**UNIVERSIDADE PAULISTA**

**CAIO PEREIRA CALADO**

**REBECCA SANTOS DO AMARAL**

**GABRIEL CAMACHO DE MEDEIROS**

**HENRIQUE ALI OLIVEIRA ALVES**

**UTILIZAÇÃO DE METODOLOGIA DATA LAKE EM CENÁRIOS DE ALTO VOLUME DE DADOS**

**SÃO PAULO**

**2023**

**CAIO PEREIRA CALADO**

**REBECCA SANTOS DO AMARAL**

**GABRIEL CAMACHO DE MEDEIROS**

**HENRIQUE ALI OLIVEIRA ALVES**

**UTILIZAÇÃO DE METODOLOGIA DATA LAKE EM CENÁRIOS DE ALTO VOLUME DE DADOS**

Trabalho de conclusão de curso apresentado à banca examinadora da Faculdade UNIP, como requisito parcial à obtenção do Bacharelado em Ciência da Computação.

Orientador: Professor Marcos Júnior

**SÃO PAULO**

**2023**

**CAIO PEREIRA CALADO**

**REBECCA SANTOS DO AMARAL**

**GABRIEL CAMACHO DE MEDEIROS**

**HENRIQUE ALI OLIVEIRA ALVES**

**UTILIZAÇÃO DE METODOLOGIA DATA LAKE EM CENÁRIOS DE ALTO VOLUME DE DADOS**

Trabalho de Conclusão de Curso para obtenção do título de Graduação em Ciência da Computação apresentado à Universidade Paulista – UNIP.

Aprovado em:

BANCA EXAMINADORA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_/\_\_\_\_\_

Prof. Nome do Professor

Universidade Paulista – UNIP

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_/\_\_\_\_\_

Prof. Nome do Professor

Universidade Paulista – UNIP

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_/\_\_\_\_\_

Prof. Nome do Professor

Universidade Paulista UNIP

**RESUMO**

Um Data Lake é uma arquitetura que permite o armazenamento de grandes volumes de dados brutos de diferentes fontes em sua forma nativa, sem a necessidade de uma estrutura pré-definida. Esse conceito é particularmente relevante em cenários de Big Data, onde lidamos com grandes volumes, variedade e velocidade de dados.

O trabalho aborda a estrutura de um Data Lake, que é projetado para exemplificar às características específicas deste conceito. O Data Lake proporciona flexibilidade na ingestão de dados, permitindo a incorporação de dados estruturados e não estruturados. Além disso, ele é capaz até de lidar com dados em tempo real, proporcionando uma base sólida para análises em larga escala.

Há também a enfatização e aplicação prática da arquitetura, destrinchada no desenvolvimento, sendo detalhados os fluxos de ingestão, processamento e visualização aplicados aos dados existentes nas camadas definidas no projeto. E a arquitetura elaborada possui processos externos de governança de dados, manutenção do ambiente e segurança do ambiente e dados armazenados mediante a restrição de acessos.

A estrutura do Data Lake envolve a divisão lógica dos dados em camadas, como as camadas Crua, Tratada e Semântica. Cada uma representando um nível distinto de refinamento e organização dos dados, permitindo o uso destas informações para aplicações diversas, pertencentes ou externas ao escopo do Data Lake.

O trabalho destaca também a importância de metadados e governança para garantir a qualidade e a integridade dos dados armazenados no Data Lake. Metadados descrevem informações sobre os dados, facilitando a compreensão, busca e gestão eficiente deles. A governança, por sua vez, estabelece políticas e procedimentos para garantir o uso adequado dos dados, incluindo questões de segurança, privacidade e conformidade regulatória.

**ABSTRACT**

A Data Lake is an architecture that allows storage of large volumes of raw data from different sources in its native form, without the need for a predefined structure. This concept is particularly relevant in Big Data scenarios, where we deal with large volumes, variety, and velocity of data.

The project addresses the structure of a Data Lake, designed to exemplify the specific characteristics of this concept. The Data Lake provides flexibility in data ingestion, allowing for the incorporation of structured and unstructured data. Moreover, it is capable of handling real-time data, providing a solid foundation for large-scale analytics.

There is also an emphasis on the practical application of the architecture, detailed in the development, with the ingestion, processing, and visualization flows applied to the data existing in the defined layers of the project. The elaborated architecture includes external processes for data governance, environment maintenance, and security of the environment and stored data through access restrictions.

The structure of the Data Lake involves the logical division of data into layers, such as Raw, Processed, and Semantic layers. Each one represents a distinct level of refinement and organization of data, allowing the use of this information for various applications, whether internal or external to the Data Lake scope.

The paper also highlights the importance of metadata and governance to ensure the quality and integrity of the data stored in the Data Lake. Metadata describes information about the data, facilitating understanding, search, and efficient management of them. Governance, in turn, establishes policies and procedures to ensure the proper use of data, including issues of security, privacy, and regulatory compliance.

**LISTA DE ILUSTRAÇÕES**

[Ilustração 1 - Gráfico representativo da quantidade de acesso à Internet 10](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920653)

[Ilustração 2 - Demonstração ilustrativa de Volume, Velocidade e Variedade dos dados 19](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920654)

[Ilustração 3 - Custo histórico do armazenamento computacional 20](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920655)

[Ilustração 4 - Exemplificação do ecossistema do Apache Spark 23](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920656)

[Ilustração 5 - Imagem representando a relação entre o mundo físico e o mundo digital 25](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920657)

[Ilustração 6 - Imagem representativa da estrutura de um Data Warehouse. 31](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920658)

[Ilustração 7 - A esquerda um Dashboard do tipo visual, a direita um do tipo funcional 33](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920659)

[Ilustração 8 - Exemplo de aplicação Serverless 37](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920660)

[Ilustração 9 - Arquitetura em Lagoa de Inmon 43](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920661)

[Ilustração 10 - Arquitetura em Zonas 43](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920662)

[Ilustração 11 - Linha de comando Hadoop para replicação de arquivos não-relacionais 45](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920663)

[Ilustração 12 - Exemplo de Diagrama de Caso de Uso UML 49](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920664)

[Ilustração 13 - Arquitetura Macro 54](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920665)

[Ilustração 14 - Função de consulta à API climática 62](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920666)

[Ilustração 15 - Fluxo de ingestão de dados climáticos 64](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920667)

[Ilustração 16 - Configurações do Banco de dados relacional 66](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920668)

[Ilustração 17 - Configurações da instância da migração de dados relacionais 67](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920669)

[Ilustração 18 - Configurações do local origem dos dados 67](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920670)

[Ilustração 19 - Configurações da base de dados de origem 68](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920671)

[Ilustração 20 - Configurações da instância de migração dos dados 68](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920672)

[Ilustração 21 - Buckets do Amazon S3 presentes na arquitetura do projeto 69](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920673)

[Ilustração 22 - Fluxo de tratamento dos dados originais da API climática 73](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920674)

[Ilustração 23 - Orquestração do fluxo de tratamento dos dados originais da API climática 74](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920675)

[Ilustração 24 - Fluxo de tratamento dos dados oriundos do banco de dados relacional 76](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920676)

[Ilustração 25 - Orquestração do fluxo de tratamento dos dados oriundos do banco de dados relacional 77](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920677)

[Ilustração 26 - Fluxo de processamento da semântica com base nos dados climáticos 79](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920678)

[Ilustração 27 - Regras de limpeza de dados da tabela 'consumers' do banco de dados 'Solar Energy Brazil' 81](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920679)

[Ilustração 28 - Função de limpeza dos dados 83](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920680)

[Ilustração 29 - Função de limpeza dos dados (cont.) 84](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920681)

[Ilustração 30 - Regra de ciclo de vida de objetos (cont.) 86](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920682)

[Ilustração 31 - Regra de ciclo de vida de objetos 86](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920683)

[Ilustração 32 - Função de movimentação de objetos 88](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920684)

[Ilustração 33 - Configurações de restrição de acesso aos dados climáticos presentes na camada tratada 90](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920685)

[Ilustração 34 - Configurações de restrição de acesso aos dados climáticos presentes na camada tratada (cont.) 91](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920686)

[Ilustração 35 - Visualização dos dados climáticos com informações omitidas pelo Lake Formation 92](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920687)

[Ilustração 36 - Configurações do Glue Crawler responsável por mapear o conjunto de dados climáticos 93](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920688)

[Ilustração 37 - Configurações do Glue Crawler responsável por mapear o conjunto de dados climáticos (cont.) 94](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920689)

[Ilustração 38 - Exemplo de utilização do Amazon Athena 95](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920690)

[Ilustração 39 - Aplicação externa manipulando os dados oriundos do Data Lake 96](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920691)

[Ilustração 40 - Função para consulta dos dados semânticos através da requisição à API 97](file:///D:\Documentos\Faculdade\TCC\Entrega_TCC_v09.docx#_Toc153920692)

**SUMÁRIO**

[1 INTrodução 9](#_Toc153920693)

[2 OBJETIVO 12](#_Toc153920694)

[2.1 Objetivos Específicos 12](#_Toc153920695)

[3 Justificativa 13](#_Toc153920696)

[4 REFERENCIAL TEÓRICO 15](#_Toc153920697)

[4.1 API 15](#_Toc153920698)

[4.2 Cloud 16](#_Toc153920699)

[4.3 Big Data 18](#_Toc153920700)

[4.3.1 As características de Big Data 19](#_Toc153920701)

[4.3.2 Modelos de dados 21](#_Toc153920702)

[4.3.3 Dados estruturados 21](#_Toc153920703)

[4.3.4 Dados não estruturados 22](#_Toc153920704)

[4.4 Apache Spark 22](#_Toc153920705)

[4.5 Internet Das Coisas 25](#_Toc153920706)

[4.6 Data Lakehouse 26](#_Toc153920707)

[4.7 Tolerância a falhas 28](#_Toc153920708)

[4.8 Data WareHouse 29](#_Toc153920709)

[4.9 Análise de Dados 31](#_Toc153920710)

[4.10 Requisitos Funcionais e Requisitos Não Funcionais 33](#_Toc153920711)

[4.11 Segurança de Dados 34](#_Toc153920712)

[4.12 Criptografia 35](#_Toc153920713)

[4.13 Lei Geral de Proteção de Dados 35](#_Toc153920714)

[4.14 Serverless 36](#_Toc153920715)

[4.15 Data Lake 37](#_Toc153920716)

[4.16 Limpeza dos dados 39](#_Toc153920717)

[4.17 Qualidade dos dados 41](#_Toc153920718)

[4.18 Arquitetura e Camadas do Data Lake 42](#_Toc153920719)

[4.18.1 Zona de Dados Brutos (Raw Zone/Landing Zone) 43](#_Toc153920720)

[4.18.2 Zona de Confiança (Trusted Zone) 45](#_Toc153920721)

[4.18.3 Zona de Análise de Dados (Data Analytics) 46](#_Toc153920722)

[4.19 UML 47](#_Toc153920723)

[4.19.1 Diagrama de Caso de Uso 48](#_Toc153920724)

[4.19.2 Diagrama de Sequência 50](#_Toc153920725)

[5 Metodologia 52](#_Toc153920726)

[5.1 Ingestão 54](#_Toc153920727)

[5.2 Armazenamento 55](#_Toc153920728)

[5.3 Processamento 57](#_Toc153920729)

[5.4 Visualização 58](#_Toc153920730)

[6 desenvolvimento 60](#_Toc153920731)

[6.1 Ingestão 60](#_Toc153920732)

[6.2 API Climática 60](#_Toc153920733)

[6.3 Banco de dados relacional 65](#_Toc153920734)

[6.4 Armazenamento 69](#_Toc153920735)

[6.5 Processamento 71](#_Toc153920736)

[6.6 API Climática 71](#_Toc153920737)

[6.7 Banco de dados relacional 75](#_Toc153920738)

[6.8 Semânticas 77](#_Toc153920739)

[6.9 Dados climáticos 78](#_Toc153920740)

[6.10 Governança de dados 80](#_Toc153920741)

[6.11 Manutenção do ambiente 85](#_Toc153920742)

[6.11.1 Ciclo de vida de objetos 85](#_Toc153920743)

[6.11.2 Movimentação de objetos 86](#_Toc153920744)

[6.12 Visualização 88](#_Toc153920745)

[6.12.1 Linguagem de consulta estruturada 92](#_Toc153920746)

[6.12.2 Interface de acesso 95](#_Toc153920747)

[Conclusão 98](#_Toc153920748)

[REFERÊNCIAS 101](#_Toc153920749)

1. INTrodução

Em um cenário abrangente, as corporações sempre possuíram a necessidade de estar um passo à frente dos concorrentes, buscando inovações a todo instante, dispor de planejamentos estratégicos para tomadas de decisões. E para ter convicção das escolhas a serem feitas, estas empresas investem em soluções tecnológicas para armazenamento, processamento e análise de dados referente aos seus processos internos, a sua relação com o cliente e o seu impacto no ambiente em que está inserida (Davenport, Dyché, 2013).

Desta forma, cada entidade necessita de uma estratégia para compor processos que serão implantados em todo o ciclo de vida das informações que forem consideradas cruciais, com intuito de permanecer relevante no mercado ao extrair percepções destes dados ao analisá-los a partir de um entendimento de qual cenário a instituição está inserida.

No entanto, não é trivial a composição de métodos que permitam tal manipulação de dados, principalmente no cenário tecnológico atual. Com a ampla difusão de aparelhos móveis, como smartphones e wearables, ocorreu a descentralização no uso de aplicações computacionais, descartando a necessidade de possuir um dispositivo centralizado para a realização de tarefas digitais. Isto é, usuários podem utilizar dos mais diversos aparelhos durante a execução de aplicativos destinados as mais variadas finalidades.

Por tanto, as organizações precisam levar em consideração quantidades cada vez mais massivas de informações provenientes de diferentes fontes, como aparelhos móveis e interfaces de programação – as APIs. Dados estes que são heterogêneos em termos de formato, estrutura e frequência de ingestão, o que torna complexo a integração e o processamento efetivo dessas informações. Além disso, sistemas convencionais de armazenamento de dados não possibilitam trabalhar com dados estruturados e não estruturados de forma conjunta e centraliza, dificultando a Gráfico, Gráfico de linhas

Descrição gerada automaticamenteinovação dos processos que organizam estes dados.

Ilustração - Gráfico representativo da quantidade de acesso à Internet

Fonte: OUR WORLD IN DATA, 2023a.

Dentro desse cenário, surge o conceito de Data Lake como uma abordagem flexível e escalável para a organização de dados de uma instituição. A partir da adoção de um repositório centralizado de informações, que não impõe que os dados sejam armazenados em uma estrutura pré-estabelecida, é possível armazenar uma grande quantidade de dados brutos provenientes de diversas fontes, em sua forma original ou modificada garantido que as informações valiosas não sejam descartadas. A abordagem de um Data Lake permite que os dados sejam segregados em camadas, nas quais é possível aplicar técnicas distintas de transformação dos dados, com intuito de tornar escalável as cargas de processamento e, dispor de dados em diferentes estágios de manipulação para possibilitar diferentes usos dessas informações simultaneamente.

Com base nessas características, o presente projeto busca desenvolver uma arquitetura de Data Lake que seja capaz de demonstrar os desafios típicos ao gerenciamento e análise de dados em larga escala, nos moldes atuais do mercado computacional, detalhando a metodologia e descrevendo as etapas relacionadas a implantação do repositório. Serão apresentados os fundamentos teóricos que embasam este projeto, discorrendo sobre conceitos-chave relacionados ao Data Lake, técnicas de armazenamento, limpeza e processamento de dados, e sua implantação baseadas em serviços em nuvem.

Por fim, serão apresentados os resultados obtidos, analisando aspectos como a eficiência da integração de dados oriundos de diferentes fontes, a utilidade dos dados armazenados, e a estrutura dos processos de limpeza e análise. Além disso, serão apresentadas considerações sobre os desafios encontrados durante o processo de implantação, e sobre a escalabilidade da arquitetura em relação ao aumento do volume de dados e a capacidade de processamento necessário.

Espera-se que este trabalho contribua para a compreensão dos benefícios e desafios associados à utilização de um Data Lake, a importância da manipulação de dados na tomada de decisões estratégicas, proporcionando uma visão abrangente e integrada dos dados, com maior flexibilidade e capacidade de análise.

1. OBJETIVO

Desenvolver uma arquitetura de Data Lake separada em camadas, capaz de realizar a ingestão de dados originados de APIs e dispositivos de IoT, realizando o armazenamento, limpeza e processamento destes dados.

* 1. Objetivos Específicos
* Realizar a integração de dados provenientes de API’s e dispositivos de IoT em um Data Lake;
* Utilizar serviços da AWS (Amazon Web Services), a fim de garantir o armazenamento desses dados no ambiente da nuvem;
* Executar a limpeza e o pré-processamento dos dados, empregando técnicas de validação, tratamento e transformação das informações.

1. Justificativa

No atual contexto do mundo dos negócios, reconhecemos a alta demanda e a complexidade das soluções de gerenciamento e análise de dados. Com membros do nosso grupo tendo experiência neste mercado, compreendemos a importância de aprofundar o estudo sobre as camadas de um Data Lake, explorando suas funcionalidades, benefícios, desafios e oportunidades de melhoria, como mencionam Singh e Ahmad (Madden, 2012). Isso é essencial para impulsionar o desenvolvimento de futuras inovações nesse campo e atender às necessidades crescentes das empresas em relação ao gerenciamento eficiente de seus dados.

À medida que as empresas lidam com volumes cada vez maiores de dados, é crucial ter uma abordagem eficaz para a ingestão, armazenamento, limpeza e processamento dessas informações. O uso de um Data Lake, que oferece uma arquitetura separada em camadas, proporciona a flexibilidade e escalabilidade necessárias para lidar com o crescimento contínuo na escrita e processamento de dados, mencionado por Pasupuleti e Purra (AGRAWAL, et al. 2011). Essa abordagem permite que as organizações centralizem e processem seus dados de forma mais eficiente, gerando insights valiosos para a tomada de decisões estratégicas, como é trazido por Misra e John (DAVENPORT, 2013).

Ao compreender em detalhes como cada camada do Data Lake funciona, podemos explorar todo o potencial dessa abordagem e identificar oportunidades para aprimoramentos. Cada camada, desde a ingestão até o processamento, desempenha um papel específico no fluxo de dados e tem um impacto direto na qualidade e utilidade das informações resultantes (AGRAWAL, et al. 2011). Portanto, é fundamental estudar e analisar essas camadas para garantir a otimização dos processos de gerenciamento e análise de dados.

Além disso, é importante ressaltar que o conceito de Data Lake está em constante evolução e desempenha um papel fundamental como base para técnicas futuras, como o Data Lakehouse, como é abordado por Hlupić e Oreščanin (Davenport, Dyché, 2013). Sendo assim, ao estudar e aprofundar o conhecimento sobre as camadas de um Data Lake, estamos contribuindo para a construção de uma base sólida que servirá de referência no desenvolvimento de futuras inovações no campo do gerenciamento e análise de dados.

Com base nessas considerações, o objetivo deste projeto foi desenvolver uma arquitetura de Data Lake separada em camadas, capaz de realizar a ingestão, armazenamento, limpeza e processamento de dados provenientes de APIs e dispositivos de IoT. Ao alcançar esses objetivos, buscamos contribuir para o entendimento e conhecimento das camadas de um Data Lake, explorando suas vantagens, desafios e oportunidades de melhoria, incentivando assim o avanço contínuo no campo do gerenciamento e análise de dados.

1. REFERENCIAL TEÓRICO
   1. API

Uma Interface de Programação de Aplicativos (API), é um conjunto de regras e protocolos definidos que permitem que diferentes aplicações se comuniquem entre si. Ela atua como uma camada intermediária que processa transferência de dados entre sistemas, permitindo que empresas disponibilizem suas aplicações a terceiros, parceiros ou entre seus departamentos.

