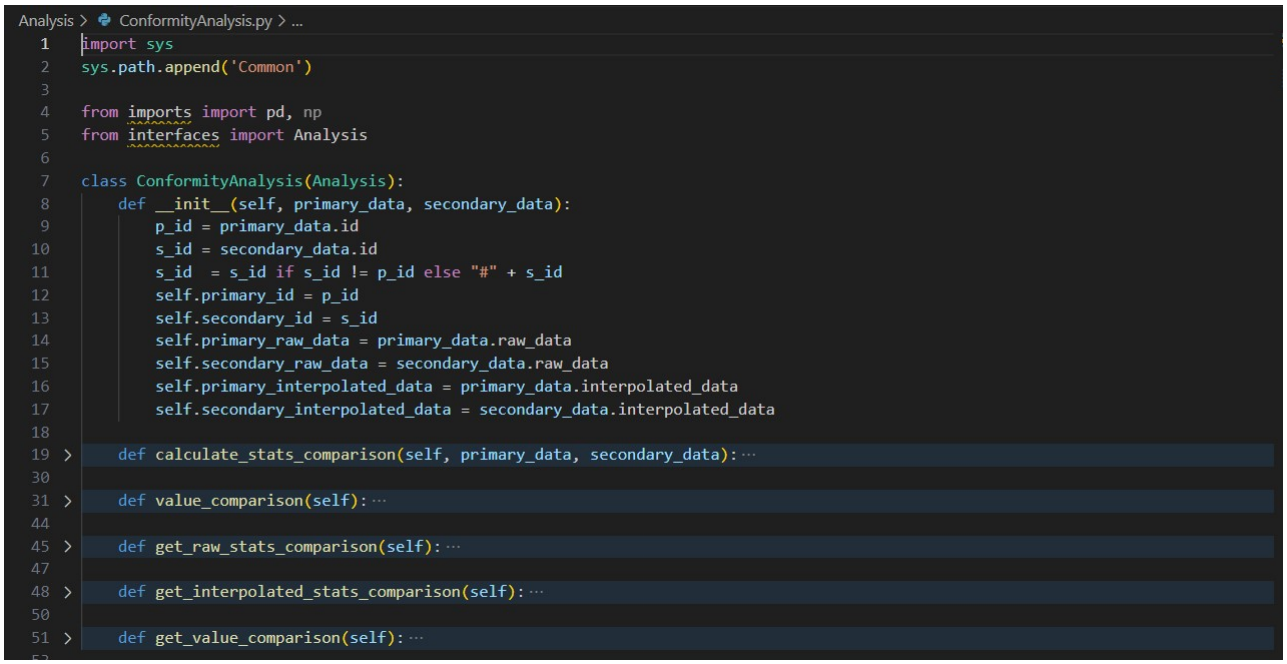
Figura 24 – Arquivos presentes na pasta ”Analysis”.



Fonte: Projeto.

5.1.3.1 ConformityAnalysis.py

Figura 25 – Código-fonte da classe ConformityAnalysis.py.



```
Analysis > ConformityAnalysis.py > ...
1   import sys
2   sys.path.append('Common')
3
4   from imports import pd, np
5   from interfaces import Analysis
6
7   class ConformityAnalysis(Analysis):
8       def __init__(self, primary_data, secondary_data):
9           p_id = primary_data.id
10          s_id = secondary_data.id
11          s_id  = s_id if s_id != p_id else "#" + s_id
12          self.primary_id = p_id
13          self.secondary_id = s_id
14          self.primary_raw_data = primary_data.raw_data
15          self.secondary_raw_data = secondary_data.raw_data
16          self.primary_interpolated_data = primary_data.interpolated_data
17          self.secondary_interpolated_data = secondary_data.interpolated_data
18
19      def calculate_stats_comparison(self, primary_data, secondary_data): ···
30
31      def value_comparison(self): ···
44
45      def get_raw_stats_comparison(self): ···
47
48      def get_interpolated_stats_comparison(self): ···
50
51      def get_value_comparison(self): ···
```

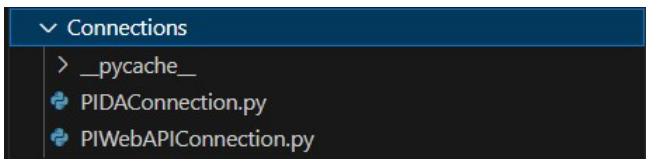
Fonte: Projeto.

Classe responsável por realizar a análise de conformidade propriamente dita, seguindo os padrões estabelecidos pela interface descrita na seção 5.3.

5.1.4 Pasta Connections

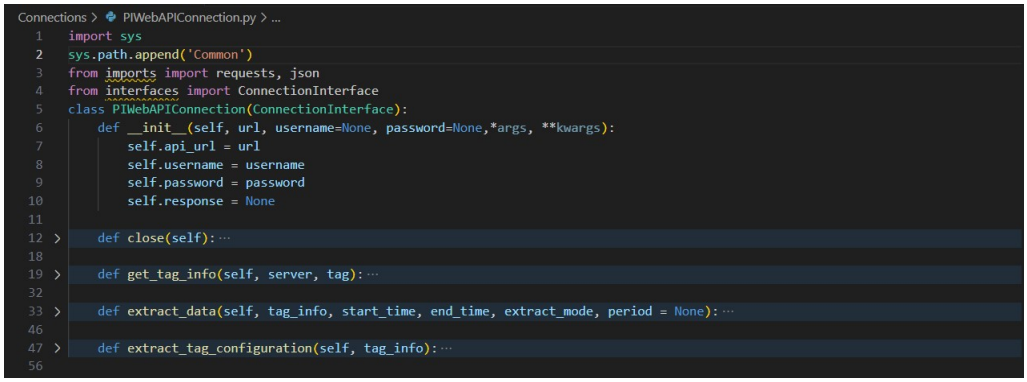
De forma semelhante à pasta de análises e buscando atender o requisito de manutenabilidade, a pasta ”Connections” contém as classes responsáveis pelos modelos de conexão com os PIMS, para a conexão com o PI System foram desenvolvidois métodos de conexão, um que utiliza o PI AF SDK e outro que utiliza o PI Web API.

Figura 26 – Arquivos presentes na pasta ”Connections”.



Fonte: Projeto.

Figura 27 – Código-fonte da classe PIWebAPIConnection.py.



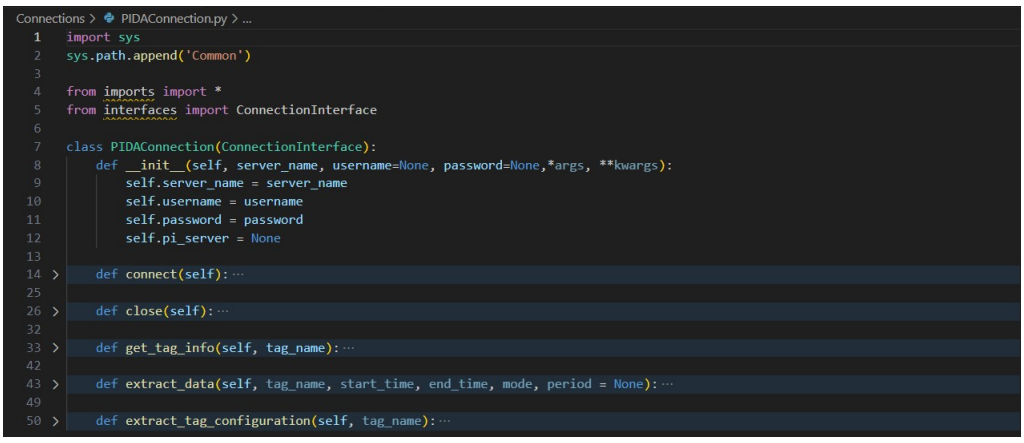
Fonte: Projeto.

5.1.4.1 PIWebAPIConnection.py

Classe apresentada na Figura 27, responsável por realizar a conexão com o PI System através do PI Web API, segue os padrões estabelecidos pela interface descrita na seção 5.3 e como modelo de conexão, permitindo que o usuário estabeleça a conexão com diferentes servidores utilizando esse método.

5.1.4.2 PIDAConnection.py

Figura 28 – Código-fonte da classe PIDAConnection.py.



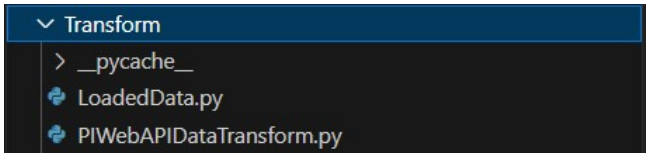
Fonte: Projeto.

