Pontifícia Universidade Católica
do Rio de Janeiro

Marcelo de Carvalho

A Data Reference Architecture for Brazilian Electrical Companies

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Marco Antonio Casanova

Rio de Janeiro
May 2024

Pontifícia Universidade Católica
do Rio de Janeiro

Marcelo de Carvalho

A Data Reference Architecture for Brazilian Electrical Companies

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Examination Committee:

Prof. Marco Antonio Casanova
Advisor
Departamento de Informática – PUC-Rio

Prof. Edward Hermann Haeusler
Departamento de Informática – PUC-Rio

Prof. Luiz André Portes Paes Leme
UFF

Rio de Janeiro, May 2nd, 2024

Marcelo de Carvalho

Graduated in Computer Science by Centro Universitário Carioca

Bibliographic data

de Carvalho, Marcelo

A Data Reference Architecture for Brazilian Electrical Companies / Marcelo de Carvalho; advisor: Marco Antonio Casanova. – 2024.

104 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2024.

Inclui bibliografia

1. Informática – Teses. 2. Data Lake. 3. Azure Databricks. 4. Arquitetura de Dados. I. Casanova, Marco Antonio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

To Mariangela Venas, who 25 years ago
accepted to share her life journey with me.

Acknowledgments

To my daughter Helena Venas, for your patience during many weekends without the family trips that she enjoys so much.

To my advisor Professor Marco Antonio Casanova for the stimulus and partnership to carry out this work, and professors Edward Hermann Haeusler and Alex de Vasconcellos Garcia for their leadership in the ENSIGHTS project, the main research source for this work.

The author thanks the team at PUC-Rio and Radix Engenharia e Desenvolvimento de Software for their support in clarifying the technological aspects of this work, especially Athos Barbosa, for sharing some of his vast knowledge in data and cloud computing, and Gabriel Resende, to reveal all the power of Python in the field of Data Science.

The author thanks colleagues from Eletrobras Furnas: Marcelo Piñeiro, Ana Claudia Rodrigues, and Ana Cristina de Freitas Marotti, for their support in coordinating the ENSIGHTS project.

The author thanks Felipe Lopes, fellow journey companion in the master's program, who, like me, experienced the challenges of balancing life between studies, family, and work.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

de Carvalho, Marcelo; Casanova, Marco Antonio (Advisor). **A Data Reference Architecture for Brazilian Electrical Companies**. Rio de Janeiro, 2024. 104p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

During the 1990s, the Brazilian electricity sector underwent profound changes in its operational model. The Brazilian state began to assume a less developmental and more regulatory role, leading to the creation of the National Electric Energy Agency (ANEEL). One of the roles of ANEEL is to ensure the quality of service provided by sector agents (energy generation, transmission, and distribution companies). When an agent does not meet established standards, ANEEL can apply penalties. In this sense, the improvement of maintenance processes plays a crucial role in ensuring the reliability and efficiency of electrical systems and consequently reducing penalties. Predictive maintenance is being adopted, in addition to more traditional methodologies (reactive and preventive). However, this methodology represents a fundamental change compared to previous ones, as it seeks to anticipate failures based on data and analysis. In this sense, the incorporation of predictive maintenance into maintenance processes presupposes the availability of operating and maintenance data of the equipment, as well as the technological resources that enable the analysis of these data. This dissertation proposes a reference technological architecture that enables the development of these analyzes, considering aspects of management, governance, and corporate compliance practiced by the sector's agents.

Keywords

Data Lake; Azure Databricks; Data Architecture.

Resumo

de Carvalho, Marcelo; Casanova, Marco Antonio. **Uma arquitetura de referência para dados de empresas do setor elétrico brasileiro**. Rio de Janeiro, 2024. 104p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Durante a década de 1990, o setor elétrico brasileiro passou por profundas mudanças em seu modelo operacional. O Estado Brasileiro passou a assumir um papel menos desenvolvimentista e mais regulatório, o que levou a criação da Agência Nacional de Energia Elétrica (ANEEL). Um dos papéis da ANEEL é assegurar a qualidade do serviço prestado pelos agentes do setor (empresas de geração, transmissão e distribuição de energia). Quando um agente não cumpre os padrões estabelecidos, a ANEEL pode aplicar penalidades. Nesse sentido, o aprimoramento dos processos de manutenção desempenham um papel crucial na garantia da confiabilidade e eficiência de sistemas elétricos e consequente redução de penalidades. A manutenção preditiva passa a ser adotada, em complemento às metodologias mais tradicionais (reativa e preventiva). Contudo, essa metodologia representa uma mudança fundamental em relação às anteriores, pois busca antecipar falhas com base em dados e análises. Nesse sentido, a incorporação da manutenção preditiva nos processos de manutenção, pressupõe a disponibilidade de dados de operação e manutenção dos equipamentos, bem como de recursos tecnológicos que viabilizem a análise desses dados. Essa dissertação propõe uma arquitetura tecnológica de referência, que habilite o desenvolvimento dessas análises, considerando aspectos de gestão, governança e conformidade empresariais praticados pelos agentes do setor.

Palavras-chave

Data Lake; Azure Databricks; Arquitetura de Dados.

Table of contents

1 Introduction **13**
1.1 Motivation 13
1.2 Service quality standards 19
1.3 Current maintenance practices 20
1.4 Data in electrical companies 22
1.5 Conclusion 24
1.6 Dissertation organization 25

2 Related Work **26**
2.1 Introduction 26
2.2 Big Data 27
2.3 Cloud Computing 29
2.4 Data Architectures 31
2.5 Data Governance 34
2.6 Conclusion 37

3 The proposed Data Reference Architecture **38**
3.1 Introduction 38
3.2 Data Reference Architecture 38
3.3 An Analysis of Implementation Alternatives 41
3.4 Azure Implementation Details 49
3.5 Conclusion 52

4 Results **53**
4.1 Introduction 53
4.2 Azure Data Lake Storage 53
4.3 Data Extraction 55
4.4 Azure Synapse Analytics Pipelines 57
4.5 Azure Databricks 70
4.6 Azure Synapse Analytics SQL Pools 76
4.7 CosmosDB 77
4.8 Azure App Services 79
4.9 PowerBI 80
4.10 Azure Purview 80
4.11 Conclusion 81

5 Conclusions and future work **82**

6 Bibliography **84**

A Code Listings **87**
A.1 Save Transactions script 87
A.2 Get All DGA Results script 92
A.3 NB Final script 101

List of figures

Figure 1.1 Maintenance Plans(RAN et al., 2019) 21
Figure 1.2 Maintenance Costs(RAN et al., 2019) 22

Figure 2.1 the NIST Big Data Reference Architecture (NIST, 2019). 28
Figure 2.2 High level design of a Data Reference Architecture (PÄÄKKÖNEN; PAKKALA, 2015). 33
Figure 2.3 Data Reference Architecture with four layers (ZBURIVSKY; PARTNER, 2021). 34
Figure 2.4 Data Governance Framework (DGI, 2024 (accessed in March 2024)). 36

Figure 3.1 Data Lake Structure. 39
Figure 3.2 Data Reference Architecture in Azure. 42
Figure 3.3 Data Reference Architecture in Amazon Web Services. 42
Figure 3.4 Data Reference Architecture in Google Cloud Platform. 43

Figure 4.1 Data Lake structure implemented in Azure 54
Figure 4.2 SAP pipelines in Synapse Analytics 59
Figure 4.3 Activities related to ingest SAP data for landing zone 59
Figure 4.4 Integration Dataset for Data lake landing zone 60
Figure 4.5 Integration Dataset for SAP Folder 60
Figure 4.6 Integration Dataset for SAP File 61
Figure 4.7 Activity related to copy SAP data for trusted zone 62
Figure 4.8 Pipeline to move SAP data from Landing Zone to Trusted zone 62
Figure 4.9 Activity related to copy SAP data for Refined zone 63
Figure 4.10 Pipeline to copy SAP data from Landing Zone to Refined zone 63
Figure 4.11 Data Lake Refined Zone for SAP Data 64
Figure 4.12 SAGE pipelines in Synapse Analytics 65
Figure 4.13 Activities related to ingesting SAGE data for landing zone 66
Figure 4.14 SAGE integration datasets 67
Figure 4.15 Pipeline to scan SAGE data files 67
Figure 4.16 Pipeline to copy SAGE data files to the landing zone. 68
Figure 4.17 Pipeline to move SAGE data from Landing Zone to Trusted zone 68
Figure 4.18 SAP and SAGE correlation. 70
Figure 4.19 Architectural patterns to access ADLS from Databricks. 72
Figure 4.20 Architectural pattern to access ADLS using Service Principal. 73
Figure 4.21 Databricks File System. 75
Figure 4.22 Databricks Secure Access Pattern. 76
Figure 4.23 A SQL query running in a Synapse SQL Pool. 77
Figure 4.24 CosmosDB Collections. 78
Figure 4.25 CosmosDB Collection in detail. 78
Figure 4.26 Custom dashboard showing CAI and EFRI indicators. 79

List of tables

Table 2.1 Addressing the Big Data challenges with cloud computing for a dust storm forecasting application. Adapted from (YANG et al., 2017) 30

Table 3.1 Cost comparison of cloud computing providers 46

Table 4.1 Hierarchical structure for mapping between fault categories, defined by the categorization algorithm, and SAGE measurements. 71

List of Abreviations

ANEEL – Agência Nacional de Energia Elétrica.

ERP – Enterprise Resource Planning.

CRM - Customer Relationship Management.

DATA - Data Management Association.

DGI - Data Governance Institute.

RPA - Robotic Process Automation.

CSV - Comma Separated Values.

ENSIGHTS - Energy Insights.

CAI - Chromatographic Assay Indicator.

EFRI - Electrical Failure Risk Indicator.

ML - Machine Learning.

IR - Integration Runtime.

BI - Business Intelligence.

DBFS - Databricks File System.

AWS - Amazon Web Services.

GCP - Google Cloud Platform.

ETL - Extract, Transform and Load.

UNC - Unified Naming Convention.

"Data is not information, information is not knowledge, knowledge is not understanding, understanding is not wisdom."

Clifford Stoll, *High-Tech Heretic: Reflections of a Computer Contrarian.*

1
Introduction

1.1
Motivation

Before the 1990s, the Brazilian electrical sector was dominated by state-owned companies. With an increasing demand for energy and the need for significant investments in infrastructure, the Brazilian government began a reform process to modernize the sector.

Restructuring the Brazilian electrical sector was a significant milestone in the country's energy history. This reform involved a series of regulatory and structural changes aimed at increasing efficiency, attracting private investments, and improving service quality. One of the main events of this time was the creation of the National Agency of Electric Energy (ANEEL), a federal regulatory agency responsible for regulating and overseeing the generation, transmission, distribution, and commercialization of electric energy. ANEEL was established to ensure transparency, fairness of tariffs, and competitiveness in the sector. The ANEEL functions include granting licenses for new energy projects, regulating electricity tariffs, overseeing sector companies, and resolving disputes between consumers and service providers.

ANEEL plays a fundamental role in ensuring the quality of service in the Brazilian electrical sector. Through Normative Resolutions (ReN), it sets quality standards that sector agents, technical term for companies operating in generation, transmission, distribution, and commercialization, must comply with. These standards include limits for the frequency and duration of interruptions in the energy supply. When an agent does not meet the established standards, ANEEL can impose penalties. However, incentives can be offered to those who exceed quality expectations.

To ensure compliance with these quality standards, agents seek to improve the maintenance processes of the energy transmission systems under their responsibility, in order to minimize interruptions in their operation to the maximum extent possible. It is important to note that ANEEL penalizes scheduled and unscheduled equipment shutdowns. However, unscheduled shutdowns incur much higher penalties than scheduled ones. In this sense, the maintenance of these systems aims primarily at preventing an unexpected failure event.

Research on maintenance effectiveness indicates that one-third of all

maintenance costs are wasted as a result of unnecessary or improper maintenance (YACOUB; CUKIC; AMMAR, 2004). The main reason for this ineffective maintenance management is the lack of factual data to quantify the actual need for equipment repair (YACOUB; CUKIC; AMMAR, 2004). Traditional maintenance methods are reactive and preventive (MOBLEY, 2002). The former is based on actual equipment failure and is the most expensive method of failure management. The latter is based on statistical trends and predetermined time intervals or operating hours to reduce the probability of failure or performance loss. Predictive methods are at the forefront because they determine maintenance actions scheduling adaptively and flexibly, according to equipment needs, rather than at fixed intervals, as in preventive maintenance.

Predictive maintenance uses direct monitoring of mechanical condition, system efficiency, and other indicators to determine the actual MTTF (mean time to failure) or loss of efficiency of each plant equipment. This predictive method aims to ensure the maximum interval between maintenance services and minimize the amount and cost of unplanned downtime caused by failures. Monitoring can employ technologies such as the Internet of Things (IoT), Cloud Computing, and Machine Learning (ML). In light of (LARIVIERE et al., 2016), maintenance has been one of the areas with the highest number of modern Predictive Analysis techniques applications. The Internet of Things (IoT) allows real-time telemetry to provide data on the operation of the system and equipment for analytical procedures to make their failures more predictable.

In the context of the electrical sector, data of interest for applying predictive maintenance techniques reside in databases of systems that support the maintenance and operation processes of generation and transmission equipment. These systems record data related to these equipment, such as: maintenance events, real-time operating conditions, and interruption events, among others. It is worth mentioning that, in many cases, these databases are isolated and inaccessible throughout the organization.

To use these data, it is necessary to build integrations that allow them to be brought into a common area, where they can be treated and analyzed together. In this sense, this dissertation aims to propose a technology-agnostic reference architecture that enables the use of these data, considering management, governance, and corporate compliance requirements practiced by agents in the electrical sector.

1.1.1 Motivation for restructuring the system

Before 1990s, the electricity sector in Brazil was structured with companies that handled all aspects of the process, from generation to distribution. State-owned companies primarily managed distribution, while federal and state entities were largely responsible for generation and transmission activities(MELO; NEVES; PAZZINI, 2011). For considerable time, this public monopoly in electricity generation, transmission, and distribution functioned effectively, resulting in a significant increase in the capacity of electricity supply, which increased by 500% over the past 50 years. However, during the 1990s, the performance of the system deteriorated significantly, prompting the federal government to intervene in 1993 to prevent a system collapse. This intervention involved assuming liabilities of US$ 26 billion in debts and increasing electricity prices by 70% to stabilize the system and prevent bankruptcy(MENDONÇA; DAHL, 1999).

Despite this initial intervention, the system continued to decline and, by 1995, the worsening transmission constraints, coupled with the government's inability to promote expansion, exacerbated a new cycle of utility defaults. This situation led to a critical scenario with a high risk of electricity blackouts.

Several factors contributed to the crisis within the electrical system. The decline in investment capacity began in the late 1970s, coinciding with the international debt crisis. This was exacerbated by government tariff policies that inflicted economic losses on the sector. Throughout the 1980s, the government consistently reduced electricity tariffs in an attempt to control inflation. During this period, pricing decisions were influenced by political and economic factors, resulting in artificially low tariffs. Furthermore, the loss of international credit further decreased investment resources in the electric sector, falling from 71% in 1974 to 29% in 1988. In the 1990s, there was widespread agreement that securing financing to expand the capacity of the system remained the primary challenge.

At this time, Brazil embarked on restructuring its electricity regulatory model, aiming to replace the previous vertically integrated system characterized by monopoly and non-differentiation among various electricity-related activities. This restructuring introduced a new system that emphasizes free pricing, competition, and the separation of generation, transmission, distribution, and commercialization activities.

In 1993, with the backing of the Congress, the federal government initiated a series of coordinated measures aimed at restructuring and setting out the principles for the electricity sector. The primary aim of this reform

was to enable the government to concentrate on its functions as a policymaker and regulator, while transferring the tasks of operations and investment to the private sector.

In 1995, with the enactment of Law 9,074, marking the initial steps toward regulatory reform. The new model introduced structural changes and new functions that were intended to stimulate investment in expanding the electrical system, improving the efficiency of concessionaire companies, and improving the overall competitiveness and quality of services in the national economy.

The reform addressed three main areas. First, it established a new legal and regulatory framework, which included adjusting existing regulations with respect to concessions, economically regulating natural monopolies, and facilitating competition. Second, it defined new trade arrangements regarding the purchase and sale of electricity in bulk, access to transmission and distribution networks, and mechanisms for sector planning and expansion. Third, it implemented institutional changes within the government and state utilities to facilitate the adoption of the proposed trading arrangements and regulatory framework. These institutional changes involved the creation of a new independent regulator (ANEEL) and the review of the role of Eletrobras. The reform also included a series of structural changes deemed necessary to establish a competitive market in bulk electricity(MENDONÇA; DAHL, 1999).

1.1.2 Restructuring Project of the Brazilian Electric Sector – RESEB

The Brazilian Electric Sector Restructuring Project, commonly referred to as RESEB (Projeto de Reestruturação do Setor Elétrico Brasileiro), was a significant initiative undertaken in Brazil to modernize and reorganize the country's electric power industry. It aimed to address challenges, promote efficiency, and promote sustainable growth within the sector.

Initiated in the late 1990s and implemented through legislation such as Law 9.427/96 and Law 10.848/04, RESEB encompassed various measures:

1. Market Liberalization: One of the central aspects of RESEB was the liberalization of the electricity market. This involved the unbundling of generation, transmission, and distribution activities, allowing for competition and private investment in previously state-dominated sectors.

2. Privatization: RESEB facilitated the privatization of state-owned utilities, particularly in the generation and distribution segments. This

change to private ownership was intended to improve efficiency, attract investment, and improve service quality.

3. Regulatory Framework: The project established a robust regulatory framework, overseen by the Agência Nacional de Energia Elétrica (ANEEL), to govern the operations of the electric power sector. This framework was designed to ensure fair competition, consumer protection, and sustainable development of energy resources.

4. Energy Auctions: RESEB introduced mechanisms such as energy auctions to promote investment in new generation capacity. These auctions provided long-term contracts and price guarantees, fostering the development of renewable energy sources and fostering a diversified energy mix.

5. Expansion of the Transmission Infrastructure: To support the growing demand for electricity and facilitate the integration of renewable energy projects, RESEB emphasized the expansion and modernization of Brazil's transmission infrastructure. This involved investments in new transmission lines, substations, and grid optimization technologies.

6. Social and Environmental Considerations: RESEB incorporated social and environmental considerations into the planning and implementation of energy projects. This included measures to mitigate the impacts of energy development on local communities and ecosystems, as well as initiatives to promote energy efficiency and conservation.

1.1.3 ANEEL - The Regulatory Agency

The "Agência Nacional de Energia Elétrica" (ANEEL), translated as the National Agency for Electric Energy, is a regulatory agency responsible for the oversight and regulation of the electric power sector. Established in 1996 by Law 9.427, ANEEL plays a crucial role in ensuring the efficiency, reliability, and sustainability of the country's electricity industry. Here are some of its key responsibilities:

1. Regulatory Oversight: ANEEL is tasked with regulating various aspects of the electric power sector, including generation, transmission, distribution, and commercialization. Establish rules, standards, and policies to govern the operations of companies operating within these segments.

2. Tariff Regulation: ANEEL is responsible for setting electricity tariffs and ensuring that they are fair, transparent, and reflect the costs associated with generating, transmitting, and distributing electricity. The agency conducts periodic review and adjustments of tariffs to align prices with market conditions and operating expenses.

3. Licensing and Authorization: ANEEL grants licenses and authorizations to companies that want to operate in the electric power sector. This includes generation plants, transmission lines, distribution networks, and commercialization activities. The agency ensures that these entities comply with technical, environmental, and legal requirements.

4. Market Monitoring and Competition Oversight: ANEEL monitors the electricity market to promote competition, prevent anti-competitive practices, and protect the interests of consumers. The agency investigates complaints, monitors market dynamics, and enforces regulations to maintain a level playing field for market participants.

5. Consumer protection: ANEEL is committed to protecting the rights and interests of electricity consumers. The agency establishes standards for service quality, reliability, and customer relations and oversees compliance by utilities. ANEEL also handles consumer complaints and disputes, mediated between consumers and service providers when necessary.

6. Renewable Energy Promotion: ANEEL promotes the development of renewable energy sources, such as wind, solar, biomass and small hydroelectric plants. The agency implements incentive programs, sets goals for the deployment of renewable energy, and facilitates the integration of renewable energy projects into the national grid.

7. Infrastructure Planning and Expansion: ANEEL participates in the planning and expansion of Brazil's electric power infrastructure, including transmission lines, substations, and interconnections. The agency assesses the need for new investments, approves infrastructure projects, and supervises their execution to ensure that they meet technical and regulatory requirements.

1.2 Service quality standards

Among other responsibilities, ANEEL establishes standards for quality service to protect consumer rights. As an example, to ensure the reliability, efficiency and safety of electricity transmission operations, ANEEL set some key standards for service quality, including:

1. Transmission System Availability: sets requirements for the availability of transmission lines and associated equipment to ensure the continuous and reliable transmission of electricity. Transmission companies must meet the minimum availability targets established by ANEEL to maintain the integrity and reliability of the transmission grid.

2. Voltage Regulation: Establishes standards for voltage regulation within the transmission system to ensure that electricity is transmitted at specified voltage levels and within allowed variations. These standards help maintain the stability of the transmission network and protect electrical equipment from damage caused by voltage fluctuations.

3. Fault Response Time: Mandates requirements for the response time of transmission companies to faults and disturbances in the transmission system. Transmission operators must promptly identify and rectify faults to minimize electrical supply interruptions and prevent cascading failures within the network.

4. Maintenance and Inspection Standards: Set standards for the maintenance and inspection of transmission infrastructure, including transmission lines, substations, and associated equipment. Transmission companies must adhere to these standards to ensure the reliability and safety of the transmission system and to prevent equipment failures.

5. Reliability Standards: Establishes reliability standards for the transmission system, including criteria to assess the reliability and adequacy of the transmission infrastructure. These standards help identify potential risks to system reliability and ensure that transmission companies take appropriate measures to maintain system stability and integrity.

6. Emergency Response and Restoration Procedures: Mandates procedures for emergency response and system restoration in the event of transmission system failures or disruptions. Transmission operators must have emergency plans in place to quickly restore service and minimize the impact of outages on electricity consumers.

7. Data Reporting and Monitoring Requirements: Requires transmission companies to regularly report operational data and performance metrics related to service quality. The agency monitors compliance with established standards and takes enforcement actions against companies that do not meet the requirements.

1.3 Current maintenance practices

Maintenance is a critical factor in the industrial sector, affecting costs and reliability and, therefore, influencing the competitiveness of a company. Unforeseen equipment failures can disrupt operations, leading to penalties and damage to the company's reputation. Detecting and addressing equipment faults is crucial to avoiding production interruptions.

Maintenance practices can be classified into three types:

1. Reactive maintenance involves fixing failures as they occur.
2. Preventive maintenance follows a schedule to prevent breakdowns.
3. Predictive maintenance uses tools and data to anticipate failures and intervene before they occur.

Advancements in technologies such as IoT, sensing, and AI have changed maintenance strategies from reactive and preventive to predictive. Reactive maintenance restores the equipment after failure, leading to delays and high downtime costs. Preventive maintenance aims to prevent breakdowns but can result in unnecessary costs. Predictive maintenance strikes a balance by identifying potential failures in real time and intervening before they occur, minimizing both reactive and preventive maintenance costs(RAN et al., 2019).

Predictive maintenance encompasses four stages of development. At the highest tier, it incorporates data analytics and real-time equipment monitoring. Analytical methods, such as machine learning algorithms, are employed to reveal obscured correlations and discern significant patterns within vast and intricate datasets, particularly in challenging industrial environments (PACHECO VAGNER PAES; MAROTTI, 2023).

In the context of the electrical sector, predictive maintenance brings a series of advantages to the management of equipment comprising large-scale energy generation and transmission systems, where reliability is crucial due to the widespread implications of failures. The following benefits stand out:

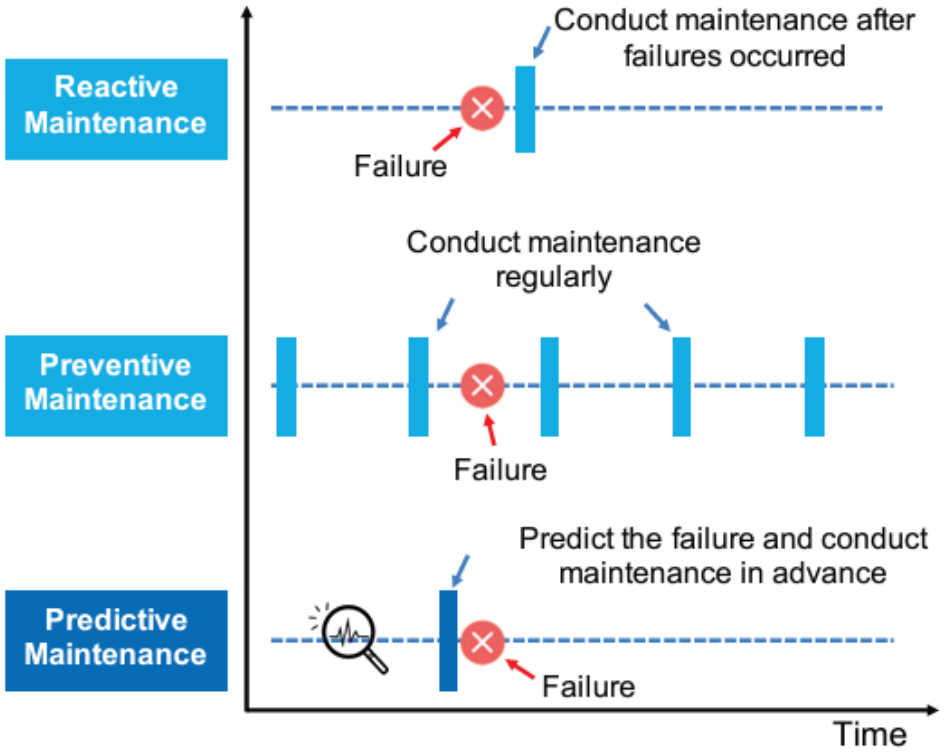


Figure 1.1: Maintenance Plans(RAN et al., 2019)

1. Improvement in service quality: Anticipating faults, Predictive Maintenance helps prevent unplanned downtime, ensuring continuous electricity supply, especially in critical sectors like hospitals and industries. This contributes to a better quality of life and economic efficiency.