A API apresenta uma função chave nas aplicações atuais e na reutilização de software, possibilitando que empresas conectem à diversas aplicações distintas que utilizam no seu dia a dia, quanto a API pode ser afirmar que:

Eles facilitam a interoperabilidade de sistemas de computação. Nos últimos anos, vimos um crescimento exponencial do uso de APIs para conectar aplicativos e serviços da Web pela Internet. Atualmente, muitos aplicativos da web usam serviços de autenticação ou armazenamento do Google, Facebook ou Dropbox graças a suas APIs públicas. Provedores de nuvem, como Amazon Web Services (AWS), oferecem interfaces de auto provisionamento e gerenciamento por meio de uma API. (RAUF, 2019, p. 2).

A interoperabilidade de uma API permite que empresas economizem tempo dos funcionários e elimina os silos que impedem a colaboração e a inovação. Para os desenvolvedores, a documentação API prove uma interface de comunicação entre aplicações, facilitando integrações/extensões.

Uma forma de entender como operam as APIs é olhar para um exemplo comum, como o processamento de pagamento por terceiros, normalmente realizado por cartão. Quando o usuário realiza uma compra em um site e-commerce, ele se depara com um botão que redireciona até a plataforma responsável para efetivação do pagamento, tendo como o exemplo, a o serviço Steam que é uma distribuidora de jogos digitais utiliza, dentro do Brasil, do “BoaCompra” para tal demanda.

A comunicação e solicitação da API ocorra decorrentes dos seguintes passos: A API faz uma chamada de informação por meio de uma requisição. A requisição é processada para o servidor web por meio da URI de uma API que contém o cabeçalho e o corpo da requisição -URI trata-se do Identificador de Recursos Universal, seu corpo é semelhante a uma URL, mas de forma simplificada para melhor entendimento, ela une o Protocolo (https://) a URL e ao URN (Nome de Recursos Universal) para que seja possível a aplicação acessar itens na web.

O servidor envia uma resposta a API com as informações solicitadas, e assim a API transfere as informações para a aplicação solicitante inicial.

O tipo de informação transitada por diferir dependendo da aplicação, porém, as respostas e requisições ocorrem por um tipo de API.

Segundo Rauf (2019, p.3) “As APIs oferecem serviços de software para públicos mais amplos que podem usar a funcionalidade oferecidos pelas APIs de maneiras além da imaginação dos desenvolvedores originais.” Pode notar-se então incontáveis finalidades para a usabilidade de uma API, como por exemplo, a Api do Google Maps é utilizada atualmente por milhares de maneira únicas e inovadoras. Como a AirBnB utiliza dessa Api para revolucionar o mercado de hospedagem.

* 1. Cloud

O termo em inglês "Cloud", cuja tradução para o português é "Nuvem", refere-se à própria internet e é uma das expressões frequentemente utilizadas para descrever o conceito mais abrangente de infraestrutura em nuvem. O princípio fundamental da Computação em Nuvem está baseado na essência dos Sistemas Distribuídos, que se dedica ao estudo da metodologia de compartilhamento de recursos computacionais geograficamente separados e distribuídos por meio de uma rede, sendo que, no caso da nuvem, essa rede é a internet. Uma característica relevante, destacada por Andrew Tanenbaum (SAGIROGLU, SINANC, 2013), é a transparência desse sistema, que se apresenta ao usuário de forma única e coerente.

Conforme a definição de Peter Mell e Timothy Grance (ZIKOPOULOS, EATON, 2011), a Computação em Nuvem é uma "shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". Deste modo, é correto afirmar que essa abordagem elimina a necessidade de que todas as empresas adquiram recursos físicos a longo prazo e dependam de uma equipe técnica especializada para sustentar a infraestrutura subjacente, apenas para lançar suas soluções no mercado.

No mesmo artigo (ZIKOPOULOS, EATON, 2011), Mell e Grance explicam as características essenciais da Computação em Nuvem, bem como os modelos de serviço e modelos de implantação. Dentre essas características, podemos citar o autoatendimento do cliente, em que os serviços são provisionados de acordo com suas necessidades; a alta disponibilidade dos recursos, proporcionada pelo amplo acesso à internet; o agrupamento de recursos para atender a vários consumidores; a elasticidade, que permite a alocação e liberação rápidas de recursos; e a automação otimizada do uso de recursos para aproveitar plenamente sua capacidade, com o nível de abstração ideal do hardware para o usuário.

Os modelos de serviço definem como esses recursos são compartilhados na nuvem e são divididos em três categorias: SaaS (Software como Serviço), que permite o uso de aplicações fornecidas pelo provedor e são executadas em um ambiente em nuvem; PaaS (Plataforma como Serviço), no qual o usuário utiliza a infraestrutura em nuvem para desenvolver suas próprias aplicações, sem se preocupar com a infraestrutura de rede ou hardware subjacente; e IaaS (Infraestrutura como Serviço), em que o usuário não gerencia o hardware utilizado, mas tem controle sobre os sistemas, armazenamento e aplicativos implantados nele, proporcionando um nível mais amplo de gerenciamento de recursos em comparação com os outros modelos (ZIKOPOULOS, EATON, 2011).

Modelos de implantação da computação em nuvem, também abordados por Meikang Qiu e Keke Gai (GÄRTNER, HIEBL, 2017), são classificados em quatro tipos distintos: a Nuvem Privada, que é provisionada exclusivamente para o uso de uma única organização; a Nuvem Comunitária, similar à privada, porém direcionada ao uso de uma comunidade específica de consumidores; a Nuvem Pública, destinada ao uso livre e aberto ao público em geral; e a Nuvem Híbrida, que representa a combinação de dois ou mais modelos (ZIKOPOULOS, EATON, 2011).

Nesse contexto, os provedores de serviços em nuvem assumem a responsabilidade pela aquisição, manutenção, disponibilidade e segurança do hardware, permitindo que as empresas concentrem exclusivamente seus esforços nas questões relacionadas ao negócio, adquirindo recursos sob demanda por meio da internet.

* 1. Big Data

No decorrer das últimas duas décadas, o termo Big Data difundiu-se no tocante de volumes de dados massivos, extremamente custosos de se armazenar, analisar e visualizar. Sumariamente, o termo aplicasse às informações que não são capazes de serem processadas ou analisadas por ferramentas e processos tradicionais (Sagiroglu; Sinanc. 2013)

Compondo a definição de Big Data, podemos considerar os dados muito grandes, muito rápidos ou muito difíceis para as ferramentas existentes processarem (S. Madden, 2012). No qual é imposto a manipulação de conjuntos de informações em enormes escalas de armazenamento provenientes de distintas origens, o processamento imediato dos dados com o intuito de gerar entendimentos rapidamente, e realizar análises que atingem o limite impostos pelas ferramentas atuais.

Embora os sistemas relacionais de dados difundidos no mercado, como os mecanismos MySQL e Postgres, sejam capazes de manipular múltiplos petabytes de informações, eventualmente sofrem para escalonar em conjunto com as demandas (S. Madden, 2012). Ou seja, sistemas relacionais de dados não se apresentam como a melhor opção em cenários que a velocidade e variedade dos dados for fator crucial para o sucesso da aplicação.

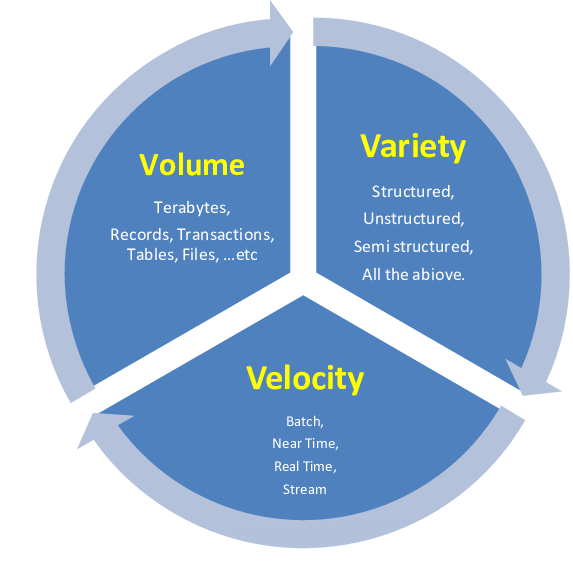
Progressivamente, instituições fundamentadas na análise de dados deparam-se com quantidades cada vez maiores de desafios relacionados a Big Data. Mesmos que possuam o acesso a um amplo conjunto de informações, dificultosamente são realizados os processos capazes de gerar entendimento com base nesses dados, pois é preciso ter em consideração que estes materiais são disponibilizados de forma bruta ou em formatos semiestruturados ou não estruturados. Consequentemente, há a dúvida de quais informações são valiosas de ser ter histórico, e se os procedimentos atuais são capazes de atender o que se é esperado como retorno.

Com as mudanças contínuas de comportamento da nossa sociedade, progressivamente mais incorporada a soluções digitais de diversas utilidades, a compreensão sobre o domínio de complexas coleções de dados recebe maior visibilidade. Através das ferramentas atuais, nós podemos expandir a gama de experiencias possíveis, por consequência, aumentamos a quantidade de registro que desejamos armazenar. Também, tornamos as pessoas e a tecnologia ainda mais interconectadas por meio dos avanços nas ferramentas de comunicação, permitindo que que incorporássemos os mais rotineiros dispositivos na geração de valor por meio de sensores.

* + 1. As características de Big Data

De acordo a IBM, Big Data é caracterizado por três vertentes distintas que se complementam para formar o novo desafio acerca do crescimento de dados globalmente, ressaltando o destaque ao volume, variedade e velocidades expressados abaixo:

Ilustração - Demonstração ilustrativa de Volume, Velocidade e Variedade dos dados

Fonte: AL-BARHAMTOSHY, EASSA, 2014

O **Volume** ou tamanho dos dados atuais ultrapassou escalas de armazenamento surreais. Exemplos deste cenário, são aplicações de interações sociais, como o Facebook, que geram mais de 10 *terabytes* diariamente (Zikopoulos, 2011).

Gráfico, Gráfico de linhas

Descrição gerada automaticamenteAtualmente, nós armazenamos praticamente tudo, informações financeiras, de saúde, de rastreamento, entre outros tópicos. Devido ao barateamento das tecnologias de armazenamento, torna-se usual empresas de médio e grande porte possuem sistemas distribuídos para acumulação desta alta quantidade de dados.

Ilustração - Custo histórico do armazenamento computacional

Fonte: OUR WORLD IN DATA, 2023b.

A grandeza dessas informações, não se limita apenas ao tamanho de cada registro ou espaço ocupado pelo conjunto desses dados, também se relaciona com a Variedade de formatos que os elementos de informações são gerados.

Com a diversidade de aparelhos capazes de gerar dados a um sistema, baseado no aumento de dispositivos inteligentes em nosso cotidiano, os processos de gestão de dados de uma organização se tornam mais complexos, fundamentos no pressuposto de que as informações não serão disponibilizadas apenas de forma relacional, mas que dados brutos, semiestruturados e não estruturados formam o novo padrão de origem de dados (SAGIROGLU; SINANC, 2013).

Compondo os pilares de Big Data, há a consideração sobre a rapidez com que os dados são enviados e registrados pelas ferramentas de armazenamento, e o tempo necessário para recuperação das informações (Zikopoulos, 2011).

O conceito de Velocidade compreende a concepção dos dados em movimento. Ao gerenciar as informações na rapidez em que se tornam disponíveis ao sistema é possível obter vantagens estratégicas ao reconhecer tendências, identificar problemas ou oportunidades em tempo (quase) real. A velocidade de é crucial ao considerar a curta vida útil dos dados atuais das organizações. Informações provenientes de sistemas embarcados e/ou aplicações mobiles devem ser analisadas rapidamente para a geração de valor.

* + 1. Modelos de dados

Um dos desafios presentes no uso da Big Data para tomada de decisão baseada em dados, é o controle das informações considerando a variedade formatos de dados produzidos atualmente e que precisam ser trabalhos de forma apartada para formar as bases de conhecimento, das quais serão extraídas as interpretações estratégicas desejadas (AGRAWAL, 2011).

As informações criadas pelas aplicações atuais não são criadas igualmente, e consequentemente são coletadas e escalonadas de formas diferentes. Desta forma, consideraremos os modelos de dados estruturados e não estruturados.

* + 1. Dados estruturados

Conforme a definição apresentada pela IBM (2021), dados estruturados são altamente organizados e decifráveis, habitualmente manipulados por meio de banco de dados relacionais para inserção, procura e manipulação de informações.

Este formato de dados necessita ser armazenados de maneira altamente consiste para manter a sua essência. Cada informação é composta por um ou múltiplos atributos, geralmente representados por colunas. Estrutura essa definida previamente, no decorrer da concepção do banco de dados. Dessa forma, independentemente se um registro não possuir valor referente a algum atributo, a estrutura padrão da informação será mantida.

Por isso, este tipo de dado é altamente utilizado em inúmeras ferramentas. Com uma compreensão prévia do modelo do dado, usuário de diferentes níveis de conhecimento técnico podem acessar e manipular essas informações estruturadas de maneira analítica ou transacional.

Entretanto, dados estruturados não possibilitam um uso altamente flexível, sendo utilizados para a finalidade a que foram destinados previamente, limitados a serem armazenados em sistemas com imposições rígidas de esquema de dados, inviabilizando atualizações baratas à estrutura das informações.

* + 1. Dados não estruturados

Por outro lado, ao manipular dados que não podem ser processados e analisados por através de métodos e ferramentas convencionais, está sendo tratado de dados não estruturados.

Fundamentando a definição desse modelo de dados, temos a ideia de informação sem uma estrutura pré-definida, oriunda de fontes de dados distintas.

Compondo mais de 80% das bases de conhecimento empresariais, de acordo com estudos recentes (IBM, 2021), dados não estruturados vêm ganhando cada vez mais destaque em questão de seu valor a partir do crescimento acelerado de soluções mobiles, embarcadas e aplicações mais consolidadas, como redes sociais.

Uma ótima opção para sistema que necessitam coletar e armazenar dados de maneira rápida e escalável, flexibilizando a adoção de estratégias de manipulação de dados.

* 1. Apache Spark

O Apache Spark é um motor de processamento de dados de código aberto, desenvolvido para atuar em cenários de *Big Data*. (IBM, 2021). Permitindo a integração com módulos integrados de consumo de dados, como aprendizado de máquina, *streaming* e processamento de gráficos. E simplificando a implantação de estratégias de alta escalabilidade, confiabilidade e tolerância a falhas.

O Spark se destaca por sua abordagem de manipulação de dados. Visando um processamento mais rápido, em comparação com demais *frameworks* disponíveis no mercado, os dados são carregados na memória primária do sistema distribuído de processamento para agilizar a consulta e relação entre as informações.

Outra particularidade relevante do Spark é o procedimento para orquestração e priorização das tarefas entre os nós de um cluster. O Spark baseia-se em gráficos acíclicos direcionados (DAG) para controlar a execução do processamento, tornando o sistema tolerante a falhas ao permitir a reversão das operações a um estado anterior saudável.

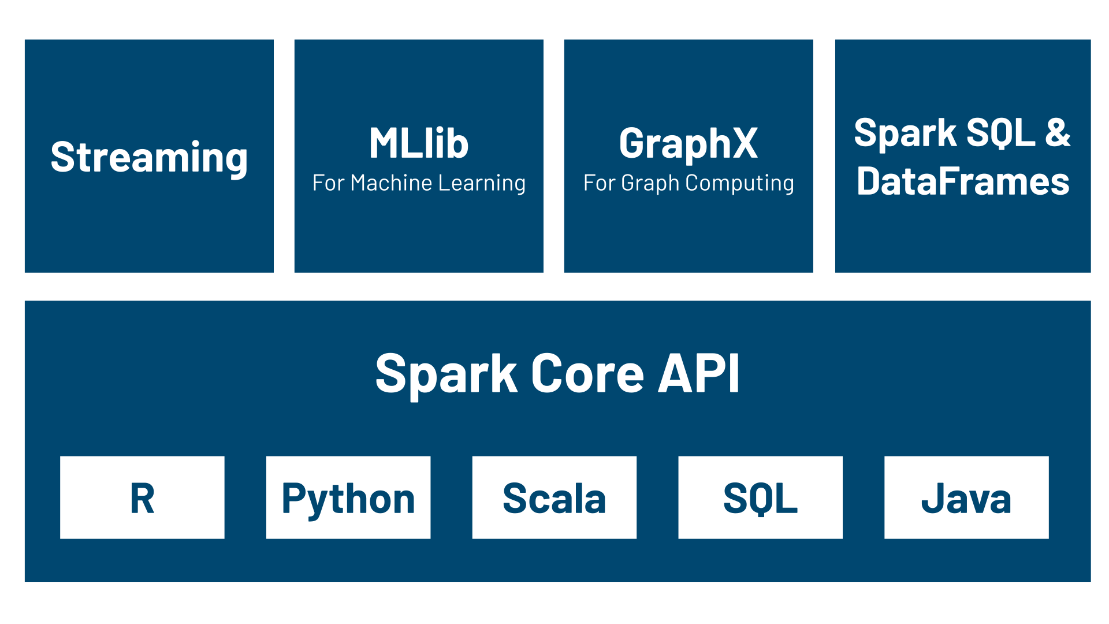
Organizando o funcionamento do motor de processamento, está estruturado os cinco componentes principais do ecossistema Spark:

Ilustração - Exemplificação do ecossistema do Apache Spark

Fonte: Silva, 2021

O Spark Core é a base do processamento distribuído de dados. Agregando todas a fundação para as funcionalidades do Spark, este componente fica a cargo da execução subjacente que orquestra e despacha tarefas e coordena operações de entrada e saída (E/S).

Este mecanismo é associado a um conjunto de funções presentes no Spark, denominado Conjunto de Dados Distribuído Resiliente - resilient distributed dataset -(RDD). Tais funções abstraem o particionamento de dados entre as máquinas do sistema distribuído, e possibilitando a criação de novos RDDs ou modificações dos já existentes,

O componente Spark SQL reúne informações sobre dados para viabilizar que os usuários otimizem o processamento de dados estruturados. Por meio deste módulo, é possível utilizar consultas em padrão SQL para induzir e visualizar operações executadas pelo Spark.

Em complemento, o recurso GraphX é um mecanismo de análise de gráfico de rede e armazenamento de dados. Ou seja, permite a construção interativa, modificação e análise de dados escaláveis estruturados em grafos direcionados com propriedades anexadas a cada vértice e aresta.

É flexível e funciona perfeitamente com gráficos e coleções, com uma variedade de algoritmos de gráfico disponibilizadas por meio de APIs.

O Spark possui como característica ser uma ferramenta que facilita a integração de aplicações, baseadas em sua estrutura, com ferramentas de inteligência artificial.

Desta forma, outro componente crítico da arquitetura Spark, o MLlib, uma biblioteca específica para aprendizado de máquina escalável que discute algoritmos de alta qualidade e alta velocidade, além de ferramentas para seleção de recursos e construção de processos de Machine Learning.

Enfim, o último componente do ecossistema é responsável pelo processamento de alto rendimento e escalonável de dados provenientes de transmissão. O mecanismo de Streaming coleta registros de diferentes fontes de dados e os divide em micro lotes para um fluxo contínuo. Desta maneira, o Spark entrega em sequência estes pequenos pacotes para os executores de processamento, redirecionando o retorno à sistemas de arquivos, bancos de dados e painéis ao vivo.

* 1. Internet Das Coisas

A Internet das Coisas (IoT) configura um ecossistema que abraça a conectividade de objetos físicos, utilizando endereços IP ou outras redes, com o propósito de trocar, armazenar e coletar dados destinados a consumidores e empresas por intermédio de aplicativos de software (CARRION, 2019).