Apresentada na Figura 28, classe desenvoldia para realizar a conexão com o PI System através do PI AF SDK, segue os padrões estabelecidos pela interface descrita na seção 5.3 e como modelo de conexão, permitindo que o usuário estabeleça a conexão com diferentes servidores utilizando esse método.

5.1.5 Pasta Transform

Por fim, a pasta transform centraliza todas as classes de transformação de dados que tem como objetivo realizar a manipulação de dados para os padrões determinados na seção 5.2. No presente projeto foram desenvolvidas três classes, sendo a classe ”LoadedData” responsável por ler arquivos que contenham dados transformados em outros momentos e as outras duas classes responsáveis por transformar dados após a extração.

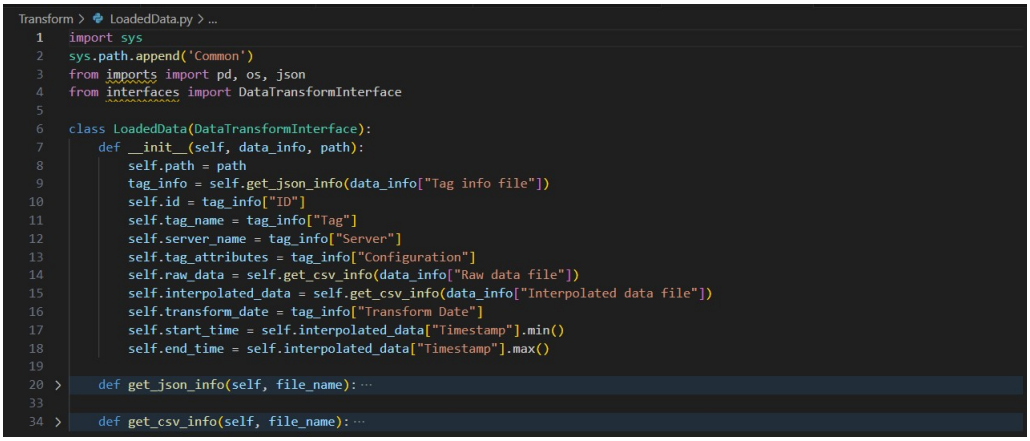
Figura 29 – Arquivos presentes na pasta ”Transform”.



Fonte: Projeto.

5.1.5.1 LoadedData.py

Figura 30 – Código-fonte da classe LoadedData.py.

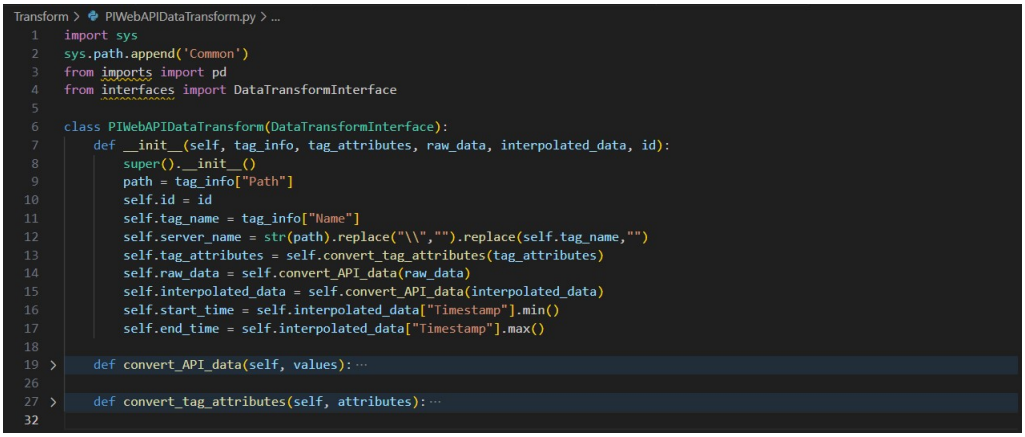


```
Transform > LoadedData.py > ...
1   import sys
2   sys.path.append('Common')
3   from imports import pd, os, json
4   from interfaces import DataTransformInterface
5
6   class LoadedData(DataTransformInterface):
7       def __init__(self, data_info, path):
8           self.path = path
9           tag_info = self.get_json_info(data_info["Tag info file"])
10          self.id = tag_info["ID"]
11          self.tag_name = tag_info["Tag"]
12          self.server_name = tag_info["Server"]
13          self.tag_attributes = tag_info["Configuration"]
14          self.raw_data = self.get_csv_info(data_info["Raw data file"])
15          self.interpolated_data = self.get_csv_info(data_info["Interpolated data file"])
16          self.transform_date = tag_info["Transform Date"]
17          self.start_time = self.interpolated_data["Timestamp"].min()
18          self.end_time = self.interpolated_data["Timestamp"].max()
19
20      def get_json_info(self, file_name): ···
33
34      def get_csv_info(self, file_name): ···
```

Fonte: Projeto.

Classe apresentada na Figura 30, responsável por realizar a leitura de arquivos gerados por extrações realizadas em outros momentos, permite que o usuário resgate dados extraídos anteriormente e segue os padrões estabelecidos pela interface descrita na seção 5.3. Das classes de transformação desenvolvidas, essa é a única que não faz parte de um modelo de conexão e é utilizada de forma independente.

Figura 31 – Código-fonte da classe PIWebAPIDataTransform.py.



Fonte: Projeto.

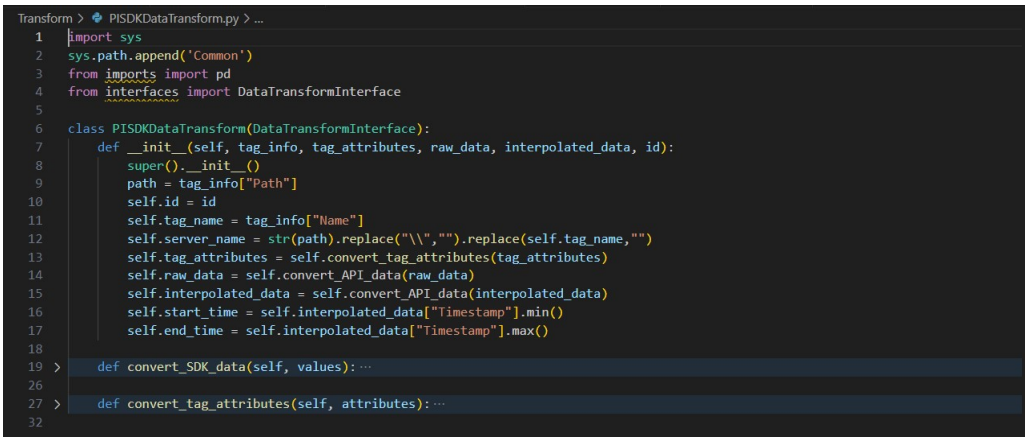
5.1.5.2 PIWebAPIDataTransform.py

A classe PIWebAPIDataTransform, complementa a classe de extração que utiliza o PI Web API para extrair dados e transforma os dados extraídos nos padrões adequados e estabelecidos para o projeto. Faz parte de um modelo de conexão e é utilizada de forma automática após a extração de valores.

Conforme apresentado na Figura 31, segue os padrões estabelecidos pela interface descrita na seção 5.3.

5.1.5.3 PISDKDataTransform.py

Figura 32 – Código-fonte da classe PISDKDataTransform.py.



Fonte: Projeto.

De forma semelhante à PIWebAPIDataTransform.py, a classe apresentada na

Figura 32 faz parte de um modelo de conexão com a classe PIDAConnection.py, transformando os dados após a extração via PI AF SDK.

5.2 PADRÕES DE DADOS

Este capítulo analisa os padrões de dados adotados no projeto, abordando desde a estruturação do arquivo de configuração até a formatação dos dados brutos e interpolados. Esses padrões são essenciais para garantir a consistência, eficiência e facilidade de manutenção do sistema, promovendo uma experiência satisfatória tanto para os desenvolvedores quanto para os usuários finais.

Com exceção do arquivo de configuração, ao se utilizar a ferramenta, os arquivos seguindo os padrões estabelecidos são gerados, de forma a permitir a consulta das informações posteriormente e realizar novas análises ou comparações utilizando um conjunto de dados extraídos previamente.