2. Cost reduction: By anticipating faults, companies can plan maintenance activities more efficiently, avoiding unplanned downtime, emergency repairs, and unnecessary equipment replacements. This reduces associated costs and optimizes resource use.

3. Accurate fault prediction: Machine learning applications enable an accurate prediction of when electrical equipment might fail, allowing proactive interventions before serious damage occurs.

4. Extension of the lifetime of the equipment: Proactive intervention through AI-based Predictive Maintenance helps prolong the lifespan of electrical equipment, reducing replacement costs, and improving the return on investment.

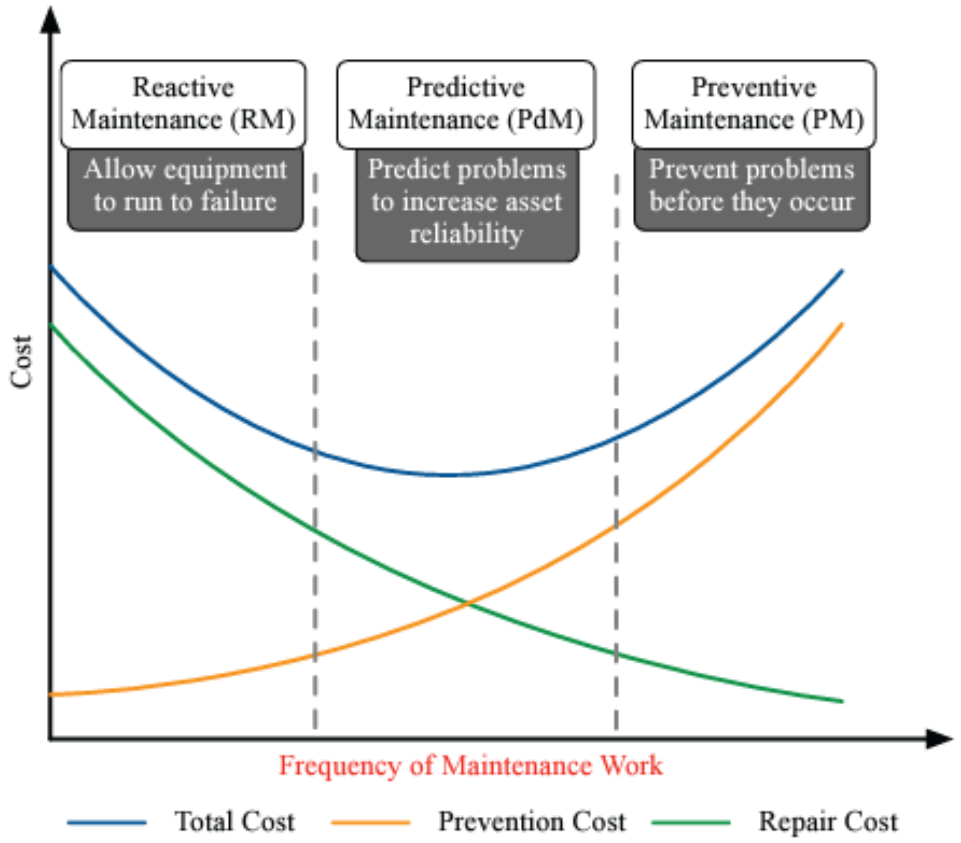


Figure 1.2: Maintenance Costs(RAN et al., 2019)

1.4 Data in electrical companies

According to (ZHU et al., 2015), in the context of electrical companies, there are different utility information systems used to support its operations, such as:

1. AMI - Advanced Metering Infrastructure: A system that allows for two-way communication between a utility and its customers' meters. AMI enables the collection of detailed consumption data and supports functions such as remote meter reading, outage detection, and demand response.

2. SCADA - Supervisory Control and Data Acquisition: A control system architecture that uses computers, networked data communications, and graphical user interfaces for high-level process supervision management. SCADA systems are used to monitor and control industrial processes, such as those in electrical power systems.

3. GIS - Geographic Information System: A system designed to capture, store, manipulate, analyze, manage, and present spatial or geographic

data. In the context of electrical companies, GIS is used to manage and analyze geographic information related to infrastructure, such as power lines, transformers, and substations.

4. WMS - Workforce Management System: A system used to manage the scheduling, dispatch, and tracking of field service workers. In the context of electrical companies, WMS helps optimize the allocation of resources for tasks such as maintenance, repairs, and installations.

5. DMS - Distribution Management System: A system that helps to monitor and control the distribution of electricity in a utility's grid. DMS includes functions such as outage management, fault detection, and load balancing to ensure efficient and reliable operation of the distribution network.

6. CIS - Customer Information System: A system used to manage customer-related information, such as billing, account management, and customer service interactions. In the context of electrical companies, CIS helps manage customer accounts, billing, and communication with customers.

In the context of the Brazilian electrical sector, some systems stand out, which are used by most companies involved in power generation, transmission, distribution, and trading.

1. SAGE: is a large-scale and high-performance SCADA/EMS system, developed and continuously updated by CEPEL. Its modular architecture allows for proper customization, enabling it to be used as a communication gateway, a data concentrator for a distribution system, a local or regional supervisory system, an operation center for a system, or even a 'multisite' system composed of multiple redundant and synchronized control centers (PEREIRA et al., 2014).

2. SAP Plant Maintenance (SAP PM): SAP is an Enterprise Resource Planning (ERP) system widely used by electrical companies in Brazil. Its modular architecture has several specialized modules to support the business processes of an entire organization. The Plant Maintenance module is used to register maintenance data on equipment such as power transformers, capacity banks, and power lines.

3. SATRA: System that supports monitoring activities of operation, maintenance tracking, and system performance statistics, and its function is to automate, in a decentralized and coordinated manner, the procedures

to determine shutdowns and temporary operational restrictions in the transmission functions of networks (ONS, 2020).

4. AMSE: Monthly transmission services and charge settlement system. AMSE calculates the financial amounts related to a specific settlement month. These amounts are allocated to services provided by transmission concessionaires and ONS; to the use of the transmission system by users, and to sectorial charges (ONS, 2020).

The wide variety of data that these systems handle poses additional challenges in achieving system interoperability. Utility information systems, usually structured as separate business entities, depict power systems and their operations from different business viewpoints. This leads to a multitude of overlapping and occasionally contradictory information models stored in numerous incompatible formats. (ZHU et al., 2015).

1.5 Conclusion

The deep transformations in the Brazilian electrical sector, driven by the federal government in the late 1990s, resulted in an increase in the capacity and reliability of the generation and transmission system. This increase is directly related to the regulatory role of ANEEL, which encouraged companies in the sector to improve the quality of their services through tax incentives or the application of penalties.

In order to reduce failures and unavailability of their equipment, companies started to improve their maintenance processes by incorporating predictive maintenance techniques, based on the analysis of data from these equipment available in their internal systems and/or from sensors attached to these equipment. However, performing this analysis presupposes the existence of a technological environment that allows the acquisition of these data, their processing, and their display in specific tools.

In a more specific context, with the use of Data Science techniques and maintenance and operation databases of transmission assets from a substation, it is possible to build predictive models that allow for the enhancement of maintenance plans for these equipment. To achieve this, historical data from these databases should be centralized in a common data environment, where analytical models will be available for data analysis and presentation of results on dashboards. The analytical models developed will complement existing predictive methods, with the objective of improving maintenance planning. To do so, technologies such as cloud computing, data structuring in data lakes

(Big Data), and the application of artificial intelligence techniques (Machine Learning) should be used.

The motivation of this work is to propose a data reference architecture capable of supporting a common data environment to the development of data science projects capable of building predictive models that can improve maintenance processes of electric company's assets.

1.6 Dissertation organization

This document is structured as follows. In Chapter 2 we investigate the literature on the most important topics of this work: Big Data, Cloud Computing, Data Architectures, and Data Governance. Each topic highlights the main concepts that guide the development of this dissertation. In Chapter 3 we present a data reference architecture suitable for data analysis and data science projects, considering aspects of data governance in the context of a Brazilian energy company. In Chapter 4 we show some results obtained from the implementation of the presented data architecture in the context of a R&D project developed for Eletrobras Furnas. Finally, in Chapter 5 we present our conclusions and future perspectives about data, analytics, and governance.

2
Related Work

2.1
Introduction

In the era of rapid digital transformation, data access, management, and analysis have become key factors driving organizational success. From enterprises to academia, there has been a concerted effort to understand and refine data architectures and governance practices to harness the full potential of data assets. This chapter delves into academic research and industry practices that have shaped understanding of data architectures and data governance over the years.

The evolution of data architectures begins in the early days of hierarchical and relational databases until the contemporary landscape characterized by distributed computing paradigms such as cloud computing and edge computing. Academic research has played a crucial role in elucidating the principles, methodologies, and technologies that underlie these architectures. The literature abounds with insights into the design, implementation, and optimization of data architectures tailored to diverse organizational needs.

In parallel, the discourse on data governance has gained prominence as organizations grapple with regulatory compliance, data privacy concerns, and the imperative to ensure data quality and integrity. Researchers have examined the multifaceted dimensions of data governance, including legal and ethical considerations, organizational structures, policies, and technologies. The interdisciplinary nature of data governance has encouraged collaborations between researchers in fields as diverse as computer science, law, management, and ethics, leading to a myriad of theoretical frameworks, best practices, and case studies.

Furthermore, the convergence of data architectures and governance has given rise to novel paradigms such as DataOps and Data Governance 2.0, which seek to integrate agile principles, automation, and collaboration into the fabric of data management practices. By synthesizing insights from academia, industry, and regulatory bodies, researchers go toward data-driven organizations characterized by agility, transparency, and accountability.

This chapter outlines what the literature says about the most important topics in the context of this study, namely: Big Data, Cloud Computing, Data Architectures, and Data Governance.

2.2 Big Data

When one thinks about the amount of information produced and consumed on the world, the term "Big Data" quickly comes to mind. But what exactly is Big Data?

According to (WARD; BARKER, 2013), the term Big Data has become ubiquitous. Due to a shared origin among academia, industry, and the media, there is no single definition, and various stakeholders provide diverse and often contradictory definitions. The lack of a consistent definition introduces ambiguity and complicates the understanding of this term.

Perhaps the first mention of Big Data dates back to the early 1990s, with an article by (GULLBERG, 1991) associating the term with cultural aspects (e.g., language) of society and the ability to extract knowledge from these aspects. However, the term Big Data began to be defined more closely to what we know today in the late 1990s. (MASHEY, 1999) cited Big Data associated with the increase in data volume and the limitations of the technological infrastructure to support this increase in volume.

Interestingly, in the literature, one of the most cited definitions on this topic is from a Meta (now Gartner) report from 2001 (LANEY et al., 2001) in which the term Big Data is not mentioned. However, this report proposes a vision that encompasses the "three Vs" - Volume, Velocity, and Variety, based on significant growth trends observed at the time with respect to the quantity of data produced, the rate at which it was produced, and the range of formats and representations employed. In this sense, although not immediately associated with the term Big Data, over time, the three Vs have come to establish a framework for the concepts that underpin this term.

In this context, it is worth noting that the National Institute of Standards and Technology (NIST), through the Big Data Public Working Group (NBD-PWG), developed several definitions related to Big Data, in addition to the term itself, from an initiative that established, together with industry, academia, and government, an extensible Big Data Interoperability Framework (NBDIF) independent of technology and infrastructure from technology vendors. It allows stakeholders in Big Data (e.g., data scientists, researchers, etc.) to use the best available analysis tools to process and gain knowledge through the use of standard interfaces between interchangeable architecture components. NIST conceptualizes Big Data as extensive datasets, primarily characterized by volume, velocity, variety, and/or variability, that require a scalable architecture for efficiency, storage, manipulation, and analysis(NIST, 2019).

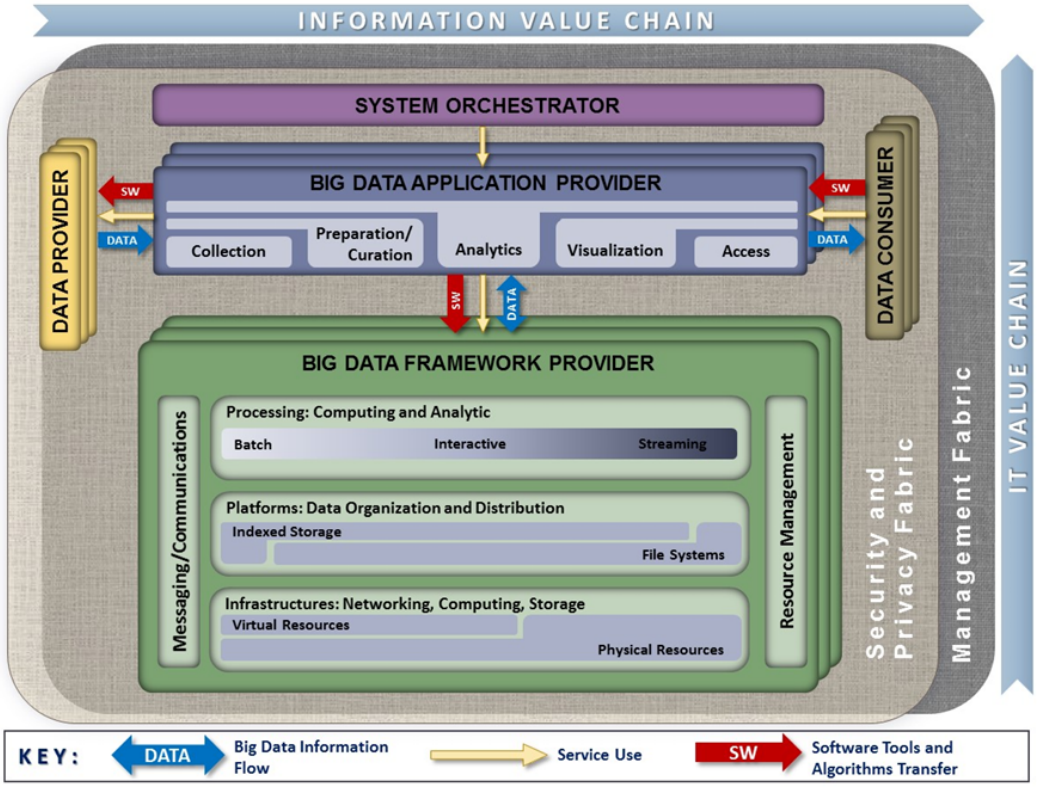


Figure 2.1: the NIST Big Data Reference Architecture (NIST, 2019).

The growing interest in Big Data has driven the market to increasingly appropriate this buzzword, resulting in a wide spectrum of concepts that further complicate the understanding of the term and its ability to precisely define something. In this sense, in their 2015 article, (MAURO; GRECO; GRIMALDI, 2015) propose a consensual definition of the term Big Data:

"Big Data represents information assets characterized by high volume, velocity, and variety that require specific Technology and Analytical Methods for their transformation into value."

This may be one of the first definitions of "Big Data" to establish a clear relationship between information and value. And to obtain value from this information, we need specific technology and analytical methods. It is the combination of these two elements that produces results that will underpin decision-making. So, we can say that the value obtained from information is directly related to the ability to make decisions based on that information.

Another important aspect related to the value of information is mentioned by (ANDERSON, 2017), referring to one of the seven principles of the "hacker ethic," published in the book "Hackers: Heroes of Computer Revolution" by Steven Levy in 1984. The principle that "All information should be free" could be interpreted as "at no cost" or "free," since the word "free" in

English can take on one meaning or the other depending on the context of the sentence. In developing the theme in his book, Anderson proposes the following understanding:

> *"Commoditized information (everyone gets the same version) wants to be free. Customized information (you get something unique that makes sense to you) is valuable."*

Thus, we can say that customized information is nothing more than commodity information that goes through specific analytical methods and produces a unique result that makes sense within a specific context, supporting decision-making.

A clear example of the relationship between information, value, and decision making can be seen in the Waze application. This application's main function is to determine the best traffic route between a pre-informed origin and destination. For this, Waze analyzes a series of information (road map of the geographical region between origin and destination, real-time traffic conditions of each street comprising one of the selected route options, which route(s) have tolls, etc.). In other words, from a series of commodity information (road map, general traffic conditions), Waze generates customized information (best route between origin and destination and estimated arrival time) that is only relevant to you for your decision-making (do I drive or opt for another transportation solution - subway, bicycle, etc.). This information has intrinsic value for those who need to take that route at that moment.

2.3 Cloud Computing

According to (YANG et al., 2017), the adoption of Big Data in industry and government expanded the meaning of this concept, including technologies and expertise to obtain value from data.

When it comes to technology, cloud computing represents a framework that facilitates widespread, easy, and immediate access to a communal reservoir of adaptable computing assets (such as networks, servers, storage, applications, and services) through the Internet. These assets can be quickly deployed and ready to use with minimal supervision or involvement of service providers (MELL; GRANCE et al., 2011).

Some features make this kind of technology the best option for supporting complex data analysis applications. Some of these features are as follows:

– Elasticity: The ability to adjust resources as needed, either automatically or manually, allows for quick and flexible scaling up or down based

on demand. Consumers may perceive the available capabilities to be limitless and accessible at any time.

– Pooled: The provider combines their computing resources to serve multiple users in a multi-tenant structure, where physical and virtual resources are allocated and reallocated based on consumer needs. Customers do not have direct control over the specific location of the resources but may be able to specify a general location (such as country or datacenter). Resources can include storage, processing power, memory, and network connectivity.

– On demand: Computing resources can be immediately adjusted to support workload variations.

– Self-service: A consumer can autonomously allocate computing resources without having to interact with individual service providers.

– Pay-as-you-go: Computing resources are billed only for the time they are used.

As proposed by (YANG et al., 2017), Big Data challenges can be addressed by cloud computing features for a given application, such as:

Table 2.1: Addressing the Big Data challenges with cloud computing for a dust storm forecasting application. Adapted from (YANG et al., 2017)

Big Data/Cloud computing	Elasticity	Pooled	On-demand	Self-service	Pay-as-you-go
Volume		X			X
Velocity	X	X			
Variety	X	X		X	
Veracity				X	X
Value	X		X		X

When it comes to the industry, Cloud computing offers several financial advantages for organizations, particularly in terms of capital expenditure (CapEx) and operational expenditure (OpEx)(RAFIQUE et al., 2011):

– Reduced Capital Expenditure (CapEx): Traditional on-premises infrastructure requires significant upfront investment in hardware, software licenses, data centers, and other infrastructure components. This expenditure is classified as capital expenditure (CapEx) and represents a substantial financial commitment. With cloud computing, organizations can eliminate or significantly reduce these upfront capital expenditures. Instead of purchasing and maintaining physical hardware, they can leverage

the infrastructure provided by cloud service providers on a pay-as-you-go basis. This shift from CapEx to operating expenditure (OpEx) allows organizations to conserve capital and allocate financial resources more effectively.

- Predictable Operational Expenditure (OpEx): Cloud computing operates on a subscription-based model, where organizations pay for the computing resources and services they consume recurring. This expenditure is classified as operational expenditure (OpEx) and is typically more predictable and manageable than the upfront costs associated with on-premises infrastructure. By paying only for the resources they use, organizations can better align their costs with their actual usage and demand. This helps to optimize cost efficiency and avoid over-provisioning or under-utilization of resources, leading to potential cost savings over time.
- Scalable Cost Structure: Cloud computing offers elastic scalability, allowing organizations to scale their computing resources up or down in response to changes in demand or workload. This scalability extends to cost, as organizations only pay for the resources they consume, whether it is an increase in capacity during peak periods or a decrease during off-peak times. This scalable cost structure enables organizations to avoid the need for costly over-provisioning of infrastructure to accommodate peak loads, reducing wastage, and optimizing cost efficiency.
- Lower Total Cost of Ownership (TCO): Cloud computing can contribute to a lower total cost of ownership (TCO) compared to traditional on-premises infrastructure. While the subscription-based pricing model of cloud services may result in higher ongoing operational costs, it often eliminates or reduces many of the hidden costs associated with maintaining on-premises infrastructure, such as hardware depreciation, maintenance, upgrades, and staffing. Additionally, cloud service providers benefit from economies of scale, allowing them to achieve efficiencies and cost savings that may be difficult for individual organizations to replicate with on-premises infrastructure.

2.4 Data Architectures

A data reference architecture serves as a blueprint or framework that guides the design, implementation, and management of an organization's data infrastructure and systems. Here are some key activities and benefits associated with using a data reference architecture.

- Standardization: A data reference architecture provides standardized guidelines, principles, and best practices for organizing and structuring data assets within an organization. Promote consistency and coherence in data management practices across different departments, projects, and systems.
- Interoperability: By defining common data standards, formats, and interfaces, a data reference architecture facilitates interoperability between disparate systems and data sources. This interoperability enables seamless data exchange, integration, and collaboration throughout the organization.
- Data Governance: A data reference architecture includes governance policies, processes, and controls to ensure data quality, integrity, security, and compliance. It helps organizations establish clear accountability and responsibilities for managing data assets and enforcing data-related policies.
- Data Integration: A data reference architecture provides guidelines for integrating and harmonizing data from diverse sources, formats, and systems. It defines standardized data integration patterns, techniques, and tools to streamline the process of aggregating, cleansing, transforming, and loading data into target systems.
- Data Storage and Management: A data reference architecture outlines strategies for storing, organizing and managing data repositories, such as data warehouses, data lakes, and databases. It helps organizations optimize data storage, retrieval, and access patterns to meet performance, scalability, and cost requirements.
- Analytics and Insights: A data reference architecture supports the implementation of analytics and business intelligence solutions by defining data modeling, querying, and analysis techniques. It enables organizations to derive meaningful insights, trends, and patterns from their data assets to support decision-making and strategic planning.
- Data Life-cycle Management: A data reference architecture encompasses the entire data life-cycle, from data acquisition and ingestion to archival and disposal. Provides guidelines for managing data throughout its life-cycle, including data retention policies, data lineage tracking, and metadata management.
- Adaptability and Flexibility: A data reference architecture is designed to be flexible and adaptable to accommodate evolving business requirements, technological advancements, and regulatory changes. It allows

organizations to iteratively refine and extend their data infrastructure and systems to meet emerging needs and challenges.

Data architectures are also a powerful tool for understanding data flows and facilitate the selection of technologies to implement a corporate big data system. (PÄÄKKÖNEN; PAKKALA, 2015) propose a framework for the design and analysis of data reference architectures, representing data stores, functionalities, and data flows in a comprehensive diagram, independent of technology.

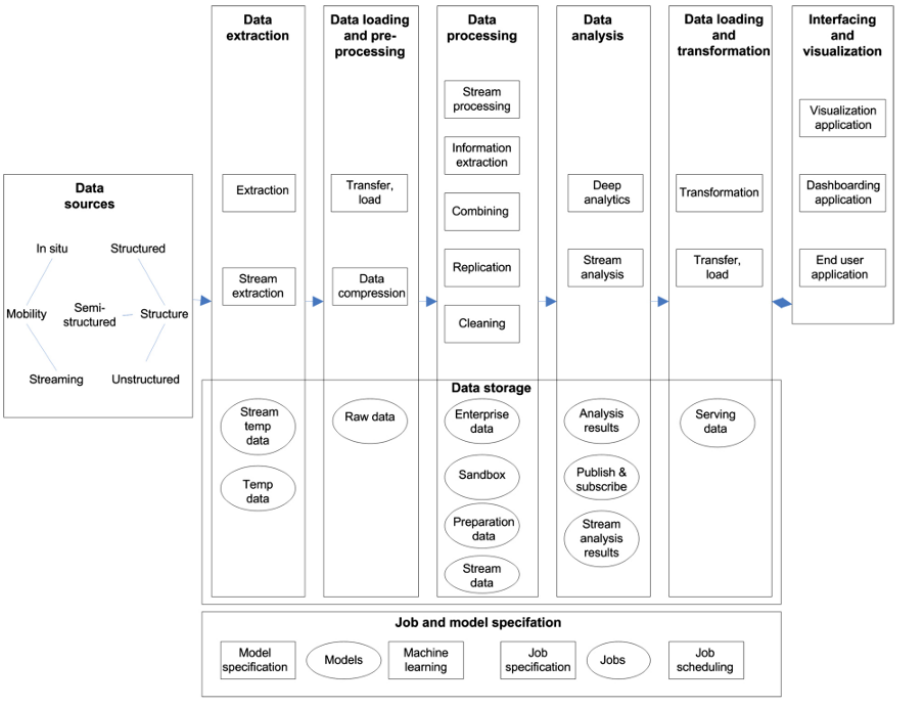


Figure 2.2: High level design of a Data Reference Architecture (PÄÄKKÖNEN; PAKKALA, 2015).