Diagrama

Descrição gerada automaticamente com confiança médiaMais do que uma simples tecnologia, a Internet das Coisas (IoT) representa uma extensão e evolução da própria internet. Essa transformação é impulsionada pelo contínuo avanço tecnológico, especialmente na miniaturização eletrônica e no desenvolvimento de diversos protocolos de comunicação. Como resultado, a IoT está desbravando novas fronteiras, permitindo a interconexão de objetos físicos de maneira cada vez mais abrangente e diversificada (MAGRANI, 2021).

Ilustração - Imagem representando a relação entre o mundo físico e o mundo digital

Fonte: FACCIONI, 2016

Atualmente, a adoção da Internet das Coisas (IoT) está experimentando uma expansão significativa na indústria, proporcionando inúmeras vantagens em diversos setores, com destaque para a indústria de mineração e o agronegócio.

Na indústria de mineração, exemplificada pela operação na mina "Casa de Pedra" da Companhia Siderúrgica Nacional (CSN) em Minas Gerais, a IoT é aplicada para otimizar operações, automatizar processos e gerar informações estratégicas. A utilização de tecnologia GPS de alta precisão e sensores possibilita o rastreamento preciso da frota de veículos, a criação de mapas da topografia da mina, a alocação eficiente de recursos, o monitoramento em tempo real da manutenção e a garantia da segurança dos operadores. Essas soluções baseadas em IoT não apenas conferem vantagens competitivas como também indicam uma tendência crescente na indústria de mineração (ALBERTIN, 2017).

No agronegócio, a aplicação da IoT é crucial para a automação, otimização e aumento da produtividade. Desde a mecanização do campo até a implementação da agricultura de precisão, o uso de sensores, drones e plataformas de análise de dados proporciona ferramentas fundamentais para uma tomada de decisão informada. A combinação estratégica da IoT com outras tecnologias, como big data e inteligência analítica, não só otimiza as decisões no setor agrícola como também impulsiona o aumento da produtividade, visando atender às demandas futuras (ALBERTIN, 2017).

* 1. Data Lakehouse

A concepção do Lakehouse, segundo Armburst (2021), caracteriza-se como um sistema de gestão de dados com ênfase em armazenamento direto e acessível, destacando a priorização de custos reduzidos. Esse modelo incorpora características analíticas tradicionais presentes nos Sistemas de Gerenciamento de Banco de Dados (DBMS), abrangendo transações ACID, versionamento de dados, auditoria, indexação, armazenamento em cache e otimização de consultas. Paralelamente, a arquitetura de Data Lakehouse, conforme estipulado por HLUPIĆ (2022), proporciona acesso a dados provenientes de data lakes e data warehouses por meio de camadas de virtualização.

Há cerca de uma década, os sistemas de warehouse, também conhecidos como sistemas de primeira geração, enfrentaram desafios substanciais relacionados à combinação de computação e armazenamento local, altos custos de provisionamento e gerenciamento de carga de usuário e dados, bem como a incapacidade de lidar eficientemente com conjuntos de dados desestruturados, como vídeo, áudio e documentos de texto. As plataformas de análise de dados de segunda geração resolveram esses problemas ao adotar data lakes, sistemas de armazenamento de baixo custo com uma API de arquivo. Essa abordagem possibilitou o armazenamento econômico de qualquer tipo de dado, mas também introduziu desafios relacionados à qualidade dos dados e governança. (JANSSEN, 2022)

As capacidades centrais de um Lakehouse incluem a capacidade de armazenar dados brutos, semelhante aos data lakes convencionais, enquanto oferece suporte a processos ETL/ELT para aprimorar a qualidade dos dados destinados à análise. Ao contrário dos data lakes tradicionais, que tratam os dados como 'apenas um conjunto de arquivos' semiestruturados, os Lakehouses superam essa limitação, possibilitando recursos de gerenciamento comuns em data warehouses, como transações, reversões e clonagem. Embora seja necessário desenvolver a lógica ETL/ELT para criar conjuntos de dados refinados em um Lakehouse, esse processo envolve menos etapas no geral. Dessa forma, os analistas podem consultar as tabelas de dados brutos com facilidade e alto desempenho, semelhante às plataformas analíticas de primeira geração. (ARMBRUST, 2021)

O Lakehouse viabiliza o armazenamento de dados não estruturados, semi-estruturados e estruturados, passando, em seguida, por uma camada de governança, filtragem e garantias de segurança, proporcionando suporte a transações ACID e ELT (Extract, Load, Transform), além de outras funcionalidades.

Destacando-se como uma vantagem significativa, o Lakehouse disponibiliza os dados por meio de APIs, permitindo que os usuários acessem e interajam programaticamente com os dados. Isso facilita a integração com outras aplicações e sistemas, sendo especialmente aplicável em contextos como machine learning (ML) e business intelligence (BI) (JANSSEN, 2022).

Os sistemas de aprendizado de máquina (ML) demonstram suporte direto aos formatos de data lake, conferindo-lhes uma posição favorável para acessar Lakehouses de maneira eficiente. As APIs dedicadas ao acesso do Lakehouse oferecem otimizações de consulta específicas para cargas de trabalho de ML, permitindo que essas se beneficiem das diversas otimizações disponíveis no ambiente Lakehouse.

Além disso, as APIs possibilitam a execução de chamadas SQL no Lakehouse, viabilizando a utilização dos dados para Business Intelligence (BI). Por meio dessas chamadas SQL, os profissionais de BI podem conduzir consultas e análises nos dados armazenados no Lakehouse, aproveitando as eficientes capacidades de processamento e consulta disponíveis.

* 1. Tolerância a falhas

A confiança em um sistema desempenha um papel crucial na sua aceitação e uso contínuo. A confiança, definida como a segurança que os usuários têm de que o sistema não falhará em condições normais de operação, é essencial para evitar afastar os usuários devido às incertezas e inseguranças. Embora confiança e utilidade sejam aspectos distintos, ambos influenciam a adoção e o uso de um sistema. Sistemas altamente confiáveis, combinados com utilidade significativa, têm mais probabilidade de atrair e reter usuários de maneira efetiva. Por outro lado, sistemas com falhas, mesmo sendo úteis, podem levar os usuários a buscarem alternativas se as falhas prejudicarem a experiência do usuário.

De acordo com Sommerville (2011), o termo 'confiança' engloba atributos como disponibilidade, confiabilidade, segurança e proteção em sistemas. A disponibilidade é crucial para atender às expectativas dos usuários e manter sua confiança, garantindo operação contínua e minimizando interrupções indesejadas. A confiabilidade, por sua vez, é a probabilidade de o sistema fornecer serviços corretos ao longo do tempo. A segurança é a análise da probabilidade de que o sistema cause danos, sendo fundamental para proteger a integridade, privacidade e confidencialidade dos usuários, além de prevenir impactos negativos no ambiente. A proteção, por sua vez, visa prevenir acessos não autorizados e assegurar a continuidade das operações, exigindo a implementação de medidas de segurança adequadas para mitigar riscos de invasões.

Assegurar a confiança de um sistema envolve a adoção de medidas de tolerância a falhas, sendo possível classificar três tipos distintos: Falha de Hardware, Falha de Software e Falha Operacional. Falhas de Hardware podem resultar de falhas estruturais, defeitos de fabricação ou alcance do tempo de vida útil do componente. Já as falhas de software podem ocorrer em diversas etapas do ciclo de vida, desde a especificação até a implementação. Atualmente, com o contínuo progresso de softwares e hardwares, as falhas operacionais tornaram-se mais frequentes, representando situações em que os usuários são incapazes de utilizar o sistema de maneira adequada (SOMMERVILLE, 2011).

É notável que essas falhas muitas vezes estão interconectadas, formando um ciclo interdependente de eventos. Por exemplo, uma falha de hardware pode desencadear uma falha de software, que por sua vez pode resultar em uma falha operacional. Essa inter-relação destaca a importância de uma abordagem abrangente no gerenciamento de falhas para identificar causas subjacentes e adotar medidas adequadas para prevenir, mitigar e corrigir eficazmente essas falhas (SOMMERVILLE, 2011).

Além disso, é crucial reconhecer que as falhas podem desencadear efeitos em cascata, pois as saídas de processos frequentemente dependem das entradas de outros. Para prevenir a propagação dessas falhas, os projetistas implementaram o conceito de "zonas de contenção". Essas zonas são estruturas delimitadas que visam evitar a disseminação de falhas entre diferentes áreas do sistema. A criação dessas barreiras de contenção permite minimizar o impacto global das falhas, limitando sua propagação e facilitando uma resposta mais efetiva e direcionada a cada incidente. Essa abordagem proativa e estratégica é fundamental para garantir a estabilidade e a confiabilidade do sistema como um todo (KOREN & KRISHNA, 2007).

Segundo Koren e Krishna (2007), "toda tolerância a falha é um exercício de exploração e gerenciamento de redundância." A redundância refere-se à prática de possuir recursos adicionais além dos necessários para a execução do sistema. Esse excedente atua como uma forma de cobertura em caso de falha, mantendo o funcionamento contínuo do sistema.

* 1. Data WareHouse

Um Data Warehouse, ou DW, representa uma ferramenta robusta de armazenamento de dados voltada para respaldar a tomada de decisões estratégicas mediante informações corporativas detalhadas. Sua concepção visa ser orientada a assuntos, integrada, não volátil e variável no tempo. Essas características possibilitam ao Data Warehouse oferecer uma visão abrangente e consistente dos dados empresariais, independentemente da sua origem original (GARDNER, 1998).

Essa plataforma abrange uma vasta gama de informações, abarcando dados transacionais e históricos, que são consolidados e organizados para proporcionar uma visão unificada e de fácil acesso. Essa abordagem avançada de integração de dados capacita as organizações a superarem os desafios associados aos bancos de dados múltiplos, distribuídos e heterogêneos, bem como outras fontes de informação (THEODORATOS, 1997; GARDNER, 1998).

Por meio do Data Warehouse, as empresas têm a capacidade de realizar consultas e análises complexas em grande escala, aproveitando a integração dos dados e a estrutura projetada para suportar o processamento eficiente de grandes volumes de informações. Isso proporciona uma compreensão mais profunda dos negócios, a identificação de tendências, o monitoramento de desempenho e o suporte à tomada de decisões informadas em diferentes níveis organizacionais.

O Data Warehouse está intrinsecamente vinculado a bancos de dados que, juntos, representam um amplo volume de dados provenientes de fontes heterogêneas, caracterizados por baixa normalização. A inserção desses dados no Data Warehouse requer uma estruturação adequada. Essa estrutura é concebida para facilitar a visualização e análise dos dados. Dada sua natureza dispendiosa, é crucial manter uma granularidade fina, implicando na subdivisão em unidades menores conhecidas como Data Marts (FERREIRA, 2010). Cada Data Mart armazena informações específicas de setores ou áreas da empresa e é alimentado pelo Data Warehouse, atuando como a fonte primária de dados (THEODORATOS, 1997).

Na figura-6, são identificados quatro setores distintos. O setor inferior refere-se aos “detalhes antigos”, geralmente alocados em armazenamentos alternativos devido ao baixo uso no sistema. Acima, estão os “detalhes atuais”, também conhecidos como atomic ou warehouse. Antes de serem inseridos no setor de warehouse, é comum passarem por um processo de ETL (Extract, Transform, Load).

No patamar acima do warehouse, encontram-se os *datamarts*, com uma normalização mais leve, responsáveis por segmentar o data warehouse de acordo com os setores da empresa, como mencionado anteriormente. No último nível, no topo da imagem, situam-se os dados altamente normalizados, temporários e já formatados para apresentação imediata ao usuário (GARDNER, 1998).

Diagrama

Descrição gerada automaticamenteFonte: INMON, 2005

Ilustração - Imagem representativa da estrutura de um Data Warehouse.

* 1. Análise de Dados

Conforme expresso por Runkler (2020), “A análise de dados permite encontrar informações, estruturas e padrões relevantes, obter novos insights, identificar causas e efeitos, prever desenvolvimentos futuros ou sugerir decisões ideais”. Esse processo fundamenta-se em métodos oriundos de diversos campos, como estatística, aprendizado de máquina, reconhecimento de padrões, teoria de sistemas, pesquisa operacional e inteligência artificial.

A análise de dados ou também conhecido como “mineração de dados” (Russom et al., 2011) é uma área multidisciplinar que envolve a aplicação de várias técnicas, incluindo métodos estatísticos e de aprendizado de máquina (Tsai et al., 2015). Inicialmente, as técnicas estatísticas eram amplamente utilizadas para analisar dados e entender fenômenos, como pesquisas de opinião pública e classificações de programas de TV (Russom et al., 2011). No entanto, com a crescente quantidade de dados e a necessidade de insights mais profundos, métodos de mineração de dados específicos para problemas de dados específicos foram desenvolvidos (Hai et al., 2023). Por exemplo, o algoritmo Apriori foi desenvolvido para o problema de regras de associação (Tsai et al., 2015).

A busca por informações e insights valiosos em conjuntos de dados é denominada Descoberta de Conhecimento em Bancos de Dados (Knowledge Discovery in Databases - KDD), conforme explicado por Runkler (2020). O KDD emprega técnicas e algoritmos para identificar padrões, tendências e relações ocultas nos dados, visando transformar essas descobertas em conhecimento acionável. O processo engloba diversas etapas, desde a seleção e pré-processamento dos dados até a interpretação dos resultados obtidos.

Ao enfrentar o desafio da big data, caracterizado pela volumetria, variedade e velocidade dos dados, conforme observado por Tsai et al. (2015) a análise de dados recorre a diversas técnicas, como algoritmos de metaheurística e computação distribuída. Isso ocorre porque a complexidade computacional dos algoritmos de mineração de dados pode ser alta, especialmente ao lidar com grandes volumes de dados. Portanto, técnicas como algoritmos genéticos e computação distribuída são frequentemente empregadas para melhorar a eficiência da mineração de dados em Data Lakes.

A análise de dados desempenha um papel central na área de Inteligência de Negócios (BI). Conforme Mashingaidze (2017), BI é definida como "um conjunto de estratégias integradas, aplicativos, tecnologias, arquiteturas, processos e metodologias utilizados para coletar, armazenar, recuperar e analisar dados, a fim de apoiar a tomada de decisões". Nesse contexto, gestores podem embasar suas decisões, utilizando a inteligência de negócios como uma ferramenta valiosa para extrair insights do desempenho passado e identificar oportunidades de aprimoramento futuro (HE, 2014).

Ao analisar dados relevantes, os gestores obtêm uma visão abrangente do cenário empresarial, identificam tendências e padrões, e identificam áreas potenciais de otimização e crescimento. Essa abordagem orientada por dados propicia uma tomada de decisão mais embasada e estratégica, contribuindo para o sucesso e a vantagem competitiva da organização.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamenteUma ferramenta destacada no contexto de BI é o OLAP (Processamento Analítico Online). Segundo Chen (2008), o OLAP permite a construção de cubos multidimensionais, proporcionando visualização e análise dos dados a partir de diversas perspectivas. Essa abordagem amplia a compreensão dos dados, facilitando a tomada de decisões com base em informações mais completas.

Ilustração - A esquerda um Dashboard do tipo visual, a direita um do tipo funcional

Fonte: SARIKAYA, Alper, 2018

Outra técnica amplamente empregada na apresentação de dados analíticos são os dashboards. De acordo com Sarikaya et al (2018), os dashboards podem ser classificados em dois tipos: visual e funcional. Os dashboards visuais representam dados estruturados de forma simplificada, enquanto os funcionais possibilitam análises interativas, oferecendo dados em tempo real e atualizados dinamicamente.

* 1. Requisitos Funcionais e Requisitos Não Funcionais

Engenharia de requisitos é uma disciplina dentro da engenharia de software que se concentra na identificação, análise, documentação e gerenciamento dos requisitos de um sistema ou software. Dito isso, segundo Eduardo (2016, p.77) “Requisitos (ou necessidades) de negócio são declarações de mais alto nível de objetivos, metas ou necessidades da organização.”

Os requisitos são segmentados em diversos tipos, sendo os principais utilizados neste trabalho os seguintes: Requisitos Funcionais e Requisitos Não Funcionais.

Há um consenso entre os autores no que diz respeito a definição de requisitos funcionais e não funcionais, “os requisitos funcionais são requisitos que expressam funções ou serviços que um software deve ou pode ser capaz de executar ou fornecer” (CYSNEIROS, 1998, p.22) e “A funcionalidade diz respeito à finalidade a que se propõe o produto de software [...].” (MASIERO, 1996, p.7).

E podemos afirmar referente aos requisitos não funcionais:

Também denominados de requisitos de qualidade, incluem tantas limitações no produto (desempenho, confiabilidade e segurança) como limitações no processo de desenvolvimento (custos, métodos a serem adotados no desenvolvimento e componentes a serem reutilizados). (SANTOS, 1996, p.8).

Nota-se então duas palavras chaves para exemplificar, requisitos funcionais denotam o que é funcionalidade do software e requisitos não funcionais abordam a qualidade para um software.

* 1. Segurança de Dados

Com o rápido avanço da tecnologia, temos presenciado tanto os benefícios quanto os desafios relacionados à segurança de dados. A era digital nos proporcionou uma conectividade sem precedentes, permitindo uma troca instantânea de informações entre pessoas em diferentes partes do mundo. No entanto, esse cenário também trouxe consigo um aumento significativo na quantidade e na importância das informações compartilhadas eletronicamente, o que despertou o interesse de indivíduos mal-intencionados que buscam explorar esses dados para benefício próprio.

A segurança de dados tornou-se uma preocupação crítica em meio a essa realidade. Afinal, é essencial garantir que as informações sejam protegidas contra ameaças como roubo, modificação ou adulteração não autorizados. Segundo a Oracle, a segurança de dados refere-se às medidas de proteções empregadas para proteger os dados contra acesso não aprovado e para preservar a confidencialidade, integridade e disponibilidade dos dados.

Olhando para a computação em nuvem, o cenário se intensifica, uma vez que no processo de virtualização, as empresas estão migrando sua infraestrutura e serviços para o cloud. Mesmo sendo vantajoso não precisar manter suas aplicações, base de dados e infraestrutura dentro da própria organização “deve-se atentar para a segurança, pois quem estará com todas as informações são as empresas que oferecem os serviços de computação em nuvem” (DIAS et al 2012, p. 57-58)

* 1. Criptografia

Desde os primórdios, houve a necessidade do homem em ocultar o conteúdo que era trocado entre os indivíduos, sendo fortemente perceptível em ambientes de guerra, onde a necessidade de trocar informações valiosas acarretou a busca de métodos para evitar que, na interceptação da comunicação, fossem descobertos planos e estratégias. Pode-se afirmam então que:

A criptografia esteve presente desde o sistema de escrita hieroglífica dos egípcios. É tão antiga quanto a própria escrita. Existem evidências, por exemplo, de que os espartanos, na Grécia, já usavam mecanismos primitivos de codificação, que permitiam que mensagens secretas fossem trocadas para comunicar seus planos de guerra, os romanos também utilizavam códigos secretos para comunicar planos de batalha. (GUIMARÃES et al, 2001, pg. 9-10)

Novas formas mais eficientes de criptografias foram surgindo, principalmente com a crescente disseminação da comunicação via internet. A criptografia em dados e navegação se faz essencial para evitar que o usuário seja exposto por conteúdos não confiáveis. Segundo Kessler (2015, pg.2) “Criptografia, então, não apenas protege os dados contra roubo ou alteração, mas também pode ser usada para autenticação do usuário[...]”