Para abordar os padrões de dados estabelecidos para o projeto, podemos dividir essa seção em quatro partes principais, cada uma descrevendo um tipo específico de padrão de dados, apresentadas nas próximas seções.

5.2.1 Arquivo de Configuração (Configuration.json)

Figura 33 – Exemplo de arquivo Configuration.json.

Fonte: Projeto.

Foi definido que o arquivo de configuração é estruturado no formato JSON e contém informações gerais do software, como fuso horário, formato de data e os modelos de conexões disponíveis para extração (conforme Figura 33). Cada modelo

de conexão é representado por um objeto dentro da lista ”Available Connection Models”, contendo atributos como nome, data de criação, nome da classe de conexão, parâmetros necessários e classe de transformação de dados associada.

5.2.2 Arquivo de Conexões (connections.json)

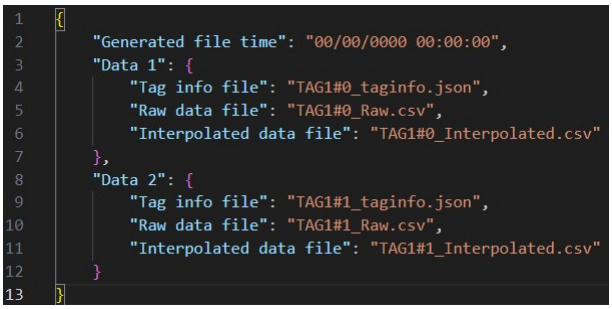
Figura 34 – Exemplo de arquivo connections.json.

Fonte: Projeto.

A Figura 34 apresenta um exemplo de arquivo de conexões, também formatado em JSON e com objetivo de armazenar os modelos de conexão criados pelo usuário através da interface gráfica (GUI). Cada modelo de conexão é representado por um objeto dentro da lista, contendo atributos como nome, modelo de conexão utilizado e os parâmetros específicos fornecidos pelo usuário.

5.2.3 Arquivo de Fonte de Análise (analysis_source.json)

Figura 35 – Exemplo de arquivo connections.json.



Fonte: Projeto.

Este arquivo contém informações sobre o momento de geração dos arquivos de fonte e faz referência aos arquivos que contêm informações mais detalhadas, conforme exemplificado na Figura 35. Ele é estruturado em JSON e inclui o tempo de geração dos arquivos, bem como informações sobre os arquivos de dados brutos e interpolados

associados a cada conjunto de dados, sendo utilizado pela classe LoadedData.py para importar dados extraídos em momentos anteriores de forma adequada.

5.2.4 Arquivos de Informações de Tags e Dados

Os arquivos com sufixo "_taginfo.json" contêm informações detalhadas sobre cada tag extraído, incluindo seu ID, servidor, nome da tag e configurações associadas. Por outro lado, os arquivos CSV com sufixos "_Raw" e "_Interpolated" contêm os dados brutos e interpolados, respectivamente, no formato Value, Timestamp.

A adoção do padrão "Value, Timestamp" é comentada na seção 5.3.2.

5.3 PADRÕES DE INTERFACE

Durante o desenvolvimento da ferramenta, foram estabelecidos padrões de interface para diferentes tipos de classes, como conexão, transformação de dados e análise. Esses padrões foram projetados para permitir que novas classes sejam facilmente integradas à ferramenta, desde que sigam as interfaces definidas. Isso promove a extensibilidade da ferramenta, facilitando a adição de novos recursos e funcionalidades.

Nas interfaces foram definidos métodos do tipo abstrato, que necessariamente precisam ser implementados na classe, métodos livres, que não necessariamente precisam ser implementados e métodos finais, que definem na própria interface qual é a função e seu retorno.

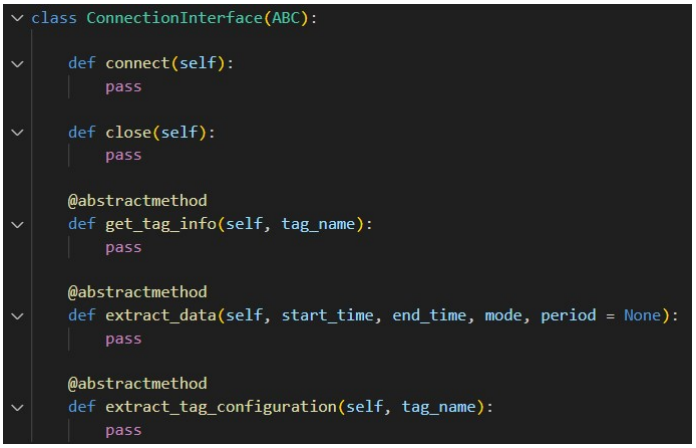
Nas próximas seções as interfaces são apresentadas destacando os seus métodos.

5.3.1 Interface de Conexão (ConnectionInterface)

A interface de conexão, apresenta na Figura 36, define os métodos necessários para estabelecer e encerrar conexões com o PIMS. As classes que implementam esta interface devem fornecer métodos para conectar-se ao sistema, fechar a conexão, obter informações sobre tags específicas, extrair dados brutos e configurar tags.

- `connect()`: Método para estabelecer a conexão com o sistema.
- `close()`: Método para encerrar a conexão com o sistema.
- `get_tag_info(tag_name)`: Método para obter informações sobre uma tag específica.
- `extract_data(start_time, end_time, mode, period=None)`: Método para extrair dados brutos do sistema dentro de um intervalo de tempo especificado.

Figura 36 – Interface de Conexão.



```python
class ConnectionInterface(ABC):

    def connect(self):
        pass

    def close(self):
        pass

    @abstractmethod
    def get_tag_info(self, tag_name):
        pass

    @abstractmethod
    def extract_data(self, start_time, end_time, mode, period = None):
        pass

    @abstractmethod
    def extract_tag_configuration(self, tag_name):
        pass
```

Fonte: Projeto.

- `extract_tag_configuration(tag_name)`: Método para extrair a configuração de uma tag específica do sistema.

A classe PIWebAPIConnection, apresentada na Figura 27, é um exemplo de implementação da interface de conexão definida anteriormente. A classe utiliza a biblioteca de requests do Python para estabelecer a conexão com o PI Web API e realizar operações de extração de dados.

As Figuras 38, 39, 40 e 41 apresentam as implementações dos métodos da classe, contendo as funções necessárias para que a PIWebAPIConnection esteja de acordo com a interface e os padrões estabelecidos.

5.3.2 Interface de Transformação de Dados (DataTransformInterface)

A interface de transformação de dados define os métodos necessários para processar e transformar os dados extraídos pelo sistema. As classes que implementam esta interface devem fornecer métodos para inicializar os dados, realizar transformações específicas e retornar os dados transformados.

Para uma melhor robustez dos processos executados pela ferramenta, foi definido que os dados históricos extraídos sempre devem ser transformados de forma a conter as informações de ”Value e Timestamp” em colunas, sendo utilizados nas estruturas gerais do código através da criação de dataframes da biblioteca pandas, conforme mencionado na seção 3.4.

- `__ini__(...)`: Método de inicialização da classe.
- `return_data()`: Método para retornar os dados transformados em um formato específico.

Figura 37 – Implementação dos métodos na classe `PIWebAPIConnection`.

```python
def close(self):
    try:
        self.response.close()
        print("Connection closed.")
    except Exception as e:
            print(f"Error closing connection: {e}")
```

Figura 38 – Exemplo do método `close()`.

```python
def get_tag_info(self, server, tag):
    if self.response != None:
        self.close()
    full_url = self.api_url + "points?path=\\\\" + server + "\\" + tag
    try:
        self.response = requests.get(full_url,
                                     auth=(self.username,
                                         self.password),
                                     verify=False,
                                     timeout=1000)
    except Exception as e:
            print(f"Error finding tag information: {e}")
    return json.loads(self.response.content)
```

Figura 39 – Exemplo do método `get_tag_info()`.

```python
def extract_data(self, tag_info, start_time, end_time, extract_mode, period = None):
    try:
        self.response = requests.get(tag_info["Links"][extract_mode],
                                     auth=(self.username, self.password),
                                     verify=False,
                                     params={'startTime': start_time,
                                             'endTime': end_time,
                                             'interval': period,
                                             'maxCount':1000000},
                                     timeout=1000)
    except Exception as e:
        print(f"Error finding tag information: {e}")
    return json.loads(self.response.content)
```

Figura 40 – Exemplo do método `extract_data()`.

```python
def extract_tag_configuration(self, tag_info):
    try:
        self.response = requests.get(tag_info["Links"]["Attributes"],
                                     auth=(self.username, self.password),
                                     verify=False,
                                     timeout=1000)
    except Exception as e:
        print(f"Error finding tag information: {e}")
    return json.loads(self.response.content)["Items"]
```

Figura 41 – Exemplo do método `extract_tag_configuration()`

Fonte: Projeto.