(ZBURIVSKY; PARTNER, 2021) propose a reference architecture with four layers, which is widely used in industry. Although these logical layers are suitable for describing architectures of different levels of complexity, they are populated and adapted with the minimum components necessary for the objectives of this proposal. The Ingestion layer is responsible for obtaining data from the original sources and loading them into the cloud data platform. Once loaded into the platform, the data is ingested into a Data Lake, a repository of data in its raw form. Automated processes for data cleaning and treatment perform the necessary transformations to feed a Structured Data Repository (which here can be implemented as a Data Warehouse, a relational database, or NoSQL). These two components form the storage layer. In the Processing layer, the data are used for exploratory analysis and for training and testing Machine

Learning models. Finally, in the Service layer, the data feeds dashboards and visualization applications.

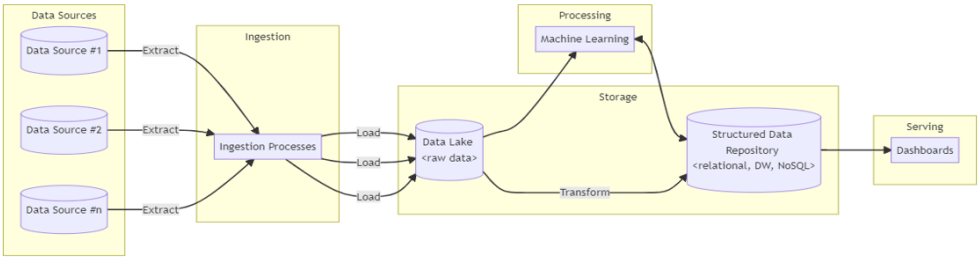


Figure 2.3: Data Reference Architecture with four layers (ZBURIVSKY; PARTNER, 2021).

2.5 Data Governance

Data governance is a set of processes, policies, standards, and controls that ensure the effective and secure management of an organization's data assets throughout their life-cycle. It encompasses the planning, supervision, and enforcement of data-related activities to ensure that data are accurate, consistent, secure, and compliant with regulatory requirements.

Key components of data governance include the following.

- Data Quality Management: Ensure that the data is accurate, complete, consistent, and reliable. This involves the establishment of standards and processes for data validation, cleaning, and enrichment.
- Data Security: Protect sensitive data from unauthorized access, disclosure, or alteration. This includes implementing access controls, encryption, authentication, and monitoring mechanisms to protect data confidentiality, integrity, and availability.
- Data Privacy: Ensure that data are handled in compliance with privacy regulations and policies, such as LGPD, GDPR, CCPA, and HIPAA. This involves defining data privacy rules, obtaining consent for data processing, and implementing measures to anonymize or pseudonymize personally identifiable information (PII).
- Data Compliance: Ensure that data management practices comply with relevant regulatory requirements, industry standards, and organizational policies. This includes conducting audits, assessments, and risk analyzes to identify and mitigate compliance risks.

- Data Stewardship: Assigning accountability and responsibility for managing specific data assets to designated individuals or teams. Data stewards are responsible for defining data requirements, resolving data issues, and ensuring data quality and integrity.
- Data Life-cycle Management: Managing data throughout its life-cycle, from creation and acquisition to archival or disposal. This involves defining policies and procedures for data retention, archival, backup, and deletion to optimize storage resources and ensure regulatory compliance.
- Metadata Management: Managing metadata, which provides contextual information about data assets, such as their structure, lineage, usage, and ownership. Effective metadata management enables data discovery, lineage tracing, and impact analysis.
- Data Governance Framework: Establish a governance framework that defines the roles, responsibilities, processes, and tools to manage data governance activities. This framework provides a structured approach to implementing and maintaining data governance initiatives within an organization.

By the early 1990s, business decisions and processes started to be driven by data and data analysis. Data governance emerged to handle the growing volume, velocity, and variety of data in corporate systems such as ERP and CRM, as well as data warehouses and other repositories. DAMA defines data governance as "the exercise of authority, control, and shared decision making (planning, monitoring, and enforcement) over the management of data assets". In other words, according to DAMA, data governance complements data management, not replaces it (AL-RUITHE; BENKHELIFA; HAMEED, 2019).

The Data Governance Institute, an organization that provides resources, guidance, and best practices to implement effective data governance programs, proposes a framework for data governance, which is a comprehensive model to understand and implement data governance within organizations. This framework consists of these key components (DGI, 2024 (accessed in March 2024)):

- Data Governance Objectives and Scope: Defines the goals, objectives, and scope of the data governance program. Clarifies the purpose of data governance and identifies the key stakeholders, data domains, and business processes that will be included in the governance initiative.
- Data Governance Roles and Responsibilities: Defines the roles and responsibilities of the individuals and groups involved in the data governance program. This includes executive sponsors, data stewards, data

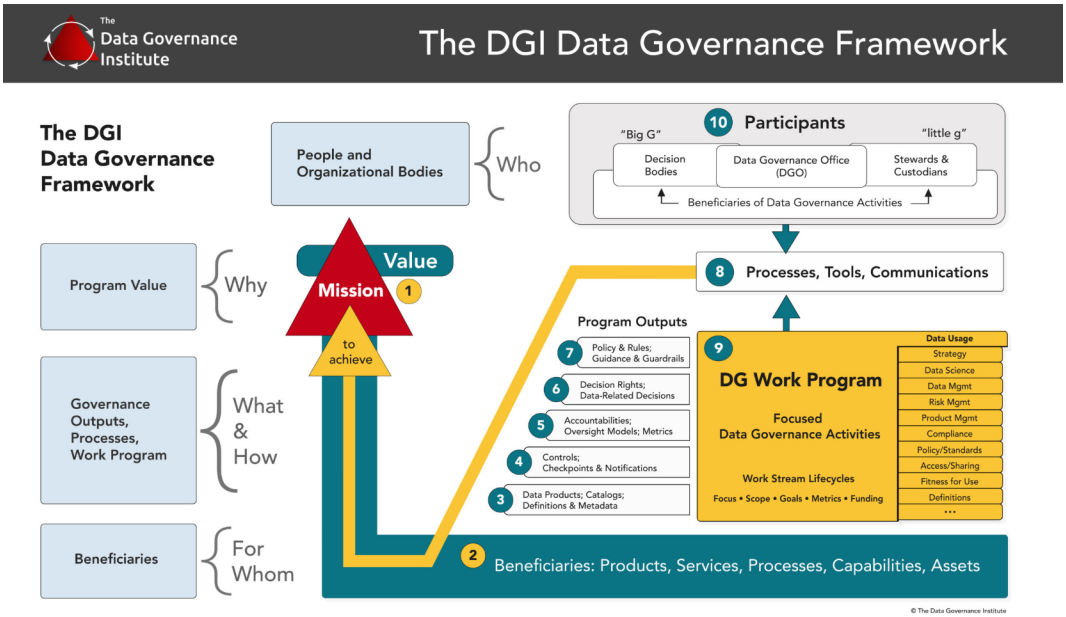


Figure 2.4: Data Governance Framework (DGI, 2024 (accessed in March 2024)).

owners, data custodians, and other stakeholders who play a role in the management and governance of data assets.

- Data Governance Policies and Standards: Establishes the policies, standards, and guidelines that govern how data is managed, accessed, and used within the organization. Covers aspects such as data quality, data security, data privacy, data classification, and data life-cycle management.
- Data Governance Processes and Procedures: Outlines the processes and procedures for implementing and executing data governance activities. This includes processes for data management, data issue management, data quality management, metadata management, and compliance monitoring.
- Data Governance Tools and Technologies: Includes recommendations for tools and technologies that support data governance activities. This could include data governance software platforms, metadata management tools, data quality tools, and other solutions that facilitate data governance implementation.
- Data Governance Metrics and Monitoring: Defines the metrics and key performance indicators (KPIs) used to measure the effectiveness of the data governance program. It includes monitoring mechanisms to track compliance, data quality, data usage, and other relevant metrics to ensure that data governance objectives are met.

– Data Governance Communication and Training: Addresses communication and training initiatives to ensure that stakeholders understand the importance of data governance and their roles in the governance process. It includes strategies to promote awareness, participation, and adoption of data governance practices throughout the organization.

2.6 Conclusion

Strategic data management has emerged as a critical driver of organizational success. The evolution of data architectures and governance, from early hierarchical and relational databases to contemporary paradigms like cloud and edge computing, plays a key role in this new era of big data analytics. Academic research has been instrumental in elucidating the principles and technologies underlying these architectures, providing insights into design, implementation, and optimization tailored to organizational needs.

Simultaneously, the discourse on data governance has gained prominence, addressing regulatory compliance, privacy concerns, and the imperative to ensure data quality and integrity. Researchers have explored legal, ethical, and organizational dimensions, fostering interdisciplinary collaborations and producing theoretical frameworks, best practices, and case studies.

By synthesizing insights from academia, industry, and regulatory bodies, researchers have paved the way for data-driven organizations characterized by agility, transparency, and accountability. Big Data, Cloud Computing, Data Architectures, and Data Governance are the key topics in this scenario.

In the next chapter, there is a proposal for a data architecture suitable for Brazilian electric companies and designed to support big data analytics and data science projects. This architecture also addresses the data governance requirements expected for this kind of organization.

3
The proposed Data Reference Architecture

3.1
Introduction

In this chapter, there is a detailed proposal for a data architecture designed to support big data analytics and data science projects in the context of a Brazilian electricity company. The proposed architecture also addresses the data governance requirements expected for this kind of organization.

3.2
Data Reference Architecture

3.2.1
Overview

The data reference architecture presented in this work is part of a research and development project, called ENSIGHTS, carried out by a large company in the electrical sector during the years 2021 and 2024.

The research and development project aimed to develop an operational prototype of a computational platform in the cloud, which would assist in optimizing the predictive maintenance process of power transformers using machine learning techniques.

In terms of prototype definition, the platform is expected to:

- Build Analytical Models: Machine learning algorithms capable of predicting the probability of failures or other relevant equipment events within the scope of the project.
- Data Structure and Integration: a system capable of autonomously and dynamically:
  - Accessing data from the company's systems.
  - Processing the data in an ETL layer with a cloud-scalable architecture.
  - Access predictive algorithms and apply them to the data.
  - The algorithm results are loaded into the system's data lake.
- User-friendly Dashboard: where maintenance analysts from the company can consume the algorithm results to aid in decision making relevant to scheduling activities.

In addition to the functional requirements, there were two important non-functional requirements: (1) the architecture must have the lowest cost in terms of storage and processing, and (2) data storage and processing must be done in the Brazilian territory, to be compliant with legislation for brazilian state companies.

To meet these requirements, the proposed data reference architecture, in terms of the structure of the data lake, is summarized in Figure 3.1.

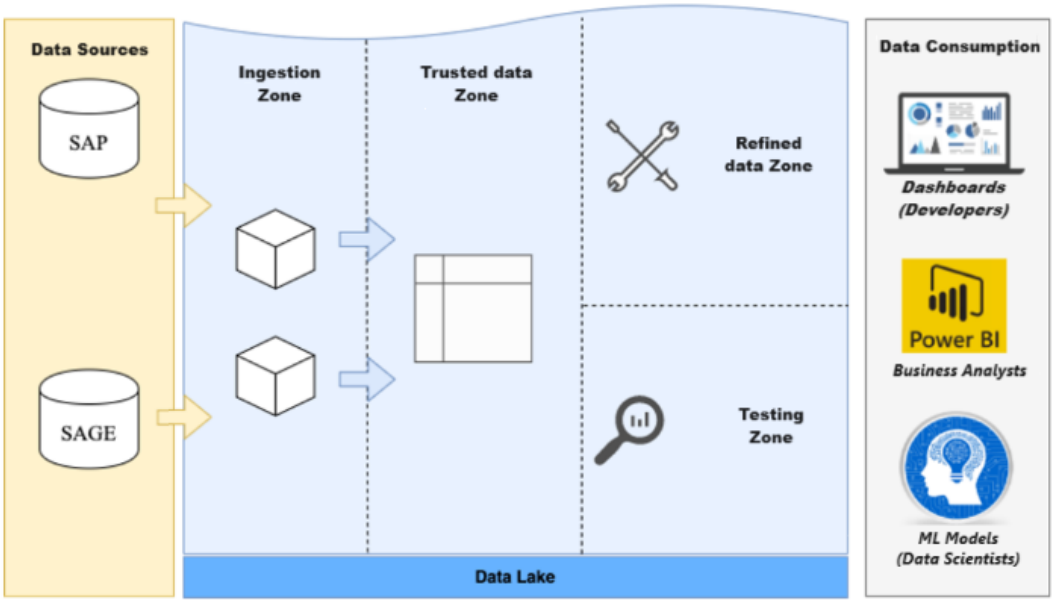


Figure 3.1: Data Lake Structure.

In a macro-view, SAP and SAGE data files are ingested from the company's on-premises environment to the cloud. Upon arrival in the cloud through pipelines configured for data ingestion from each system, these data files are sent to the Landing Zone of the Data Lake and then processed by specific data pipelines.

After being properly processed, the original replicas of the data present in the Landing Zone are transferred to the Raw Data Zone for backup, while the preprocessed data are sent to the Trusted Data Zone.

The Landing and Raw Data Zones are represented in Figure 3.1 by the Ingestion Zone. Upon arrival in the Trusted Data Zone, the data may still undergo further processing to more promptly serve specific services such as feeding indicators and dashboards. For this purpose, there is the Refined Data Zone, which acts as a Data Mart, specifically to store data related to a specific business domain, providing processed data sources for specific purposes.

Lastly, there is the Testing Zone, whose objective is to act as a data sandbox for conducting tests, primarily aiming at the development of new

machine learning models and other data science experiments. Arrows indicate data flows between components.

3.2.2 Data Reference Architecture Details

In Data Sources layer, all data sources of interest to the company are identified. According to (PÄÄKKÖNEN; PAKKALA, 2015), data sources should be classified along two dimensions: mobility and structure:

- Mobility: whether the data is *in situ*, which refers to data that exists at its original location, such as files stored on a local system or data residing in a database or *streaming* which refers to continuous flows of data generated in real-time, often from sources like social media platforms, sensors, or IoT devices.
- Structure: whether the data is structured (e.g., a database table), semistructured (e.g., XML or JSON files), or unstructured (e.g., images, videos, audio, etc.).

Data extraction from *in situ* and streaming data sources involves retrieval of raw data from these sources for further processing and analysis.

Extracting *in situ* data typically involves accessing and copying files or querying databases to retrieve the required data. This process may involve reading text files, executing SQL queries, or using APIs provided by the data source to fetch the data. For example, extracting data from a text file stored on a server involves reading the file directly from its location and transferring its contents to the processing environment.

Extracting streaming data involves capturing and processing data as it is generated, usually without initially storing it in a persistent storage medium. This process requires establishing connections to the streaming source and continuously receiving and processing data in near-real-time. For example, extracting streaming data from a social media platform involves subscribing to relevant data streams provided by the platform's API, receiving real-time updates, and processing them as they arrive.

In Ingestion layer, the raw data obtained from the data extraction are sent to the Data Lake landing zone. This operation is performed using pipelines.

In the Storage layer, the storage services planned for use in the proposed architecture are configured.

In the Processing Layer, the services chosen to process the data are configured.

In the Presentation Layer, services and applications are planned to consume data available in the refined and test data zones of Data Lake, as well as in available databases, and display the results of the analysis through dashboards, reports, or data science projects.

In the Cloud Monitoring, Data Management, and Governance layer are the services available operational monitoring of cloud-enabled services, as well as services for managing and governing data in this environment.

3.3 An Analysis of Implementation Alternatives

3.3.1 Implementation Alternatives

To choose the best cloud solution to implement the prototype requirements, a technology and cost evaluation was conducted with three cloud computing providers: Microsoft, Google and Amazon. These providers were chosen because they are considered hyperscaler cloud providers, due to the size of the computational infrastructure they possess, which ensures the delivery of virtually unlimited processing and storage capacity to their customers (DIGITALOCEAN, 2024 (accessed in April 2024)).

Below is an outline of the architecture considering the environments of Microsoft (Figure 3.2), Amazon (Figure 3.3) and Google (Figure 3.4).

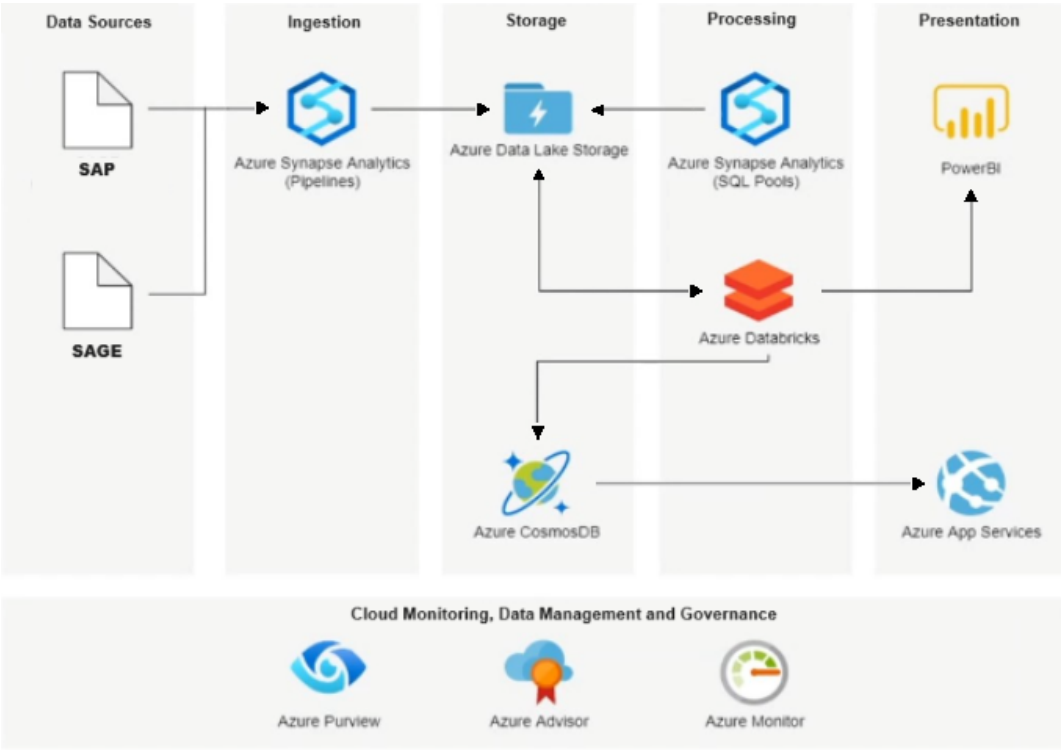


Figure 3.2: Data Reference Architecture in Azure.

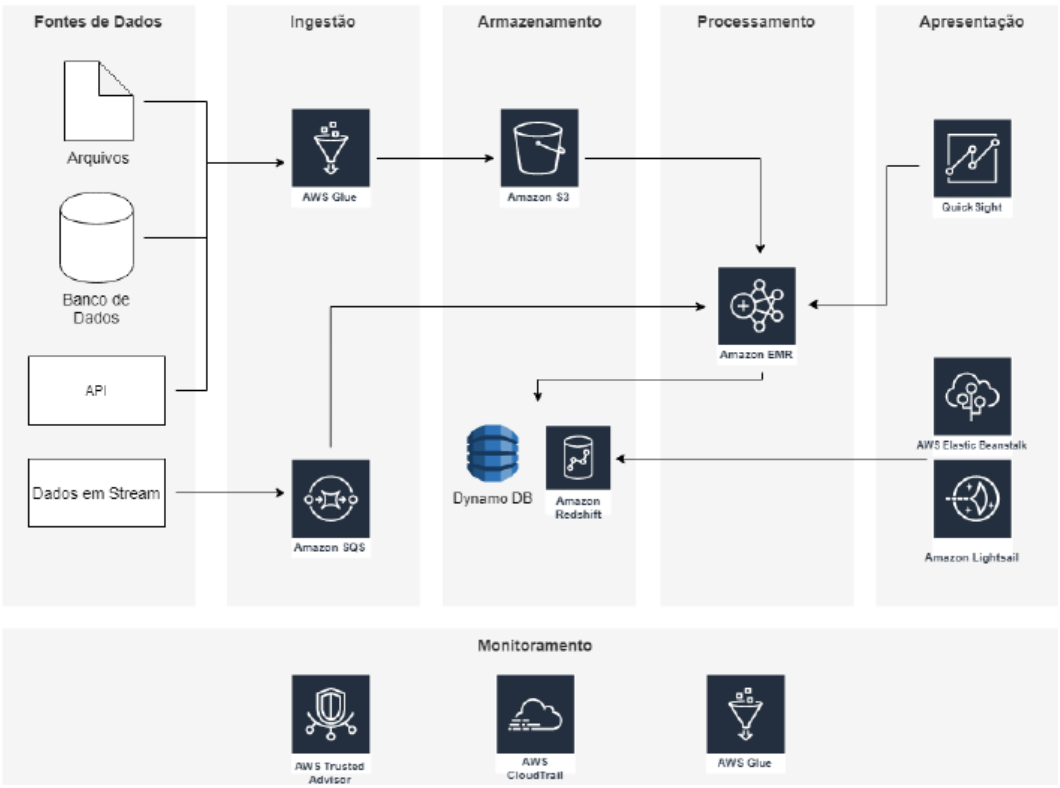


Figure 3.3: Data Reference Architecture in Amazon Web Services.

It is noticeable from the above figures that the architectures, regardless of the platform, have very similar topologies, with only the services used being different. The components of the ingestion layer are responsible for

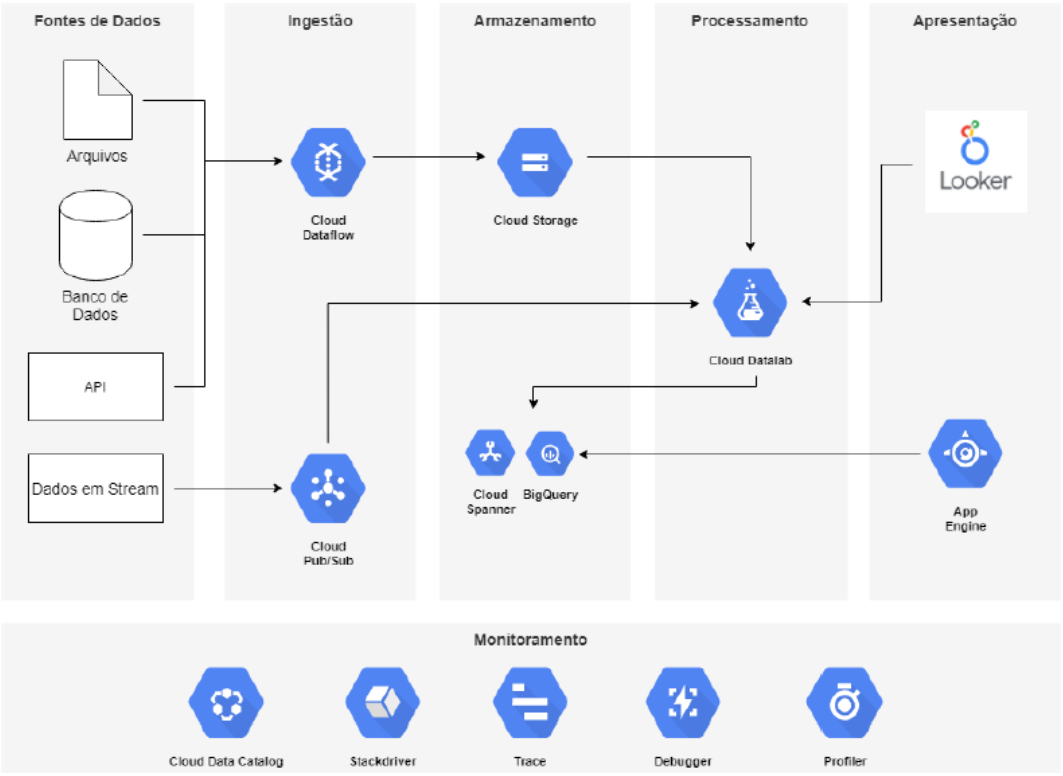


Figure 3.4: Data Reference Architecture in Google Cloud Platform.

integrating the various data sources, either batch or streaming, and performing ETL processes. The storage layer consists of the data lake, which will be the main repository of the data needed for the solution's intelligence, and other databases (transactional and analytical) to store internal application information and post-processed hot data. The processing layer is where all the intelligence of the solution should be developed with tools that allow for the creation of algorithms and data science analyses. The presentation layer has components for hosting dashboards and applications with access to stored data, which can be used to create customized reports. Finally, the monitoring layer consists of governance and data security components that span the entire application.

It is important to note that it is a simplified diagram; additional auxiliary services may be necessary to integrate between components depending on the provider.

3.3.2 Technology analysis

Although very similar, there are small differences that can lead to choosing one provider over another.

In terms of regions, all three providers have extensive coverage, with

servers in North America, South America, Europe, and Asia. In Brazil, all the required services can be instantiated, eliminating the need to host them outside the national territory. An exception is highlighted for one of the services provided by GCP (Cloud Spanner), which should be replaced by another service.

Among the three providers, AWS serves the largest number of customers (VISUAL-CAPITALIST, (accessed in April 2024)), offering resources for the vast majority of industry use cases, with a complete solution stack. As it represents about 16% of its holding's revenue (CRN, (accessed in April 2024)), AWS support is guaranteed. In recent years, some issues have arisen regarding the company's view of open source software, which impacts the usability and integration of its resources. Furthermore, the constant emergence of new services that replicate the functionalities of existing services disrupts the cohesion in integration (GARTNER, 2021 (accessed in July 2022)).

GCP is the smallest of the three providers. The cloud services offered still represent a very small portion of the company's revenue, raising questions about the longevity of the support provided (VISUAL-CAPITALIST, (accessed in April 2024)). It is also the provider that suffers the most from outages and instability, especially in the last year (HARMONY, (accessed in April 2021)). It has the least integration with third-party services, mainly due to a lack of partnerships.