* 1. Lei Geral de Proteção de Dados

O advento da Internet das Coisas (IoT) e do Big Data, torna a imersão no mundo virtual mais recorrente. Dentre o grupo de pessoas, a preocupação se centra nas crianças e adolescentes que nasceram no tempo em que as tecnologias já estavam inseridas na rotina do indivíduo, tornando-as mais suscetíveis aos problemas decorrentes da sociedade digital. Segundo Botelho (2020) “à proteção especial desses grupos de pessoas levou o legislador a conferir previsão específica na Lei Geral de Proteção de Dados (LGPD) no que se refere ao tratamento de dados pessoais de crianças e adolescentes.”

Embora publicada em 14 de agosto de 2018, A Lei Geral de Proteção de Dados Pessoais (LGPD) atualizou a legislação brasileira de proteção de dados pessoais, e teve sua vigência em agosto de 2020. Porém, as sanções previstas em Lei passaram a valer apenas em agosto de 2021. Podemos concluir então que:

A LGPD aplica-se ao tratamento de dados pessoais por pessoas singulares ou coletivas de direito privado ou autoridades públicas (artigo 1º). O objetivo desta lei é proteger o direito fundamental à privacidade, por um lado, e o livre desenvolvimento da personalidade de uma pessoa interessada (art. 1º), por outro. A lei aplica-se, portanto, tanto ao setor privado quanto à administração pública. (HOEREN, 2020, p.28)

* 1. Serverless

O acelerado ritmo de inovação dos datacenters, softwares neles contido, e infraestrutura como um todo, vem mudando a maneira como construímos, publicamos, e gerenciamos aplicações e serviços. Antigamente, as aplicações eram compiladas em seu próprio dispositivo físico, e o alto custo para manter, escalar, e manutenir esses equipamentos levaram a um grande marco: virtualização.

No entanto, embora a virtualização em hardware não seja uma solução completa, surgem tecnologias mais leves para abordar seus pontos fundamentais. Uma solução líder nesse quesito são os containers, “uma reformulação de processos no estilo Unix voltada para servidores, com virtualização adicional de namespaces¹” (HENDRICKSON 2016, p.1). Continua então Hendrickson (2016) “Combinados com ferramentas de distribuição como o Docker, os containers permitem que os desenvolvedores criem rapidamente novos serviços sem as lentidões de provisionamento e os custos operacionais das máquinas virtuais”.

Quanto a virtualização de hardware e contêiner baseado em servidor, ambos possuem o ponto comum que é o servidor. Os servidores são fundamentais para suportar aplicações online, porém, a computação em nuvem prevê o fim dos modelos tradicionais. Segundo Castro et al (2019) “Até 2020, 67% da infraestrutura e software de TI corporativos os gastos serão para ofertas baseadas em nuvem”.

Servidores são notoriamente conhecidos pela sua complexidade de configuração e maneabilidade. Dessa forma, é instituído um novo paradigma, o Serverless, que promete a inovação da forma como hoje são tratadas as aplicações. Referente a Serverless, Pacífico et al (2020) descrevem como:

Um novo paradigma crucial para computação em nuvem que permite funções serem implementadas facilmente de modo escalável. Neste paradigma, milhares de funções são executadas em paralelo sem o usuário preocupar com a infraestrutura. (PACÍFICO 2020, p.1)

Essa tecnologia idealiza remover do desenvolvedor a preocupação no que diz respeito a infraestrutura, sendo provida integralmente pelo serviço de nuvem. O AWS Lambda (2014) é conhecido como o primeiro serviço Serverless, assim nomeado (DAVENPORT, 2013). Quanto a Serverless podemos então concluir que:

Diagrama

Descrição gerada automaticamenteA computação Serverless está surgindo como um paradigma novo e atraente para a implantação de aplicações em nuvem, em grande parte devido à recente mudança das arquiteturas de aplicativos empresariais para containers e micros serviços. O uso da computação sem servidor oferece um modelo de pagamento conforme o uso, sem a necessidade de iniciar e parar servidores, e está mais próximo das expectativas originais para o uso da computação em nuvem como uma utilidade. Os desenvolvedores que utilizam a computação sem servidor podem obter economia de custos e escalabilidade sem a necessidade de possuir um alto nível de expertise em computação em nuvem.” (CASTRO et al, 2019)

Ilustração - Exemplo de aplicação Serverless

Fonte: EISMANN, et al. 2020

* 1. Data Lake

Dada as características do Big Data, instituições despreparadas para receber uma quantidade exorbitante de dados tiveram um grande impacto de manter ou atualizar sua governança de dados. Isto, devido a massa de informações gerados constantemente e de forma ilimitada, por sistemas conectados a redes de internet e de comunicações. "Os sistemas Data Lake foram propostos como uma solução para esse problema, fornecendo um repositório sem esquema para dados brutos com uma interface de acesso comum." (STEIN e MORRISON, p. 18, 2014). Uma das características do Data Lake é a habilidade de receber dados de diversas origens diferentes, conseguindo armazená-los em um único repositório, mantendo os dados exatamente como foram ingeridos (sua forma "crua") e preservando as linhagens dos dados dentro de seu pipeline (FARID et al., p. 2089-2092, 2016).

"Os data lakes podem ajudar a resolver o problema persistente de acessibilidade e integração de dados. Usando infraestruturas de big data, as empresas estão começando a reunir volumes crescentes de dados para análises ou simplesmente armazenar para uso futuro indeterminado." (FARID et al., p. 2089-2092, 2016).

Esta realidade possibilita que agora as empresas desfrutem da totalidade de seus dados, não apenas os estruturados, os quais são menos complexos de utilizar, como também os semiestruturados e desestruturados, "[...] os Data Lakes (DLs) foram conceituados como repositórios de big data que armazenam dados brutos e fornecem funcionalidade para integração sob demanda com a ajuda de descrições de metadados (Alrehamy, Walker, 2015; Chessell, Scheepers, 2014; Quix, Hai, Vatov, 2016; Terrizzano, Schwarz, Roth, Colino, 2015)." (STEIN e MORRISON, p. 18, 2014).

Este novo cenário, de um repositório unificado, possibilita que os dados antes considerados descartáveis, possam ser manipulados e estudos em conjunto, para então gerarem informações "[...] a avaliação da qualidade dos dados e a limpeza de grandes volumes de fontes de dados heterogêneas tornam-se tarefas essenciais para revelar o valor da big data.” (NARGESIAN et al., p. 1986-1989, 2019.) "Para a ciência de dados, os data lakes fornecem uma camada de armazenamento conveniente para dados experimentais, tanto a entrada quanto a saída da análise de dados e das tarefas de aprendizado. A criação e uso de dados podem ser feitos de forma autônoma sem coordenação com outros programas ou analistas. [...] o armazenamento compartilhado de um data lake, juntamente com uma estrutura computacional (normalmente distribuída), fornece a infraestrutura rudimentar necessária para compartilhamento e reutilização de conjuntos de dados massivos."(MILOSLAVSKAYA e TOLSTOY, p. 300-305, 2016).

O Data Lake é um esquema de armazenamento, pensado na possibilidade de obter maior economia em lidar com dados de estruturas imprevisíveis e diferentes fontes, com uma forma de armazenamento em camadas para maior flexível, e versatilidade de consumo para os usuários ou sistemas finais. "Pode ser imaginado como um grande pool de dados para reunir todos os dados históricos acumulados e novos dados (estruturados, não estruturados e semiestruturados, mais binários de sensores, dispositivos e assim por diante) quase em tempo real em um único local, no qual o esquema e os requisitos de dados não são definidos até que os dados sejam consultados («schema-on-read» é usado) (ALEKSANDER, 2019).

* 1. Limpeza dos dados

Dado o cenário em que as empresas estão produzindo cada vez mais dados, e compreendendo a importante destes para a etapa de tomada de decisões, é necessário ter a certeza de que as informações consideradas no processo, estão corretas e condizentes com a realidade "Qualquer subconjunto de dados que não esteja em conformidade com as regras definidas é considerado errôneo, o que também é referido como uma violação." (HAI, GEISLER e QUIX, p. 2097-2100, 2016). Entendendo está complicação, surgiu a necessidade da etapa crucial de limpeza dos dados "refere-se ao processo de detecção e correção de erros nos dados. Vários tipos de regras de qualidade de dados foram propostos para esse objetivo e grandes esforços foram feitos para melhorar a eficácia e eficiência de seus algoritmos de limpeza [...]" (ILYAS et al., p. 281-393, 2015).

É inegável que dados despadronizados, que não conversam corretamente entre si, impactam diretamente nos resultados de seu uso:

"[...] a qualidade desses dados é comprometida por fontes de ruído difíceis de remover no ciclo de vida dos dados: a imprecisão dos extratores na aquisição de dados assistida por computador pode levar a valores ausentes, a heterogeneidade nos formatos na integração de dados de várias fontes pode introduzir registros duplicados e erros humanos na entrada de dados podem violar as restrições de integridade declaradas. Esses problemas comprometem as tarefas de consulta e análise, com prejuízos de bilhões de dólares.". (Eckerson, 2002)

Dito isso, o outro ponto importante é como este processo deve ser feito, "limpeza de dados tem três componentes: auditoria de dados para encontrar discrepâncias, escolha de transformações para corrigi-las e aplicação das transformações no conjunto de dados." (CHU, ILYAS e PAPOTTI, p. 458-469, 2013), ou seja, não é uma etapa trivial que possa ser ignorada. Isto, além de estar alerta a respeito da complexidade deste processo, "Os dados geralmente têm muitos casos especiais difíceis de encontrar, portanto, esse processo de auditoria e transformação deve ser repetido até que a “qualidade dos dados” seja boa o suficiente. Essa abordagem tem dois problemas:

* Falta de interatividade: a transformação normalmente é feita como um processo em lote, operando em todo o conjunto de dados sem qualquer feedback. Isso leva a atrasos longos e frustrantes durante os quais os usuários não têm ideia se uma transformação é eficaz.
* Necessidade de muito esforço do usuário: tanto a transformação quanto a detecção de discrepâncias exigem um esforço significativo do usuário, tornando cada etapa do processo de limpeza penosa e propensa a erros." (CHU, ILYAS e PAPOTTI, p. 458-469, 2013).

"Rahm e Do (Rahm, H. Do, 2000) classificam os diferentes tipos de erros que podem ocorrer em um processo Extract-Transform-Load (ETL), e levantam as ferramentas disponíveis para limpeza de dados em um processo ETL; alguns enfocam o efeito de dados incompletos na resposta a consultas (Grahne, 1991) e o uso de um procedimento Chase para lidar com dados incompletos (Spezzano, Molinaro, 2012); Hellerstein (Hellerstein, 2008) enfoca a limpeza de dados quantitativos, como números inteiros e pontos flutuantes, usando principalmente técnicas estatísticas de detecção de outliers. Bertossi (Bertossi, 2011) fornece resultados de complexidade para reparar dados inconsistentes e executar consultas consistentes em dados consistentes; Fan e Geerts (Geerts, Fan, 2012) discutem o uso de regras de qualidade de dados na consistência de dados, atualização de dados e integridade de dados, como diferentes aspectos de problemas de qualidade de dados podem interagir; e Dasu e Johnson (Johnson, Dasu, 2003) resumem como as técnicas de mineração de dados exploratórios podem ser integradas ao gerenciamento de qualidade de dados." (HAI, GEISLER e QUIX, p. 2097-2100, 2016). O processo de limpeza de dados ou "data clean", como discutido entre os estudiosos da área, é crucial para considerá-lo na camada final da aplicação.

* 1. Qualidade dos dados

A ausência de qualidade dos dados pode acarretar diversos impactos em uma empresa. Entre esses impactos, podemos citar a insatisfação dos clientes, o aumento dos custos operacionais, a tomada de decisões menos eficaz e a redução da capacidade de elaborar e executar estratégias. De maneira mais sutil, é possível observar que a baixa qualidade dos dados afeta a moral dos funcionários, gera desconfiança dentro da organização e dificulta a harmonização dos objetivos da empresa (REDMAN, 1998). A qualidade dos dados é um dos fatores decisivos para a criação de um data Warehouse (SHENG & MYKYTYN JR, 2002).

A exatidão dos dados é de extrema importância para atender aos requisitos de qualidade estabelecidos. No entanto, é importante ressaltar que a qualidade dos dados não se limita somente à sua exatidão. Ela está intrinsecamente ligada à sua adequação para um uso específico, ou seja, os dados devem ser apropriados para uma determinada aplicação ou finalidade. Isso implica que o conceito de qualidade dos dados é relativo e varia de acordo com as necessidades e contextos individuais (SHENG & MYKYTYN JR, 2002).

Além da exatidão, outros aspectos como completude, consistência e relevância desempenham um papel crucial na determinação da qualidade dos dados. A completude refere-se à presença de todos os dados necessários e relevantes para uma determinada análise ou processo. A consistência diz respeito à coerência e uniformidade dos dados em diferentes fontes e momentos. Já a relevância está relacionada à pertinência dos dados em relação aos objetivos e requisitos estabelecidos.

Data quality tem 2 principais métodos sendo eles avaliação (Assessment) e melhoria (Improvement). Avaliação, que mede a qualidade das coleções de dados em relação a dimensões relevantes de qualidade. O termo "medição" é usado para abordar a questão de quantificar o valor das dimensões de qualidade dos dados. O termo "avaliação" é utilizado quando essas medições são comparadas a valores de referência, possibilitando a identificação da qualidade dos dados. Utiliza técnicas de análise de dados, análise de requisitos de qualidade dos dados, identificação de área críticas, modelagem de processos e avaliação da qualidade (BATINI, 2009).

O processo de melhoria envolve a seleção criteriosa de etapas, estratégias e técnicas que visam atingir novos patamares de qualidade dos dados. Essa fase é fundamental para identificar as áreas que necessitam de aprimoramento, implementar ações corretivas e realizar ajustes necessários nos processos e sistemas envolvidos na gestão dos dados. Entre suas técnicas temos, avaliação dos custos, atribuição de responsabilidades dos processos, identificação das causas de erros, entre outros (BATINI, 2009).

* 1. Arquitetura e Camadas do Data Lake

Parte essencial de uma Data Lake é sua arquitetura, que ilustra o processo de “transição” dos dados em uma jornada de transformação para efetivo consumo de forma analítica. É comum a prática da incorporação de arquiteturas de referência, as quais oferecem recomendações relativas às "Zona/Camada" destinadas ao alojamento de dados. Essas arquiteturas podem ser mutáveis, visto que não existe uma única e “exata” podendo ser diferente a depender da proposta de construção do Data Lake, que pode ser distinta de organização e o que se busca com ele atingir e resolver.

Algumas dessas arquitetura que são mais difundidas, são: Arquitetura em Lagoa (Pond Architecture); e Arquitetura em Zona (Zone Architectures).

Inmon traz o Data Lake como um conjunto de “lagoas” de dados, podendo ser entendida como uma subdivisão do próprio Data Lake. Cada lago é associado a um sistema específico de armazenamento, processamento, condicionamento e análise, Diagrama

Descrição gerada automaticamenteInmon atribui a arquitetura de cinco níveis, conforme a ilustração seguinte:

Ilustração - Arquitetura em Lagoa de Inmon

Fonte: SAWADOGO, DARMONT, 2021

Linha do tempo

Descrição gerada automaticamente com confiança baixaNa arquitetura em zonas, tem-se princípios semelhantes de divisão, operando com um padrão de quatro camadas: Transitório (Transient), Bruto (Raw), Confiável (Trusted) e Refinado (Refined). A quantidade de camadas não é fixa, podendo variar para a qual melhor se adapte aos objetivos da organização.

Ilustração - Arquitetura em Zonas

Fonte: Datalakers Tecnologia, 2021

* + 1. Zona de Dados Brutos (Raw Zone/Landing Zone)

Umas das áreas cruciais de um data lake, é a camada de “chegada” dos dados, ou zona de dados brutos. Geralmente é a primeira camada da arquitetura (certos casos as arquiteturas podem possuir camadas transitórias), nessa área são armazenados todos os dados brutos, sem processamento ou com transformações mínimas. Esses dados podem ser oriundos de diversas fontes, como sensores, bancos de dados, vídeos, imagens, redes sociais, aplicativos, e assim por diante.

A zona de dados brutos é a pedra angular do data lake, garantindo a preservação dos dados originais, crucial para auditorias, rastreabilidade e conformidade regulatória. Possui-se também, nessa área, flexibilidade de análise, possibilitando a aplicação de diferentes estruturas de dados conforme necessário, provendo ao usuário fácil adaptação do banco de dados, sem se limitar a uma única estrutura.

Pasupuleti, acrescenta também que:

Esta zona permite reduzir os prazos de integração. No modelo tradicional de data warehouse, as informações são consumidas após terem sido enriquecidas, agregadas e formatadas para atender às necessidades específicas da aplicação. Você só pode consumir os dados pré-processados e agregados de um Data Warehouse. O Data Lake é arquitetado de forma diferente para ser modular, consistindo em várias zonas distintas. Essas zonas oferecem múltiplas oportunidades de consumo, resultando em flexibilidade para o consumidor. (PASUPULETI, Pradeep. Pg 27.)

Dessa forma, aplicação que necessitam de dados em estados de transformação mínimo, podem acessá-los direto da zona bruta, pulando assim as subsequentes etapas e se mostrando um ganho em tempo ao acesso da informação, sendo significantes e criando mais valor para o consumidor.

Os dados chegam à camada bruta por meio de processos de ingestão, que garantem a disponibilidades dos mesmo para análises e tomadas de decisões. A ingestão de dados é o processo que coleta informações em sua origem, processa e armazena em um destino, possibilitando a disponibilização para uso geral, conforme Armoogum e Li (2019, apud Casagrande).

A ingestão pode ocorrer em diferentes modos e formas, como exemplo, na arquitetura lambda, na camada de lote (batch layer), os dados podem ser armazenados utilizando Sqoop para dentro de um HDFS, ou propriamente a utilização do Apache Hadoop ingerindo diretamente no HDFS.

HDFS (*Hadoop Distributed File System*) é o sistema de arquivos distribuídos Hadoop, otimizado para leitura e escrita de dados. Aqui, os dados são divididos por MapReduce, e replicado através dos nós de servidores. O Sqoop é uma ferramenta de código aberto desenvolvida pela Apache Software Foundation que é usada para transferir dados entre bancos de dados relacionais (como MySQL, Oracle, SQL Server) e o HDFS, de forma eficiente e automatizada.

Já para os dados não estruturados, o Hadoop possui um utilitário de linha de comando “hadoop fs”, que permite interagir com o HDFS. Com esse utilitário, é possível fazer a ingestão de dados por meio da replicação do arquivo diretamente no sistema HDFS.

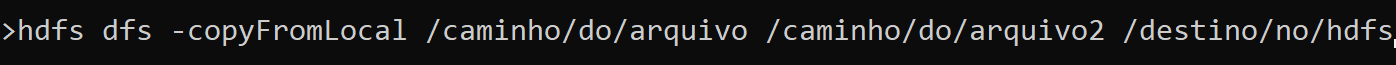
Fonte: Autoria própria

Ilustração - Linha de comando Hadoop para replicação de arquivos não-relacionais

* + 1. Zona de Confiança (Trusted Zone)

A zona trusted representa um ambiente onde os usuários desfrutam de total controle sobre os dados, podendo adaptá-los conforme suas necessidades, os resultados dessa adaptação são armazenados. Existem duas principais abordagens para o processamento de dados: o processamento em lote, ideal para grandes volumes de dados, mas menos eficaz para dados em alta velocidade. Para solucionar essa limitação, empregamos o processamento em tempo real. Ambos os métodos, em lote e em tempo real, podem ser integrados em um data lake.

Nesta zona, os usuários também têm a capacidade de realizar processos de data clean e assegurar a data quality, garantindo a precisão e confiabilidade das informações, conforme ressaltado por Sawadogo e Darmont, que afirmam que "A zona confiável é onde os dados são transferidos após serem padronizados e limpos.". Esta etapa é crucial para que os dados estejam em condições ótimas para análise.