Figura 42 – Interface de Transformação de Dados.

```python
class DataTransformInterface(ABC):
    @abstractmethod
    def __init__(self, tag_info = None, tag_attributes = None, raw_data = None, interpolated_data = None):
        self.id = None
        self.tag_name = None
        self.server_name = None
        self.tag_attributes = None
        self.raw_data = None
        self.interpolated_data = None
        self.transform_date = str(datetime.now(timezone.utc).strftime(date_format))

    @final
    def return_data(self):
        data_dic = {"ID":self.id,
                    "Server":self.server_name,
                    "Tag":self.tag_name,
                    "RawData":self.raw_data,
                    "InterpolatedData":self.interpolated_data}
        return data_dic
```

Fonte: Projeto.

A classe PIWebAPITransform, apresentada na Figura 31, exemplifica uma interface de transformação de dados. A classe utiliza a biblioteca pandas para converter os dados para o formato padronizado.

É possível observar que a classe `return_data()` não é implementada na classe, visto que sua definição está atrelada à interface. As Figuras 44 e 45 apresentam as implementações dos métodos da classe.

Figura 43 – Implementação dos métodos na classe `PIWebAPITransform`.

```
def convert_API_data(self, values):
    df = pd.DataFrame(values["Items"], columns=['Value','Timestamp'])
    df['Timestamp'] = pd.to_datetime(df['Timestamp'])
    if not df['Timestamp'].empty:
        df['Timestamp'] = df['Timestamp'].dt.tz_convert(time_zone).dt.strftime(date_format)
    return df
```

Figura 44 – Exemplo de método de conversão de dados.

```
def convert_tag_attributes(self, attributes):
    tag_attributes = {}
    for attribute in attributes:
        tag_attributes[attribute["Name"]]=attribute["Value"]
    return tag_attributes
```

Figura 45 – Exemplo de método de manipulação de dados.

Fonte: Projeto.

5.3.3 Interface de Análise (Analysis)

Figura 46 – Interface de Conexão.

```
class Analysis(ABC):
    @abstractmethod
    def get_raw_stats_comparison(self):
        pass

    @abstractmethod
    def get_interpolated_stats_comparison(self):
        pass

    @abstractmethod
    def value_comparison(self):
        pass

    def generate_insights(self):
        pass
```

Fonte: Projeto.

A interface de análise define os métodos necessários para realizar análises estatísticas e comparações com os dados extraídos pelo sistema. As classes que implementam esta interface devem fornecer métodos para calcular estatísticas sobre os dados brutos e interpolados, bem como comparar os valores entre diferentes conjuntos de dados.

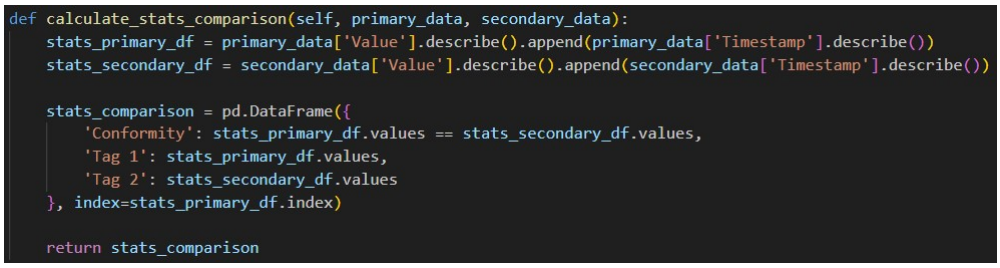
- `get_raw_stats_comparison()`: Método para obter estatísticas sobre os dados brutos.
- `get_interpolated_stats_comparison()`: Método para obter estatísticas sobre os dados interpolados.
- `value_comparison()`: Método para comparar os valores entre diferentes conjuntos de dados.
- `generate_insights(self)()`: Método para gerar insights a partir das análises realizadas.

Esses padrões de interface foram desenvolvidos com o objetivo de facilitar a integração de novas funcionalidades à ferramenta, promovendo a coesão e a modularidade do código. Ao seguir esses padrões, novas classes podem ser facilmente implementadas na ferramenta, proporcionando maior flexibilidade e extensibilidade ao sistema.

A classe ConformityAnalysis, apresentada na Figura 25, exemplifica a classe desenvolvida utilizando a interface de análise. A classe é responsável por realizar análises de conformidade entre dois conjuntos de dados, representados por instâncias de objetos de alguma classe que respeite a interface DataTransformInterface e que contenha informações sobre dados brutos, interpolados e do tag propriamente dito.

As Figuras 47 e 48 apresentam métodos desenvolvidos para a classe, de forma a analisar os dados extraídos e gerar indicadores de conformidade.

Figura 47 – Método de levantamento de dados estatísticos da classe ConformityAnalysis.



Fonte: Projeto.

O método `calculate_stats_comparison()`, por exemplo, realiza a comparação entre pares de dados e retorna as estatísticas de comparação dos dados brutos, para isso ele utiliza um método da biblioteca pandas que retorna o seguinte conjunto de informações:

- **count**: O número de elementos não nulos no conjunto de dados.
- **mean**: A média dos valores no conjunto de dados.
- **std**: O desvio padrão dos valores no conjunto de dados.
- **min**: O valor mínimo no conjunto de dados.
- **25%**: O primeiro quartil dos valores no conjunto de dados.
- **50%**: O segundo quartil dos valores no conjunto de dados (mediana).
- **75%**: O terceiro quartil dos valores no conjunto de dados.

Figura 48 – Método de comparação de dados da classe ConformityAnalysis.

```
def value_comparison(self):
    df_merged = pd.merge(self.primary_raw_data, self.secondary_raw_data, on='Timestamp', suffixes=('_df1', '_df2'))
    df_merged['Conformity'] = df_merged['Value_df1'] == df_merged['Value_df2']
    df_merged['Error'] = abs(df_merged['Value_df1'] - df_merged['Value_df2'])

    comparison = {
        "Simultaneous values": df_merged.shape[0],
        "Similar values": df_merged['Conformity'].sum(),
        "Different values": df_merged.shape[0]-df_merged['Conformity'].sum(),
        "Average error": df_merged['Error'].mean()
    }

    return comparison
```

Fonte: Projeto.

- **max**: O valor máximo no conjunto de dados.

Já o método `value_comparison()` realiza uma comparação direta dos valores dos dados brutos entre os conjuntos primário e secundário, calculando o número de valores simultâneos, valores semelhantes, valores diferentes e a média do erro absoluto.

Esses dados são utilizados pela classe para gerar análises de conformidade levando em conta certas diretrizes definidas no código como, por exemplo:

- Se os dados possuem o mesmo número de elementos no dataframe, seus valores estatísticos são semelhantes e a comparação direta indica que não existem valores diferentes e uma média de erro absoluto igual à zero, então os dados estão em conformidade.
- Por outro lado, se o que está exposto acima não é verdadeiro, a ferramenta considera que os dados não estão conformes e realiza algumas análises como, por exemplo:
  - Se um conjunto de dados possui mais elementos do que o outro, porém na comparação direta dos valores brutos o erro é zero, isso pode indicar que a frequência de coleta dos dados está diferente, visto que o conjunto de dados maior (que possui maior frequência) possui os mesmos valores do conjunto de dados menor, ou seja, um conjunto está armazenando os valores de ”x” em ”x” período e o outro conjunto só em múltiplos de ”x”. Um caso hipotético é um tag estar recebendo valores da interface de trinta em trinta segundos e o outro de um em um minuto. Nesse caso a indicação da ferramenta seria conferir a configuração da interface de coleta.
  - Se um conjunto de dados possui mais elementos do que o outro, a a comparação direta dos valores brutos o erro é diferente à zero, mas os dados

estatísticos não estão tão diferentes entre si, isso pode indicar que os tags possuem diferentes taxas de compressão e um conjunto está deixando de armazenar certos valores mas não com um impacto tão grande. A indicação da ferramenta seria verificar a configuração do tag no PI SMT.