In terms of integration, Azure has the most comprehensive solution, with partnerships with SAP, Oracle, among others, and is openly participative in the open-source software scene. It also has the most complete services, with various functionalities included in some products. Of the three, it has the most expensive specialized support and the least coverage of regions; however, these points do not impact the project since the support service is not planned and the region chosen for architecture implementation is offered by the provider. As observed in the diagrams from the previous section, in general, the required functionalities can be implemented in any of the three providers. It should be noted that Azure typically bundles many functionalities into single services, causing the need for two or more services from other providers to replicate the total functionality of a single service in Azure. However, often due to the concentration of functions, many of them are not used, despite being included in the service's value. Especially in GCP, the integration between components often requires the use of managed services, which complicates their configuration, maintenance, and introduces more failure points.

Databricks, present in Azure's architecture, is one of the most comprehensive solutions for creating machine learning models, large-scale data trans-

formation, and analytics, among others, being an optimized version of Spark, with performance improved up to 50 times faster (DATABRICKS, (accessed in April 2024)). Additionally, the use of Delta Lake, an optimized version of Parquet for Databricks, ensures greater performance in read and write operations and the history of table changes. Databricks can be used in conjunction with AWS and GCP services. However, in Azure, integration is complete with other services, while in AWS, external configurations are required, and in GCP, it is completely separate from its platform. Azure and AWS also offer more security features, utilizing security tools integrated into the platforms and facilitating integration with data ingestion and processing services.

In terms of databases, Azure's Cosmos DB is the only multimodel database platform that supports OLTP and OLAP. This is advantageous because it can be used both as a transactional application and as a data warehouse model. Additionally, despite being a proprietary database, it predominantly uses open source drivers, facilitating a possible component or architecture change, unlike, for example, AWS Dynamo, which uses a proprietary driver and is only document-oriented. For example, depending on the data model chosen in Cosmos DB, it is possible to use the native core (SQL) API, MongoDB API, Cassandra API, Gremlin API, and Table API (COSMOSDB, (accessed in April 2024)). In the case of AWS, it would also be necessary to use Redshift in conjunction with Dynamo to meet the need for analytical data. Redshift may have auto-scaling functionality depending on the type of machine used. By aggregating various types of data model, Cosmos DB consolidates the functionalities of two separate services in other providers. Because it encompasses various functionalities, it has a proportionally higher value.

Additionally, cloud providers offer an on-premise solution if there is no desire to store or transfer a large amount of data to the cloud, which can be advantageous if you want to turn the solution into a product. It should be noted that Azure's on-premise solution has more components compared to its competitors, making it possible to transform most of the architecture into a local solution. Azure's presentation solution, Power BI, is the most common business intelligence tool on the market, with extensive documentation and easy usability, and it is the company's choice to develop reports from BI analysis. Its competitors, Looker and Quicksight, although less used, also provide complete dashboarding solutions and integrations with key data services.

Regarding the documentation of all services, Azure also excels in completeness with manuals and diagrams of all publicly disclosed services. GCP does not publicly disclose the complete documentation of its services. AWS,

although also publicly presenting its documentation, is a bit less organized and more challenging to navigate.

3.3.3 Cost analysis

For the price estimation, the calculators of each provider were used. Each of them has a different pricing method. Details are shown in Table 3.1. The estimates were made based on the already known aspects of the project and experiences from other projects.

Table 3.1: Cost comparison of cloud computing providers

Layers\Cloud Providers	Azure	GCP	AWS
Ingestion	Data Factory, Service Bus	Cloud Dataflow, Cloud Pub/Sub	Glue, SQS
Storage	Datalake Storage, Cosmos DB	Cloud Storage, Cloud Spanner, BigQuery	S3, Dynamo DB, Redshift
Processing	Databricks	Cloud Datalab, CloudDataproc	EMR ou Databricks with specific contract
Presentation	PowerBI, App Services	Looker, App Engine	Quicksight, Elastic Beanstalk, Lightsail
Total cost estimated	3649,62 USD	4426,42 USD	4038,00 USD

3.3.3.1 Azure

The data factory is priced based on the type of agent for collecting data from the source, the number of activities executed in the workflows created within it, the use of data flows, and the quantity of read, write, and monitoring operations. In the estimate, it was considered that the data would be extracted from the four data sources five times per hour, using the agent installed locally (self-hosted Integration Runtime) and without the use of data flows. The other agent option would be for public endpoints, which is not the case for the project.

Estimates for storage and operation of the data lake and service bus were inferred from company's experience with projects that used the same type of architecture. A reasonable value of 5TB of storage was considered for the data lake and ten million operations for the service bus.

Cosmos DB is priced through its own processing unit (RU/s) and storage capacity. Azure capacity calculator was used for the value of 400 RU/s, and 2000GB of transactional storage and 2000GB of analytical storage were inferred.

For Databricks, both the machine configuration and the number of machines were defined based on the amount of data expected to be processed simultaneously and on past project experiences. For presentation, in the app service, two instances were considered, one for the front-end and another for the back-end, using Linux as the operating system and the basic level of app with a single-core instance with 1.75GB of RAM. The free version of Power BI meets the project's development demands, and hence does not add extra cost to the estimate.

Finally, Azure allows some services to be reserved in advance, which guarantees discounts of up to 60%.

3.3.3.2 AWS

Data Glue is priced by its own processing unit (DPU), the duration of the jobs executed, the provision of development endpoints, and Databrew sessions. The quantity of DPUs was estimated by equivalence with 8 vCPUs and 32GB of memory, a value inferred from experience. Two development endpoints were considered to meet demand, along with 5 interactive Databrew sessions.

SQS is priced by queue requests and data transfer. 5 million standard queue requests and 5 million FIFO queue requests were considered, totaling 10 million operations. For data transfer, a volume of 1TB input and 50GB output per month was inferred.

For storage components, 5TB of storage for S3 was estimated, with a data transfer rate of 1TB for both input and output. For DynamoDB, a capacity of 2000GB was inferred, along with Redshift also having 2000GB of storage and autoscaling functionality.

Similarly to Databricks, EMR configurations were defined on the basis of experiences from projects where similar amounts and characteristics of data were processed.

Presentation through QuickSight is charged based on the number of readers, authors, and SPICE capacity. An estimate of 50 readers with the company's name was raised, with 2 authors considered and 10GB of SPICE capacity.

3.3.3.3 GCP

Cloud Dataflow is priced based on the type of job, its duration, and its worker nodes. The execution of batch jobs throughout the month using 1 worker node of the type equivalent to 8 vCPUs and 30GB of memory was considered.

For storage, Cloud Storage with a capacity of 5TB was considered as the data lake, for analytics, BigQuery with 2000GB of storage and a volume of 1TB of consumed queries, and an additional 2000GB from Cloud Spanner.

Cloud Pub/Sub considers the volume of data and backlog storage for its pricing. A data volume of 1TB was inferred, 10GB for the retained acknowledged backlog, and an additional 10GB for snapshot backlog.

Similarly to Databricks, Cloud Dataproc configurations were defined based on experiences from projects where similar amounts and characteristics of data were processed. For presentation, two instances of the App Engine of type F4_1G were considered, which equates to a memory limit of 2GB and 2.4GHz CPU with compatible automatic scaling types.

The price estimates were made based on the information already presented in the project and may change as understanding of the databases and systems presented, as well as their integration methods and data volume, progresses. Additionally, the pricing methods of the services differ among providers. For example, a database may be priced based on storage space with one provider and on processing time and access quantity with another. This means that it may not be possible to exactly replicate configurations between providers, leading to some differences in prices.

3.3.4 Conclusion

Given the considerations outlined in the previous sections, it is understood that Microsoft Azure is the most suitable provider to meet the requirements necessary for the execution of the project. In addition to being widely used on the market, it is the solution that requires the fewest components and offers the greatest simplicity in integration, maintenance, and monitoring. Additionally, the solution used in the presentation layer, Power BI, is the most well known and well documented among the three providers, making it easier for nontechnical users to use. Any potential drawbacks, such as availability in certain regions, do not affect the project requirements. Despite being smaller than the others, Azure still has extensive coverage.

Finally, it is important to note that this analysis was performed in 2021 when the Azure Synapse Analytics service was not yet available. With the release of this service, the decision was made to replace Data Factory, as Synapse offers slightly lower costs and incorporates a larger set of features.

3.4 Azure Implementation Details

3.4.1 Ingestion Layer

As mentioned above, in this logical layer, the raw data obtained from the data extraction are sent to the Data Lake landing zone. This operation is performed using pipelines implemented in Azure Synapse Analytics.

Azure Synapse Analytics Pipelines is a cloud-based service that enables users to orchestrate and automate data integration workflows within the Azure Synapse Analytics environment. These pipelines streamline the movement and transformation of data across various data sources and destinations, facilitating data processing, analytics, and reporting tasks.

3.4.2 Storage Layer

As mentioned above, in this logical layer, the storage services planned to be used in the proposed architecture are configured.

Azure Data Lake Storage is a scalable and secure cloud-based storage solution, designed to handle massive amounts of data in various formats, including structured, semi-structured, and unstructured data. The data lake structure in Figure 3.1 is implemented using this service.

Azure Cosmos DB is a globally distributed, multi-model database service, designed to support highly responsive and scalable applications with low-latency access to data.

3.4.3 Processing Layer

As mentioned above, in this logical layer, the services chosen to process the data are configured.

Azure Synapse Analytics SQL Pools, formerly known as SQL Data Warehouse, is a cloud-based data warehouse service designed to store and analyze large volumes of structured and semistructured data for business intelligence and analytics purposes.

Azure Databricks is a cloud-based analytics platform built on top of Apache Spark. It offers a unified analytics workspace for data engineers, data scientists, and machine learning practitioners to collaborate and work on big data and AI projects efficiently.

All data processing is executed through pipelines and/or Python scripts from Azure Databricks. The output of Azure Databricks can be stored in one of the data lake zones or in the Cosmos DB database.

3.4.4 Presentation Layer

As mentioned above, in this logical layer, services and applications are planned to consume data available in the refined and test data zones of Data Lake, as well as in Cosmos DB, and display the results of the analysis through dashboards, reports, or data science projects.

Azure App Service is a fully managed platform-as-a-service (PaaS), designed to build, deploy, and scale web applications and APIs quickly and easily.

Power BI is a business analytics application that enables users to visualize and analyze data quickly and easily.

3.4.5 Cloud Monitoring, Data Management and Governance

In this logical layer are the services available in Azure for operational monitoring of cloud-enabled services, as well as services for managing and governing data in this environment.

Azure Purview is a unified data governance service designed to help discover, govern, and analyze data assets across on-premises, multicloud, and Software as a Service (SaaS) environments. Here are some key features and characteristics of Azure Purview:

- Unified Data Governance: Azure Purview provides a centralized platform for managing data governance activities, including data discovery, classification, lineage, and policy enforcement. enabling organizations to gain visibility into their data assets and ensure compliance with regulatory requirements.
- Data Discovery and Cataloging: Azure Purview automatically discovers and catalogs data assets across various data sources, including databases, data lakes, file shares, and SaaS applications. Create a comprehensive inventory of data assets and metadata, making it easy for users to search, explore, and understand their data landscape.

- Data Classification and Sensitivity Labeling: Azure Purview allows organizations to classify and label their data based on sensitivity, regulatory requirements, or business policies. It supports automated classification using machine learning algorithms and integration with Microsoft Information Protection (MIP) to apply sensitivity labels.
- Data Lineage and Impact Analysis: Azure Purview provides visibility into the lineage of data, showing how data flows through systems and processes from source to destination. It enables users to track the lineage of the data, understand the dependencies of the data, and perform an impact analysis to assess the potential impact of changes on downstream systems and processes.
- Data Governance Policies: Azure Purview enables organizations to define and enforce data governance policies to ensure data compliance, security, and privacy. Supports policy definition for data classification, access control, data retention, data masking, and data lineage tracking.
- Integration with Azure Services: Azure Purview integrates seamlessly with other Azure services, including Azure Data Lake Storage, Azure SQL Database, Azure Synapse Analytics, Azure Blob Storage, and Azure Key Vault. Using Azure's built-in security, compliance, and management capabilities, it provides a secure and scalable data governance solution.
- Collaboration and insights Azure Purview facilitates collaboration among data stewards, data owners, and data consumers through a centralized data catalog and collaboration features. Provides insights into data usage, access patterns, and data quality metrics to drive data-driven decision making and improve data governance practices.
- Scalability and performance: Azure Purview is a fully managed service that automatically scales to handle large volumes of data and metadata. It offers high availability, reliability, and performance to support mission-critical data governance and analytics workloads.

Azure Advisor is a custom cloud consultant. It offers best practices, recommendations, and actionable insights to help optimize Azure resources, improve security, enhance performance, and reduce costs.

Azure Monitor is a comprehensive monitoring and management service that monitors the performance, availability, and health of applications, infrastructure, and resources running in Azure and on-premise environments.

3.5 Conclusion

The proposed architecture addresses the requirements defined by the company within the scope of the ENSIGHTS project. It is possible to implement this architecture on the three main cloud computing providers: Microsoft, Amazon, and Google.

The architecture consists of a set of technology components with strong integration among them, as well as a storage framework in zones within the company's Data Lake. This architecture includes tools for monitoring cloud resources, as well as data management and governance.

The components of the architecture are responsible for ingesting data from the company's on-premises systems into the cloud environment, processing the data for each zone of the Data Lake, and managing the flow of data between these different zones.

The next chapter presents the results obtained from the deployment of the proposed architecture, aiming to address the business case studied within the scope of the ENSIGHTS project.

4
Results

4.1
Introduction

During the development of the R&D project ENSIGHTS, the data architecture presented in this document was implemented in an experimental environment distinct from the company's production environment. After implementation, data flows from the SAP and SAGE systems were created, along with pipelines for data ingestion, storage, processing, and presentation. This chapter presents the results of the experiment and the results of using the cloud monitoring and the data management and governance tools proposed in the reference architecture.

The chapter mentions two predictive indicators for ML, the chromatographic assay indicator (CAI) and the Electrical Failure Risk Indicator (EFRI), which use chromatographic and sensor data from power transformers, respectively. These models were developed and trained with SAP and SAGE data for this kind of equipment. CAI evaluation showed a significant improvement in predicting failures compared to classical methods used in the maintenance of this kind of equipment, while the EFRI tests showed that it can be useful in helping maintenance teams identify potential defects that can lead to equipment failure. However, the development of these two indicators is outside the scope of this work.

4.2
Azure Data Lake Storage

4.2.1
Key Features

Azure Data Lake Storage (ADLS) is a highly scalable and secure cloud-based storage service provided by Microsoft Azure. It is designed to handle large volumes of data in various formats, including structured, semi-structured, and unstructured data. ADLS is optimized for big data analytics and Data Lake scenarios, offering features such as high throughput, low latency, and built-in integration with Azure services like Azure Data Lake Analytics, Azure Databricks, and Azure HDInsight.

Key features of ADLS (ADLS, (accessed in April 2024)) include:

- Scalability: ADLS can handle massive amounts of data, scaling up or down as needed to accommodate changing data storage requirements.
- Performance: provides high throughput and low latency for data access, enabling fast data ingestion, processing, and analytics.
- Security: ADLS offers robust security features to protect data at rest and in transit, including encryption, access control, and role-based security policies.
- Integration: It seamlessly integrates with other Azure services such as Azure Data Factory, Azure Synapse Analytics, and Azure Machine Learning, enabling organizations to build end-to-end data pipelines and analytics solutions.
- Analytics: Azure Data Lake Storage supports a wide range of analytics and data processing tools, including Apache Spark, Hadoop, and machine learning frameworks.

The Data Lake structure was implemented following the structure proposed in Figure 3.1, as shown in Figure 4.1.

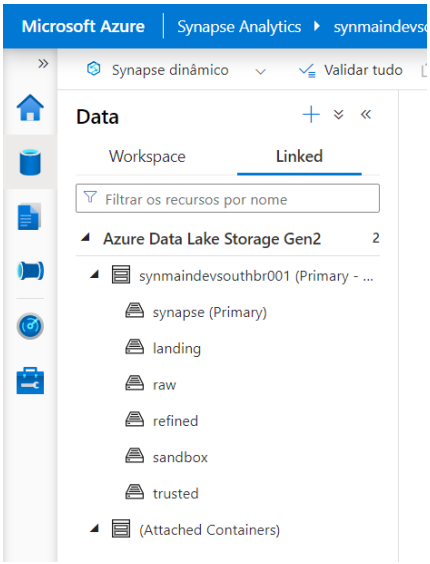


Figure 4.1: Data Lake structure implemented in Azure

4.2.2 Delta Lake

Data are stored on Data Lake in Delta Lake format. Delta Lake is an open source storage layer that provides reliability to data lakes. Databricks developed it to address some of the challenges faced by organizations when dealing with big data in cloud storage environments. Delta Lake builds on top

of Apache Spark and expands its capabilities to provide ACID transactions, schema enforcement, and time travel capabilities to data stored in cloud storage systems like Amazon S3, Microsoft Azure Blob Storage, and Google Cloud Storage.

One of the key features of Delta Lake is its ACID (Atomicity, Consistency, Isolation, Durability) transaction support, which ensures that data operations are atomic and consistent, even when multiple users are accessing the data concurrently. This helps prevent issues such as data corruption or inconsistent reads that can occur in distributed data processing environments.

Schema enforcement is another important feature of Delta Lake. It ensures that the data ingested into the lake comply with a predefined schema, thus improving data quality and making downstream data processing more reliable. Delta Lake also supports schema evolution, allowing for the evolution of schemas over time without breaking existing pipelines or requiring manual intervention.

Time travel is perhaps one of the most powerful features of Delta Lake. It allows users to query data at different points in time, providing a historical view of the data, and enabling easy rollback to previous versions if necessary. This feature is particularly useful for debugging data quality issues, auditing changes, and recovering from errors.

Delta Lake has gained popularity among organizations looking to build reliable and scalable data lakes in the cloud. Its integration with Apache Spark makes it easy to use for data engineers and data scientists familiar with Spark, while its support for ACID transactions, schema enforcement, and time travel makes it a robust choice for building production-grade data pipelines. As organizations continue to deal with increasingly large volumes of data, Delta Lake offers a promising solution to manage and analyze data on a scale.

4.3 Data Extraction

4.3.1 Overview

In the proposed architecture, the two main data sources within the scope of the project stand out:

– SAP PM: This SAP module helps to manage equipment inspections, repairs, and preventive activities. The SAP PM (Plant Maintenance) module can also plan material and labor activities, record costs, and even control automatic repairs and maintenance requests. At the company,

SAP PM is mainly used by the maintenance sector to administer the company's technical database. More specifically, the following data, which were relevant to the objective of the project, are stored in the database of this module:

1. Technical information for equipment identification, such as installation location, classes, and their electrical characteristics.
2. Historical records of equipment management since its conception (installation and energization).
3. Information regarding maintenance notes for each equipment, as well as their respective executions (maintenance orders).

– SAGE: This system records data from sensors and operation terminals present in substation equipment, such as power transformers and reactors. These include data such as current, voltage, power, temperature, among others, collected in real time.

These systems were deployed in the company's on-premises infrastructure and operated in a production environment. Therefore, to obtain the project's relevant data, it was necessary to build integrations between each of the systems and the prototype's cloud environment.

For each system, a specific method of making its data available in files was developed, and these files were saved in specific folders on a server in the company's on-premises environment. However, the integration process followed a common logic for both systems, where the cloud components would be responsible for accessing these files and ingesting them into the prototype environment in Azure.

When it comes to transfer data from on-premises to cloud, Microsoft recommends Integration Runtime (IR). IR is a component of Azure Synapse Analytics, a cloud-based data integration service that facilitates the movement and transformation of data between different data stores, both on-premises and in the cloud.

4.3.2 SAP PM data extraction

The data extraction of Orders and Maintenance Notes of Power Transformers and Reactors data from SAP PM was performed through an RPA that extracts the data from each of the project's substations of interest and makes it available in XLSX files in a folder on the company's on-premises server environment.

Using IR for integrating the on-premises environment with the cloud environment, the generated files can be imported into Azure using the following parameters:

- Host - path of the server's file folder;
- User Name - login of a user with access to the server;
- Password - password of the user with access to the server.

4.3.3 SAGE data extraction

Data extraction from SAGE is performed through an automated scripted routine that periodically accesses the historical database in PostgreSQL and transfers the necessary files to the server with the IR installed.

The historical base of SAGE is continuously updated with gathered data in real-time, and the process of updating these data in the Data Lake is done daily.

4.4 Azure Synapse Analytics Pipelines

4.4.1 Overview

Upon reaching the cloud through preconfigured processes in the IR, SAP and SAGE data are sent to the Data Lake Landing Zone and then processed by specific pipelines implemented in the Azure Synapse Analytics component. Once properly processed, the original replicas of the data in the Landing Zone are transferred to the Raw Data Zone for backup, while the preprocessed data are sent to the Trusted Data Zone. The landing zone and the raw data zone are represented by the Ingestion Zone (figure 3.1).

After reaching the Trusted Data Zone, the data can still undergo further processing to more promptly meet specific services, such as feed indicators and dashboards developed via Power BI. For this purpose, there is the Refined Data Zone, which acts as a Data Mart, providing processed data sources for specific purposes. Lastly, there is the Test Zone, whose objective is to act as a data sandbox for conducting tests, mainly aiming at the development of new ML models.

All data processing is executed through pipelines or Python scripts from Azure Databricks. Azure Databricks utilizes the Apache Spark engine, geared toward distributed processing of large amounts of data (Big Data). The output

of Azure Databricks can be stored in one of the Data Lake zones or in the CosmosDB database. CosmosDB is a universally distributed, highly available database that offers relational features as well as a wide range of NoSQL features, from document-based to graph-based modeling. Initially in this project, its goal is to serve as a database focused on serving a Web application hosted on the Azure App Services component; however, its versatility will provide future evolution in the architecture prototype and dashboard being developed throughout the project without significant impacts.

Regarding data flows in the Data Lake, they can theoretically be divided into three parts for each system. In practice, there is no clear identification of where each flow begins and ends. The flows are:

- Ingestion Flow: responsible for transporting data from the server to the Landing Zone of the Data Lake;
- Cleaning and Standardization Flow: responsible for processing the received file in the Landing Zone and treating this data to be stored in the Trusted Data Zone, and subsequently storing the original form in the Raw Data Zone;
- Refinement Flow: responsible for refining the treated files present in the Trusted Data Zone and storing them in the Refined Data Zone, where data visualization options will typically retrieve them.

The next sections provide a detailed description of what has been built involving each of these flows for each of the systems mentioned above. It is important to note that any validation failure in the pipelines will result in an email notification sent to a defined group of specific individuals so that necessary actions can be taken to correct the issue.

4.4.2 SAP Pipelines

SAP data flows between the data lake zones are implemented in four pipelines, as shown in Figure 4.2.

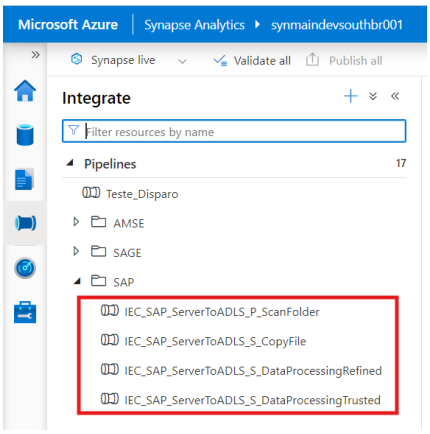


Figure 4.2: SAP pipelines in Synapse Analytics

Processing begins in pipeline "IEC_SAP_ServertoADLS_P_ScanFolder". This is the main pipeline, which calls the others in a logical sequence of activities.

4.4.2.1 SAP data to landing zone

The first three activities are responsible for the ingestion of data to the Data Lake landing zone. Processing starts by searching for files in the UNC path of the company's on-premises server, where, in the case of SAP, there should be XLSX files in this folder with the transaction name appended to them, such as *"2023-10-23_Transformers_IH08.xlsx"*. The rest of the name does not need to adhere to any other specific pattern.

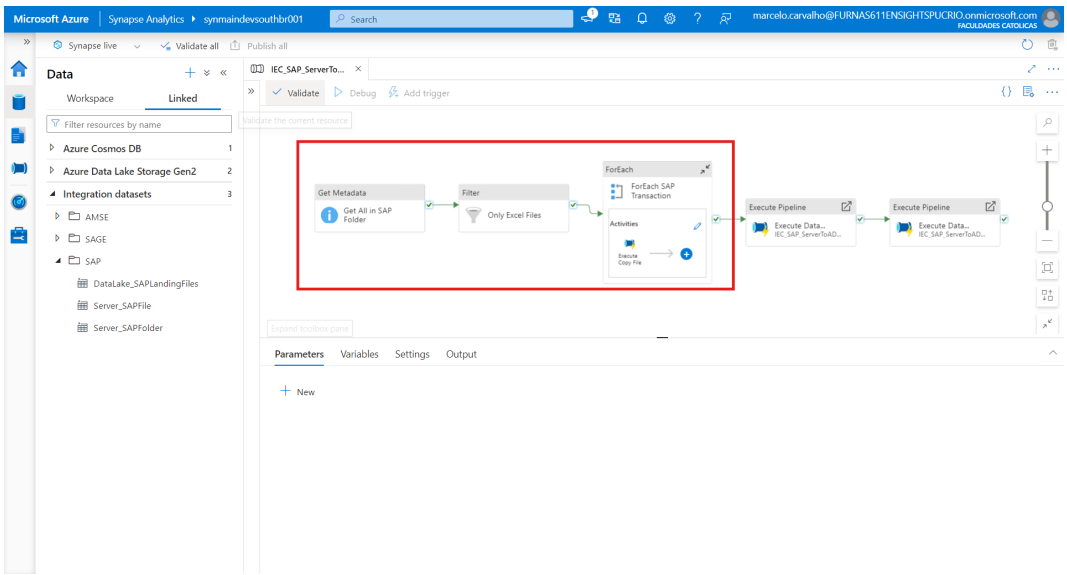


Figure 4.3: Activities related to ingest SAP data for landing zone

The access to the Data lake landing zone and the SAP folder and file

data is implemented as integration datasets, as shown in Figures 4.4, 4.5 and 4.6. Integration datasets are a feature in Synapse Analytics that allows you to create and manage datasets that can be used in various parts of the Synapse environment. These datasets serve as centralized definitions of data that can be accessed and utilized by different components within Synapse, such as SQL scripts, Spark jobs, and Power BI reports.