Além disso, é viável efetuar transformações nos dados para atender às exigências do usuário. Assim, funcionalidades como seleção, projeção, junção e agregação são amplamente empregadas, facilitando o consumo eficiente desses dados para análises posteriores. Dessa forma, a "zona confiável" não apenas oferece controle e adaptabilidade, mas também garante a qualidade e utilidade dos dados para análises mais precisas e confiáveis.

* + 1. Zona de Análise de Dados (Data Analytics)

A camada de Data Analytics em um Data Lake é de extrema importância, pois é nesse estágio que os dados são transformados em informações valiosas e insights que podem orientar decisões estratégicas. Como mencionado por Hai et al. (2023), a análise de dados é responsável por encontrar os padrões, regras e informações ocultas nos dados, transformando-os em "nuggets de ouro", ou seja, em conhecimento valioso. Neste contexto, a análise de dados é frequentemente referida como "mineração de dados" (Russom et al., 2011) e é fundamental para extrair valor dos grandes conjuntos de dados armazenados em um Data Lake.

A etapa de entrada de dados envolve a coleta de informações brutas, enquanto a análise de dados tem como objetivo identificar padrões e relacionamentos nos conjuntos de dados. Além disso, a construção e atualização de regras são operações iterativas que refinam as informações extraídas dos dados brutos. Uma outra abordagem consiste em utilizar dados brutos e combiná-los com informações já processadas, bem como resultados de análises anteriores - consultas que são executadas e registradas na camada de Data Analytics. Ao realizar novas consultas complexas sobre esses dados, essas informações são armazenadas na camada de Data Analytics, tornando-as prontamente disponíveis para consultas rápidas ou para uso futuro, combinadas com outros conjuntos de dados.

Dentro da análise de dados em Data Lakes, existem diferentes tipos de problemas que podem ser abordados. Por exemplo, a clusterização é um problema bem conhecido que envolve agrupar dados não rotulados em diferentes grupos, como mencionado por Tsai et al. (2015). Isso pode ser útil para entender novos padrões nos dados de entrada. Por outro lado, a classificação é o oposto da clusterização, onde os dados são rotulados com base em um conjunto de dados de treinamento conhecido. Métodos como árvores de decisão, classificação bayesiana ingênua e máquina de vetores de suporte são amplamente utilizados para esse fim (Tsai et al., 2015).

Além da clusterização e classificação, também existem problemas de mineração de regras de associação e padrões sequenciais, que se concentram em encontrar relacionamentos entre os dados de entrada (Tsai et al., 2015). O algoritmo Apriori é um exemplo de método amplamente utilizado para descobrir regras de associação em grandes conjuntos de dados.

A camada de Data Analytics em um Data Lake desempenha um papel crucial na transformação de dados em conhecimento valioso. Utiliza uma variedade de técnicas, incluindo métodos estatísticos, aprendizado de máquina e algoritmos de mineração de dados, para analisar e extrair informações dos dados armazenados. Dito isso, a análise de dados desempenha um papel fundamental na extração de valor de um Data Lake.

* 1. UML

A Linguagem de Modelagem Unificada (UML) é uma notação gráfica amplamente reconhecida e adotada, fundamental para especificar e documentar sistemas de informação (Fuentes-Fernández & Vallecillo-Moreno, 2004). Ela oferece suporte em todas as fases do desenvolvimento de software, desde a análise de requisitos até a implantação (Fuentes-Fernández & Vallecillo-Moreno, 2004; Khawar & Cary, 2001).

A UML é uma notação de propósito geral, mas pode ser estendida para se adequar a domínios específicos, utilizando mecanismos de extensão, como os Perfis UML (Fuentes-Fernández & Vallecillo-Moreno, 2004). Isso é particularmente relevante quando se trata de modelagem em domínios complexos, como a arquitetura J2EE (Khawar & Cary, 2001).

A UML teve suas raízes em notações anteriores, como as de Booch, Rumbaugh e Jacobson, e foi unificada no final dos anos 1990 (Khawar & Cary, 2001). Ela continua a evoluir sob a supervisão da OMG, com extensões recentes para modelagem de dados, aplicações web e mapeamento de construções J2EE (Khawar & Cary, 2001).

A UML é projetada para capturar detalhes significativos de um sistema, facilitando a compreensão do problema, o desenvolvimento da arquitetura da solução e a identificação da implementação escolhida (Khawar & Cary, 2001). A notação UML não apenas descreve os blocos de construção básicos, mas também permite expressar relacionamentos complexos entre eles (Khawar & Cary, 2001).

Para representar esses aspectos, a UML oferece diversos tipos de diagramas, cada um com sua área de enfoque específica, chamados de visões (Khawar & Cary, 2001). Os principais tipos de diagramas UML incluem diagramas de caso de uso, diagramas de classe, diagramas de objeto, diagramas de máquina de estados, diagramas de atividade, diagramas de interação, diagramas de componentes e diagramas de implantação (Khawar & Cary, 2001).

A UML é uma linguagem de modelagem essencial para o desenvolvimento de software, com mecanismos de extensão que a tornam flexível para adaptação a diferentes domínios (Fuentes-Fernández & Vallecillo-Moreno, 2004; Khawar & Cary, 2001).

* + 1. Diagrama de Caso de Uso

O diagrama de caso de uso UML é uma ferramenta amplamente utilizada na engenharia de software, sendo fundamental para modelar a funcionalidade de um sistema em termos de atores, casos de uso e suas relações (ARIFIN & SIAHAAN, 2020). Durante a análise de requisitos, esse diagrama auxilia na identificação de atores e na definição do comportamento do sistema por meio dos casos de uso (GEPPERT & SCHMID, 2002).

Diagrama

Descrição gerada automaticamenteFonte: LUCIDCHART, S/D

Ilustração - Exemplo de Diagrama de Caso de Uso UML

Como exemplifica a Figura 1 os casos de uso representam as funcionalidades ou serviços oferecidos por um sistema, sendo essenciais para definir o escopo do projeto (JILANI et al., 2011). São uma parte crucial do processo de modelagem de sistemas orientados a objetos e facilitam a descrição clara do comportamento do sistema para todas as partes envolvidas (ALERYANI, 2016).

Além de sua representação gráfica, o diagrama de caso de uso desempenha um papel importante na modelagem de sistemas orientados a objetos, identificando atores e definindo o comportamento do sistema (ARIFIN & SIAHAAN, 2020). É amplamente aceito na indústria de desenvolvimento de software (GEPPERT & SCHMID, 2002).

Outra aplicação do diagrama de caso de uso é sua adaptação para a Engenharia de Linhas de Produto de Software (SPL), possibilitando a modelagem de características comuns e variáveis em sistemas complexos (ARIFIN & SIAHAAN, 2020). Isso é crucial para o desenvolvimento de sistemas que lidam com diferentes cenários e variações de funcionalidade (JILANI et al., 2011).

Para avaliar a similaridade entre diagramas de caso de uso, é essencial considerar tanto a similaridade estrutural quanto a semântica (ARIFIN & SIAHAAN, 2020). A similaridade estrutural envolve as relações entre casos de uso, como extensões e inclusões (GEPPERT & SCHMID, 2002). Já a similaridade semântica leva em conta informações semânticas dos diagramas (JILANI et al., 2011).

O diagrama de caso de uso UML é uma ferramenta crucial na engenharia de software, permitindo a modelagem clara da funcionalidade de um sistema, identificação de atores e adaptação para sistemas complexos com características comuns e variáveis (ALERYANI, 2016). A avaliação da similaridade entre diagramas de caso de uso envolve considerar aspectos estruturais e semânticos (ARIFIN & SIAHAAN, 2020).

* + 1. Diagrama de Sequência

O Diagrama de Sequência UML (Unified Modeling Language) é uma ferramenta essencial no desenvolvimento de software, projetada para especificar sistemas orientados a objetos. O UML define duas formas principais de descrever interações entre objetos: diagramas de sequência e diagramas de colaboração (LI et al., 2004; TSIOLAKIS, 2001).

Os diagramas de sequência enfocam a sequência explícita de interações entre objetos durante a execução de uma operação ou resultado desejado. Em contraste com os diagramas de colaboração, não explicitam as relações entre objetos que desempenham diferentes papéis na interação (TSIOLAKIS, 2001). A representação ocorre ao longo de um eixo vertical, onde o tempo progride de cima para baixo, com os objetos dispostos ao longo do eixo horizontal, comunicando-se por meio de mensagens (LI et al., 2004).

A formalização da semântica dos diagramas de sequência é um tópico relevante. Li et al. (2004) apresentam uma formalização detalhada, destacando a importância dessa formalização no design do sistema. A formalização, representada por metamodelos, descreve a estrutura abstrata das soluções, oferecendo um arcabouço para a compreensão dos padrões de interação (LI et al., 2004).

A abordagem de Aredo (2000) para a semântica dos diagramas de sequência, utilizando o ambiente PVS (Specification and Verification System), propõe uma modelagem formal das noções básicas, como ação, evento, mensagem, objeto e operação (AREDO, 2000). Essa formalização estabelece propriedades do sistema, considerando invariáveis e restrições (AREDO, 2000).

A integração da informação de modelos nos diagramas de sequência UML é abordada por Tsiolakis (2001), que destaca a importância da consistência e análise semântica desses diagramas. A autora propõe uma análise semântica e verificação de consistência, utilizando o PVS para a criação de modelos mais robustos (TSIOLAKIS, 2001). Essa abordagem é necessária para validar o comportamento do sistema modelado e garantir que ele atenda aos requisitos especificados durante a análise e design (TSIOLAKIS, 2001).

Os diagramas de sequência UML desempenham um papel importante no desenvolvimento de software, oferecendo uma representação gráfica detalhada do comportamento dinâmico dos sistemas. A formalização da semântica, como proposta por Li et al. (2004) e Aredo (2000), juntamente com a análise semântica e a verificação de consistência de Tsiolakis (2001), contribuem para a robustez e confiabilidade desses diagramas, tornando-os ferramentas essenciais na engenharia de software.

1. Metodologia

A composição de um Data Lake robusto, que seja eficaz e resiliente em todos os seus processos, demanda um bom planejamento e investimentos consideráveis. Ao elencar todos os mecanismos passiveis de serem anexados na arquitetura de processamento de dados e sistemas adicionais anexados a um Data Lake, é comumente necessário inúmeros serviços e ferramentas funcionando de forma conjunta.

Num cenário concreto, em que o tratamento e disponibilização das informações trazem valor agregado para determinadas aplicações externas é fundamental a utilização de metodologias adicionais com intuito de tornar o fluxo de dados tolerante a falhas.

Um projeto dessa escala possui inúmeros engenheiros e cientistas de dados construindo novos meios de ingestão, processamento e visualização de informações no mesmo ambiente. Desta forma, é indispensável a padronização do uso de repositórios de código para cada escopo interno no Data Lake. Assim, cada novo recurso a ser implementado, correção a ser feita, e ajuste emergencial é elaborado de forma individual, evitando conflito com as alterações realizadas por outros programadores.

Após a composição do código, é necessário fazer a manutenção dos artefatos utilizados no fluxo de processamento para consolidar a alteração arquitetada pelo programador. Tal processo é o mais propicio de se ocorrer falhas, já que mudanças são realizadas na infraestrutura do ambiente em que os usuários finais interagem com o produto. Para isso, estabelece-se a prática de integração e entrega contínuas, Continuous Intergration/Continuous Delivery, que visa tornar a integração de artefatos mais eficiente por meio de esteiras e testes automatizados. Esta prática implica o monitoramento e automação de todo o ciclo de vida das manutenções, e consequentemente, reduzindo as possíveis falhas humanas na execução deste tipo de demanda.

Além do aprimoramento do Data Lake por meio da adição de novos recursos e correção de problemas que possam surgir, a arquitetura deve ser capaz de monitorar o fluxo de dados, permitindo o acompanhamento do trajeto e a validação da integridade em cada etapa. De tal forma que cada eventual falha possa ser identificada o mais rápido possível, agilizando a atuação sobre o problema.

Este processo é extremamente crucial, pois qualquer inconsistência seja em qual for o procedimento do produto pode impactar a informação apresentada ao usuário final, diminuindo permanentemente a confiabilidade sobre o produto.

Tendo em vista o quão complexo pode ser tornar a arquitetura de um possível Data Lake, este trabalho de conclusão de curso se propõe a demonstrar a construção de um fluxo de processamento de dados com as etapas fundamentais de ingestão, processamento e disponibilização de dados. O intuito é não apenas apresentar um mero esboço teórico, mas sim demonstrar de forma prática e aplicada como estabelecer um ambiente eficiente de gerenciamento e uso de dados no contexto de um Data Lake.

O escopo de arquitetura proposto por este trabalho é limitado pelos custos associados à construção de uma solução abrangente utilizando serviços disponíveis pela Amazon Web Services. Desta forma, serão utilizados serviços e procedimentos aplicáveis à cenários de qualquer escala, apropriados a demonstrar a capacidade de escalabilidade de um Data Lake construído em ambiente Cloud, em suas configurações mais básicas compatíveis à proposta demonstrativa das capacidades de um Data Lake.

Isto posto, o fluxo de processamento de dados concebido está disposto abaixo:

Tela de jogo de vídeo game

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Arquitetura Macro

* 1. Ingestão

Um dos diferenciais do Data Lake quando comparado a outros métodos de armazenamento de dados é a variedade dos tipos de dados que podem compor o fluxo de processamento de informações, possibilitando a composição de aplicações que desfrutem de dados provenientes de inúmeras fontes.

Para demonstrar tal capacidade, a arquitetura elaborada propõe a coleta de dados de duas fontes de dados distintas.

Periodicamente consulta-se uma *API* externa, coletando informações referentes ao clima em pontos dispersos pelo Brasil. Dessa forma, armazenando um histórico da mudança de temperatura, umidade, e qualidade do ar ao longo do tempo em cada localidade.

Agregando a esta ingestão de dados ao ambiente, um banco de dados relacional atua com outra fonte de dados. Este tipo de sistema de gerenciamento de banco de dados é amplamente utilizado no quesito de armazenamento de informações transacionais e históricas, e por isso é geralmente a principal origem de dados em Data Lake dos mais variados escopos.

Assim sendo, sucede-se a migração em lote e transacional de múltiplos esquemas de tabelas que arquivam informações ambientais e sociais sobre o território brasileiro, visando possuir relações entre as informações de modo que seja construída visualizações agregadas desses dados.

* 1. Armazenamento

O ponto principal na metodologia de um Data Lake descende da sua técnica de armazenamento de informações pois, por meio de um sistema de armazenamento de objetos, permite a retenção de dados de variadas fontes em um repositório centralizado e escalável capaz de distribuir estas informações entre camadas apartadas, correspondentes aos objetivos impostos pela arquitetura do projeto.

Assim sendo, baseando nos dados coletados pelo processo inicial presente no escopo do Data Lake, está imposta a segregação das informações em três camadas distintas.

A primeira camada, denominada “Crua”, é responsável pelo armazenamento de todos os dados provenientes da etapa de ingestão. Neste local, as informações são retidas em seus formatos originais, sem qualquer processamento ou transformação significativa. Isso permite preservar a integridade original dos dados, independentemente do seu tamanho, formato ou tipo. A ideia é reter toda a informação disponível, mesmo que não se saiba previamente como ela será utilizada ou se será relevante para análises futuras.

Esta camada é fundamental na arquitetura de um Data Lake pois oferece flexibilidade, possibilitando o processamento e a transformação dos dados conforme a necessidade surgir. Ao preservar os dados em seu estado bruto, os setores responsáveis têm a liberdade de aplicar diferentes modelos de processamento, análises e interpretações posteriormente, adaptando-os aos requisitos específicos de cada situação ou análise.

Seguidamente, há a imposição da camada “Tratada” na qual os dados brutos da camada inicial passam por processos de limpeza, transformação e estruturação para se tornarem mais utilizáveis, compreensíveis e aptos para análises e aplicações específicas.

Nessa etapa, os dados coletados previamente, que estão na sua forma original, são refinados e preparados para atender às necessidades dos usuários finais, aplicativos ou sistemas que irão consumi-los. Tornando os dados mais acessíveis, compreensíveis e aptos para análises avançadas facilitando a extração de valor a partir do vasto volume de informações disponíveis no ambiente.

Culminando na última separação de armazenamento de dados, a camada “Analítica” corresponde a consolidação dos resultados de processamentos analíticos dos dados previamente tratados, visando materializar informações valiosas para suportar a visualização desses conhecimentos.

Essa camada envolve a aplicação de diversas técnicas e ferramentas analíticas para explorar e compreender os dados de maneira mais profunda, estruturando os dados de forma a fornecer uma visão mais clara das relações, significados e conexões entre diferentes conjuntos de dados. Assim sendo, facilitando que consumidores de variados contextos entendam e sejam capazes de usar os mesmos conjuntos de dados de maneira consistente. Isso promove a reutilização e o compartilhamento de dados entre diferentes setores e evitando o reprocessamento constante dessas visualizações.

Ademais, todas as categorias de dados impõem mecanismos extras no tocante ao consumo dessas informações. É de suma importância que durante a manipulação e visualização de dados exista moderações acerca da restrição desses dados ao público geral.

Portanto, em cada camada há restringimentos sobre o acesso as informações armazenadas, impondo controle na leitura, manipulação e escritas de dados dependendo do setor e nível de acesso do usuário.

* 1. Processamento

O processamento de dados em um Data Lake é uma etapa crucial que envolve uma série de operações sequenciais relacionadas as camadas estipuladas na arquitetura do Data Lake.

Em todas as etapas de manipulação de dados entre as camadas, deve-se respeitar a estrutura e tipo de arquivo estabelecidos previamente ao estipular o processamento de dados, e dentro desse contexto, aspectos fundamentais como regras de negócio, limpeza de dados, conversão de formatos e agregação desempenham papéis vitais no processo.

As regras de negócio são diretrizes que definem como os dados devem ser tratados e processados para atender aos objetivos específicos da arquitetura. Elas determinam critérios como validações, restrições, cálculos e procedimentos que são aplicados aos dados para garantir sua precisão, consistência e conformidade com a veracidade da informação. A aplicação rigorosa dessas regras no processamento de dados no Data Lake é essencial para garantir a qualidade e a integridade dos dados utilizados em análises e tomadas de decisão.

A limpeza de dados é outra fase crucial no processamento de dados. Envolve a identificação, correção e remoção de inconsistências, erros, valores ausentes, duplicatas e informações irrelevantes nos conjuntos de dados. Essa etapa é essencial para garantir a confiabilidade dos dados, reduzindo o risco de distorções nos resultados das análises. A limpeza de dados também está associada à padronização de formatos, normalização de dados e tratamento de outliers, contribuindo para uma base de dados mais consistente e confiável.

A conversão de formatos é realizada para garantir a compatibilidade e a uniformidade dos dados dentro do Data Lake. Isso inclui a conversão de tipos de dados, transformação de formatos de arquivos, adaptação de dados para diferentes sistemas e ambientes. Ao padronizar os formatos, os dados tornam-se mais acessíveis e interoperáveis, facilitando a integração e o compartilhamento entre diferentes fontes e sistemas.

E por fim, a agregação de dados consiste na combinação e consolidação de informações de diversas fontes para criar conjuntos de dados maiores e mais abrangentes. Isso pode envolver a sumarização de dados, cálculos de métricas, agrupamentos e a criação de visões mais amplas e estratégicas dos dados. A agregação permite analisar e compreender tendências e padrões em níveis mais abrangentes, proporcionando novas compreensão acerca dos dados.

