MARCOS PAULO FERREIRA RODRIGUES

USO DO ADD E ARQUITETURA DE REFERÊNCIA BIG DATA NO PROCESSO DE DESIGN DE UMA ARQUITETURA DE SISTEMA DE CONSOLIDAÇÃO DE DADOS

São Paulo
2024

MARCOS PAULO FERREIRA RODRIGUES

USO DO ADD E ARQUITETURA DE REFERÊNCIA BIG DATA NO PROCESSO DE DESIGN DE UMA ARQUITETURA DE SISTEMA DE CONSOLIDAÇÃO DE DADOS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Especialização em Engenharia de Dados e Big Data.

São Paulo
2024

MARCOS PAULO FERREIRA RODRIGUES

USO DO ADD E ARQUITETURA DE REFERÊNCIA BIG DATA NO PROCESSO DE DESIGN DE UMA ARQUITETURA DE SISTEMA DE CONSOLIDAÇÃO DE DADOS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Especialização em Engenharia de Dados e Big Data.

Área de Concentração:

PECE

Orientador:

Msc. Ana Claudia Rossi

São Paulo
2024

AGRADECIMENTOS

Aos meus familiares e amigos pelo companheirismo durante a vida.

À República Maternidade por toda acolhida e vivência.

À Stephanie por todo amor e companheirismo compartilhado nos últimos anos.

Ao Chico e a Brisa pela pureza do ser vivo.

À minha orientadora Ana por toda paciência e suporte.

À USP pela excelente qualidade do ensino.

"Nenhum homem pode banhar-se duas vezes no mesmo rio... pois na segunda vez o rio já não é o mesmo, nem tão pouco o homem!"

-- Heráclito de Éfeso

RESUMO

O mercado de pagamentos brasileiro cresce a cada ano que se passa, atingindo recorde histórico de aproximadamente 120 milhões de pagamentos diários com cartão. No ecossistema do mercado de pagamentos, destacam-se as credenciadoras, que são responsáveis por processar os pagamentos relacionados as vendas dos estabelecimentos comerciais. Entre 2010 e 2023, houve um crescimento de 1250% no número de concorrentes no mercado de credenciamento, influenciando na necessidade das empresas obterem insights rápidos e dinâmicos sobre seus dados, e para isso, é priorizado o desenvolvimento de requisitos funcionais enquanto os requisitos não funcionais são ignorados ou adicionados futuramente ao sistema. Portanto, neste trabalho propomos a concepção e elaboração de uma arquitetura de sistema de big data para uma credenciadora projetada para atender a atributos de qualidade essenciais (como desempenho, escalabilidade e extensibilidade), a partir da aplicação do método Attribute Driven Design 3.0 em conjunto com a metodologia de desenvolvimento Scrum para produzir o design de arquitetura de software se adaptando rapidamente às mudanças de negócios, e instanciamos o NIST Reference Architecture para designar as responsabilidades de cada componente do sistema. A vantagem deste método é que evidenciamos um processo iterativo para tratar tanto requisitos funcionais quanto requisitos não funcionais. Como resultado, obtivemos um design de arquitetura de sistema de big data componentizado que contempla tanto os atributos de qualidade propostos quanto os casos de uso primários, oferecendo uma abordagem prática para a implementação de sistemas de big data destacando a importancia de metodologias e arquiteturas referenciais,

Palavras-Chave – Big Data, Arquitetura de Referência NIST, Design Orientado a Atributo 3.0, Arquitetura de Software

ABSTRACT

The Brazilian payments market has been growing every year, reaching a record high of approximately 120 million daily card payments. In the payments market ecosystem, the acquirers stand out, as they are responsible for processing payments related to sales at commercial establishments. Between 2010 and 2023, there was a 1250% growth in the number of competitors in the acquirer market, influencing the need for companies to obtain fast and dynamic insights into their data. To this end, the development of functional requirements is prioritized while non-functional requirements are ignored or added to the system in the future. Therefore, in this work we propose the design and elaboration of a big data system architecture for an acquirer agency designed to meet essential quality attributes (such as performance, scalability and extensibility), from the application of the Attribute Driven Design 3.0 method in conjunction with the Scrum development methodology to produce the software architecture design that adapts quickly to business changes, and we instantiate the NIST Reference Architecture to designate the responsibilities of each system component. The advantage of this method is that we evidence an iterative process for dealing with both functional and non-functional requirements. As a result, we obtained a componentized big data system architecture design that addresses both the proposed quality attributes and the primary use cases, offering a practical approach to the implementation of big data systems highlighting the importance of reference methodologies and architectures.