With integration datasets, you can define schema-on-read datasets, which means that the data schema is defined at the time of querying rather than being enforced at data ingestion. This provides flexibility to work with different types of data sources and formats.

Integration datasets help streamline data management and access within Azure Synapse Analytics by providing a unified way to define, organize, and consume data across different analytical workflows.

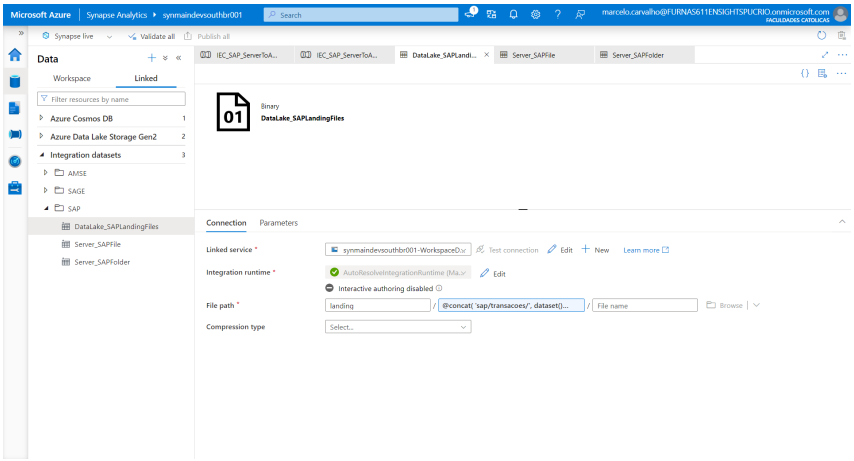


Figure 4.4: Integration Dataset for Data lake landing zone

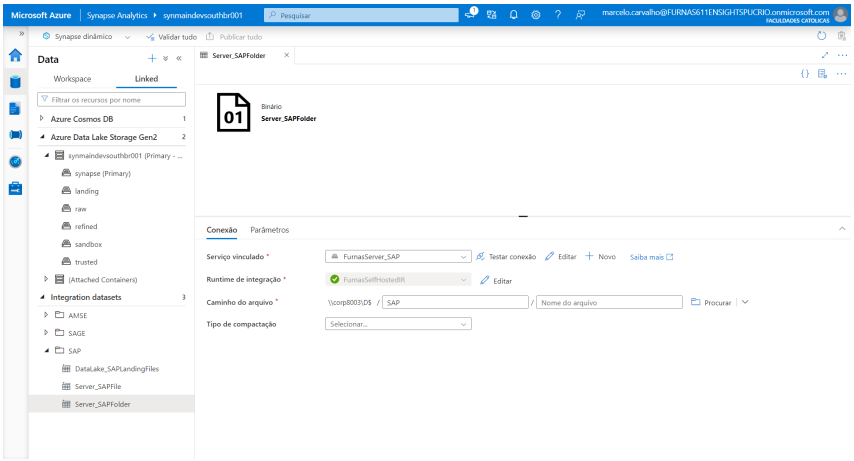


Figure 4.5: Integration Dataset for SAP Folder

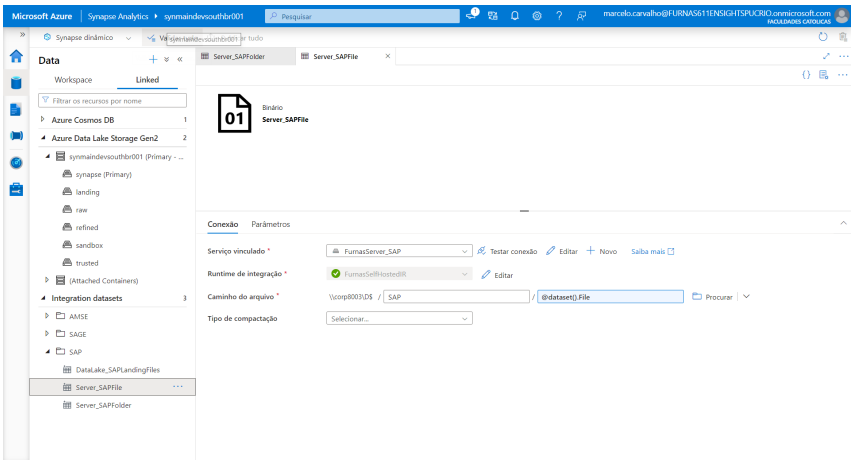


Figure 4.6: Integration Dataset for SAP File

The XLSX files in the folder will contain the transaction history for all SAP company substations that were used in the extraction. The IH08 transaction will return the master data for all the company's equipment, while other transactions will be limited only to transformers and reactors.

The second activity validates whether the files meet the requirements described above. After validation, the third activity implements a foreach loop and invokes pipeline "IEC_SAP_ServertoADLS_S_CopyFile" for copy each file to the Data Lake Landing Zone, thus completing the data ingestion flow of SAP transactions.

The landing data in Data Lake are saved in "landing/sap/transactions/-transaction/date: year-month-day", where the *transaction* represents the code of the processed transaction, while the *date* indicates when the pipeline persisted in the file. Files that were successfully copied to the cloud can be deleted by the pipeline itself. The components involved in this flow include only Azure Synapse Analytics and Data Lake.

4.4.2.2 SAP data to trusted zone

With the files in the Landing Zone, the fourth activity invokes pipeline IEC_SAP_ServerToADLS_S_DataProcessingTrusted.

The main activity of this pipeline is to invoke the script /Dev/preprocessing/SAP/transmissao-transacoes/save_transaction stored in a Azure Databricks notebook to process the data. In the case of SAP transactions, special characters present in column names were removed to avoid errors during queries on these data. After the script finishes processing, the data is saved in Delta format in the Trusted Data Zone of the Data Lake, while the original files that were in the Landing Zone are moved to the Raw

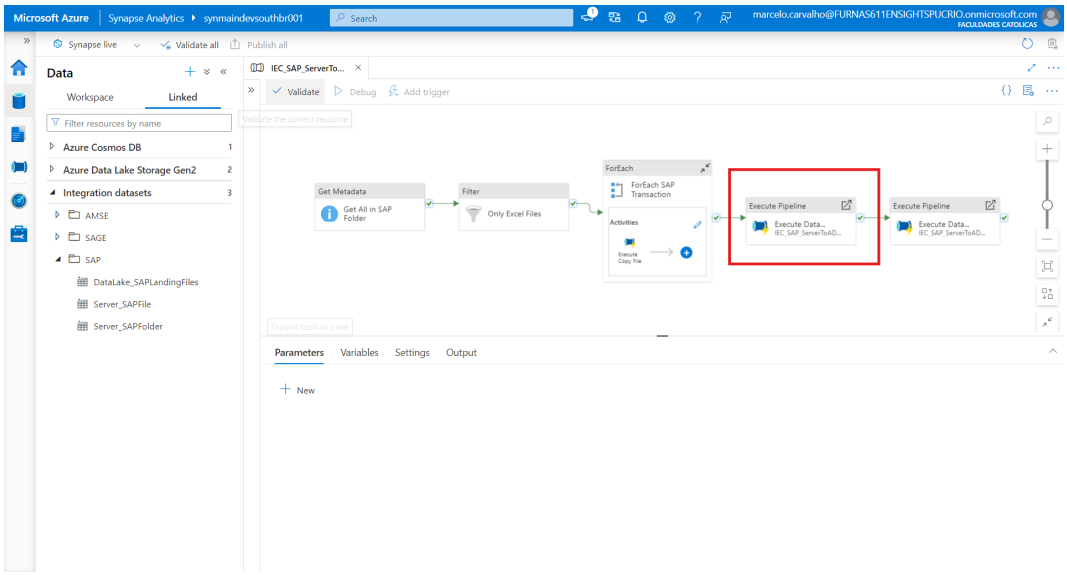


Figure 4.7: Activity related to copy SAP data for trusted zone

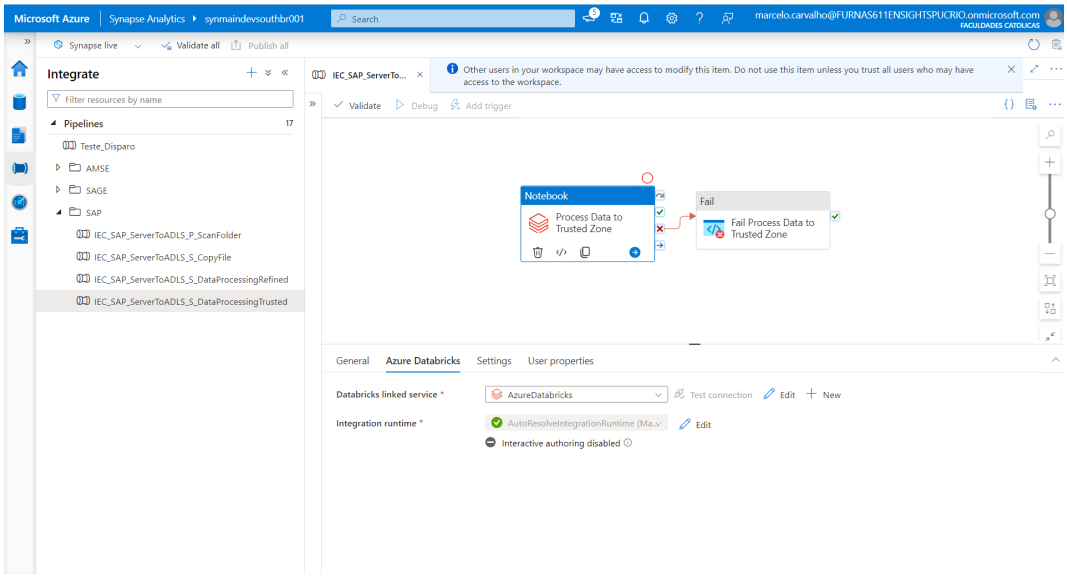


Figure 4.8: Pipeline to move SAP data from Landing Zone to Trusted zone

Data Zone, where they are available for auditing if necessary. This code is in Section A.1. Due to confidentiality reasons, SAP data have been hidden in the code presented.

Data in the trusted data zone are saved in “trusted/sap/transactions/transaction”, where the transaction represents the code of the transaction that is saved in this zone of the Data Lake. Note that the data in the Raw Data Zone are saved in the same pattern as in the Landing Zone, for example, “raw/sap/transactions/transaction/date: year-month-day”. At this point, the cleaning and standardization flow is concluded, where, in addition to the previously used components, Azure Databricks is included in the orchestration performed by the pipeline.

4.4.2.3 SAP Data to refined zone

After the files persist in the Trusted Data Zone without errors, the pipeline continues its flow, with the fifth activity invoking the pipeline IEC_SAP_ServerToADLS_S_DataProcessingRefined.

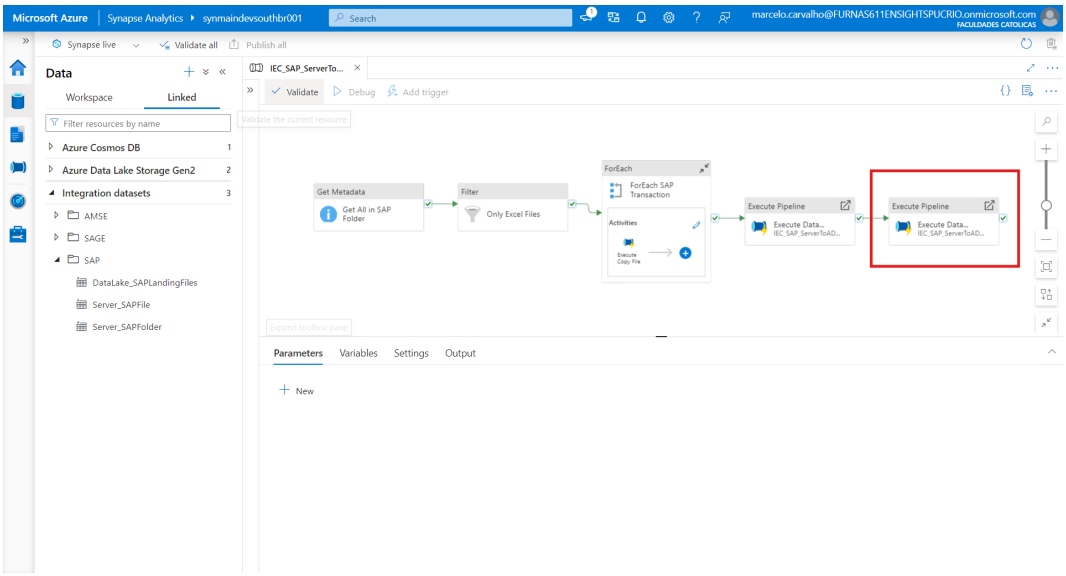


Figure 4.9: Activity related to copy SAP data for Refined zone

The main activity of this pipeline is to invoke the script */Dev/models/iec/get_all_dga_results* stored in a Azure Databricks notebook that will refine the data to feed, mainly, the dashboard developed for the project. This code is in Section A.2. Due to confidentiality reasons, SAP data have been hidden in the code presented.

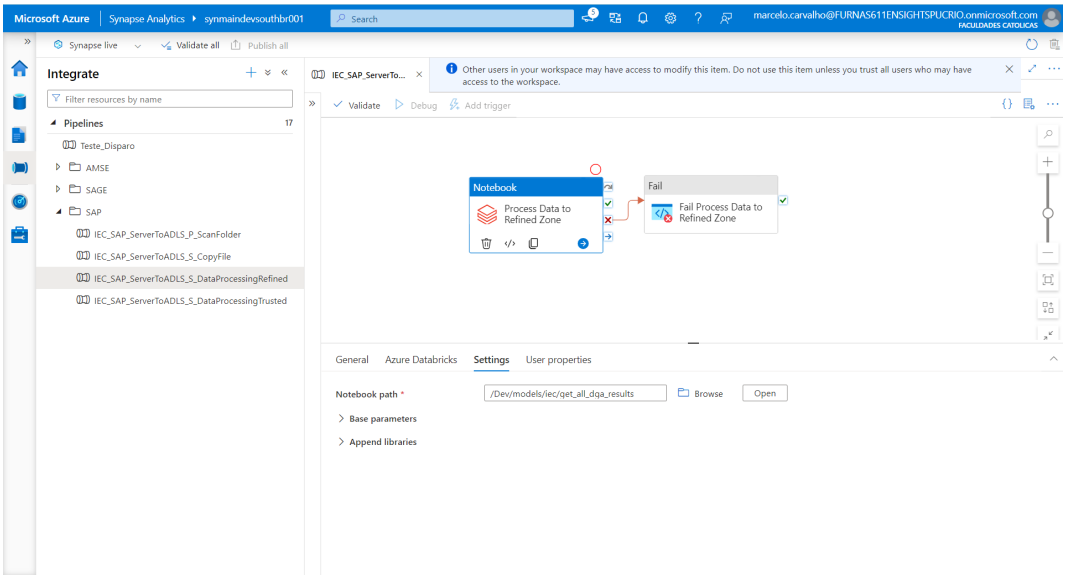


Figure 4.10: Pipeline to copy SAP data from Landing Zone to Refined zone

At this stage, four datasets are created and persisted in the Refined Data Zone of the Data Lake:

- dga: dataset created from chromatographic data originating from the SAP IK17 transaction. Feeds the Chromatographic Indicators area (Chromatographic Test Indicator, other chromatographic diagnostics, and Duval Triangle) of the dashboard's indicators tab.
- idade-equipamento: dataset created from data from the SAP IH08 transaction. Provides equipment age information.
- lista-equipamento: data set created from data from the IH08 transaction with results of analyses using the IK17 transaction. Feeds the equipment list table and the alert counter in the indicator tab.
- oleos-processados: data set created from data from the IK17 transaction, feeding the Chromatographic Indicators area (Gas Concentration and Test History) of the dashboard's indicators tab.

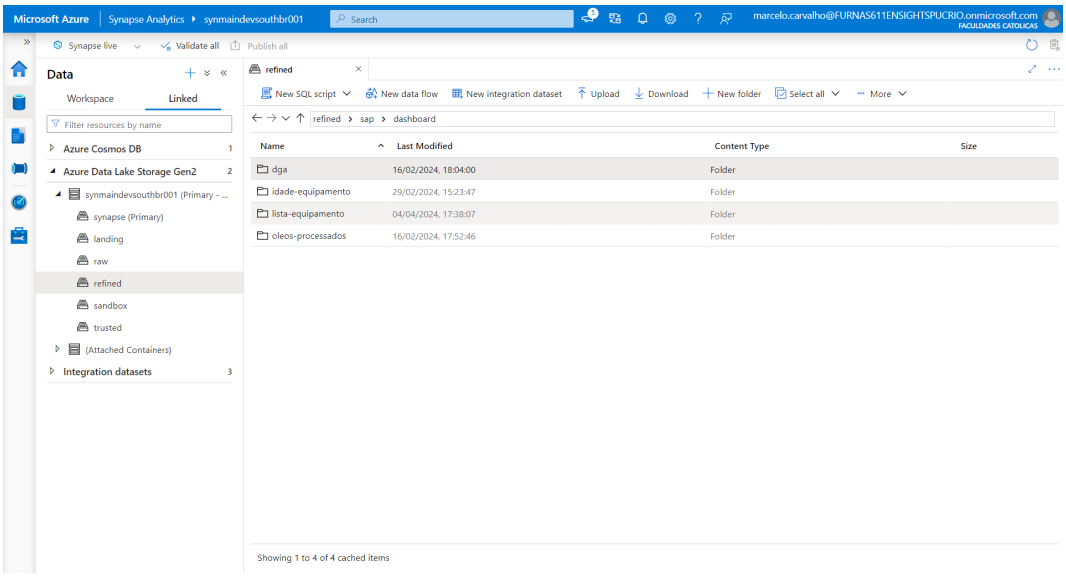


Figure 4.11: Data Lake Refined Zone for SAP Data

The scripts use the trained CAI (Chromatographic Assay Indicator) model saved in the Data Lake Refined Data Zone, specifically in refined/sap/-models/iec_model.sav. All the forementioned datasets are saved in refined/s-ap/dashboard, also in Delta format, just like the data in the Trusted Data Zone. The backend application, developed for the project, can directly access these data through the Serverless SQL Pools functionality of Azure Synapse Analytics. The source code related to the application backend is hosted on the Azure App Services component.

After all processing steps are executed, the refinement flow is completed, and the pipeline execution is finished.

It should be noted that, up to this point, pipeline processing activities have traversed the flow among all components listed in the Ingestion, Storage, Processing, and Presentation layers, illustrated in Figure 3.2, with the latter responsible for housing the Azure App Services component, which manages both the backend of the solution, which provides data to the frontend, responsible for visualizing these data through a dashboard.

4.4.3 SAGE Pipelines

SAGE data flows between the data lake zones are implemented in four pipelines, as shown in Figure 4.12.

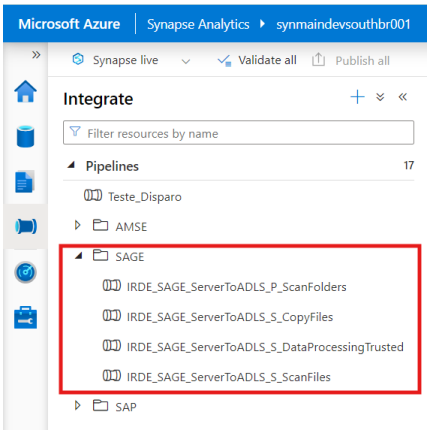


Figure 4.12: SAGE pipelines in Synapse Analytics

Processing begins in pipeline “IEC_SAGE_ServertoADLS_P_ScanFolders”. This is the main pipeline, and it calls the others in a logical sequence of activities.

4.4.3.1 SAGE data to landing zone

The first three activities are responsible for the ingestion of data to the landing zone of the data lake. Processing starts with data retrieval from the company's on-premises server. It is worth noting that, specifically for SAGE, the .pas, .alr, and .sde data files are brought to the server through a routine that accesses a network address on-premises daily, referring to the historical SAGE database. In this folder, there should be files in the formats pas, alr, and sde, separated by folders corresponding to the substations. All file extensions must follow the naming pattern mmmddyy.ext, where mmm = jan, feb, ..., dec, dd = 01, 02, ..., 31, yy = ... 14, 15, 16, ..., and ext = .pas, .alr, .sde. For example, the file "ago2419.pas" can be cited. Since the files in this case are provided automatically, the pipeline is configured for daily execution at 8 a.m. It is worth mentioning that files generated after midnight on the same day are filtered out.

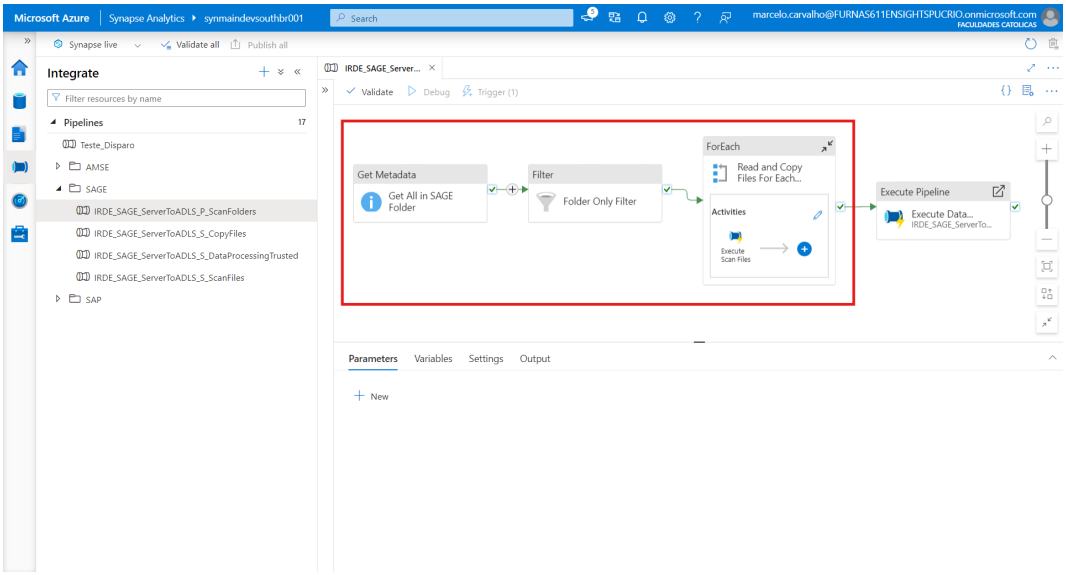


Figure 4.13: Activities related to ingesting SAGE data for landing zone

Similarly to the SAP pipelines, access to the data lake landing zone and the SAP folder and file data is implemented as integration datasets, as shown in Figure 4.14.

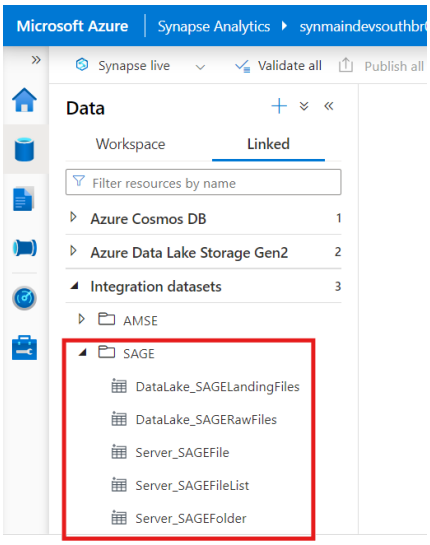


Figure 4.14: SAGE integration datasets

The second activity selects only the substations folders, and the next activity implements a foreach loop and invokes pipeline "IRDE_SAGE_ServertoADLS_S_ScanFiles". This pipeline verifies the existence of the files to be ingested and confirms their naming format, as shown in Figure 4.15.