* 1. Visualização

Culminando os processos de armazenamento, tratamento e análise de dados, a visualização de dados consolida todos as etapas anteriores ao comunicar as informações contidas em cada uma das camadas do Data Lake. Este processo envolve a representação gráfica ou visual dos dados de forma compreensível e intuitiva, transformando conjuntos complexos de informações em representações visuais fáceis de serem interpretadas.

As ferramentas de visualização permitem a criação de gráficos, dashboards, mapas, diagramas e outras representações visuais interativas que facilitam a análise e a interpretação dos dados. Além de possibilitar que os usuários executem consultas SQL em conjuntos de dados diretamente armazenados em qualquer camada da arquitetura.

Concluindo a disponibilização dos dados para usuários finais, outra forma de compartilhamento dos dados é através de uma *API*, Interface de Programação de Aplicações, no qual os dados presentes no Data Lake podem ser acessados por outros sistemas ou aplicativos de maneira padronizada e eficiente.

No ambiente de um Data Lake, onde uma vasta quantidade de dados de diferentes fontes e formatos está disponível, a visualização desempenha um papel crucial na apresentação e no entendimento dos padrões, tendências e relações existentes entre os dados.

Além disso, a visualização de dados também é essencial para comunicar informações de maneira clara e eficaz para diferentes públicos dentro de uma organização. Por meio de dashboards interativos e relatórios visuais, os gestores, analistas e outras partes interessadas podem facilmente compreender e interpretar os resultados das análises, o que facilita a colaboração, a tomada de decisão.

1. desenvolvimento
   1. Ingestão

Compondo a ingestão de dados na arquitetura do Data Lake estipulada para este Trabalho de Conclusão de Curso, há duas fontes de dados distintas para exemplificar a variedade de tipos de dados capazes de consolidarem um Data Lake.

Essa variedade propositada visa ilustrar a amplitude de tipos de dados que podem ser consolidados em um Data Lake, refletindo a natureza abrangente e adaptável do projeto. Cada fonte de dados é cuidadosamente incorporada, considerando as particularidades de sua estrutura, formatos e requisitos específicos, demonstrando a flexibilidade e a capacidade de acomodação da arquitetura proposta.

* 1. API Climática

A primeira ingestão ocorre a partir da consulta periódica das informações provenientes da API externa “*IQAiR*” (IQAIR, [S/D]a), que disponibiliza dados climáticos como temperatura, umidade, velocidade do vento e outra métricas sobre diversas localidades dispersas pelo mundo. Foram consideradas uma porção de cidades do território brasileiros para a consulta às informações, utilizando-se o critério de uma cidade por estado para que as informações armazenadas na camada “Crua” abrangessem, de certa forma, todo o território nacional. Sendo assim, serão requisitadas as informações acerca das seguintes localidades:

* Rio Branco – AC
* Macapá – AP
* Manaus – AM
* Camaçari – BA
* Fortaleza – CE
* Serra – ES
* Brasília – DF
* Cuiabá – MT
* Timóteo – MG
* Santarém – PA
* Piancó – PB
* Colombo – PR
* Recife – PE
* Rio de Janeiro – RJ
* Porto Velho – RO
* Boa Vista – RR
* São Paulo – SP
* Aracaju – SE
* Palmas – TO.

Para realizar a consulta, utilizou-se o serviço AWS Lambda para execução de uma lógica, desenvolvida na linguagem Python, responsável por solicitar as informações atuais sobre uma cidade, e armazenar os dados na camada “Crua” do Data Lake.

Ao solicitar a execução da função Lambda, dois argumentos são informados ao serviço, “state” e “city”, que detalham qual localidade será consultada de acordo com os valores do nome do estado e nome da cidade, respectivamente.

A requisição à API ocorre por meio da função “load\_data\_api”, que estrutura a solicitação por meio da biblioteca “requests”. As informações da localidade desejada são agregadas ao endereço de requisição da API e a chave de acesso disponibilizada para a construção desta demonstração.

O retorno à requisição é disponibilizado na sintaxe padrão JSON, e por isso é necessário o uso de outra biblioteca externa para manipulação das informações de retorno. Por meio da biblioteca “json” a resposta da API é convertida para um dicionário da linguagem Python, tornado a sua manipulação mais simples.

Devido ao escopo da arquitetura, não ocorre nenhuma tratativa sobre o dado no momento da coleta, desta forma, o último processo a ser realizado é a gravação das informações na camada de destino. Para isso a biblioteca de autoria da própria AWS, a biblioteca “boto3”, é utilizada para relacionar as informações processadas pela função com os procedimentos de armazenamento de arquivos no Amazon S3.

Texto

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Função de consulta à API climática

Embora a requisição as métricas referentes a localidade sejam satisfeitas pelo processo descrito acima, a manipulação destas informações demanda um fluxo de etapas definidos para que a performance da ingestão não seja impactada por fatores externos. A principal adversidade presente neste cenário, é o conflito apresentado entre as condições impostas pelo servidor da API durante a requisição e as limitações da função Lambda, ao utilizá-la em suas configurações mais baratas.

O serviço AWS Lambda custeia o usuário por cada segundo em que a função está em execução (AMAZON, 2023), ou seja, ao utilizar este serviço que o código a ser executado deve ter o menor tempo de processamento possível. Porém, a API disponibilizada pela equipe da “IQAir” impõe restringimentos a respeito do período entre cada requisição. De acordo com a documentação disponível em seu site (IQAIR, [S/D]b), a empresa declara que há o limite máximo de dez requisições por segundo aos seus servidores.

Dessa forma, é imposta uma divergência entre as lógicas de programação a serem aplicadas para a função Lambda e para a requisição aos dados.

Interface gráfica do usuário

Descrição gerada automaticamenteAssim sendo o seguinte fluxo de processos foi definido:

Ilustração - Fluxo de ingestão de dados climáticos

Fonte: Autoria própria

Por tanto, realizou-se a utilização de outro serviço para orquestração do fluxo de ingestão, segmentando a lógica de programação ao escopo de cada serviço. O serviço Amazon Step Function, permitiu que a imposta pelo limite máximo de requisições à API aparta-se dos procedimentos internos da função Lambda.

No qual a ingestão ocorre da seguinte maneira, todas as localidades a serem requisitadas são informadas na etapa “Definição de Localidades”, definindo os valores de nome do estado e nome da cidade. O processo denominado “Orquestrador” é responsável por iterar sob cada localidade solicitando a execução da função Lambda para os parâmetros de localidade atual. Caso a função finalize com sucesso, será imposta uma espera de 15 segundos para impossibilitar que alguma das requisições à *API* falhe por ultrapassar o limite máximo imposto pelo servidor. Entretanto, caso a função *Lambda* aborte, há uma etapa para tratativa de falhas evitando que o ciclo repetido para cada localidade se interrompa.

Estruturado o fluxo de coleta às informações para cada localidade, complementa-se a arquitetura com o uso do serviço Amazon EventBridge, que propicia engatilhar o fluxo de ingestão de forma periódica, pois o intuito é que estes dados sejam utilizados para análise da mudança climática histórica em pontos distribuídos no país.

* 1. Banco de dados relacional

Para a adoção de dados provenientes de gerenciadores de dados relacionais na arquitetura imposta por este Trabalho de Conclusão de Curso, realizou-se uma triagem de conjuntos de dados que coincidissem com a proposta ambiental do Data Lake.

Sendo assim, as seguintes estruturas de dados foram escolhidas para agregar o banco de dados relacional de origem da arquitetura, visando a variedade entre as informações:

Iniciando pelo conjunto de dados “Solar Energy Brazil”, o qual detalha o consumo e produção de energia solar pelo território brasileiro ao sintetizar informações históricas para cada estado ao longo do tempo; em seguida, apresenta-se a coleção de dados “Hydrometeorological Brazil”, que descreve entre diversas tabelas informações sobre os níveis de chuva, nível da capacidade de reservatórios de agua, vazão dos reservatórios e métricas climáticas para diversos locais do país; em diante, há informações acerca dos incêndios ambientais ocorridos ao longo dos anos no Brasil, através do conjunto “Forest Fire Brazil”, ao segregar esse histórico para cada estado brasileiro; enfim, por meio dos relatórios do “Climate Weather Surface of Brazil”, é armazenado a coleta histórica de dados climáticos em cada uma das cincos regiões do Brasil em intervalos de uma hora a cada coleta, tornando-se o conjunto de dados mais robusto da arquitetura; e finalizando a diversidade de informações a serem consideradas, realiza-se a ingestão de dados de dois conjunto de dados auxiliares, um acerca da poluição atmosférica e os seus impactos na saúde da população, e outro com referências sobre cada cidade do país, coletadas pelo IBGE.

Compatível com volumetria e estrutura dos dados, o serviço Amazon RDS teve o comportamento desejado ao armazenar todos estes conjuntos de dados. À vista disso, o banco de dados descrito abaixo foi construído.

No escopo gratuito do serviço, é possível armazenar até vinte gigabytes de dados, que dado o escopo do projeto é suficiente para o armazenamento das informações de origem. E para concluir esta etapa, o MySQL atua como sistema de gerenciamento de banco de dados utilizado no banco de dados, já que é uma interface de código aberto confiável e performática, capaz de oferecer todas as funcionalidades necessárias para a migração dos dados de maneiro total, em lotes, ou de forma incremental, ao se analisar os registros de alterações realizadas às bases de dados.

Tela de celular

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Configurações do Banco de dados relacional

Concluindo o armazenamento dos dados de origem no banco de dados descrito acima, elaborou-se uma rotina de extração destas informações e ingestão dos dados na camada “Crua” do Data Lake por intermédio do serviço AWS Database Migration Service.

Tela de celular com publicação numa rede social

Descrição gerada automaticamenteCom o banco de dados em execução, é necessário a criação de uma instância de replicação que conectará à base de dados desejada e controlará a migração de maneira contínua. Dessa forma a seguinte instância realiza esta função.

Ilustração - Configurações da instância da migração de dados relacionais

Fonte: Autoria própria

Em seguida, dentro do serviço é necessário criar “pontos de extremidades” que detalham a conexão com os locais de origem e destino da informação a ser migrada. Assim sendo, dois pontos fazem parte da arquitetura. O primeiro sendo o ponto de origem detalhando os parâmetros de conexão ao banco de dados, desde as credenciais para acesso as bases de dados até a descrição do endereço público do banco de dados RDS.

Tela de celular com aplicativo aberto

Descrição gerada automaticamenteRessaltando que o banco de dados deve liberar que usuários externos o acessem através de credenciais, para tanto deve-se listar os endereços IP e portas de acessos livres na configuração de rede do banco de dados

Ilustração - Configurações do local origem dos dados

Fonte: Autoria própria

Finalizando na criação do ponto de extremidade de destino, que no caso é responsável por conectar a instância de replicação do AWS Database Migration Service à camada “Crua” presente no Data Lake. As seguintes configurações foram estabelecidas:

Ilustração - Configurações da base de dados de origem

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamenteFonte: Autoria própria

E para concluir a ingestão dos dados proveniente do banco de dados relacional ao Data Lake, configura-se uma “tarefa de migração” que relaciona todos os pontos elencados acima, declarando qual instância de migração será usada para movimentação dos dados, qual ponto de extremidade de origem e de destino irão compor a tarefa, e quais tabelas de dados serão migradas.

Culminando na ingestão total dos dados presentes na base de dados no momento que a tarefa de migração inicia, e na ingestão de qualquer alteração de registro realizada posteriormente.

Ilustração - Configurações da instância de migração dos dados

Tela de celular com publicação numa rede social

Descrição gerada automaticamenteFonte: Autoria própria

* 1. Armazenamento

A etapa de armazenamento é um pilar fundamental na construção de um Data Lake robusto, e a escolha do sistema de armazenamento desempenha um papel crítico na eficiência e na flexibilidade do ambiente. Diante desse desafio, nossa decisão estratégica foi adotar o Amazon S3 (Simple Storage Service) da AWS como a solução principal para o armazenamento dos dados do Data Lake.

A escolha do Amazon S3 foi uma estratégia ponderada, guiada pela sua essência como um sistema de armazenamento de arquivos distribuído. Essa decisão foi fundamentada na capacidade distintiva do S3 de lidar com a complexidade proveniente de utilizar fontes distintas de dados. A arquitetura distribuída do S3 oferece uma escalabilidade automática, assegurando que o Data Lake possa gerenciar grandes volumes de dados de maneira eficiente.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamenteA integração nativa do S3 com outros serviços da AWS é uma vantagem estratégica, simplificando o desenvolvimento de *pipelines* de dados e facilitando o processamento subsequente. Essa escolha não apenas proporciona uma infraestrutura robusta, mas estabelece uma base sólida para enfrentar os desafios associados a cenários de alto volume de dados. A flexibilidade, escalabilidade e integração do Amazon S3 desempenham um papel crucial na criação de um ambiente de Data Lake que não apenas armazena dados, mas os gerencia de maneira inteligente, preparando o terreno para análises avançadas e insights valiosos.

Ilustração - Buckets do Amazon S3 presentes na arquitetura do projeto

Fonte: Autoria própria

Nesta arquitetura de Data Lake, os *buckets* do Amazon S3 assumem papéis distintos, delineando as diversas etapas do processamento de dados e evidenciando uma organização meticulosa para a gestão eficiente dessas informações. O bucket “*tcc-data-lake-raw*" surge como o ponto de ingresso, abrigando dados brutos provenientes de múltiplas fontes. Nesse cenário, cada fonte é segregada em sua pasta dedicada, com uma subdivisão adicional para cada tabela específica, preservando assim a integridade original dos dados e garantindo a manutenção da diversidade de formatos e estruturas. Dentro de cada tabela na camada *"Raw"*, há também uma subpasta "processed/" que armazena todos os arquivos que já foram transformados e armazenados na camada de *"Stage”*.

No âmbito da etapa intermédia, o “*tcc-data-lake-stage*" desempenha um papel essencial, oferecendo residência aos dados após passarem por transformações e limpezas. Semelhante à camada *"Raw"*, cada fonte tem sua pasta, e dentro dela, as pastas de cada tabela são ampliadas para incluir subpastas "*temp/*" e "*processed/*". Essa organização estratégica destina-se ao armazenamento temporário e final, respectivamente. No contexto analítico, o “*tcc-data-lake-semantic*” surge como o epicentro das tabelas geradas a partir dos dados limpos da camada *"Stage"*. Essas tabelas surgem da integração dos dados processados durante as etapas analíticas e de geração de insights, estabelecendo uma base sólida para análises avançadas.

Os artefatos de desenvolvimento do Data Lake são dispostos nos domínios designados do “*tcc-data-lake-artefatos*". Scripts, arquivos e ferramentas, utilizados na construção do ambiente, são organizados por assunto, garantindo uma gestão eficiente e acessível desses elementos essenciais. Encerrando o ciclo, o “*tcc-data-lake-logs"* emerge como um repositório vital, guardando os registros detalhados de todas as atividades ao longo do fluxo do Data Lake. Essa abordagem logística não apenas viabiliza monitoramento e auditoria eficazes, mas também se torna um recurso para a resolução de problemas e otimização contínua. A concepção dessa estrutura de buckets reflete a eficácia do ambiente de Data Lake, capacitando a manipulação ágil e inteligente de volumes consideráveis de dados.

* 1. Processamento

Na fase de processamento dos dados, destaca-se a implementação do AWS Glue como peça central. Este serviço fundamenta-se na aplicação da ferramenta Apache Spark, específico para transformação e padronização dos dados em variadas volumetrias e tipos de arquivo, simplificando a gestão dos componentes de infraestrutura necessários para compor o poder computacional em formato de *cluster* demandado para o funcionamento do Spark.

A escalabilidade inerente ao AWS Glue é um diferencial significativo. Sua capacidade de dimensionar recursos automaticamente em resposta à demanda assegura a eficiência no processamento, mesmo em cenários de alto volume de dados. E em vista da escala do projeto, as rotinas de processamento focalizam na manipulação da estrutura dos dados e na sintetização de informações ao analisar os conjuntos de dados presente no Data Lake, visto que o AWS Glue concilia o poder computacional do cluster Spark à cada volumetria de dados processada.

Dessa forma, as rotinas de processamento são criadas inicialmente com as configurações mínimas de poder computacional, ao definir a quantidade e tipo de instância que será alocada no cluster a ser criado no momento de processar os dados.

Para a arquitetura neste trabalho, construiu-se duas formas de processamento de dados distintas, visando replicar a lógica de programação para as informações que compartilhassem das mesmas características, e diminuindo assim a complexidade ao reduzir a quantidade de processos distintos presentes no ambiente. Pois, é possível parametrizar os fluxos de processamento presentes no AWS Glue customizados a rotina de processamento para cada conjunto de dados.

Ademais, estão estipulados processos apartados de governança de dados e manutenção do ambiente que podem ser anexados aos fluxos de processamento de dados de acordo com as suas demandas e particularidades de execução.

* 1. API Climática

A primeira forma de processamento construída atende as particularidades presentes nos dados provenientes do processo de ingestão da API climática. Neste processo, o objetivo é considerar os novos presentes na camada “Crua” e padronizá-los ao realizar a conversão para o formato de dados colunar “parquet”. Visando tornar mais performático os demais processos que podem vir a consumir estas informações.

Inicialmente, ocorre a leitura dos arquivos em seu formato original, selecionando e convertendo a estrutura encadeada do JSON para a disposição de dados padrão do Spark. Logo em seguida, ocorre a seleção das informações úteis e o armazenamento de um arquivo unificado na camada “Tratada” do Data Lake. Abaixo está disposto o processo de conversão dos dados originais para a versão tratada.

Texto

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Fluxo de tratamento dos dados originais da API climática

Sintetizando, o processo associado a este *job* do AWS Glue é projetado para buscar os arquivos na raiz do caminho definido, mapear as colunas necessárias e realizar a transformação para o formato parquet.

Este processamento de dados é encadeado e gerenciado pelo mesmo orquestrador externo utilizado no processo de ingestão de dados. Dessa forma, os dados da camada “Crua” são padronizados e inseridos na camada “Tratada” logo após a ingestão de novos arquivos no Data Lake. Além de inicializar os processos de limpeza dos dados tratados e manutenção da camada “Crua” estipulada na arquitetura deste trabalho.

Tela de computador com imagem de vídeo game

Descrição gerada automaticamenteSegue a estrutura do orquestrador contendo o processo de ingestão seguido pelo processamento dos dados entre as camadas, e imposição da governança de dados e manutenção do ambiente.

Ilustração - Orquestração do fluxo de tratamento dos dados originais da API climática

Fonte: Autoria própria

* 1. Banco de dados relacional

Contemplando a segunda forma de processamento de dados presente no escopo da arquitetura deste projeto, há a padronização dos dados provenientes do processo de ingestão do banco de dados relacional disposto no ambiente. Neste processo, o objetivo é considerar os novos presentes na camada “Crua” e padronizá-los ao realizar a conversão para o formato de dados colunar “parquet”. Visando tornar mais performático os demais processos que podem vir a consumir estas informações.

Entretanto, em comparação ao fluxo de processamento documentado anteriormente, há alguns particulares que obrigaram a criação de uma segunda lógica de tratamento apenas para os dados provenientes do banco de dados relacional.

Nesta tratativa os dados originais estão disponibilizados na estrutura “CSV”, então após serem lidos, estes dados são convertidos para a estrutura padrão do Spark permitindo a execução do restante das transformações. Que se resumem a adição de uma coluna adicional para versionamento dos registros com base na data em que foram processados pelo fluxo.

Finalizando, com a seleção das informações úteis e o armazenamento de um arquivo unificado na camada “Tratada” do Data Lake. Abaixo está disposto o processo de conversão dos dados originais para a versão tratada.