5.4 MODELOS DE CONEXÃO

Para o projeto, foram definidos o que seriam modelos de conexão, entidades que desempenham um papel fundamental no processo de extração de dados do PIMS. Cada modelo de conexão consiste em um par de classes que respeitam os padrões apresentados na seção anterior: uma classe de conexão, responsável por estabelecer a comunicação com o PIMS e extrair os dados, e uma classe de transformação, responsável por processar e preparar os dados para análise posterior.

Esses modelos são projetados para fornecer uma estrutura padronizada e modular que facilita a integração de diferentes sistemas de gerenciamento de dados com a ferramenta de extração e análise. Ao separar a lógica de conexão e extração da lógica de transformação de dados, os modelos de conexão permitem uma maior flexibilidade e reusabilidade do código.

Os modelos de conexão são utilizados para realizar a extração dos dados do PIMS, seguindo uma abordagem consistente e eficiente. Cada modelo é projetado para lidar com os requisitos específicos do sistema de destino, garantindo que os dados sejam extraídos de forma precisa, confiável e transformados posteriormente de acordo com o padrão definido.

Um exemplo de modelo de conexão desenvolvido consiste no par de classes exemplificados nas seções 5.3.1 e 5.3.2, que permitem a conexão com o PI System via PI Web API.

5.5 ANÁLISE DE CONFORMIDADE

Para verificar a conformidade entre os dados extraídos, foram definidas algumas diretrizes que a ferramenta considera ao realizar a análise de conformidade, de forma a indicar se os dados estão de acordo com o esperado ou não.

De forma geral, os dados extraídos contém informações estatísticas (apresentadas na seção 5.3.3), informações da configuração do tag no PI System, dados brutos e dados interpolados. Durante a análise de conformidade o software realiza comparações entre os conjuntos de dados de cada tag extraído de forma a realizar algumas verificações como, por exemplo, verificar se os dois tags possuem o mesmo desvio padrão, média, mediana, se possuem os mesmos valores ponto a ponto quando interpolados ou se os dados brutos de mesmo *timestamp* possuem o mesmo valor.

Assim, a ferramenta é capaz de identificar determinados padrões de comparação e classificar a conformidade do par de tags. O quadro 2 apresenta alguns exemplos de diretrizes consideradas pelo software.

Quadro 2 – Exemplos de diretrizes de conformidade.

Classificação	**Diretriz**
Conforme	Os tags possuem os mesmos valores estatísticos e os valores brutos são iguais em valor e quantidade.
Conforme	Os tags possuem os mesmos valores estatísticos, quantidade diferente de valores brutos, porém os valores ponto a ponto são iguais.
Não conforme	Os tags possuem a mesma quantidade de valores brutos porém os valores estatísticos são diferentes.
Não conforme	Os tags apresentam valores ponto a ponto iguais, porém um tag apresenta uma quantidade maior de valores e os valores estatísticos são diferentes.
Não conforme	Os tags possuem os mesmos valores estatísticos, porém os valores ponto a ponto são diferentes.

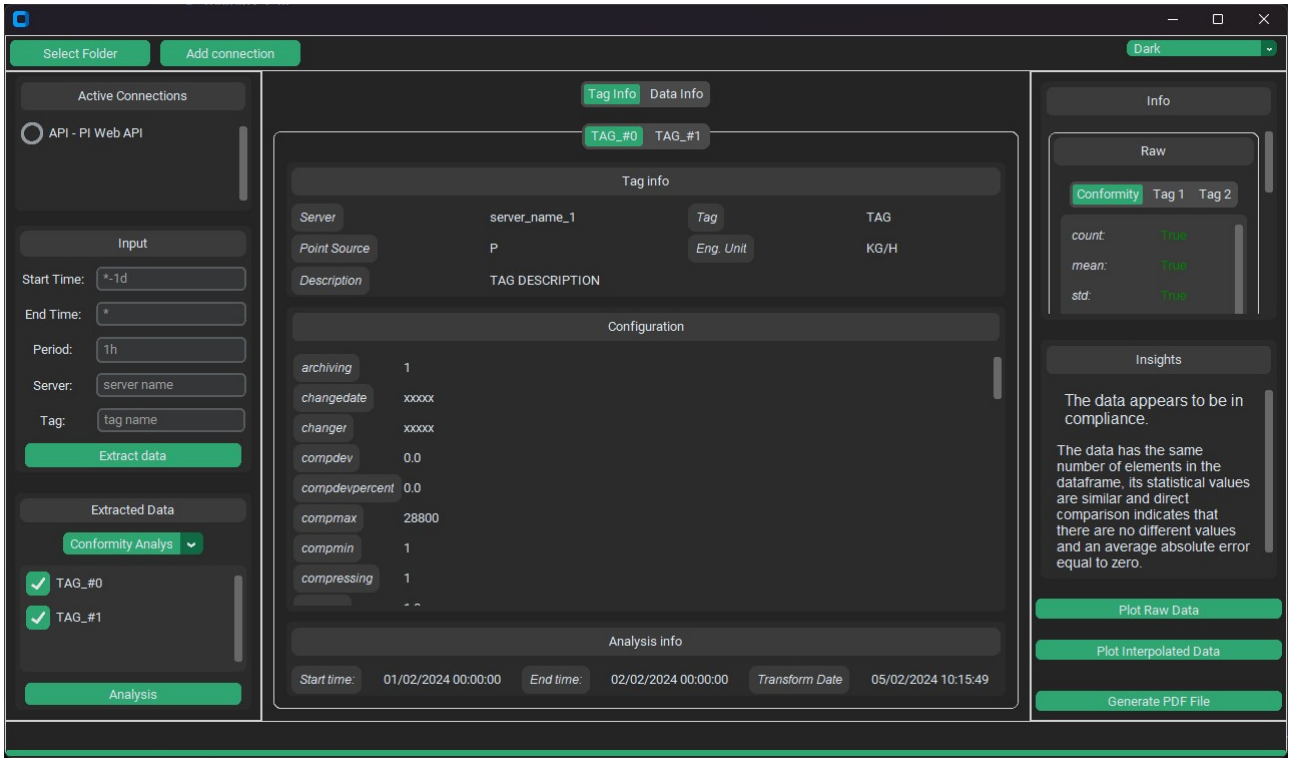
Fonte: Projeto.

Utilizando essa diretrizes que consideram os valores extraídos e os dados estatísticos calculados, o software faz a análise dos pares de tags e indica sua conformidade. No caso específico do PI System a ferramenta também considera atributos da configuração do tag em si para realizar a análise como, por exemplo, verificar se o atributo de compressão está ligado em um tag e não no outro.

5.6 INTERFACE GRÁFICA

A ferramenta é composta por uma tela principal da interface gráfica desenvolvida utilizando a biblioteca *CustomTkinter* de forma a oferecer uma experiência intuitiva e organizada para os usuários, permitindo o gerenciamento e visualização eficientes de dados provenientes de diversas fontes. Dividida em seções distintas, cada uma com funcionalidades específicas, a interface proporciona uma experiência de usuário agradável e simplificada. Além disso, a ferramenta possui janelas específicas que abrem de acordo com a interação do usuário para a configuração do modelo de conexão a ser utilizado e visualização dos gráficos.

Figura 49 – Tela principal da interface gráfica.



Fonte: Projeto.