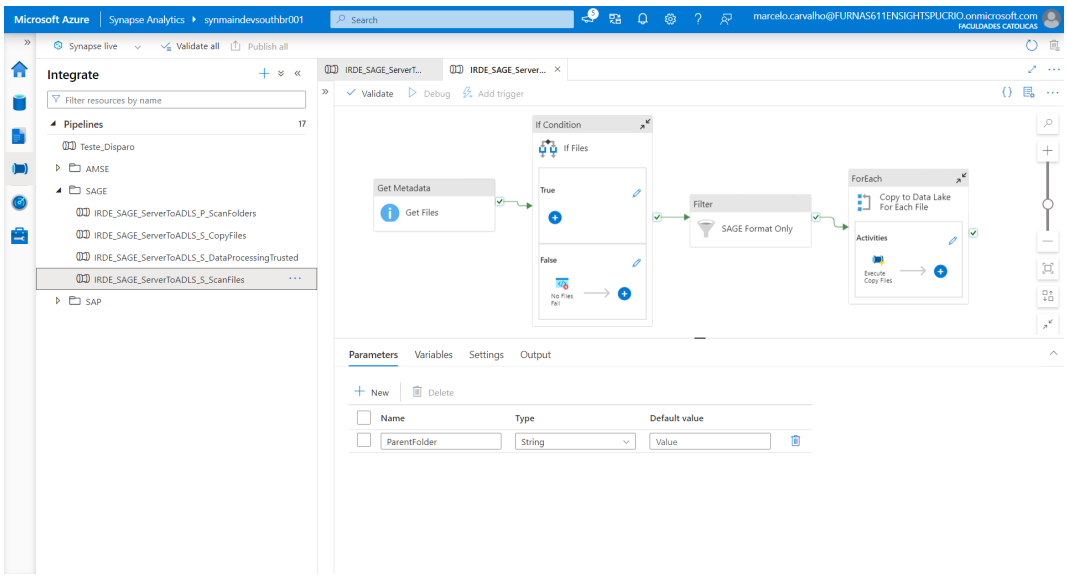


Figure 4.15: Pipeline to scan SAGE data files

After verification, the pipeline proceeds to the last activity, invoking pipeline "IRDE_SAGE_ServertoADLS_S_CopyFiles", which copies the files to the Data Lake Landing Zone, thus completing the data ingestion flow from SAGE to the Azure cloud architecture.

Data in the Data Lake is saved in the location *"landing/sage/substation/year/month"*, where the substation represents the code of the processed

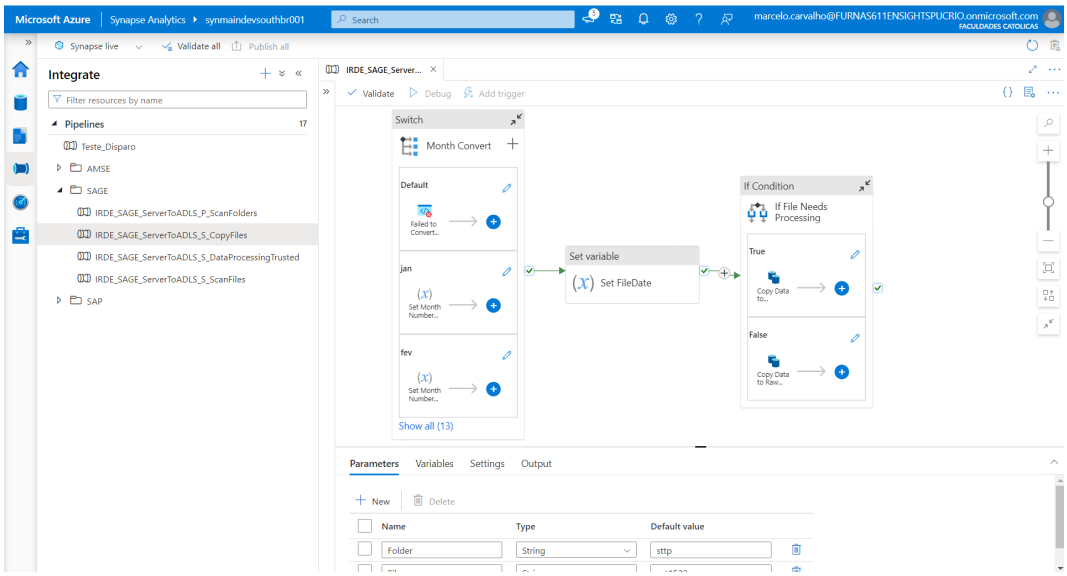


Figure 4.16: Pipeline to copy SAGE data files to the landing zone.

substation, while the year and month indicate the year and month when the pipeline persisted in the file.

4.4.3.2 SAGE data to trusted zone

With the files in the Landing Zone, the last activity of the pipeline in Figure 4.13 is invoking pipeline "IEC_SAP_ServerToADLS_S_DataProcessingTrusted", as shown in Figure 4.17.

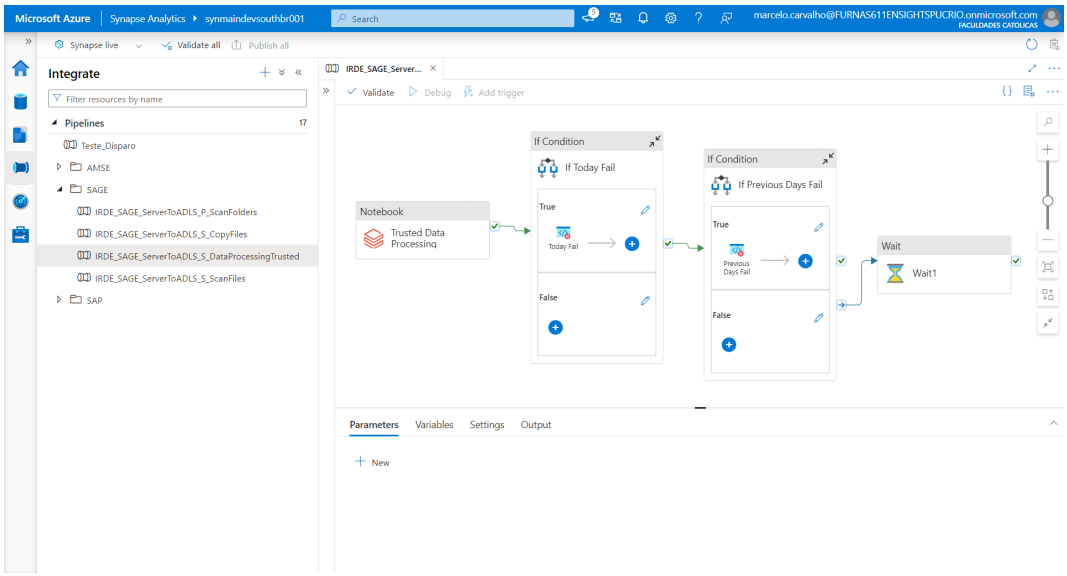


Figure 4.17: Pipeline to move SAGE data from Landing Zone to Trusted zone

The main activity of this pipeline is to invoke the script */Dev/preprocessing/SAGE/mover_trusted/nb_final* stored in an Azure Databricks notebook

to process the data, except for the sde files, which are not used as data sources in the project. For this reason, .sde files are transferred directly to the Raw Data Zone. On the other hand, files with pas and alr extensions, as they are semi-structured files, need to undergo specific processing before being saved in the Trusted Data Zone, in a tabular format. This code is in Section A.3. Due to confidentiality reasons, SAGE data have been hidden in the code presented.

After the script finishes processing, the data is saved in Delta format in the Trusted Data Zone of the Data Lake, while the original files, which were present in the Landing Zone, are moved to the Raw Data Zone, where they are available for audit if necessary.

In the Trusted Data Zone, the data are saved in the location "trusted/sage/substation/file", where the substation represents the substation code and the file represents the.alr and.pas extensions. In the Raw Data Zone, the files are saved in "raw/sage/substation/year/month". The.pas files are partitioned by year, whereas the alr files do not have partitions because they are very small. Logs of problems arising from the processing of alr and pas files are saved in Delta format in "trusted/sage/error_processing_logs'. Files that showed any abnormality during pipeline execution are moved to the Raw Data Zone immediately after notifying the users of interest about the problem. These logs may eventually be part of a dashboard area under development.

At this point, the cleaning and standardization flow is concluded, where, in addition to the previously used components, the Azure Databricks component is included in the orchestration performed by the pipeline.

4.4.3.3 SAGE data to refined zone

Once the files are saved in the Trusted Data Zone without errors, the pipeline continues its execution to run another Azure Databricks script, which will refine the data to meet the needs of the dashboard developed for the project primarily. In the case of SAGE, the datasets that will make up this zone are still under discussion, but it is certain that, as in the case of the CAI indicator, there will be a folder saved with the EFRI (Electrical Failure Risk Indicator) model in "refined/sage/models". The results of the EFRI predictions will be saved in The "sap/refined/dashboard/equipment-list" table, despite being located in the SAP area, it displays the equipment list present in the dashboard, showing the results of both indicators from the indicators tab.

After all processing steps are executed, the refinement flow is completed, and the SAGE pipeline execution is finished. It should be noted that, up to this point, the pipeline processing steps have traversed the flow between

all components listed in the Ingestion, Storage, Processing, and Presentation layers, illustrated in Figure 3.2. The latter is responsible for housing the Azure App Services component, which manages both the backend of the solution responsible for data provisioning and the frontend responsible for data visualization through a dashboard.

4.5 Azure Databricks

4.5.1 SAP and SAGE Correlation

As mentioned earlier, during the execution of Azure Synapse Analytics pipelines for SAP and SAGE data, Azure Databricks notebooks are activated in some of the data processing steps. Although steps such as data cleansing and transformation can be performed in Synapse Analytics, Azure Databricks offers greater flexibility and lower processing costs for the same operations.

To build the data set used in the EFRI model, one of the most important operations that involved SAP and SAGE data was the correlation between these two data sources for a given asset. SAP and SAGE identifiers differ in type and format, which prevents a direct relationship between the equipment records contained in both systems, as shown in Figure 4.18. In the case of power transformers, SAP uses an 8-digit format identifier to distinguish equipment, while in SAGE, this identification is made by combining the transformer's location identifier with its phase. From this unified identifier it is possible to access the analog measurements of that equipment.

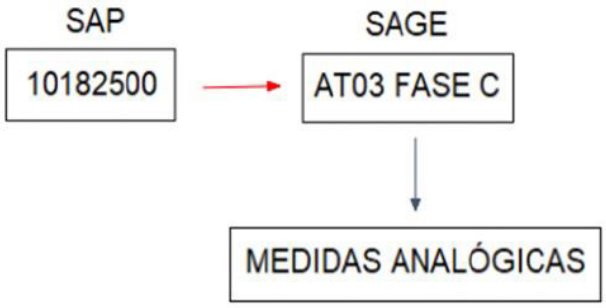


Figure 4.18: SAP and SAGE correlation.

To solve this problem and thus fill the measurement columns with the respective measurement values of each equipment, it was necessary to proceed in three steps:

1. Extraction of transformer location data and phase from the columns *Denominação 1, Denominação 2 and Descrição*, present in the file *Notas de Manutenção - Trafos.xlsx*, originating from SAP PM.

2. Definition of a mapping structure between fault categories and measurements. For this purpose, a hierarchical structure similar to that represented by Table 4.1 was created.

3. Execution of a query using regular expressions in the file containing the analog moving averages calculated from one of the file-restructuring procedures in the PAS format. The SAGE measurement variables are typically formed by (1) the substation acronym (e.g., STSB or SB), (2) the transformer, (3) the measurement, represented by one of the numerous acronyms in quotations present in the Measures column of Table 4.1, and (4) the phase. To find the variables corresponding to the chosen fault categories for each of the seven correspondences, regular expressions were used.

Table 4.1: Hierarchical structure for mapping between fault categories, defined by the categorization algorithm, and SAGE measurements.

Failure category	Measurement type	Measuments
Cooling	Current	“AT_I”, “BT_I”, “TE_I”
	Temperature	“AT_TMP”, “BT_TMP”, “TE_TMP”
Termometer	Temperature	“AT_TMP”, “BT_TMP”, “TE_TMP”
Electric	Current	“AT_I”, “BT_I”, “TE_I”
	Frequency	“AT_HZ”, “BT_HZ”, “AT_DIF_HZ”, “BT_DI_HZ”
	Power	“AT_MR”, “BT_MR”, “AT_MVA”, “BT_MVA”, “AT_MVAP”, “BT_MVAP”, “AT_MVAR”, “BT_MVAR”, “AT_MW”, “BT_MW”, “AT_PF”, “BT_PF”
	Voltage	“AT_KV_ANT”, “AT_KV_POS”, “BT_KV_ANT”, “BT_KV_POS”, “AT_KV_AVG”, “BT_KV_AVG”

4.5.2 Databricks Security

4.5.2.1 Overview

As part of this research, an architectural pattern has been developed to enable secure access to data stored in Azure Data Lake Storage (ADLS) by notebooks developed in Azure Databricks. This pattern is aligned with the proposed data architecture, as it integrates databricks with the Azure key vault, a service that provides a secure store and management of passwords used to access resources from Azure.

The implementation of this architectural pattern uses a service principal as a mediator of access, allowing data access using a *mount point* syntax, without exposing sensitive information from the Azure environment in the notebooks, which can be accessed by multiple developers in a team. The solution is intended for data engineers and data scientists who use the Azure Databricks platform for their data science projects and need to access data from an ADLS.

It should be noted that due to the rapid development of Azure Databricks, it is estimated that other solutions, such as Unity Catalog, may replace the architecture presented in this project in the coming years, as an alternative for secure data access control.

4.5.2.2 Implementation

There are some architectural patterns to enable access to ADLS from Databricks, as shown in Figure 4.19.

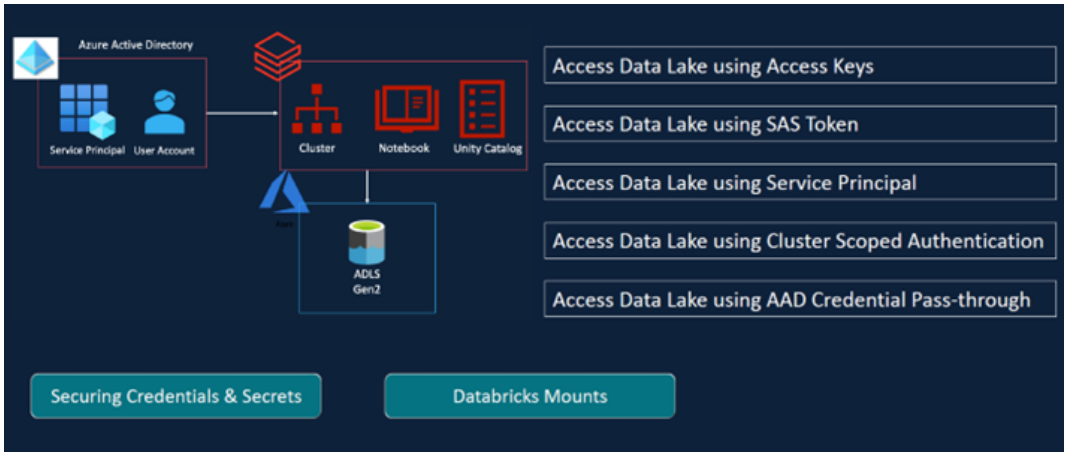


Figure 4.19: Architectural patterns to access ADLS from Databricks.

Of all available patterns, the one that ensures the highest security in access is the use of the Service Principal. A Service Principal is a resource very similar to a user account in Azure. It is registered in the Azure Active Directory and is associated with access permissions to resources within an Azure subscription through RBAC (Role Based Access Control). It is the recommended method to be used in Azure Databricks jobs and processing pipelines, as it provides greater security and traceability. In a good architecture, each application should have its own Service Principal, and each Service Principal should only have the necessary permissions for the application it is linked to.

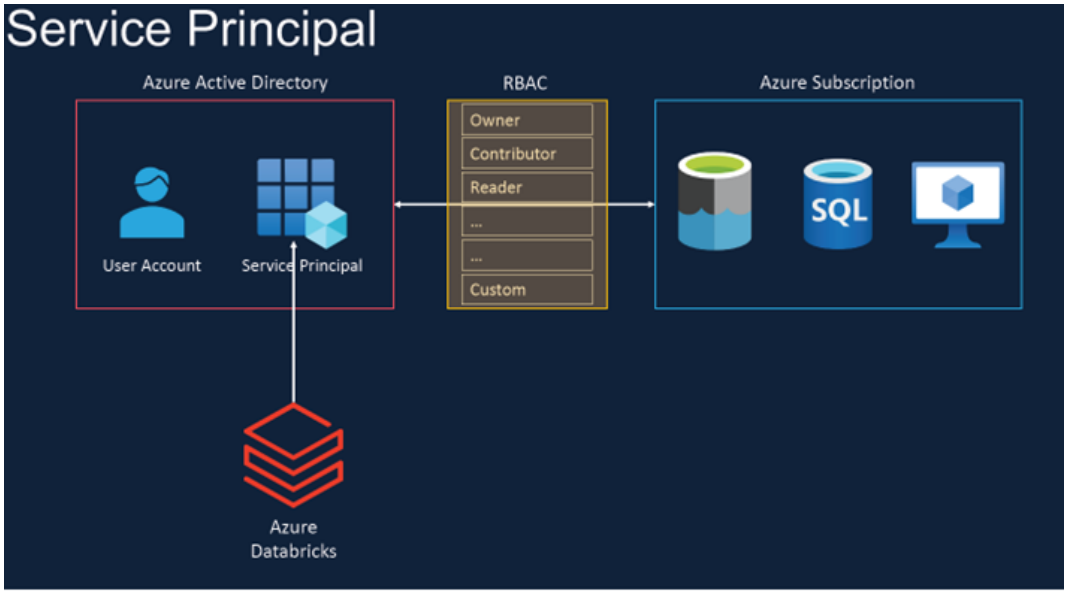


Figure 4.20: Architectural pattern to access ADLS using Service Principal.

To implement a Service Principal in the proposed architecture, the following steps are necessary:

1. Register the service principal (or application) in Azure Active Directory.
2. Generate a secret for the application.
3. Configure Spark with the Client ID, Tenant ID, and the secret.
4. Associate the "Storage Blob Data Contributor or Reader" role with the Data Lake, so that the Service Principal can access the data lake blobs.

4.5.2.3 Databricks File System and Databricks Mounts

The Databricks File System is an abstraction layer that simplifies access to object storage, providing a file-system-like experience for Databricks users and applications. It plays a key role in the integration of Azure Databricks with the available storage frameworks in Azure, such as Data Lake Storage. Its main characteristics include:

- Through DBFS, users and applications can interact with stored data as if they were working with a local file system or HDFS. This simplifies reading and writing data to object storage.
- DBFS stores data in Azure Blob Storage, which offers durability and cost-effectiveness. DBFS makes this interaction transparent to the user.
- You can "mount" Azure Blob Storage containers or directories onto specific points in DBFS. This allows you to access your data in Azure Blob Storage through paths in DBFS.
- Since DBFS is built on Azure Blob Storage, it inherits its durability and availability characteristics. Additionally, DBFS ensures read-after-write consistency, which is essential for data analysis workloads.
- DBFS is optimized to support streaming and large data analysis workloads. It can handle large volumes of data and supports parallel read and write operations.
- DBFS works well with tools and frameworks from the Hadoop/Spark ecosystem. The data in DBFS can be read directly by Apache Spark applications, facilitating processing and analysis.

Although DBFS can store data, it is not recommended for storing organizational data because if the workspace is deactivated, the data is lost. In this sense, Databricks allows mounting a storage account as a mount point in DBFS (Databrick mount). Access credentials are provided at the time of mounting. Once done, everyone with access to the workspace will have access to the data from the mounted storage account as shown in Figure 4.21. Access to the data is done through a file and folder semantics, different from long URLs. The main benefits of this approach are:

- Access to data without need for credentials.
- Access to files using a folder semantics instead of URLs (e.g., /mnt/storage1).
- Stores files in Blob Storage objects.

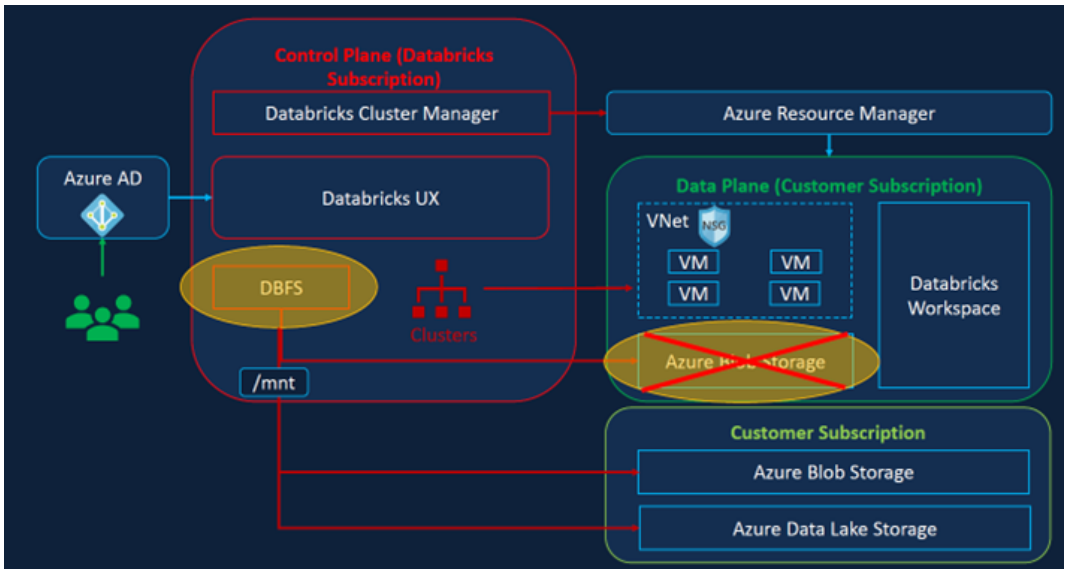


Figure 4.21: Databricks File System.

Note that the "Unity Catalog" feature has been adopted as the standard for data access management in the most recent projects. Unity Catalog is Databricks' data governance solution that offers the ability to manage the availability, usability, integrity, and security of data within an organization. Since this product was launched at the end of 2022, its adoption is not yet widespread, and the architectural patterns presented in Figure 4.19 are still the most commonly used in projects involving the use of ADLS and Azure Databricks.

With the use of Service Principal and DBFS, it is possible to implement data access in a simple and secure manner, as shown in Figure 4.22. Data stored in blob containers in Azure Data Lake Gen2 are made available through mount points in DBFS. The access control mediation is ensured by the Service Principal. Notebooks gain access to the data through mount points in DBFS, without needing to expose sensitive tenant data in plain text within the notebooks.

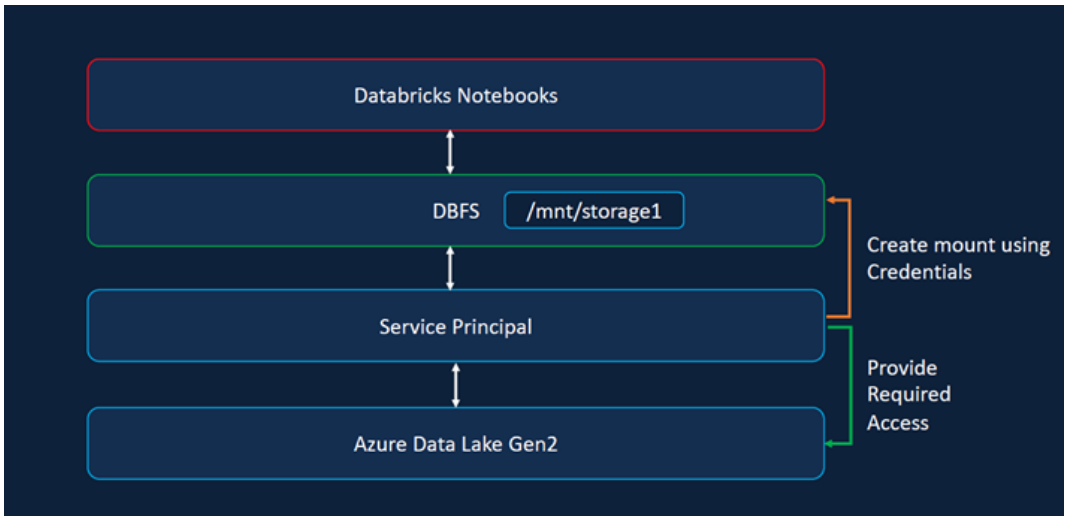


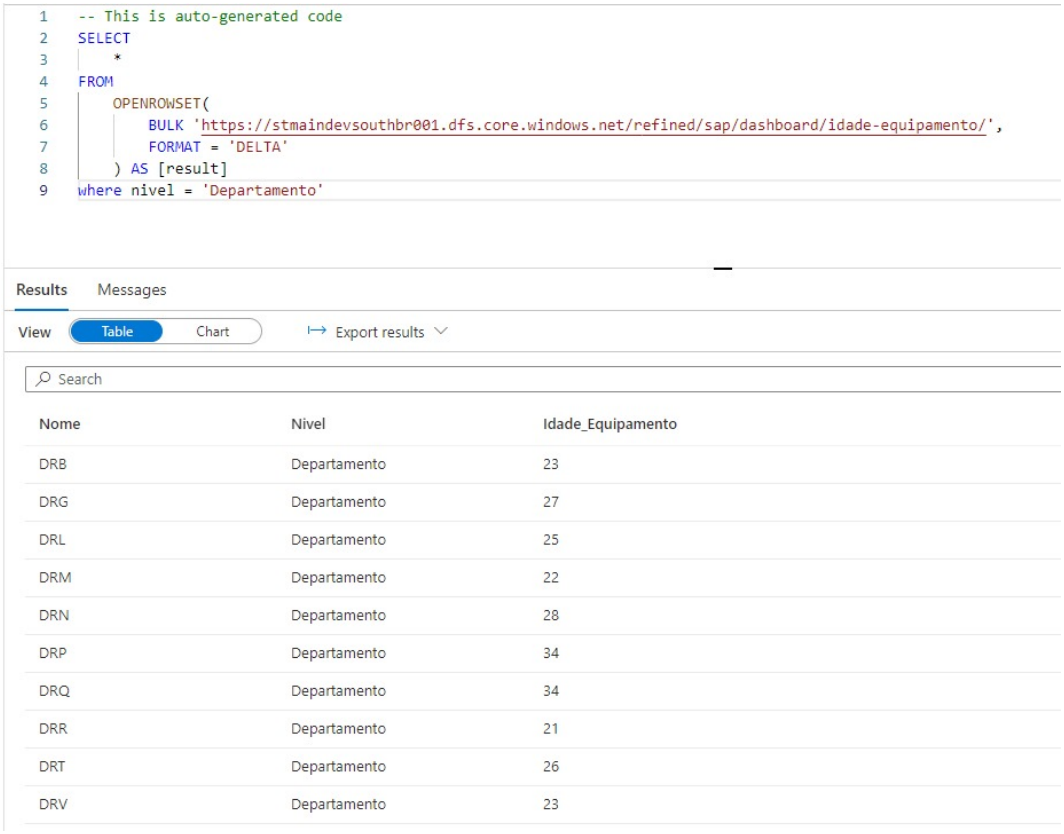
Figure 4.22: Databricks Secure Access Pattern.