Texto

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Fluxo de tratamento dos dados oriundos do banco de dados relacional

Este processamento de dados é encadeado e gerenciado pelo mesmo orquestrador externo utilizado no processo de ingestão de dados. Dessa forma, os dados da camada “Crua” são padronizados e inseridos na camada “Tratada” logo após a ingestão de novos arquivos no Data Lake. Além de inicializar os processos de limpeza dos dados tratados e manutenção da camada “Crua” estipulada na arquitetura deste trabalho.

Segue a estrutura do orquestrador contendo o processo de ingestão seguido pelo processamento dos dados entre as camadas, e imposição da governança de dados e manutenção do ambiente.

Interface gráfica do usuário

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Orquestração do fluxo de tratamento dos dados oriundos do banco de dados relacional

* 1. Semânticas

Contemplando a sintetização dos dados presentes na camada “Tratada” da arquitetura, há a imposição de fluxos de processamento de dados voltados para consolidação das informações em visualizações que contemplam a unificação de tabelas, agrupamentos de dados por algum período ou característica, ou aplicação de processos externos como o uso de inteligência artificial ou análise preditiva.

Os processamentos pertencentes à camada “Semântica” têm a sua importância fundamentada na execução de processos, geralmente extremamente custosos computacionalmente, de maneira singular. Ou seja, elabora-se uma lógica que produza visualizações concretas de dados através de um processamento único, e que o consumo destas informações seja compartilhado. Desta forma, os usuários finais de um fluxo de processamento semântico acessam informações centralizadas descartando a necessidade de executar o fluxo a cada nova demanda.

Assim sendo, no escopo estipulado para a arquitetura deste projeto há a composição de semânticas que consolidam os dados processados e armazenados previamente pelos fluxos de tratamento de dados.

* 1. Dados climáticos

Com intuito consolidar os dados provenientes da API climática, coletados periodicamente utilizando o processo de ingestão, elaborou-se um tratamento de dados que sintetiza as informações disponíveis ao classificar os dados referentes a cada localidade relativos ao dia atual em um registro único.

Assim, é possível apresentar o histórico de dados de maneira a compor médias, contagens e outros técnicas agregativas com base nos dados agrupados por localidade.

Por conseguinte, disponibiliza-se aos usuários desta semântica uma visualização mais sucinta e direta de todas as métricas coletadas ao longo do dia. Facilitando a compreensão das mudanças climáticas ocorridas em todo o território nacional brasileiro.

O fluxo de processamento da semântica está disposto abaixo.

Texto

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Fluxo de processamento da semântica com base nos dados climáticos

Sumarizando a atuação do fluxo de processamento, são considerados os dados armazenados na camada “Tratada”, referentes ao dia em que ocorre a execução da semântica, para realizar os agrupamentos destas informações baseando-se na localidade e momento que as métricas foram coletadas.

Logo em seguida, são definidas as informações que retornaram do agrupamento dos registros por meio de funções agregativas para obtenção de médias, mínimas e máximas. Culminando na gravação do resultado gerado na camada “Semântica”.

* 1. Governança de dados

A governança de dados estipulada na arquitetura do Data Lake é fundamentada no processo meticuloso de limpeza dos dados. Essa prática não apenas assegura a qualidade e a consistência das informações, mas também estabelece um fluxo estruturado desde a análise da estrutura da informação até a gravação do arquivo final.

A análise da estrutura da informação é o ponto de partida, envolvendo uma investigação macro de cada tabela. Esse processo abrangente busca identificar lacunas nos dados, necessidades de conversões de tipagem, remoção de caracteres indesejáveis, tratamento de valores nulos, entre outras ações. A compreensão global de todas as tabelas é crucial para definir um padrão consistente entre os dados, proporcionando uma base sólida para consultas e análises.

A definição das regras de limpeza, como forçar tipos de dados, remover linhas com valores específicos, substituir pontuações, entre outras, é uma etapa crítica e foi codificada em Python. Cada tabela possui suas regras específicas, e para facilitar a parametrização dessas regras, optou-se pelo uso do AWS DynamoDB. Este serviço de banco de dados não relacional permitiu a criação de um sistema de chave-valor, em que a chave representa a regra e o valor corresponde às colunas ou valores associados à regra.

Abaixo é possível conferir a estrutura de um dos itens de parametrização da limpeza de dados, armazenados na tabela “tcc\_dc\_parameter” do DynamoDB. O registro define quais processos de limpeza de dados que serão aplicadas à tabela *consumers* da base de dados *Solar Energy Brazil* presente na camada “Tratada” do Data Lake.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Regras de limpeza de dados da tabela 'consumers' do banco de dados 'Solar Energy Brazil'

É possível conferir no registro acima, parâmetros para identificação da origem dos dados, orientação de manipulação de colunas e conversão de tipos de dados, culminando na revisão de todos os valores armazenados para tratamento de duplicidade, nulidade e particularidade acerca da estrutura das informações.

Dessa forma, todas as bases de dados presentes na camada “Tratada” do Data Lake devem ter suas características mapeadas, na tabela do DynamoDB, visando o aumento da confiabilidade do usuário final que consome tais informações.

A execução eficiente dessas regras é realizada por meio do AWS Lambda, um serviço *serverless* que garante uma resposta rápida às demandas de limpeza. O AWS Lambda também é responsável por salvar os arquivos já limpos na camada *"Stage",* na pasta "processed/" dentro de suas respectivas tabelas. Este processo de limpeza é feito por tabela, e a expectativa é que seja concluído em no máximo 15 minutos.

A exclusão dos arquivos processados da camada "temp/" na tabela é feita após a limpeza, uma vez que esses arquivos foram apenas transformados e possuem uma versão original armazenada historicamente na camada "Raw". Esta abordagem não apenas mantém a integridade dos dados, mas também oferece uma trilha de auditoria e histórico para análises retroativas.

Ressaltando que, por ser um processo apartado desenvolvido visando impor um padrão de qualidade de informações armazenadas nas camadas do Data Lake, a execução dessa rotina de limpeza de dados não possui orquestração própria. Ou seja, não possui estrutura que define a periodicidade e fluxo de execução. Portanto, na arquitetura deste projeto, este recurso é anexado aos fluxos de processamentos presentes no ambiente, tal qual documentado nos tópicos anteriores de API Climática e Banco de dados relacional.

Assim, a governança de dados no Data Lake é intrinsecamente vinculada ao processo de limpeza, envolvendo uma análise cuidadosa, codificação de regras, parametrização eficiente e execução ágil, garantindo a qualidade e a confiabilidade dos dados ao longo do tempo.

Texto

Descrição gerada automaticamenteA seguir está disposto a estrutura da Função Lambda responsável por executar o processo parametrizado de limpeza de dados no ambiente.

Ilustração - Função de limpeza dos dados

Fonte: Autoria própria

Texto

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Função de limpeza dos dados (cont.)

* 1. Manutenção do ambiente

A manipulação de volumetrias de dados, em constante crescimento, e o controle de execução de processos apartados no mesmo ambiente, demanda prudências aptas a conservar a estrutura estipulada pela arquitetura do Data Lake.

Incessantemente, haverá novos dados sendo armazenados nas camadas, novas gravações de resultados provenientes da execução dos processos de ingestão, processamento e visualização, além do uso do ambiente para manipulação de usuários, permissões e outras funcionalidades. Em vista disto, é evidente que sem nenhuma ação sobre isso a qualidade da solução construída será degrada ao longo do tempo devido ao excesso de arquivos históricos e reprocessamento de arquivos.

Para tanto, dois métodos de manutenção do ambiente foram incorporados à arquitetura do projeto com intuito de conservar a performance e estimativa de custos do Data Lake.

* + 1. Ciclo de vida de objetos

Em prol de reduzir a quantidade de arquivos, referentes ao controle de execução dos processos em execução, armazenados no repositório “tcc-data-lake-logs” foi criada uma monitoria sob todas as possíveis subpastas para impor um período limite de armazenamento, expirando e deletando qualquer arquivo que ultrapasse o ciclo de vida estipulado.

Devido ao escopo, o período máximo de conservação de arquivos no repositório foi definido como trinta dias corridos. Dessa forma, qualquer arquivo que ultrapasse essa idade será automaticamente deletado, evitando o acúmulo de objetos obsoletos para o escopo deste Trabalho de Conclusão de Curso.

Abaixo evidencia-se a configuração do ciclo de vida definida no repositório para atender a demanda de manutenção do ambiente.

Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamenteInterface gráfica do usuário, Texto

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Regra de ciclo de vida de objetos (cont.)

Ilustração - Regra de ciclo de vida de objetos

Fonte: Autoria própria

* + 1. Movimentação de objetos

A volumetria constante do Data Lake ocasiona desafios acerca da manutenção da performance das aplicações processante de dados no ambiente. Devido a sua característica incremental, os dados armazenados nas camadas da arquitetura devem ser preservados para que seja montado um histórico dessas informações.

Por isso, um dos diferenciais na arquitetura de um Data Lake é a construção de uma lógica que evite o processamento de informações fora do escopo estipulado para algum fluxo de processamento de dados. E para contemplar este atributo neste Trabalho de Conclusão de Curso, há a imposição de um processo para movimentação de arquivos, dentro das camadas, que foram consumidos pelos fluxos de processamento presentes no escopo do projeto.

Este procedimento foi construído utilizando uma função do AWS Lambda, que facilita a parametrização e replicação de sua lógica para qualquer demanda da arquitetura. Na qual é solicitado a migração de objetos presentes em uma localidade de origem para outra localidade de destino. Assim, esse processo é crucial para marcar os arquivos como processados e manter uma trilha histórica essencial para a gestão e análise retroativa dos dados no Data Lake.

Desta maneira, os processos de ingestão de dados e fluxos de processamentos de dados utilizam desta lógica para evitar que arquivos desnecessários sejam considerados em suas execuções, evitando assim, a decadência da performance no decorrer do tempo.

Ressaltando que, igualmente ao processo de governança de dados documentado anteriormente, a execução dessa manutenção do ambiente não possui orquestração própria. Ou seja, não possui estrutura que define a periodicidade e fluxo de execução. Portanto, na arquitetura deste projeto, este recurso é anexado aos fluxos de processamentos presentes no ambiente, tal qual documentado nos tópicos anteriores de API Climática e Banco de dados relacional.

Abaixo está o demonstrativo da versão final do processo, que a cada solicitação considera os parâmetros informados e realiza a movimentações dos objetos encontrados na localidade de origem para a localidade de destino.

Texto

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Função de movimentação de objetos

* 1. Visualização

A visualização das informações armazenadas entre as camadas da arquitetura do Data Lake é a forma de consumo mais valioso destes dados, pois consolida todos os fluxos de ingestão, processamento e armazenamento executados anteriormente ao consumo do dado. É o momento que o usuário final ou processos externos ao ambiente estipulado pela arquitetura, acessam os dados e beneficiam-se das informações disponíveis.

Nesta etapa, os dados, após passarem pelo processo completo de ingestão, armazenamento e processamento no Data Lake, tornam-se acessíveis para análise e exploração.

Desta forma, é necessário estipular regras de acesso a todas as informações retidas no Data Lake, pois pode haver conjuntos de dados com informações pessoalmente identificáveis ou informações sensíveis de qualquer escala.

Para isso, é fundamental destacar a integração do AWS Lake Formation, que desempenha um papel crucial na restrição dos dados em todas as camadas do Data Lake com base no nível de acesso do usuário. Essa ferramenta proporciona uma camada adicional de segurança, garantindo que apenas usuários autorizados possam acessar e manipular dados específicos dentro do ambiente do Data Lake.

Impor restrições de acesso através da interface do AWS Lake Formation apenas a criação de uma “permissão ao Data Lake”, no qual é descrito qual usuário ou grupos de usuários serão afetados pelas restrições impostas na permissão.

Exemplificando este passo, foram impostas restrições à leitura do conjunto de dados climáticos presente na camada “Tratada” a um dos usuários existentes no Data Lake. Abaixo é possível analisar as configurações presentes na permissão.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Configurações de restrição de acesso aos dados climáticos presentes na camada tratada

Interface gráfica do usuário

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Configurações de restrição de acesso aos dados climáticos presentes na camada tratada (cont.)

Após a ativação desta política de restrição, caso o usuário afetado acesse essas informações, através de uma interface de acesso como o Amazon Athena, certas informações são omitidas da visualização.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Visualização dos dados climáticos com informações omitidas pelo Lake Formation

Com essas capacidades de visualização e restrição de acessos às informações, o Data Lake se transforma em um recurso dinâmico para análises avançadas e tomada de decisões informadas.

* + 1. Linguagem de consulta estruturada

A forma de visualização de dados mais consolidada atualmente é por meio da linguagem SQL, que permite acessar dados relacionados através de uma sintaxe simples e composta por inúmeras maneiras de manipulação de dados.

Objetificando a implantação desta forma de acessos aos dados na arquitetura deste projeto, o serviço Amazon Athena integrado à arquitetura do Data Lake, oferecendo uma abordagem flexível e poderosa para visualização por meio de consultas SQL.

A capacidade de executar consultas diretamente sobre os dados armazenados no Amazon S3 proporciona uma maneira dinâmica de extrair insights e realizar consultas complexas, utilizando diferentes fontes de dados, enriquecendo assim a análise exploratória. Além disso, as estruturas de dados geradas neste processo de consulta sobre os dados, podem ser salvas na camada "Semântica" do Data Lake. Essa integração amplia as possibilidades de análises, permitindo que essas tabelas sejam reconsultadas e combinadas com dados adicionais da camada "Stage" e "Semântica". Isso resulta em análises mais complexas e insights mais profundos, contribuindo para uma compreensão abrangente do ambiente de dados consolidados.

Para atender esta demanda, utilizou-se ferramentas fornecidas pelo serviço AWS Glue denominados Glue Crawler e Glue Data Catalog, capazes de mapear os objetos armazenados nas camadas do Data Lake, permitindo que os processos internos do Amazon Athena possuam de antemão a estrutura do dado presente no arquivo e onde o seu endereço de acesso.

Sendo assim, é necessário criar um Glue Crawler para cada conjunto de dados que se deseja visualizar e manipular pela interface do Athena. Definindo a origem do conjunto de dados e destino de armazenamento no Glue Data Catalog, além de configurações de permissão de acesso e execução.

Ilustração - Configurações do Glue Crawler responsável por mapear o conjunto de dados climáticos

Tela de celular com publicação numa rede social

Descrição gerada automaticamenteFonte: Autoria própria

Tela de celular com publicação numa rede social

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Configurações do Glue Crawler responsável por mapear o conjunto de dados climáticos (cont.)

Acima está disposta as configurações necessárias para mapeamentos da estrutura de dados climáticos armazenados na camada “Tratada” do Data Lake.

Ao término do mapeamento realizado pelo Glue Crawler sob os objetos do conjunto de dados, ficará disponível, por meio do Glue Data Catalog, uma referência a estrutura destas informações e localidade dos objetos que retem os dados. Possibilitando que a interface do Amazon Athena apresente as informações e aplique as manipulações construídas pelo usuário.

Abaixo está o demonstrativo do uso do Amazon Athena para composições de consultas SQL, considerando os dados climáticos armazenados na camada “Tratada” proveniente do fluxo de processamento das informações coletadas da API climática externa. Possibilitada pelo mapeamento aos objetos de origem, realizado pelo Glue Crawler

Tela de computador com jogo

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Exemplo de utilização do Amazon Athena

* + 1. Interface de acesso

Com intuito de oferecer um diferencial acerca dos acessos as informações presentes no Data Lake, há a disponibilização de conjuntos de dados através de uma API que constrói uma interface entre os dados presentes nas camadas da arquitetura com aplicações externas que desejem utilizar esses dados para incrementar suas lógicas internas.

Para isso, é possível o uso do serviço Amazon API Gateway para construir e manter uma estrutura que intermedia os dados do Data Lake com os solicitantes pela interface de uma API.

O Amazon API Gateway se destaca como uma peça fundamental na consolidação dos dados do Data Lake, atuando como a interface central para aplicações externas. Ao fornecer uma fonte unificada de dados, o API Gateway facilita a integração desses dados com sistemas externos. Essa ponte entre o Data Lake e aplicações externas promove uma distribuição eficiente e controlada das informações, garantindo que os dados sejam utilizados de maneira otimizada por diferentes sistemas e plataformas.

Desta forma, foi elaborada uma API que consulta os dados resultantes do fluxo de processamento semântico dos dados climáticos presentes no Data Lake. E esta API, foi implementada a lógica da Atividade Prática Supervisionada “Desenvolvimento de aplicação web para conscientização do impacto ao meio ambiente do cotidiano atual”, desenvolvida para contemplar a APS do 8º semestre de Ciência da Computação na Universidade Paulista (Autoria própria, 2023).

Mapa

Descrição gerada automaticamenteNo qual os dados gerados pelo processo de semântica, são considerados para compor um mapa interativo no qual todas as localidades consultadas ao longo do fluxo de processamento são apresentadas ao usuário, que pode agir sobre o mapa para visualizar as informações climáticas da localidade interagida.

Ilustração - Aplicação externa manipulando os dados oriundos do Data Lake

Fonte: Autoria própria

Para compor a lógica que intermedia a requisição feita ao API Gateway, pela aplicação externa, e os dados semânticos armazenados no Data Lake, é necessário o uso de uma Função Lambda que consulta os dados desejados e retorna ao solicitante da requisição.

Abaixo é possível conferir a lógica de programação aplicada a função.

Texto

Descrição gerada automaticamenteFonte: Autoria própria

Ilustração - Função para consulta dos dados semânticos através da requisição à API

Conclusão

O foco primordial desta pesquisa é conceber uma arquitetura de Data Lake organizada em camadas, visando a eficiente ingestão de dados provenientes de APIs e bancos de dados on-premise. Essa abordagem abrangente engloba desde a fase inicial de integração dos dados até o armazenamento seguro na nuvem, utilizando os serviços da AWS (Amazon Web Services). A proposta visa realizar a integração de dados provenientes de diversas fontes, empregando técnicas especializadas para garantir a qualidade e coerência dessas informações no contexto do Data Lake.

Adicionalmente, a arquitetura proposta busca executar a limpeza e o pré-processamento dos dados, aplicando métodos avançados de validação, tratamento e transformação. Esta etapa crucial tem como objetivo assegurar a integridade e a utilidade dos dados armazenados, preparando-os para análises subsequentes. Ao alcançar esses objetivos específicos, a intenção é estabelecer uma infraestrutura robusta e escalável que não apenas acomode uma variedade de dados, mas também otimize a utilidade e a acessibilidade dessas informações para análises futuras.

A efetiva implementação do Data Lake na AWS reflete a concretização dos objetivos propostos, estabelecendo uma infraestrutura sólida e eficiente para a gestão de dados provenientes de APIs e de um banco de dados on-premise. A estratégia adotada envolveu a utilização do Amazon S3 como a plataforma central de armazenamento, estruturada em três camadas distintas: dados brutos, camada de limpeza e qualidade de dados, e camada de análise de dados.

A camada de dados brutos no Amazon S3 demonstrou ser uma solução eficaz para a contínua ingestão de dados, provenientes de diversas fontes. A flexibilidade e escalabilidade oferecidas pelo Amazon S3 permitiram a coleta eficiente e o armazenamento seguro dos dados, mantendo a integridade e acessibilidade necessárias para um Data Lake eficaz. O AWS Glue desempenhou um papel crucial no processamento dos dados, facilitando a limpeza e o pré-processamento das informações antes de serem armazenadas na camada de qualidade de dados. A integração eficaz desses serviços resultou em uma camada intermediária que garante a consistência e a confiabilidade dos dados armazenados.