5.6.1 Tela principal

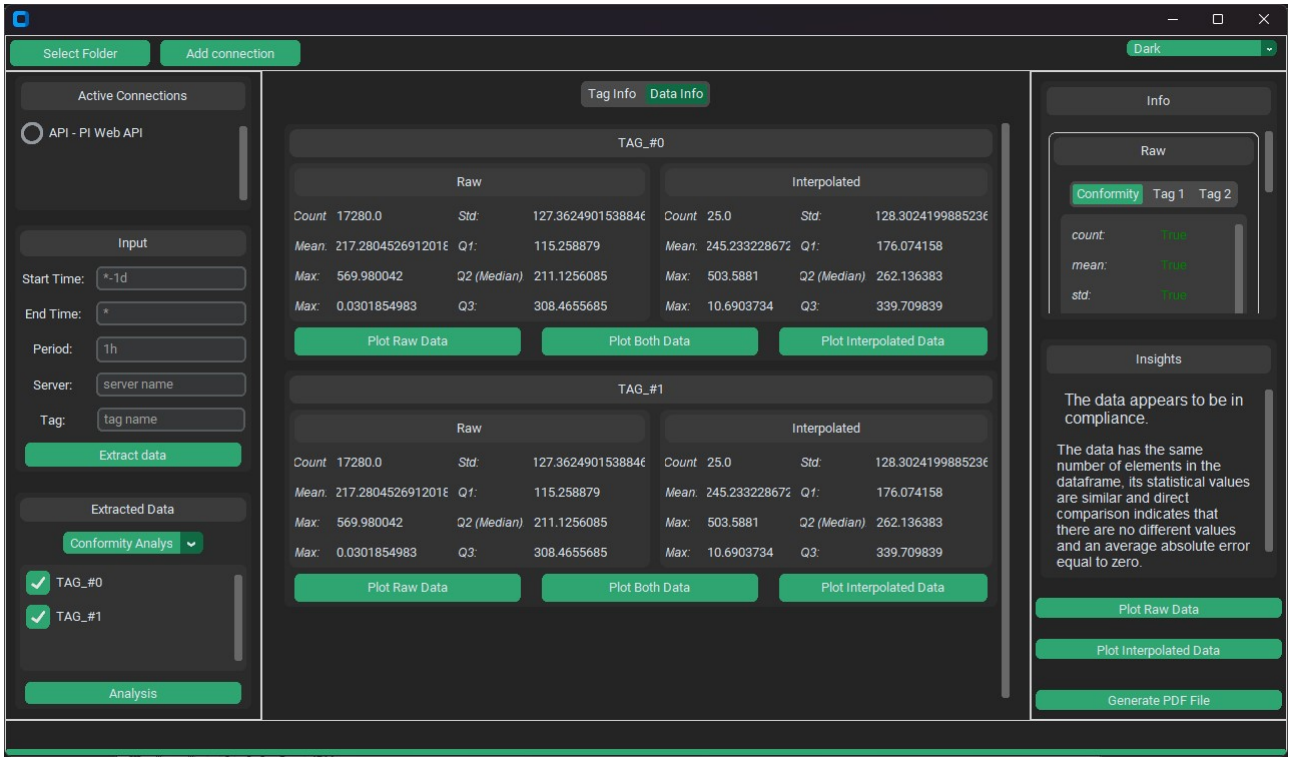
A tela principal é composta por cinco grande áreas, sendo elas:

- **Parte superior:** Contém os botões de seleção de pasta, adição de modelo de conexão e escolha do tema, fornecendo ao usuário acesso rápido a funcionalidades essenciais de gerenciamento e personalização.
- **Parte central esquerda:** Apresenta as informações de entrada do usuário referentes à extração e análise de dados. Aqui, o usuário pode selecionar os parâmetros para a extração de dados e os conjuntos para análise, além de disparar essas funcionalidades, proporcionando uma interação intuitiva e direta com o sistema.
- **Parte central do meio:** Nesta área, são exibidas as informações dos dados transformados, incluindo as informações de configuração dos tags e os dados históricos. O usuário tem a possibilidade de visualizar gráficos ao selecionar essa a opção ”Data Info” (conforme representado na Figura 50), facilitando a análise e interpretação dos dados.

- **Parte central direita:** Aqui, são apresentadas as informações da análise de conformidade e insights da análise. Além disso, são disponibilizados botões para visualizar os dados comparados e gerar um PDF, proporcionando ao usuário uma visão detalhada dos resultados e a capacidade de compartilhá-los de forma eficiente.
- **Parte inferior:** Contém uma barra com mensagens para auxiliar o usuário no uso da ferramenta, fornecendo *feedback* e orientações durante a interação com o sistema, aumentando a usabilidade e facilitando a resolução de problemas.

Por padrão, ao iniciar a ferramenta somente o botão de seleção de pasta está habilitado, forçando ao usuário selecionar uma pasta em que serão gerados todos os arquivos mencionados na seção 5.2, de forma a garantir uma maior flexibilidade no uso da ferramenta.

Figura 50 – Tela principal da interface gráfica com possibilidade de plot dos dados.



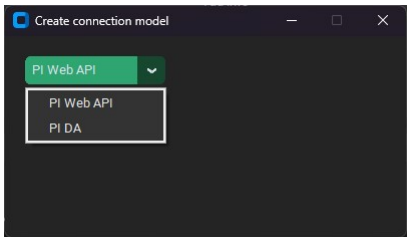
Fonte: Projeto.

5.6.2 Tela de configuração de conexão

Ao selecionar a opção de ”Add connection”, a janela da Figura 51 é aberta e permite ao usuário adicionar uma nova conexão a partir dos modelos de conexão

disponíveis. No exemplo, estão disponíveis os modelos de conexão via PI Web API e PI AF SDK, cabendo ao usuário definir qual o modelo desejado.

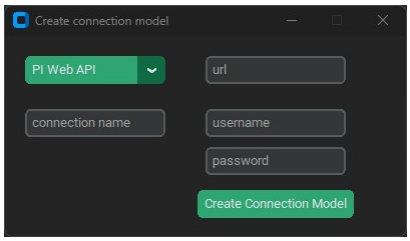
Figura 51 – Janela para configuração de conexões com o PIMS.



Fonte: Projeto.

Após a seleção do modelo, os campos com os parâmetros necessários aparecem na tela (Figura 52), permitindo ao usuário finalizar a configuração e adicionar o modelo ao uso.

Figura 52 – Janela para configuração de conexões do tipo PI Web API.



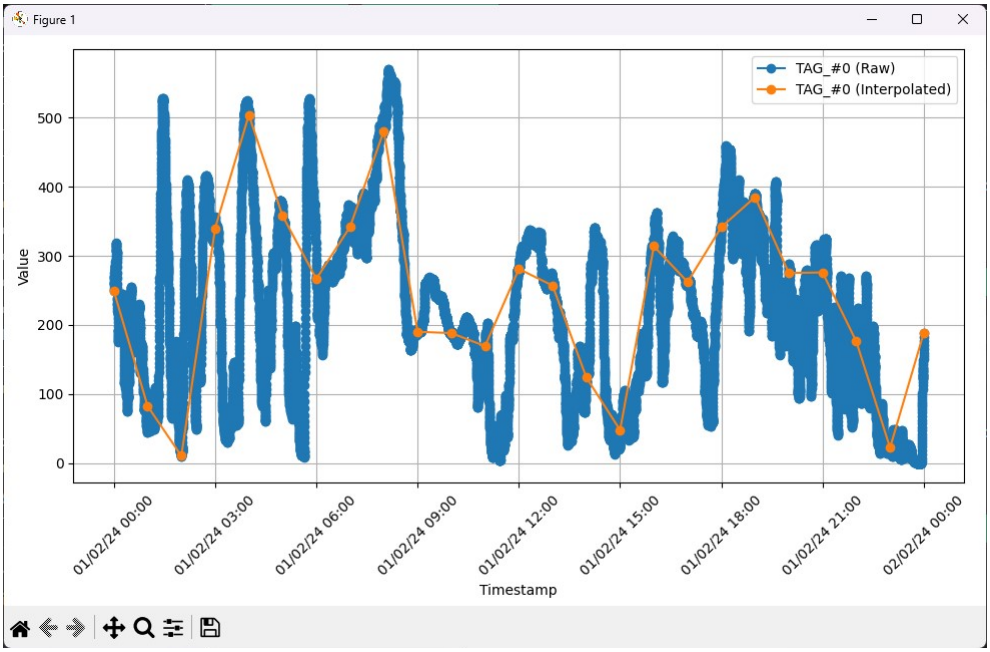
Fonte: Projeto.

Após a primeira adição da conexão, ela fica disponível no campo de ”Active Connections” da tela principal e é gerado o arquivo ”connections.json”, apresentado na seção 5.2.1, permitindo que o usuário continue com a conexão salva e disponível para uso em momentos futuros.

5.6.3 Tela de visualização de dados

Utilizando a biblioteca matplotlib em conjunto com a biblioteca pandas, os botões de ”plot” permitem ao usuário visualizar os dados históricos em uma nova janela, exemplificada na Figura 53. A ferramenta possibilita que o usuário visualize os dados transformados de maneira bruta, interpolada, comparando os dois entre o tag e, após a realização da análise, comparando os dados brutos ou interpolados dos tags analisados.

Figura 53 – Janela para visualização dos dados extraídos.



Fonte: Projeto.

Com um design intuitivo e organizado, a interface gráfica desenvolvida proporciona uma experiência de usuário fluida e eficiente, facilitando a navegação e o acesso às funcionalidades do software.