4.6 Azure Synapse Analytics SQL Pools

As mentioned earlier, the backend application, developed for the project, can directly access these data through the Serverless SQL Pools functionality of Azure Synapse Analytics.

Figure 4.23 shows one of the queries built in the back-end code to list the ages of the equipments grouped by the company departments responsible for their maintenance. Data are from SAP transaction IH08.

The request to perform the query is made through an Synapse's endpoint.



```
-- This is auto-generated code
SELECT
    *
FROM
    OPENROWSET(
        BULK 'https://stmaindevsouthbr001.dfs.core.windows.net/refined/sap/dashboard/idade-equipamento/',
        FORMAT = 'DELTA'
    ) AS [result]
where nivel = 'Departamento'
```

Results Messages

View Table Chart Export results

Search

Nome	Nivel	Idade_Equipamento
DRB	Departamento	23
DRG	Departamento	27
DRL	Departamento	25
DRM	Departamento	22
DRN	Departamento	28
DRP	Departamento	34
DRQ	Departamento	34
DRR	Departamento	21
DRT	Departamento	26
DRV	Departamento	23

Figure 4.23: A SQL query running in a Synapse SQL Pool.

4.7 CosmosDB

CosmosDB is the main database of the proposed architecture, and it is used by applications in the presentation layer. It has a copy of the data in the refined zone, in a more suitable format than in the data lake zones, when it comes to use in reports, dashboards, or machine learning models.

CosmosDB can also be used to perform direct analysis, utilizing the built-in analytical capabilities of the service along with its supported APIs and query languages. You can choose one of the APIs based on the data model (e.g. SQL, MongoDB, Cassandra or Gremlin) and write analytical queries using the query language associated with the chosen API. The built-in functions and operators can be used in queries for various tasks such as filtering, aggregation, sorting, and data transformation.

Although data analysis operations can be performed in Synapse Analytics, carrying them out in Cosmos DB presents a significantly lower cost, which was a premise of the project and guided the development of the proposed architecture.

In the context of the project, the data are stored as collections and accessed via a MongoDB API, as shown in Figure 4.24.

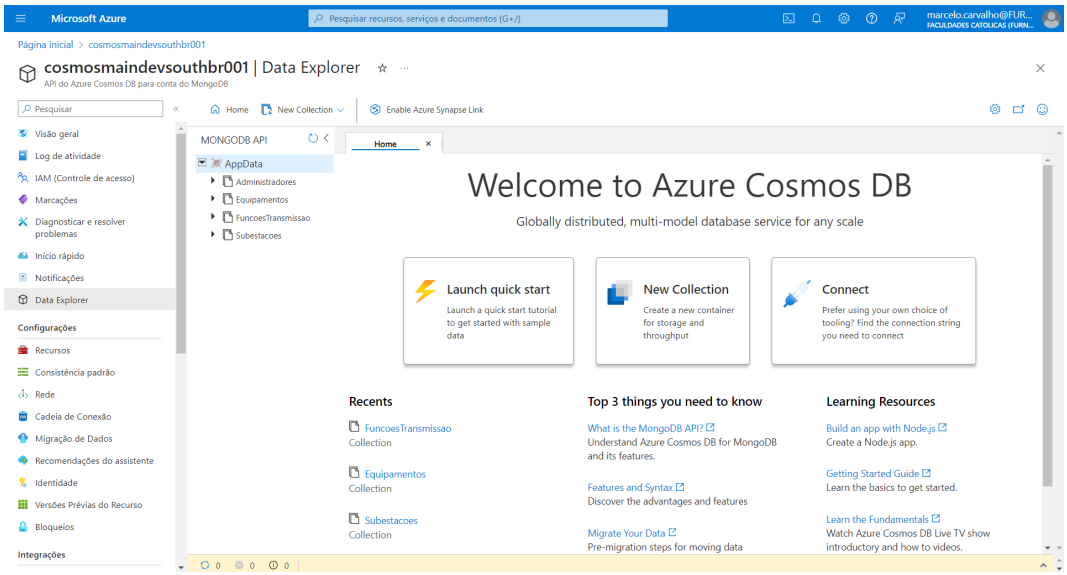


Figure 4.24: CosmosDB Collections.

MongoDB is a document-oriented NoSQL system. This means that a MongoDB record is a document which, in turn, is a data structure composed of field-value pairs. MongoDB documents are similar to JSON objects. The field values can include other documents, arrays, and arrays of documents (MONGODB, (accessed in September 2022)), as shown in Figure 4.25.

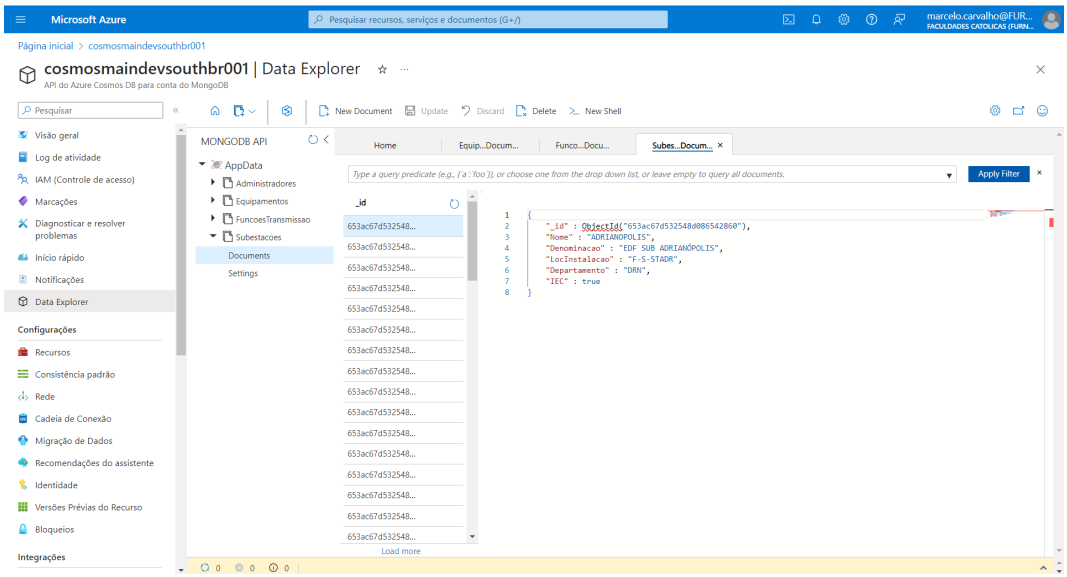


Figure 4.25: CosmosDB Collection in detail.

In the context of the ENSIGHTS project, CosmosDB stores some of the data (e.g. Equipments, Transmission Functions, Substations) to be presented in a custom dashboard designed to help maintenance team's decisions, as shown in Figure 4.26:

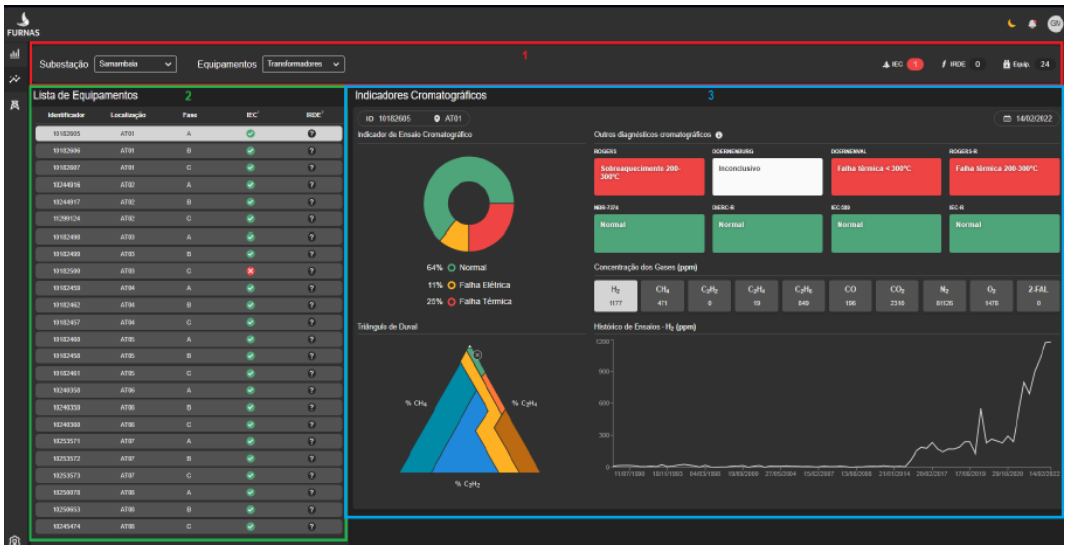


Figure 4.26: Custom dashboard showing CAI and EFRI indicators.

4.8 Azure App Services

This component manages both the backend and the frontend of the solution, responsible for publishing data in a custom dashboard, as shown in Figure 4.26.

4.8.1 The frontend

This is the layer responsible for the user interface, namely the graphical presentation of the dashboard. Through the front-end, requests are made to the back-end of the solution, which returns the requested data for the correct construction of its visualization.

For the development of the front-end, the React framework and the TypeScript language were chosen. These technologies are widely used in similar applications to be performant and have active communities that collaborate in the evolution and support of their respective tools.

4.8.2 The backend

The back-end is the layer responsible for providing information from the data sources used in the project to user requests via the front-end application, securely.

For the development of the back-end, the .Net Core framework was chosen, which, due to its robustness and scalability, proved to be a more attractive option for the solution's productization. Managed by Microsoft, it has exten-

sive documentation, strong community support, and high compatibility with the technologies offered by the Azure platform.

Access to data sources by the back-end can be done through two scenarios: (i) via Azure CosmosDB or (ii) via Azure Synapse Analytics, depending on the storage strategy. A hybrid approach that adopts both components may also be adopted according to the strategy defined after receiving and validating the data.

The first scenario was conceptualized at the beginning of the project and uses Azure CosmosDB as the data source. In this way, queries would be made aiming to provide data to the front-end through the language's own libraries, created to facilitate communication and use of Azure CosmosDB in solution development.

With the inclusion of Azure Synapse Analytics in the project architecture, there is the option to query Data Lake data directly through this component, without the need to transfer data to another base where such queries can be executed.

4.9 PowerBI

PowerBI is part of the company's strategy to implement a self-service culture with the empowerment of people outside IT, to build their own reports using this application.

Although nothing was developed with this tool in the context of the project, the proposed architecture provides all the necessary resources to the data citizens to explore, analyze, and present reports without IT support.

4.10 Azure Purview

Microsoft Purview is a comprehensive set of solutions that can help your organization govern, protect, and manage data wherever it lives. Microsoft Purview solutions provide integrated coverage and help address data fragmentation between organizations, the lack of visibility that hampers data protection and governance, and the blurring of traditional IT management roles (PURVIEW, (accessed in April 2024)).

The article "Microsoft Purview and Azure Synapse: Enabling End-to-End Data Governance and Quality Management" (CHAUDHARY, (accessed in April 2024)) explores the importance of data governance and quality management in a data-driven world. Highlights how Azure Purview and Azure Synapse can be used to achieve comprehensive data governance, including data

discovery, cataloging, classification, data lineage, and security. In addition, it describes how Azure Synapse can be used to perform automated data quality checks, integration, and team collaboration.

The article also presents the project phases for implementing data governance and quality management, including the identification of crucial attributes, classification of quality check components, and automation of the quality check process. It emphasizes the importance of automating quality checks and how it can enhance organizations' consistency, accuracy, and efficiency, as well as provide a simplified view of the process through Power BI reports.

This project can be a good starting point for establishing data governance practices considering the proposed architecture.

4.11 Conclusion

The implementation based on Microsoft Azure made it possible to validate the proposed architecture with the development of ML models based on SAP and SAGE data, which were collected, ingested, treated, and presented in a custom dashboard.

Synapse Analytics' pipelines were responsible for orchestrating the activities and invoking Databricks, which played a key role in bringing the raw data from the on-premise sources, processing, and saving the data in a structured data lake. Azure Databricks also performed all the data transformations required to correlate SAP and SAGE data, which was necessary to run the predictions of the CAI and EFRI machine learning models.

5
Conclusions and future work

The main contributions of this dissertation were:

- The definition of a data reference architecture for the electrical sector, consisting of a set of technology components with a strong integration among them, as well as a storage framework organized in zones within a data lake.
- The description of an operational prototype of a computational platform in the cloud, which would assist in optimizing the predictive maintenance process of power transformers using Machine Learning techniques.

The prototype was developed within the scope of the ENSIGHTS Research and Development project. Data storage, processing and presentation flows were implemented and delivered the expected results.

The implementation of the proposed architecture made possible the development and operation of the CAI and ERFI machine learning models, due to the data integration flows between the SAP and SAGE databases and the data lake. Prediction models need data to be updated on a regular basis to improve the model's accuracy.

During the development of the ERFI model, a limitation in the model's generalization capacity was observed, due to the different implementations of SAGE for each substation considered in the project context. In future work, it is suggested to investigate a possible alignment of schemes that allow for expanding the data set used in the training of this model.

Before the R&D project began, Microsoft Azure was already chosen by the company's infrastructure department as cloud technology to preferably adopt to develop new cloud-based solutions. The result of the evaluation, indicating Microsoft as the best option, facilitates the onboarding of the prototype into the company's cloud infrastructure.

Although the project did not implement data management and governance practices, it was possible to describe some scenarios using the proposed technologies to validate their use.

The article "on the use of machine learning for the predictive maintenance of power transformers" (PACHECO VAGNER PAES; MAROTTI, 2023) presented the results of the Data Science project developed within the scope of ENSIGHTS, using an implementation of the proposed architecture.

During the development of this dissertation, Microsoft released Fabric (FABRIC, (accessed in April 2024)), an Azure cloud all-in-one analytics

solution designed for enterprises. It encompasses a wide range of services, from data movement to Data Science, Real-Time Analytics, and Business Intelligence. Data Lake, Data Engineering, and Data Integration are combined in a single platform, eliminating the need to use multiple services from different vendors. Power BI, Azure Synapse, and Azure Data Factory can be used in a unified environment, providing a seamless user experience across different analytics tasks. The service also offers centralized administration and governance, where IT teams can centrally manage enterprise capabilities, and permissions are automatically applied across all services. It is a promising solution for future work considering the potential to simplify the proposed architecture.

As mentioned earlier, Unity Catalog is Databricks' data governance solution, released in late 2022. Considering the proposed architecture, it is also a promising solution, which can be explored in future work.

Finally, although the architecture proposal has focused on electricity companies, it can be adapted for other industry sectors, with a preliminary assessment of the requirements and typical data aspects of a company in the sector under study. This assessment can be conducted with the support of a framework, as proposed in (BRINCH, 2018). This preliminary study may indicate, for example, the need to incorporate technologies for processing streaming data flows into the proposed architecture. It is important to note that the proof-of-concept study described in Section 3.2, conducted with two other cloud computing providers, showed that the proposed architecture can be deployed on technologies other than Microsoft, further expanding its usage in the industry.

6
Bibliography

ADLS. **ADLS Features**. (accessed in April 2024). <https://azure.microsoft.com/en-us/solutions/data-lake/>. Cited in page 53.

AL-RUITHE, M.; BENKHELIFA, E.; HAMEED, K. A systematic literature review of data governance and cloud data governance. **Personal and Ubiquitous Computing**, Springer, v. 23, p. 839–859, 2019. Cited in page 35.

ANDERSON, C. **Free: grátis: o futuro dos preços**. [S.l.]: Elsevier Brasil, 2017. Cited in page 28.

BRINCH, M. Understanding the value of big data in supply chain management and its business processes: Towards a conceptual framework. **International Journal of Operations & Production Management**, Emerald Publishing Limited, v. 38, n. 7, p. 1589–1614, 2018. Cited in page 83.

CHAUDHARY. **Enabling End-to-End Data Governance and Quality Management**. (accessed in April 2024). <https://www.linkedin.com/pulse/microsoft-purview-azure-synapse-enabling-end-to-end-data-chaudhary/>. Cited in page 80.

COSMOSDB. **Cosmos DB Features**. (accessed in April 2024). <https://azure.microsoft.com/pt-br/products/cosmos-db#features>. Cited in page 45.

CRN. **AWS Revenues**. (accessed in April 2024). <https://www.crn.com/news/cloud/2024/amazon-ceo-aws-100b-run-rate-more-than-any-other-cloud-provider>. Cited in page 44.

DATABRICKS. **Azure Databricks**. (accessed in April 2024). <https://www.databricks.com/product/azure>. Cited in page 45.

DGI. **Data Governance Framemork**. 2024 (accessed in March 2024). <https://datagovernance.com/the-dgi-data-governance-framework/>. Cited 3 times in pages 9, 35, and 36.

DIGITAL-OCEAN. **Hyperscaler cloud**. 2024 (accessed in April 2024). <https://www.digitalocean.com/resources/article/hyperscaler-cloud>. Cited in page 41.

FABRIC. **Fabric Overview**. (accessed in April 2024). <https://learn.microsoft.com/en-us/fabric/get-started/microsoft-fabric-overview>. Cited in page 82.

GARTNER. **Magic Quadrant for Cloud Infrastructure and Platform Services**. 2021 (accessed in July 2022). <https://www.gartner.com/doc/reprints?id=1-1ZDZDMTF&ct=200703&st=sb>. Cited in page 44.

GULLBERG, A. Den samhälleliga självreflexionens möjligheter: Big data på 1980-talet. Nordiska institutet för samhällsplanering, 1991. Cited in page 27.

HARMONY, C. **Cloud Providers Outages**. (accessed in April 2021). <https://cloudharmony.com>. Cited in page 44.

LANEY, D. et al. 3d data management: Controlling data volume, velocity and variety. **META group research note**, Stanford, v. 6, n. 70, p. 1, 2001. Cited in page 27.

LARIVIERE, J. et al. Where predictive analytics is having the biggest impact. **Harvard business review**, v. 25, 2016. Cited in page 14.

MASHEY, J. R. Big data and the next wave of {InfraStress} problems, solutions, opportunities. In: **1999 USENIX annual technical conference (USENIX ATC 99)**. [S.l.: s.n.], 1999. Cited in page 27.

MAURO, A. D.; GRECO, M.; GRIMALDI, M. What is big data? a consensual definition and a review of key research topics. In: AMERICAN INSTITUTE OF PHYSICS. **AIP conference proceedings**. [S.l.], 2015. v. 1644, n. 1, p. 97–104. Cited in page 28.

MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National ..., 2011. Cited in page 29.

MELO, É.; NEVES, E. M. A.; PAZZINI, L. H. A. Brazilian electricity sector restructuring: From privatization to the new governance structure. In: IEEE. **2011 8th International Conference on the European Energy Market (EEM)**. [S.l.], 2011. p. 905–910. Cited in page 15.

MENDONÇA, A. F.; DAHL, C. The brazilian electrical system reform. **Energy Policy**, Elsevier, v. 27, n. 2, p. 73–83, 1999. Cited 2 times in pages 15 and 16.

MOBLEY, R. K. **An introduction to predictive maintenance**. [S.l.]: Elsevier, 2002. Cited in page 14.

MONGODB. **Introduction to MongoDB**. (accessed in September 2022). <https://www.mongodb.com/docs/manual/introduction/>. Cited in page 78.

NIST. **Big Data at NIST**. 2019. Https://www.nist.gov/itl/big-data-nist. Cited 3 times in pages 9, 27, and 28.

ONS. **Submódulo 18.2 Relação dos sistemas e modelos computacionais**. 2020. Https://www.ons.org.br/. Cited in page 24.

PÄÄKKÖNEN, P.; PAKKALA, D. Reference architecture and classification of technologies, products and services for big data systems. **Big data research**, Elsevier, v. 2, n. 4, p. 166–186, 2015. Cited 3 times in pages 9, 33, and 40.

PACHECO VAGNER PAES, M. d. C. F. L. G. M. A. G. E. N. J. S. E. H. C.; MAROTTI, A. On the use of machine learning for predictive maintenance of power transformers. In: SIMAS, E.; FERREIRA, D. D.; OLIVEIRA, L. R. (Ed.). **Anais do XVI Congresso Brasileiro de Inteligência Computacional (CBIC'2023)**. Salvador, BA: SBIC, 2023. p. 1–8. Cited 2 times in pages 20 and 82.

PEREIRA, L. A. et al. **SAGE-Um Sistema Aberto para a Evolução**. [S.l.]: CEPEL, 2014. Cited in page 23.

PURVIEW. **Learn about Microsoft Purview**. (accessed in April 2024). <https://learn.microsoft.com/en-us/purview/purview>. Cited in page 80.

RAFIQUE, K. et al. Cloud computing economics opportunities and challenges. In: IEEE. **2011 4th IEEE International Conference on Broadband Network and Multimedia Technology**. [S.l.], 2011. p. 401–406. Cited in page 30.

RAN, Y. et al. A survey of predictive maintenance: Systems, purposes and approaches. **arXiv preprint arXiv:1912.07383**, 2019. Cited 4 times in pages 9, 20, 21, and 22.

VISUAL-CAPITALIST. **Cloud Market Share**. (accessed in April 2024). <https://www.visualcapitalist.com/worlds-biggest-cloud-computing-service-providers/>. Cited in page 44.

WARD, J. S.; BARKER, A. Undefined by data: a survey of big data definitions. **arXiv preprint arXiv:1309.5821**, 2013. Cited in page 27.

YACOUB, S.; CUKIC, B.; AMMAR, H. H. A scenario-based reliability analysis approach for component-based software. **IEEE transactions on reliability**, v. 53, n. 4, p. 465–480, 2004. Cited in page 14.

YANG, C. et al. Big data and cloud computing: innovation opportunities and challenges. **International Journal of Digital Earth**, Taylor & Francis, v. 10, n. 1, p. 13–53, 2017. Cited 3 times in pages 10, 29, and 30.

ZBURIVSKY, D.; PARTNER, L. **Designing Cloud Data Platforms**. [S.l.]: Simon and Schuster, 2021. Cited 3 times in pages 9, 33, and 34.

ZHU, J. et al. A framework-based approach to utility big data analytics. **IEEE Transactions on Power Systems**, IEEE, v. 31, n. 3, p. 2455–2462, 2015. Cited 2 times in pages 22 and 24.