A integração com o Amazon API Gateway viabilizou o envio eficiente de informações para um site externo, demonstrando a capacidade de interoperabilidade do Data Lake com outros sistemas e serviços. Essa abordagem não apenas valida de forma concreta os objetivos propostos, mas também evidencia a aplicação prática da arquitetura implementada em cenários do mundo real. Em síntese, a implementação bem-sucedida do Data Lake na AWS não apenas atende, mas supera as expectativas, proporcionando uma estrutura sólida para a gestão eficaz de dados e análises avançadas.

Apesar do êxito na implementação do Data Lake na AWS e no alcance dos objetivos propostos, é imperativo reconhecer e examinar as limitações encontradas durante o desenvolvimento do projeto. Uma das restrições primárias identificadas está nos custos associados à utilização dos serviços da Amazon Web Services (AWS). A escalabilidade e flexibilidade proporcionadas pela AWS, embora valiosas, podem apresentar desafios financeiros, especialmente quando a infraestrutura cresce em consonância com as necessidades do projeto. Ao longo do projeto, foi necessário um gerenciamento cuidadoso dos recursos da AWS para evitar impactos financeiros inesperados.

É crucial salientar que, embora a questão financeira tenha sido identificada como uma limitação, as medidas de mitigação adotadas contribuíram para manter os custos dentro de limites gerenciáveis. Este desafio específico ressalta a importância de uma abordagem estratégica ao planejar e implementar a solução de um Data Lake, assegurando um equilíbrio apropriado entre os benefícios proporcionados pelos serviços e as considerações orçamentárias do projeto com a realidade da organização.

Portanto, a título de mérito, há espaço para aplicação do processamento em tempo real (real time), que por questões financeiras, não fora aplicado no presente projeto, visto que, também, não iria gerar ganhos para a proposta do trabalho se confrontado ao seu custo. Dessa forma, caso a organização tenha um escopo de fundos maior, e haja a real necessidade, fica aberto a implementação e estudo.

Adicionalmente, há possibilidades de implementação do Amazon Athena e do Amazon QuickSight proporcionando uma camada de análise de dados acessível e robusta. Essas ferramentas permitiriam a execução de consultas SQL diretamente nos dados armazenados no Amazon S3, simplificando a análise e facilitando a criação de visualizações e dashboards de maneira intuitiva. Um possível próximo passo seria explorar mais as funcionalidades dessas ferramentas e comparar os resultados obtidos com outras soluções de análise de dados disponíveis no mercado.

Em suma, o Data Lake apresenta-se como uma solução concreta e viável a vertente emergente do Big Data, se mostrando capaz de operar e lidar com uma quantidade massiva de dados, permitindo assim melhor ciclo de vida dos dados, desde sua ingestão, até seu resgate analítico, mantendo a interoperabilidade e escalabilidade características da arquitetura. E que apesar dos impeditivos que podem ser decorrentes de questões financeira, a depender do nível da organização, a técnicos, é descomplicado sua construção quando se há um planejamento correto por parte dos interessados

REFERÊNCIAS

AGRAWAL, Divyakant et al. Challenges and opportunities with Big Data 2011-1. 2011.

AL-BARHAMTOSHY, Hassanin M.; EASSA, Fathy E. A Data Analytic Framework for Unstructured Text. Life Science Journal, 2014. p. 340. Ilustração 2.

ALBERTIN, Alberto Luiz; ALBERTIN, Rosa Maria de Moura. A Internet das Coisas irá muito além das Coisas. 2017.

ALEKSANDER. The Rise of Serverless Computing. DOI: https://dl.acm.org/doi/abs/10.1145/3368454. November 2019.

ALVES, Diego M. Começando com Spark. linkedin, 2021. Disponível em: <https://www.linkedin.com/pulse/come%C3%A7ando-com-spark-diego-alves/?originalSubdomain=pt>.

AMAZON. Definição de preço do AWS Lambda. Amazon Web Services, 2023. Disponível em: <https://aws.amazon.com/pt/lambda/pricing/>.

AREDO, Demissie B. Semantics of UML sequence diagrams in PVS. In: UML 2000 Workshop on Dynamic Behavior in UML Models, Semantic Questions, York, UK. 2000.

ARMBRUST, Michael et al. Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. In: Proceedings of CIDR. 2021. p. 8.

ARIFIN, Mohammad Nazir; SIAHAAN, Daniel. Structural and Semantic Similarity Measurement of UML Use Case Diagram. Lontar Komputer: Jurnal Ilmiah Teknologi Informasi, v. 11, n. 2, p. 88, 2020.

AUTORIA PRÓPRIA. Conscientização do impacto ao meio ambiente, 2023. Disponível em: <https://d2li0v5slthobx.cloudfront.net/>.

BATINI, Carlo et al. Methodologies for data quality assessment and improvement. ACM computing surveys (CSUR), v. 41, n. 3, p. 1-52, 2009.

BOTELHO, Marcos César. A LGPD E A PROTEÇÃO AO TRATAMENTO DE DADOS PESSOAIS DE CRIANÇAS E ADOLESCENTES. DIREITOS SOCIAIS E PULITICAS PUBLICAS (UNIFAFIBE). Junho 2020.

CARRION, Patrícia; QUARESMA, Manuela. Internet da Coisas (IoT): Definições e aplicabilidade aos usuários finais. Human Factors in Design, v. 8, n. 15, p. 049-066, 2019.

CASTANHO, Matheus Salgueiro; NACIF, José Augusto Miranda; VIEIRA , Marcos Augusto Menezes. Sistema de processamento de pacotes Serverless. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS (SBRC), 38., 2020, Rio de Janeiro. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2020.

CASTRO, Paul; ISHAKIAN, Vatche; MUTHUSAMY, Vinod; SLOMINSKI, ALEKSANDER. The Rise of Serverless Computing. DOI: https://dl.acm.org/doi/abs/10.1145/3368454. November 2019.

CHEN, Chen et al. Graph OLAP: Towards online analytical processing on graphs. In: 2008 eighth IEEE international conference on data mining. IEEE, 2008. p. 103-112.

CHU, Xu; ILYAS, Ihab F.; PAPOTTI, Paolo. Holistic data cleaning: Putting violations into context. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE). IEEE, 2013. p. 458-469.

CYSNEIROS, Luiz Marcio. Requisitos Não Funcionais: Da Elicitação ao Modelo Conceitual. Pontifícia Universidade Católica do Rio De Janeiro. P. 22.

DAMO, Juan Vinícius Casagrande; LUNARDELLI, Fernando. Sistema de Ingestão de Dados para Data Lakes. Taquara, pg. 5.

DATALAKERS TECNOLOGIA. O que são as zonas de um Data Lake?, 2021. Disponível em: <https://medium.com/datalakers-blog/o-que-s%C3%A3o-as-zonas-de-um-data-lake-5c00af387a4e>. Ilustração 10.

DAVENPORT, Thomas H.; DYCHÉ, Jill. Big data in big companies. International Institute for Analytics, v. 3, n. 1-31, 2013.

DIAS, Jean Miguel F.; RODRIGUES, Rita de Cássia M. C.; PIRES, Daniel Facciolo. A SEGURANÇA DE DADOS NA COMPUTAÇÃO EM NUVENS NAS PEQUENAS E MÉDIAS EMPRESAS. UniFACEF. 2012.

EISMANN, Simon; SCHEUNER, Joel; EYK, Erwin van; SCHWINGER, Maximilian; GROHMANN, Johannes; HERBST, Nikolas; ABAD, Cristina L.; IOSUP, Alexandru. Serverless Applications: Why, When, and How?. Disponível em: <https://arxiv.org/pdf/2009.08173.pdf>. Setembro, 2020. Ilustração 8.

FACCIONI, Mauro. Internet das Coisas. UnisulVirtual: Palhoça, 2016. Cap. 2, p. 29. Ilustração 5.

FARID, Mina et al. CLAMS: bringing quality to data lakes. In: Proceedings of the 2016 International Conference on Management of Data. p. 2089-2092. 2016.

FERREIRA, João et al. O processo etl em sistemas data warehouse. In: INForum. sn, 2010. p. 757-765.

FOX, Stephen; DO, Tuan. Getting real about Big Data: applying critical realism to analyse Big Data hype. International Journal of Managing Projects in Business, v. 6, n. 4, p. 739-760, 2013.

FUENTES-FERNÁNDEZ, Lidia; VALLECILLO-MORENO, Antonio. An introduction to UML profiles. UML and Model Engineering, v. 2, n. 6-13, p. 72, 2004.

Fuentes-Fernández, L., & Vallecillo-Moreno, A. (2004). An introduction to UML profiles. UML and Model Engineering, 2(6-13), 72.

GÄRTNER, Bernhard; HIEBL, Martin RW. Issues with big data. In: The Routledge companion to accounting information systems. Routledge, 2017. p. 161-172.

GARDNER, Stephen R. Building the data warehouse. Communications of the ACM, v. 41, n. 9, p. 52-60, 1998.

GEPPERT, Birgit; SCHMID, Klaus. International Workshop on Requirements Engineering for Product Lines. ALR, v. 2002, 2002.

GU, Xiaodong; ZHANG, Hongyu; ZHANG, Dongmei; KIM, Sunghun. Deep API Learning. The Hong Kong University of Science and Technology. Julho, 2017.

GUIMARÃES, Carla Rocha. Criptografia para Segurança de Dados. Centro Universitário do Triângulo – Unit. Dezembro 2021.

HADOOP VS. SPARK: WHAT'S THE DIFFERENCE? IBM, 2021. Disponível em: < https://www.ibm.com/cloud/blog/hadoop-vs-spark>. Acesso em: 20 de mai. De 2023.

HAI, Rihan et al. Data Lakes: A Survey of Functions and Systems. IEEE Transactions on Knowledge and Data Engineering, 2023.

HAI, Rihan; GEISLER, Sandra; QUIX, Christoph. Constance: An intelligent data lake system. In: Proceedings of the 2016 international conference on management of data. 2016. p. 2097-2100.

HENDRICKSON, Scott; STURDEVANT, Stephen; HARTER, Tyler; VENKATARAMANI, Venkateshwaran; ARPACI-DUSSEAU, Andrea C.; ARPACI-DUSSEAU, Remzi H. Serverless Computation with OpenLambda. University of Wisconsin, Madison 2016.

HE, Xin James. Business intelligence and big data analytics: An overview. Communications of the IIMA, v. 14, n. 3, p. 1, 2014.

HOEREN, Thomas; PINELLI, Stefan. A NOVA LEI BRASILEIRA DE PROTEÇÃO DE DADOS - UMA VISÃO CRÍTICA. PROTEÇÃO DE DADOS PESSOAIS EM PERSPECTIVA: LGPD e RGPD NA ÓTICA DO DIREITO COMPARADO. 2020.

ILYAS, Ihab F. et al. Trends in cleaning relational data: Consistency and deduplication. Foundations and Trends® in Databases, v. 5, n. 4, p. 281-393, 2015.

INMON, William H. Building the Data WareHouse. 4ª. ed. [S.l.]: [s.n.], 2005. Cap. 2, p. 34. Ilustração 6.

IQAIR. Air quality API. IQAir, [S/D]a. Disponível em: <https://www.iqair.com/commercial-air-quality-monitors/api>.

IQAIR. AirVisual API DOC, [S/D]b. Disponível em: <https://api-docs.iqair.com/>.

JANSSEN, Nathalie E. The Evolution of Data Storage Architectures: Examining the Value of the Data Lakehouse. 2022. Dissertação de Mestrado. University of Twente.

JILANI, Atif AA; USMAN, Muhammad; NADEEM, Aamer. Comparative study on DFD to UML diagrams transformations. arXiv preprint arXiv:1102.4162, 2011.

KESSLER, Gary C. An Overview of Cryptography . DOI: https://www.academia.edu/download/38411944/An\_Overview\_of\_Cryptography.pdf. Setembro 1998.

KHAWAR, Z. A., & Cary, E. U. (2001). Developing Enterprise Java Applications with J2EE and UML.

KOREN, ISRAEL; KRISHNA, C. MANI. Fault Tolerant Systems. Morgan Kaufmann Publishers. p. 2-4, 2007.

LI, Xiaoshan; LIU, Zhiming; JIFENG, He. A formal semantics of UML sequence diagram. In: 2004 Australian Software Engineering Conference. Proceedings. IEEE, 2004. p. 168-177.

LIN, P. Paul. What CPAs need to know about Big Data. The CPA Journal, v. 84, n. 11, p. 50, 2014.

LUCIDCHART. UML Use Case Diagram Tutorial. Lucidchart, [S/D]. Disponível em: <https://www.lucidchart.com/pages/uml-use-case-diagram>. Ilustração 12.

MAGRANI, Eduardo. A internet das coisas. BOD GmbH DE, 2021.

MALTBY, Dylan. Big data analytics. In: 74th Annual Meeting of the Association for Information Science and Technology (ASIST). 2011. p. 1-6.

MASHINGAIDZE, Kenneth; BACKHOUSE, Judy. The relationships between definitions of big data, business intelligence and business analytics: a literature review. International Journal of Business Information Systems, v. 26, n. 4, p. 488-505, 2017.

MILOSLAVSKAYA, Natalia; TOLSTOY, Alexander. Big data, fast data and data lake concepts. Procedia Computer Science, v. 88, p. 300-305, 2016.

NARGESIAN, Fatemeh et al. Data lake management: challenges and opportunities. Proceedings of the VLDB Endowment, v. 12, n. 12, p. 1986-1989, 2019.

OUR WORLD IN DATA. Number of people using the Internet, 2023a. Disponível em: <https://ourworldindata.org/grapher/number-of-internet-users?country=~OWID\_WRL>. Ilustração 1.

OUR WORLD IN DATA. Historical cost of computer memory and storage, 2023b. Disponível em: <https://ourworldindata.org/grapher/historical-cost-of-computer-memory-and-storage>. Ilustração 3.

PACÍFICO, Racyus Delano Garcia; DUARTE, Lucas Ferreira Dos Santos; CASTANHO, Matheus Salgueiro; NACIF, José Augusto Miranda; VIEIRA , Marcos Augusto Menezes. Sistema de processamento de pacotes Serverless. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS (SBRC), 38., 2020.

PASUPULETI, Pradeep. Data Lake Development with Big Data. Birmingham: Editora Packt Publishing Ltd, 2015.

PRADO, Eduardo N. A. Data Warehouse: Armazenamento e Análise de Grandes Volumes de Dados. 2015.

QUINLAN, Michael J. Induction of decision trees. Machine learning, v. 1, n. 1, p. 81-106, 1986.

RAUF, Irum; TROUBITSYNA, Elena; PORRES, Ivan. A Systematic Mapping Study of API Usability Evaluation Methods. The Open University. Junho 2019, p.3-4.

RAVAT, Franck; ZHAO, Yan. Data lakes: Trends and perspectives. In: Database and Expert Systems Applications: 30th International Conference, DEXA 2019, Linz, Austria, August 26–29, 2019, Proceedings, Part I 30. Springer International Publishing, 2019. p. 304-313.

REDMAN, Thomas C. The impact of poor data quality on the typical enterprise. Communications of the ACM, v. 41, n. 2, p. 79-82, 1998.

RUSSOM, Philip et al. Big data analytics. TDWI best practices report, fourth quarter, v. 19, n. 4, p. 1-34, 2011.

RUNKLER, Thomas A. Data analytics. Wiesbaden: Springer Fachmedien Wiesbaden, 2020.

SAGIROGLU, Seref; SINANC, Duygu. Big data: A review. In: 2013 international conference on collaboration technologies and systems (CTS). IEEE, 2013. p. 42-47.

SARIKAYA, Alper et al. What do we talk about when we talk about dashboards?. IEEE transactions on visualization and computer graphics, v. 25, n. 1, p. 682-692, 2018. Ilustração 7.

SAWADOGO, Pegdwendé; DARMONT, Jérôme. On data lake architectures and metadata management. Journal of Intelligent Information Systems, v. 56, p. 97-120, 2021.

SAWADOGO, Pegdwendé; DARMONT, Jérôme. On Data Lake Architectures and Metadata, p. 5, 2021. Ilustração 9.

S. Madden, "From Databases to Big Data", IEEE Internet Computing, June 2012, v.16, pg.4-6.

SHENG, Yi Hua; MYKYTYN JR, Peter P. Information Technology Investment and Firm Performance: A Perspective of Data Quality. In: ICIQ. 2002. p. 132-141.

SIVA, Vidhya Vijayakumari; ARUMUGAM, Suganya. A comprehensive survey on big data. Future Computing and Informatics Journal, v. 3, n. 1-2, p. 159-172, 2018.

SIVASHANMUGAM, Karthik; DEVIPRIYA, T. Big Data Analytics: A Review. In: 2014 International Conference on Recent Trends in Information Technology. IEEE, 2014. p. 1-6.

SOMMERVILLE, IAN. Engenharia de Software. Pearson Education do Brasil Ltda. V. 9, p. 203-213, 2011.

STEIN, Brian; MORRISON, Alan. The enterprise data lake: Better integration and deeper analytics. PwC Technology Forecast: Rethinking integration, v. 1, n. 1-9, p. 18, 2014.

STRUCTURED VS. UNSTRUCTURED DATA: WHAT’S THE DIFFERENCE? IBM, 2021. Disponível em: https://www.ibm.com/cloud/blog/structured-vs-unstructured-data. Acesso em: 20 de mai. De 2023.

TSAI, Chun-Wei et al. Big data analytics: a survey. Journal of Big data, v. 2, n. 1, p. 1-32, 2015.

TSAI, Chun-Wei et al. Big data analytics: a survey. Journal of Big data, v. 2, n. 1, p. 1-32, 2015.

TSIOLAKIS, Aliki. Integrating model information in uml sequence diagrams. Electronic Notes in Theoretical Computer Science, v. 50, n. 3, p. 266-274, 2001.

TSIOLAKIS, Aliki. Semantic analysis and consistency checking of UML sequence diagrams. Leiter der Fachbibliothek Informatik, Sekretariat FR 5-4, 2001.

TURINE, Marcelo Augusto Santos; MASIERO, Paulo Cesar. ESPECIFICAÇÃO DE REQUISITOS: UMA INTRODUÇÃO. Instituto de Ciências Matemáticas de São Carlos Universidade de São Paulo São Carlos. Março 1996, p. 8-9.

VAZQUEZ, Carlos Eduardo; SIMÕES, Guilherme. Engenharia de Requisitos: Software Orientado ao Negócio. 2016, p.78 – 79.

VIEIRA André G.; PANTUZA, Gustavo; FREIRE, Jean H. F.; DUARTE, Lucas F. S; PACÍFICO, Racyus D. G.; VIEIRA, Marcos A. M.; VIEIRA, Luiz F. M.; NACIF, José A. M. Computação Serverless: Conceitos, Aplicações e Desafios, 2020. Disponível em: https://sol.sbc.org.br/livros/index.php/sbc/catalog/download/50/232/469-1.

VIEIRA, André G. Computação sem servidor: Uma revisão sistemática. 2019.

WALKER, Coral; ALREHAMY, Hassan. Personal Data Lake with Data Gravity Pull. In: 2015 IEEE Fifth International Conference on Big Data and Cloud Computing. Dalian, China, 2015. p. 160-167.

WHAT IS DATA WAREHOUSE? **IBM**, 2021. Disponível em: https://www.ibm.com/cloud/learn/data-warehouse. Acesso em: 20 de mai. De 2023.

WHAT IS SERVERLESS COMPUTING? **IBM**, 2021. Disponível em: https://www.ibm.com/cloud/learn/serverless. Acesso em: 20 de mai. De 2023.

WHAT IS THE INTERNET OF THINGS (IOT)? **IBM**, 2021. Disponível em: https://www.ibm.com/cloud/learn/internet-of-things. Acesso em: 20 de mai. De 2023.

ZIKOPOULOS, Paul; EATON, Chris. Understanding big data: Analytics for enterprise class hadoop and streaming data. McGraw-Hill Osborne Media, 2011.