6 TESTES REALIZADOS E RESULTADOS OBTIDOS

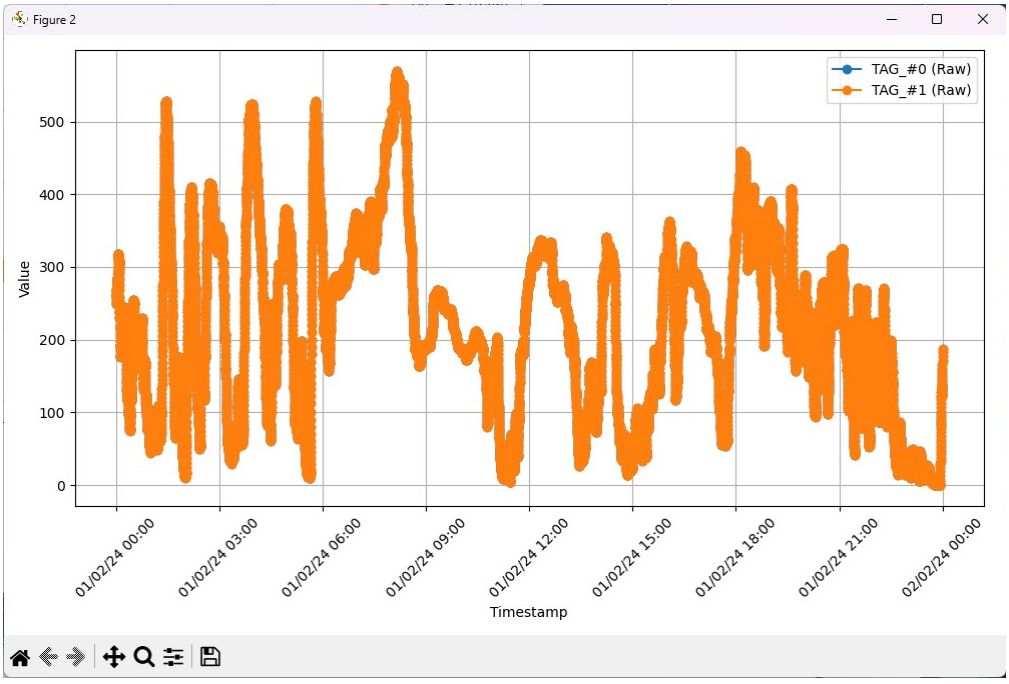
Este capítulo abordará os testes realizados para validar a ferramenta e os resultados obtidos durante esses testes, neles a ferramenta foi utilizada para extrair tags com um comportamento conhecido em determinado período de tempo e validado se os insights finais da ferramenta fazem sentido.

6.1 VALIDAÇÃO DA CONFORMIDADE ENTRE TAGS

O primeiro teste realizado teve como objetivo validar a capacidade da ferramenta de detectar a conformidade entre tags provenientes de diferentes servidores, mas que eram réplicas e tinham sua conformidade conhecida. Os tags eram essencialmente iguais, e o teste visava verificar se a ferramenta seria capaz de identificar essa conformidade.

A Figura 49, utilizada para exemplificar a interface gráfica, apresenta o final do teste gerado, já a Figura 54 apresenta a comparação dos dados brutos extraídos gerada pela ferramenta.

Figura 54 – Comparação dos dados históricos de tags conformes de teste.



Fonte: Projeto.

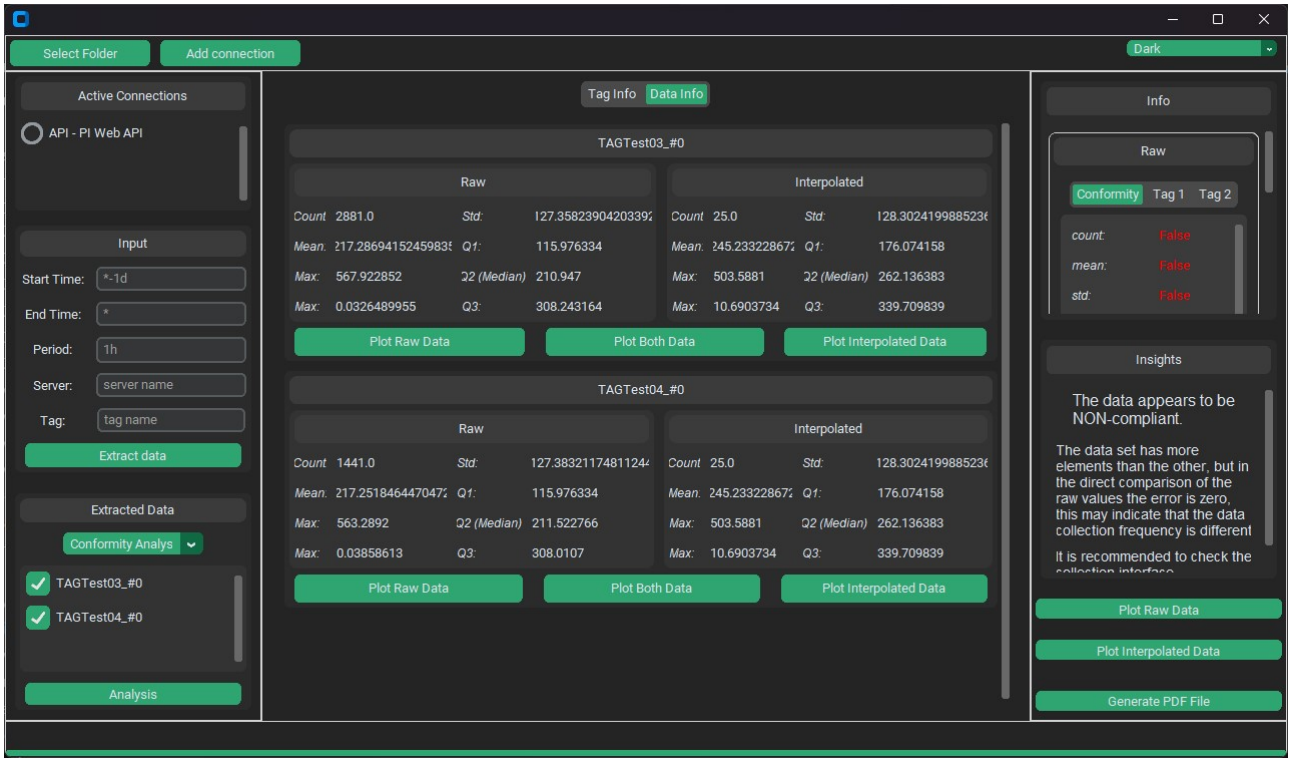
Durante o teste, a ferramenta foi capaz de detectar a conformidade entre as tags e indicou isso de forma clara e precisa. Isso demonstrou que a ferramenta é capaz de reconhecer padrões e detectar similaridades entre diferentes conjuntos de dados.

6.2 IDENTIFICAÇÃO DE DIFERENÇAS NAS FREQUÊNCIAS DE ATUALIZAÇÃO

O segundo teste realizado envolveu tags provenientes de campo, mas com frequências de atualização diferentes. O objetivo deste teste era verificar se a ferramenta seria capaz de identificar essas diferenças e fornecer sugestões para resolver o problema.

A Figura 55 apresenta a tela principal após a execução do teste com os dados não conformes, os insights gerados estão de acordo com o esperado visto que para o teste a frequência de atualização dos tags foi alterada propositalmente.

Figura 55 – Tela principal após o teste com dados não conformes.

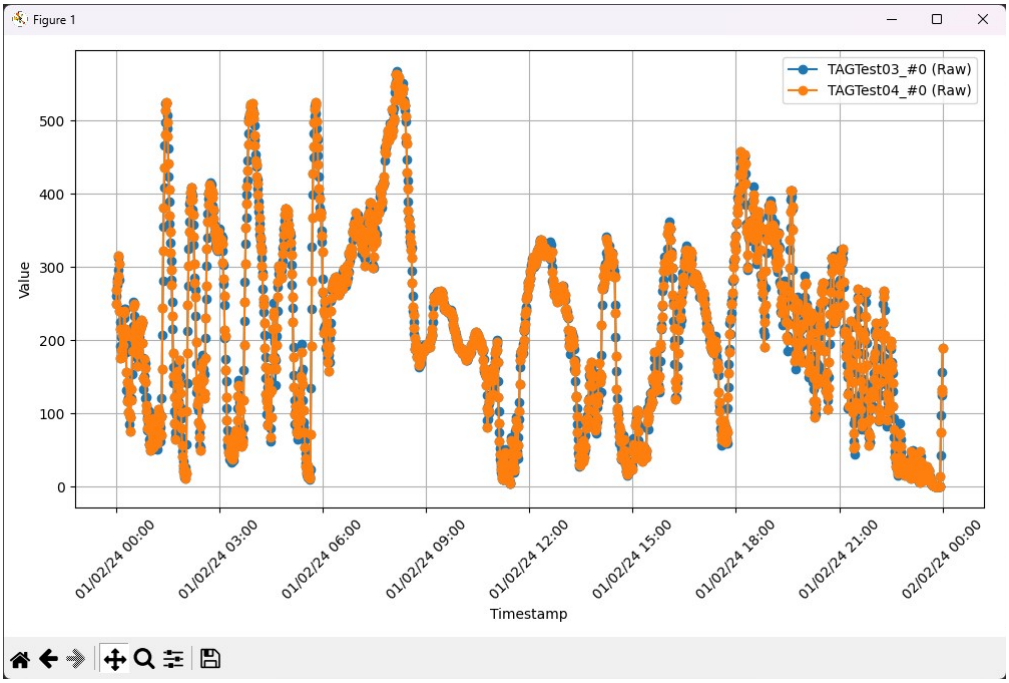


Fonte: Projeto.