A
Code Listings

A.1
Save Transactions script

databricks save_transactions

(https://databricks.com)

Esse script lê os arquivos de excel da pasta landing e salva no trusted.

```
import pandas as pd
from datetime import datetime
import re
from pathlib import Path
from delta.tables import DeltaTable
from pyspark.sql.functions import lit, when, split, concat
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, TimestampType, FloatType
import numpy as np
import pyspark.pandas as ps
import unidecode
import re
import os
import shutil

from warnings import simplefilter
simplefilter(action='ignore', category=FutureWarning)


#CAMINHO_SAP_LANDING = '/dbfs/mnt/landing/sap/'
CAMINHO_SAP_LANDING = '/dbfs/mnt/landing/sap/transacoes'
# CAMINHO_SAP_REFINED_EQUIPMENTS = "/dbfs/mnt/refined/sap/dashboard_teste/lista-equipamento"
CAMINHO_DELTA_ERROS = '/dbfs/mnt/refined/_logs'


dbutils.widgets.text("subestacao", "stsb")
dbutils.widgets.text("transacao", "iw29")


subestacao = dbutils.widgets.get('subestacao')
transacao = dbutils.widgets.get('transacao')


def limpar_nome_colunas(df):
    df.columns = [(lambda s: s[:-1] if s[-1] == "_" else s)(string) for string in [re.sub(r'[^\w]+', '_', string) for string in
    return df


def tratamento_IK17(df):

    df["Data"] = df["Data"].astype(str)
    df["Ordem"] = df["Ordem"].astype(float)
    df["ValMed_PosTCont"] = df["ValMed_PosTCont"].astype(float)
    df["Equipamento"] = df["Equipamento"].astype(str)
    df['Item_medicao'] = df['Item_medicao'].astype(str)
    df['Unid_med_doc'] = df['Unid_med_doc'].astype(str)
    df["Valor_medido"] = df["Valor_medido"].astype(float)

    return df
```

```
# DICIONARIO_SCHEMA = {'Loc.instalação': str, #  'Denominação': str, #  'Valor m ...
```

Show cell

```
def tratamento_IH08(df):
    df['Equipamento'] = df['Equipamento'].astype(int)
    df['Ano_construcao'] = df['Ano_construcao'].astype(int)
    df['Dt_entr_servico'] = df['Dt_entr_servico'].astype('datetime64[ns]')
    df['Grp_plnj_PM'] = df['Grp_plnj_PM'].astype(int)
    df['Tipo_de_objeto'] = df['Tipo_de_objeto'].astype(str)

    return df
```

```
def determine_dtype(col, nome_transacao):
    if nome_transacao == 'IH08':

        if col in ["Ano_construcao"]:
            return FloatType()
        elif col in ['Dt_entr_servico',]:
            return TimestampType()
        else:
            return StringType()

    elif nome_transacao == 'IK17':

        if col in ['Ordem', 'Valor_medido']:
            return FloatType()
        else:
            return StringType()

    else:
        return StringType()

def mover_para_raw(arquivo):

    caminho_arquivo = '/dbfs' + arquivo

    print((caminho_arquivo).replace('landing', 'raw'))
    print(caminho_arquivo)


    raw_path = '/'.join(((caminho_arquivo).replace('landing', 'raw')).split('/')[:-1])
    print('raw_path', raw_path)

    if not os.path.exists(raw_path):
        os.makedirs(raw_path)

    if not os.path.exists((caminho_arquivo).replace('landing', 'raw')):
        shutil.move(caminho_arquivo, (caminho_arquivo).replace('landing', 'raw'))
        #if not os.listdir(Path(caminho_arquivo).parent):  os.rmdir(Path(caminho_arquivo).parent)


import os
import shutil

def delete_empty_subfolders(directory):
    # Walk through all subfolders in the given directory
    for root, dirs, files in os.walk(directory, topdown=False):
        # Check each subfolder from the deepest to the shallowest
        for name in dirs:
            dir_path = os.path.join(root, name)
            # Check if the subfolder is empty
            if not os.listdir(dir_path):
                # If empty, delete it
                shutil.rmtree(dir_path)
                print(f"Deleted empty folder: {dir_path}")
```

```
def tratamento_transacao(transacao):

    nome_transacao = transacao.split("/")[-1]

    CAMINHO_TRUSTED = f'/mnt/trusted/sap/transacoes/{nome_transacao}/'

    print('transacao', transacao)

    print(f'Processando transação {nome_transacao}')

    if len(os.listdir(transacao)) == 0:
        print(f'Transação {transacao} não contem nenhum arquivo no landing.')
        return

    arquivo = glob.glob(transacao + '/*')[0].replace('/dbfs', "")
    print(arquivo)

    try:

        dataframe_landing_pandaspyspark = ps.read_excel(arquivo, header = 0, dtype = str)#, dtype=DICIONARIO_SCHEMA)

        if nome_transacao == 'IK17': #essas funções (cast) são específicas para o Ik17
            dataframe_landing_pandaspyspark = tratamento_IK17(limpar_nome_colunas(dataframe_landing_pandaspyspark))

        elif nome_transacao == 'IH08':

            dataframe_landing_pandaspyspark = (limpar_nome_colunas(dataframe_landing_pandaspyspark))
            dataframe_landing_pandaspyspark = tratamento_IH08(dataframe_landing_pandaspyspark)

        else:
            dataframe_landing_pandaspyspark = (limpar_nome_colunas(dataframe_landing_pandaspyspark))

    except Exception as e:

        print(e)
        print()

        dt_string = datetime.now()
        #dt_string = now.strftime("%d/%m/%Y %H:%M:%S")

        dicionario_erro = {'Erro':[str(e.args[0])], 'Data_execucao_erro': [dt_string], 'Sistema': "SAP",
'Nome_arquivo_erro':[str(arquivo).replace('landing', 'raw').replace('/dbfs/mnt/','').replace('dbfs:/mnt/', '')]}

        df_erros = ps.DataFrame(dicionario_erro)

        df_erros.to_delta(CAMINHO_DELTA_ERROS, mode = 'append')

        mover_para_raw(arquivo)

        dbutils.notebook.exit(e.args)



    if (DeltaTable.isDeltaTable(spark, CAMINHO_TRUSTED)):
        print("Delta Table já criado !")
        dataframe_pyspark_trusted = ps.read_delta(CAMINHO_TRUSTED)

        #Melhor ordenar essa parte para garantir que está vindo na mesma order.
        #dataframe_pyspark_trusted.columns = dataframe_landing_pandaspyspark.columns

        dataframes_diff = ps.concat([dataframe_landing_pandaspyspark, dataframe_pyspark_trusted,
dataframe_pyspark_trusted]).drop_duplicates(keep=False)

        if len(dataframes_diff):
            schema_string_nullable = StructType([StructField(col, determine_dtype(col, nome_transacao), nullable=True) for
col in dataframe_landing_pandaspyspark.columns])
            df_pandas = dataframes_diff.to_pandas()
            sdf = spark.createDataFrame(df_pandas, schema=schema_string_nullable)
            sdf.write.format("delta").option("mergeSchema", "true").mode("append").save(CAMINHO_TRUSTED)
        else:
            print(f'Nenhuma linha nova.')
```

```
    else:
        schema_string_nullable = StructType([StructField(col, determine_dtype(col, nome_transacao), nullable=True) for col
in dataframe_landing_pandaspyspark.columns])
        print(type(dataframe_landing_pandaspyspark))
        display(dataframe_landing_pandaspyspark)
        df_pandas = dataframe_landing_pandaspyspark.to_pandas()
        # df_pandas['Ano_construcao'] = df_pandas['Ano_construcao'].astype('Int64')
        sdf = spark.createDataFrame(df_pandas, schema=schema_string_nullable)
        sdf.write.format("delta").option("mergeSchema", "true").mode("append").save(CAMINHO_TRUSTED)

    mover_para_raw(arquivo)
```

```
Deleted empty folder: /dbfs/mnt/landing/sap/transacao
```

A.2
Get All DGA Results script

databricks get_all_dga_results

(https://databricks.com)

```
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, TimestampType, FloatType, DateType
from pyspark.sql.functions import lit, when, regexp_replace
from itertools import groupby
import datetime
from delta.tables import DeltaTable
import numpy as np
from pathlib import Path
import pyspark.pandas as ps
import itertools
import pickle
import pyspark.pandas as ps
import os
import sys
import shutil
from random import randint
import pandas as pd

import pymongo
from dotenv import load_dotenv


%run ./dga_methods


%run ../../preprocessing/SAP/estruturacao_planilha_oleos/estruturacao_IK17_from_trusted
```

Show result

```
%run /Dev/reprocessamento-sap/create-equipment-list
```

Show result

```
CAMINHO_SAP_REFINED_DGA = '/mnt/refined/sap/dashboard/dga/'
CAMINHO_SAP_REFINED_OLEOS = '/mnt/refined/sap/dashboard/oleos-processados/'
CAMINHO_SAP_REFINED_EQUIPAMENTOS = '/mnt/refined/sap/dashboard/lista-equipamento/'
CONNECTION_STRING =
'mongodb://cosmosmaindevsouthbr001:n5UzI5Gc96SudZj7Vl3Zij4qogg7RQYohMGWQxfxCFUA1qHDk76DjTmDd4NTAhLZrJ2B5Cig5miQACDbwbrhkw==
@cosmosmaindevsouthbr001.mongo.cosmos.azure.com:10255/?
ssl=true&replicaSet=globaldb&retrywrites=false&maxIdleTimeMS=120000&appName=@cosmosmaindevsouthbr001@'
CAMINHO_IH08 = '/mnt/trusted/sap/transacoes/IH08/'
CAMINHO_IDADE_EQUIPAMENTOS = '/mnt/refined/sap/dashboard/idade-equipamento'
```

```
df_oleos_processados = ps.read_delta(CAMINHO_SAP_REFINED_OLEOS)
df_oleos_processados.head()
```

Show result

```
np.seterr(divide='ignore', invalid='ignore')
data = []
erros_lista = []
erro=0
sucess1 = 0
for item in df_oleos_processados.to_numpy():
    gases = item[4:] #Do 4 para frente porque não tem mais a coluna UnidadeMedida
    equip = item[1]
    sub = item[2]
    data_medicao = item[0]

    for dga_type in DGAType:
        try:
            sucess1 +=1
            result = get_dga_diagnose(dga_type,gases)
            #print(f'Sucesso número {sucess1, dga_type,gases}', end='\r')
        except:
            erro += 1
            print(f'Erro número {erro, dga_type,gases}', end='\r')
            erros_lista.append((dga_type, gases, item))
            result = [-1, 'Não foi possível calcular']

        if(dga_type == DGAType.DGA_MODEL):
            result = result.tolist()
            pn = result[0][0]

            if pn >= 0.6:
                result.insert(0, 0)
            elif pn < 0.4:
                result.insert(0, 2)
            else:
                result.insert(0, 1)

        result.insert(0, dga_type.name)
        result.insert(1, equip)
        result.insert(2, sub)
        result.insert(3, data_medicao)

        data.append(result)
```

Show result

Salvar DGA

```
for row in data:
    if row[0] == 'DGA_MODEL':
       row[5] = str(row[5])
df_data = ps.DataFrame(data)
df_data.columns = ["Diagnostico_cromatografico", "Equipamento", "Loc_instalacao", "Data_medicao", "Codigo_diagnostico",
"Resultado"]
```

Show result

```
df_data.to_delta(CAMINHO_SAP_REFINED_DGA, mode="overwrite")
```

Show code

Show result

Processamento df_lista_equipamentos

```
df_data = df_data[df_data['Diagnostico_cromatografico'] == 'DGA_MODEL']
df_data.head()
```

Show result

```
df_data = df_data.sort_values(['Equipamento','Data_medicao'], ascending = False)
df_data = df_data.drop_duplicates(subset='Equipamento')[['Codigo_diagnostico', 'Data_medicao', 'Equipamento']]
df_data.head()
```

Show result

```
df_refined_equipamentos_ps = ps.read_delta(CAMINHO_SAP_REFINED_EQUIPAMENTOS)
df_merged = df_refined_equipamentos_ps.merge(df_data, on='Equipamento', how='left', suffixes=('', '_updated'))
df_merged.head()
```

Show result

```
df_merged['IEC'] = df_merged['Codigo_diagnostico']
df_merged['Data_medicao'] = df_merged['Data_medicao_updated']
df_merged.head()
```

Show result

```
df_merged = df_merged.drop(columns={'Codigo_diagnostico','Data_medicao_updated'})
df_merged.head()
```

Show result

```
df_merged.IEC = df_merged.IEC.astype('int32')
```

```
df_merged.to_delta(CAMINHO_SAP_REFINED_EQUIPAMENTOS, mode= 'overwrite')
```

Atualizar a flag IEC no Cosmos

```
df = ps.read_delta(CAMINHO_SAP_REFINED_EQUIPAMENTOS)
df.head()
```

Show result

```
lista_loc_instalacao = list(set(df.Loc_instalacao.to_list()))
lista_loc_instalacao
```

Show result

```
load_dotenv()

DB_NAME = "AppData"
COLLECTION_NAME = "Subestacoes"

client = pymongo.MongoClient(CONNECTION_STRING)

db = client[DB_NAME]
collection = db[COLLECTION_NAME]

documentos = list(collection.find({}))

for documento in documentos:
    if documento['LocInstalacao'] in lista_loc_instalacao:
        documento['IEC'] = True
```

Show result

```
for update in documentos:
  collection.update_one({"_id": update["_id"]}, {"$set": update})
```

Calcular idade dos equipamentos

```
import datetime

df_planilha = ps.read_delta(CAMINHO_IH08).to_pandas()
df_planilha.head()
```

Show result

Não há nenhum valor nulo em Dt_entr_servico

```
df_planilha.columns
```

```
Out[128]: Index(['Loc_instalacao', 'Denominacao', 'Ctg_equipamento', 'GrpAutorizacoes',
       'Equipamento', 'Denominacao_1', 'Tipo_de_objeto', 'No_serie',
       'Denom_do_tipo', 'Fabricante', 'Ano_construcao', 'Dt_entr_servico',
       'Pais_produtor', 'Perf_catalogo', 'Status_sistema', 'Status_usuario',
       'Area_operacion', 'Cen_manutencao', 'CenTrab_respon', 'Grp_plnj_PM',
       'Centro_trabalho', 'Centro_planej', 'Contrato_equipamento'],
      dtype='object')
```

```
df_planilha['Dt_entr_servico'] = df_planilha['Dt_entr_servico'].astype('datetime64[ns]').dt.year

df_planilha.loc[df_planilha['Ano_construcao'] == '', 'Ano_construcao'] = np.nan
df_planilha['Ano_construcao'].astype(float)
```

Show result

```
df_planilha[(df_planilha['Dt_entr_servico'].isna()) | (df_planilha['Dt_entr_servico'] == '')]
```

Show result

```
df_cosmos_sub = pd.DataFrame(data = collection.find({}))
df_cosmos_sub
```

Show result

```
df_planilha['locinst'] = df_planilha['Loc_instalacao'].str.slice(0,9)
```

```
df_merged = df_planilha.merge(df_cosmos_sub[['Nome', 'LocInstalacao', 'Departamento']], left_on = 'locinst', right_on=
'LocInstalacao', how = 'left').rename(columns = {'Nome':'Subestacao'})
df_merged.LocInstalacao.isna().sum()
```

```
Out[133]: 333
```

```
df_merged.head()
```

Show result

Filtro de anos 1959 - 2023:

```
df_merged['Ano_construcao'].isna().sum()
```

```
Out[135]: 2286
```

```
df_merged.loc[(df_merged['Ano_construcao'].isna()) | (~df_merged['Ano_construcao'].between(1956,2023, inclusive = 'both')),
'Ano_construcao'] = df_merged['Dt_entr_servico']
df_merged.loc[~df_merged['Ano_construcao'].between(1956,2023, inclusive = 'both'), 'Ano_construcao'].value_counts()
```

```
Out[136]: 1900.0    1082
Name: Ano_construcao, dtype: int64
```

```
df_merged = df_merged[df_merged['Ano_construcao'].between(1956,2023, inclusive = 'both')]
```

Groupby Departamento

```
df_grouped_dept = df_merged.groupby('Departamento')['Ano_construcao'].mean().reset_index()
df_grouped_dept['Nivel'] = 'Departamento'
df_grouped_dept['Idade_Equipamento'] = round(datetime.date.today().year - df_grouped_dept['Ano_construcao'])
df_grouped_dept = df_grouped_dept.rename(columns = {'Departamento': 'Nome'})
df_grouped_dept = df_grouped_dept.drop(columns = 'Ano_construcao')
df_grouped_dept
```

Show result

Agrupado por Subestacao

```
df_grouped_sub = df_merged.groupby('Subestacao')['Ano_construcao'].mean().reset_index()
df_grouped_sub['Nivel'] = 'Subestacao'
df_grouped_sub['Idade_Equipamento'] = round(datetime.date.today().year - df_grouped_sub['Ano_construcao'])
df_grouped_sub = df_grouped_sub.rename(columns = {'Subestacao': 'Nome'})
df_grouped_sub = df_grouped_sub.drop(columns = 'Ano_construcao')
df_grouped_sub
```

Show result

Agrupamento por Tipo

Filtro CE

```
df_ce = df_merged[((df_merged['Loc_instalacao'].str.upper().str.contains('-CE'))&
(df_merged['Loc_instalacao'].str.upper().str.contains('-MC')))|(df_merged['Tipo_de_objeto'] == 'COES')]
df_ce_grouped = df_ce.groupby('Subestacao').mean()['Ano_construcao'].reset_index()
df_ce_grouped['Subestacao'] = 'CE_' + df_ce_grouped['Subestacao']
df_ce_grouped['Nivel'] = 'Tipo'
df_ce_grouped['Idade_Equipamento'] =  round(datetime.date.today().year - df_ce_grouped['Ano_construcao'],1)
df_ce_grouped = df_ce_grouped.drop(columns = 'Ano_construcao').rename(columns = {'Subestacao':'Nome'})
df_ce_grouped
```

Show result

Filtro BC

```
df_bc = df_merged[df_merged['Tipo_de_objeto'] == 'BCCP']
df_bc_grouped = df_bc.groupby('Subestacao').mean()['Ano_construcao'].reset_index()
df_bc_grouped['Subestacao'] = 'BC_' + df_bc_grouped['Subestacao']
df_bc_grouped['Nivel'] = 'Tipo'
df_bc_grouped['Idade_Equipamento'] =  round(datetime.date.today().year - df_bc_grouped['Ano_construcao'],1)
df_bc_grouped = df_bc_grouped.drop(columns = 'Ano_construcao').rename(columns = {'Subestacao':'Nome'})
df_bc_grouped
```

Show result

Filtro BS

```
df_bs = df_merged[(df_merged['Tipo_de_objeto'] == 'BCCP')&(df_merged['Loc_instalacao'].str.upper().str.contains('-BS'))]
df_bs_grouped = df_bs.groupby('Subestacao').mean()['Ano_construcao'].reset_index()
df_bs_grouped['Subestacao'] = 'BS_' + df_bs_grouped['Subestacao']
df_bs_grouped['Nivel'] = 'Tipo'
df_bs_grouped['Idade_Equipamento'] =  round(datetime.date.today().year - df_bs_grouped['Ano_construcao'],1)
df_bs_grouped = df_bs_grouped.drop(columns = 'Ano_construcao').rename(columns = {'Subestacao':'Nome'})
df_bs_grouped
```

Show result

Filtro CR

```
df_reat = df_merged[df_merged['Tipo_de_objeto'] == 'REAT']
df_reat_grouped = df_reat.groupby('Subestacao').mean()['Ano_construcao'].reset_index()
df_reat_grouped['Subestacao'] = 'CR_' + df_reat_grouped['Subestacao']
df_reat_grouped['Nivel'] = 'Tipo'
df_reat_grouped['Idade_Equipamento'] =  round(datetime.date.today().year - df_reat_grouped['Ano_construcao'],1)
df_reat_grouped = df_reat_grouped.drop(columns = 'Ano_construcao').rename(columns = {'Subestacao':'Nome'})
df_reat_grouped
```

Show result

Filtro CS

```
df_cs = df_merged[df_merged['Loc_instalacao'].str.upper().str.contains('-CS')]
df_cs_grouped = df_cs.groupby('Subestacao').mean()['Ano_construcao'].reset_index()
df_cs_grouped['Subestacao'] = 'CS_' + df_cs_grouped['Subestacao']
df_cs_grouped['Nivel'] = 'Tipo'
df_cs_grouped['Idade_Equipamento'] =  round(datetime.date.today().year - df_cs_grouped['Ano_construcao'],1)
df_cs_grouped = df_cs_grouped.drop(columns = 'Ano_construcao').rename(columns = {'Subestacao':'Nome'})
df_cs_grouped
```

Show result

Filtro MG

```
df_mg = df_merged[df_merged['Loc_instalacao'].str.upper().str.contains('-MG')]
df_mg_grouped = df_mg.groupby('Subestacao').mean()['Ano_construcao'].reset_index()
df_mg_grouped['Subestacao'] = 'MG_' + df_mg_grouped['Subestacao']
df_mg_grouped['Nivel'] = 'Tipo'
df_mg_grouped['Idade_Equipamento'] =  round(datetime.date.today().year - df_mg_grouped['Ano_construcao'],1)
df_mg_grouped = df_mg_grouped.drop(columns = 'Ano_construcao').rename(columns = {'Subestacao':'Nome'})
df_mg_grouped
```

Show result

Filtro RS

```
df_rs = df_merged[df_merged['Loc_instalacao'].str.upper().str.contains('-RS')]
df_rs_grouped = df_rs.groupby('Subestacao').mean()['Ano_construcao'].reset_index()
df_rs_grouped['Subestacao'] = 'RS_' + df_rs_grouped['Subestacao']
df_rs_grouped['Nivel'] = 'Tipo'
df_rs_grouped['Idade_Equipamento'] =  round(datetime.date.today().year - df_rs_grouped['Ano_construcao'],1)
df_rs_grouped = df_rs_grouped.drop(columns = 'Ano_construcao').rename(columns = {'Subestacao':'Nome'})
df_rs_grouped
```

Show result

Filtro TR

```
df_tr = df_merged[df_merged['Tipo_de_objeto'].isin(('TFFR', 'TFRL'))]
df_tr_grouped = df_tr.groupby('Subestacao').mean()['Ano_construcao'].reset_index()
df_tr_grouped['Subestacao'] = 'TR_' + df_tr_grouped['Subestacao']
df_tr_grouped['Nivel'] = 'Tipo'
df_tr_grouped['Idade_Equipamento'] =  round(datetime.date.today().year - df_tr_grouped['Ano_construcao'],1)
df_tr_grouped = df_tr_grouped.drop(columns = 'Ano_construcao').rename(columns = {'Subestacao':'Nome'})
df_tr_grouped
```

Show result

Filtro RT

```
df_rt = df_merged[(df_merged['Tipo_de_objeto'] == 'REAT') & (df_merged['Loc_instalacao'].str.upper().str.contains('-RT')) &
((df_merged['Denominacao'].str.upper().str.contains('FASE')))]
df_rt_grouped = df_rt.groupby('Subestacao').mean()['Ano_construcao'].reset_index()
df_rt_grouped['Subestacao'] = 'RT_' + df_rt_grouped['Subestacao']
df_rt_grouped['Nivel'] = 'Tipo'
df_rt_grouped['Idade_Equipamento'] =  round(datetime.date.today().year - df_rt_grouped['Ano_construcao'],1)
df_rt_grouped = df_rt_grouped.drop(columns = 'Ano_construcao').rename(columns = {'Subestacao':'Nome'})
df_rt_grouped
```

Show result

```
df_final = pd.concat([df_grouped_dept,df_grouped_sub, df_ce_grouped, df_bc_grouped, df_bs_grouped, df_reat_grouped,
df_cs_grouped, df_mg_grouped, df_rs_grouped, df_tr_grouped, df_rt_grouped])
display(df_final)
```

Show result

```
/databricks/spark/python/pyspark/pandas/internal.py:1572: FutureWarning: iteritems is deprecated and will be removed in a future
version. Use .items instead.
  fields = [
/databricks/spark/python/pyspark/sql/pandas/conversion.py:656: FutureWarning: iteritems is deprecated and will be removed in a f
uture version. Use .items instead.
  [(c, t) for (_, c), t in zip(pdf_slice.iteritems(), arrow_types)]
```

A.3
NB Final script

databricks nb_final

(https://databricks.com)

```
import numpy as np
import pandas as pd
from pathlib import Path
from enum import Enum, auto
from dataclasses import dataclass
from abc import ABC, abstractmethod
from os import path, rename, listdir, stat
from datetime import date, datetime, timedelta
import hashlib, csv, re, locale, calendar, glob
import pyspark.pandas as ps
import re
from os import listdir, path, getcwd, stat
from datetime import date, datetime, timedelta
import os, glob
from pathlib import Path
import time
import itertools, re
from concurrent.futures import ThreadPoolExecutor, ProcessPoolExecutor
pd.options.mode.chained_assignment = None
import time
import json
from datetime import datetime
#import caminhos_pastas
```

Show result

```
%run "/Workspace/Dev/preprocessing/SAGE/mover_trusted/SAGE_utils.ipynb"
```

Show result

```
def processAnIntegerNumber(sub):
  dbutils.notebook.run(path = caminho_notebook_parent,
                                     timeout_seconds = 14400,
                                     arguments = {'subestação': sub})
```

Show result

```
def executar_notebooks_ano(subs):
    with ThreadPoolExecutor() as executor:
        results = executor.map(processAnIntegerNumber, subs)
```

Show result

```
subestacoes = os.listdir(f'/dbfs{caminho_landing}/sage/')
```

Show result

```
executar_notebooks_ano(subestacoes)
```

Show result

Notebook Workflows

Start time	End time	Notebook path	Duration	Status	Error code	Run parameters

```
month_translation = {
    'jan': 'jan',
    'fev': 'feb',
    'mar': 'mar',
    'abr': 'apr',
    'mai': 'may',
    'jun': 'jun',
    'jul': 'jul',
    'ago': 'aug',
    'set': 'sep',
    'out': 'oct',
    'nov': 'nov',
    'dez': 'dec',
}

def translate_month(date_string):
    month_portuguese = date_string[:3]
    month_english = month_translation.get(month_portuguese, '')
    return date_string.replace(month_portuguese, month_english)

def transform_date(date_string):
    given_date = datetime.strptime(translate_month(date_string), "%b%d%y").date()
    current_date = datetime.now().date()
    return given_date == current_date

def falha_na_data_atual(arquivo_log_caminho):
    datas = []
    df = pd.read_csv(arquivo_log_caminho, header = None)
    for arquivo in df[3]:
        datas.append(arquivo.replace('.pas', '')[-7:])

    for data in datas:
        if transform_date(data):
            return True
    return False

def is_dataframe_empty(arquivo_log_caminho):
    return pd.read_csv(arquivo_log_caminho).empty

def falhas_datas_atual_e_anteriores(arquivo_log_caminho):
    df = ps.read_delta(arquivo_log_caminho, header = None).to_pandas()
    df = df[df['Sistema'] == 'SAGE']
    data_hoje = datetime.now().date()
    erro_na_data_atual = False
    erro_datas_passadas = False
    for index, row in df.iloc[1:].iterrows():

        #data_execucao_erro = datetime.strptime(row[1].replace('.pas', ''), '%d/%m/%Y %H:%M:%S').date()
        data_execucao_erro = row['Data_execucao_erro'].date()
        data_arquivo_erro = datetime.strptime(translate_month(row['Nome_arquivo_erro'].replace('.pas', '').split('/')[-1]),
"%b%d%y").date()
        if (data_arquivo_erro == data_hoje):
            erro_na_data_atual = True
        if (data_execucao_erro == data_hoje) and (data_arquivo_erro != data_hoje):
            erro_datas_passadas = True
    return erro_na_data_atual, erro_datas_passadas
```

Show result

```
# arquivo_log_caminho = '/dbfs/mnt/sandbox/trusted/sage/arquivos_com_problemas_indexacao.csv'
# retorno = falha_na_data_atual(arquivo_log_caminho), not is_dataframe_empty(arquivo_log_caminho)
```

Show result

```
#arquivo_log_caminho = f'/dbfs{caminho_trusted}/sage/arquivos_com_problemas_indexacao.csv'
arquivo_log_caminho = '/mnt/trusted/_logs'
retorno = falhas_datas_atual_e_anteriores(arquivo_log_caminho)
```

Show result

Show result