Conforme apresentado na Figura 56, os dados extraídos apresentam taxas de atualização diferentes, embora sejam provenientes da mesma fonte de campo.

Durante o teste, a ferramenta identificou as diferenças nas frequências de atualização entre as tags e sugeriu verificar a configuração da interface de coleta de dados. Essa indicação foi útil para os usuários, pois apontou diretamente para a possível causa do problema e forneceu orientações sobre como resolvê-lo.

Figura 56 – Comparação dos dados históricos de tags não conformes de teste.



Fonte: Projeto.

6.3 RESULTADOS

Os testes realizados demonstraram que a ferramenta é capaz de identificar conformidades e não conformidades. Isso confirma a eficácia da ferramenta na análise e validação de dados provenientes de diferentes fontes, porém mais testes serão realizados para validar diferentes considerações e casos de não conformidade, não foi possível realizar um número maior de validações durante o projeto por restrições de tempo e priorização de demandas.

Conclusão

O desenvolvimento de uma ferramenta para extração, comparação e validação de conformidades entre dados históricos armazenados em um PIMS representou uma tarefa desafiadora. Uma das dificuldades enfrentadas foi seguir as diretrizes de segurança dos sistemas PIMS e respeitar as boas práticas associadas a isso.

Embora tenham sido estabelecidas diretrizes específicas que facilitaram o trabalho, o esforço para criar uma ferramenta abrangente capaz de lidar com uma variedade de sistemas complexos foi considerável. Ao concentrar os esforços na compatibilidade com o PI System, a ferramenta foi concebida com a flexibilidade necessária para acomodar diferentes PIMS.

Os resultados alcançados até o momento demonstram uma estrutura robusta, permitindo uma fácil adaptação da ferramenta para extrair dados de outras fontes e por meio de diferentes métodos, como conexões API, SDKs de outros fornecedores e conexões SQL.

É importante ressaltar que nem todos os requisitos foram totalmente atendidos até o momento. A ferramenta ainda está disponível apenas em um idioma, a documentação interna do código ainda não foi finalizada e há espaço para melhorias na interface gráfica e na configuração da ferramenta.

No entanto, os resultados finais obtidos foram satisfatórios, apesar de ainda demandarem mais testes e validações. A ferramenta demonstrou ser um instrumento valioso para a automação da análise de dados, permitindo que novas análises sejam desenvolvidas e aplicadas facilmente, mesmo por usuários sem conhecimento específico em softwares PIMS.

Como atividades e melhorias futuras, seria continuar com testes, desenvolver outros modelos de conexões visando outros softwares PIMS disponíveis no mercado, novas análises e diretrizes para a verificação de conformidade e geração de insights. Além disso, realizar uma validação da interface e da ferramenta em si com um número considerável de usuários e disponibilizar a ferramenta para uso amplo e geral da empresa.

Em suma, o trabalho realizado apresenta uma contribuição significativa para o campo da análise de dados, fornecendo uma ferramenta poderosa e flexível que simplifica e agiliza processos de análise em sistemas PIMS e outras fontes de dados.

REFERÊNCIAS

AVEVA. **PI System Administration**. 2024a. Disponível em: https://osicdn.blob.core.windows.net/learningcontent/pdfs/PI%20System%20Administration.pdf. Acesso em: 18 fev. 2024.

AVEVA. **PI System Documentation**. [*S.l.*], 2024b. Disponível em: https://docs.aveva.com/category/pi-system. Acesso em: 18 fev. 2024.

CORPORATION, Microsoft. **VSCode FAQ**. 2024. Disponível em: https://code.visualstudio.com/docs/supporting/faq. Acesso em: 18 fev. 2024.

ENGINEERING, RADIX; SOFTWARE. **A RADIX é destaque no Top 3 de melhores empresas para se trabalhar em Tecnologia da Informação**. 2023a. Disponível em: https://www.radixeng.com.br/insights/melhores-empresas/a-radix-e-destaque-no-top-3-de-melhores-empresas-para-se-trabalhar-em-tecnologia-da-informacao. Acesso em: 18 fev. 2024.

ENGINEERING, RADIX; SOFTWARE. **RADIX é destaque mais uma vez no ranking do GPTW Brasil**. 2023b. Disponível em: https://www.radixeng.com.br/insights/melhores-empresas/radix-e-destaque-mais-uma-vez-no-ranking-do-gptw-brasil. Acesso em: 18 fev. 2024.

FOWLER, Martin. **Model-View-Controller**. 2024. Disponível em: https://martinfowler.com/eaaCatalog/modelViewController.html. Acesso em: 18 fev. 2024.

HUNTER, J. D. Matplotlib: A 2D graphics environment. **Computing in Science Engineering**, v. 9, n. 3, p. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

IEEE Recommended Practice for Software Requirements Specifications. [*S.l.*: *s.n.*], 1998. DOI: 10.1109/IEEESTD.1998.88314.

LIMITED, AVEVA Solutions. **AVEVA Framework SDK Overview**. 2024a. Disponível em: https://docs.aveva.com/bundle/af-sdk/page/html/af-sdk-overview.htm. Acesso em: 18 fev. 2024.

LIMITED, AVEVA Solutions. **PI Server Data Access Administration**. 2024b. Disponível em:

https://docs.aveva.com/bundle/pi-server-da-admin/page/1022851.html. Acesso em: 18 fev. 2024.

LIMITED, AVEVA Solutions. **PI System Architecture, Planning and Implementation Workbook**. 2024c. Disponível em: https://cdn.osisoft.com/learningcontent/pdfs/PISystemArchitecturePlanningAndImplementationWorkbook.pdf. Acesso em: 18 fev. 2024.

LIMITED, AVEVA Solutions. **PI System SDK Getting Started**. 2024d. Disponível em: https://docs.aveva.com/bundle/af-sdk-getting-started/page/1011019.html. Acesso em: 18 fev. 2024.

LIMITED, AVEVA Solutions. **PI to PI Interface**. 2024e. Disponível em: https://docs.aveva.com/bundle/pi-to-pi-interface/page/1012207.html. Acesso em: 18 fev. 2024.

LIMITED, AVEVA Solutions. **PI Web API**. 2024f. Disponível em: https://docs.aveva.com/category/pi-web-api. Acesso em: 18 fev. 2024.

LIMITED, AVEVA Solutions. **PI Web API Reference**. 2024g. Disponível em: https://docs.aveva.com/bundle/pi-web-api-reference/page/help.html. Acesso em: 18 fev. 2024.

MICROSOFT. **Visual Studio Code - Code Editing. Redefined**. 2024. Disponível em: https://code.visualstudio.com/. Acesso em: 19 fev. 2024.

PYTHON Reference Manual. [*S.l.*], 2024. Disponível em: https://docs.python.org/3/reference/. Acesso em: 18 fev. 2024.

REITZ, Kenneth. **Requests: HTTP for Humans**. 2024. Disponível em: https://pypi.org/project/requests/. Acesso em: 18 fev. 2024.

SCHIMANSKY, Tom. **CustomTkinter Documentation**. 2024. Disponível em: https://customtkinter.tomschimansky.com/documentation/. Acesso em: 18 fev. 2024.

TEAM, The pandas development. **pandas: Python Data Analysis Library**. 2024. Disponível em: https://pandas.pydata.org/. Acesso em: 18 fev. 2024.