Keywords – Big Data, NIST Reference Architecture, Attribute Driven Design 3.0, Software Design

LISTA DE FIGURAS

LISTA DE TABELAS

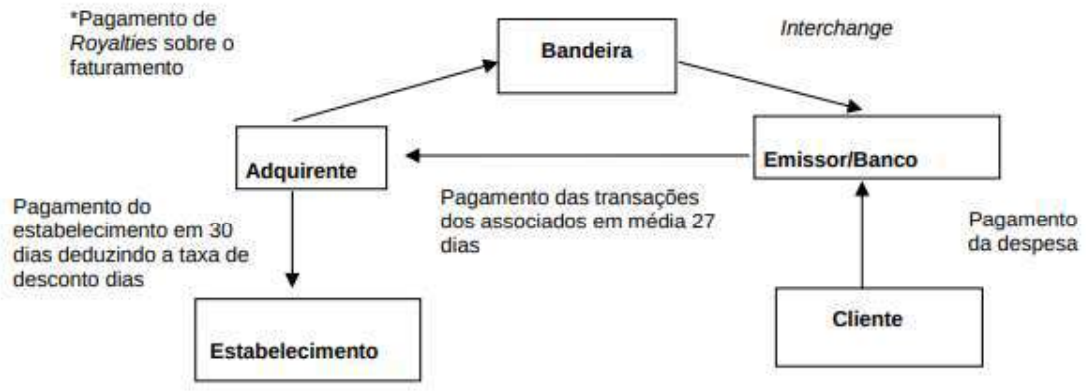
SUMÁRIO

1 INTRODUÇÃO

O mercado de pagamentos brasileiro cresce a cada ano que se passa. Só no primeiro semestre de 2024 foram mais de 22 bilhões de operações de cartão que totalizaram 2 trilhões de reais, o que representa um crescimento de 10,3% na quantidade e 11,2% no volume transacionado, e atingindo o recorde histórico de aproximadamente 120 milhões de pagamentos diários com cartão, segundo a Associação Brasileira das Empresas de Cartão de Crédito e Serviços - ABECS (Serviços (s.d.)). Em complemento, de acordo com a Federação Brasileira de Bancos - FEBRABAN - as transações PIX, desconsideradas pela ABECS, cresceram 61% em quantidade e registram 12 trilhões de reais durante o primeiro semestre de 2024 (Febraban (s.d.)).

Este ecossistema do mercado de pagamentos é composto por agentes como clientes, lojistas, emissores de cartão, bandeiras e credenciadoras, sendo os clientes e lojistas os clientes finais e os demais são considerados agentes da industria, de acordo com Furini (2020). A Figura 1 apresenta a dinâmica do negócio de cartões de crédito, de acordo com Lorey (2008).

Figura 1: Dinâmica do Negócio de Cartões de Crédito.



Fonte: Lorey (2008)

Os clientes são pessoas ou empresas interessados em adquirir um produto ou serviço, enquanto que os estabelecimentos comerciais são, em sua maioria, empresas que vendem ou alugam determinado produto ou serviço. Os emissores de cartão são tradicionalmente bancos que oferecem crédito a clientes por meio de cartões, e possuem comunicação pa-

dronizada através das bandeiras. Por fim, as credenciadoras são as empresas responsáveis por intermediar a comunicação entre o estabelecimento comercial e a bandeira (Lorey (2008), Furini (2020)).

Dentro deste ecossistema, destacam-se as credenciadoras, que são responsáveis por processar os pagamentos relacionados as vendas dos estabelecimentos comerciais, oferecendo alternativas de pagamento como cartão de crédito, pix, boletos, link de pagamento, etc. De acordo com o Banco Central do Brasil (Brasil (s.d.)), em 13 anos houve um crescimento de 1250% na quantidade de concorrentes no mercado de credenciamento, onde em 2010 o mercado se concentrava em dois principais competidoroes, que resultava em um alto indice de monopólio de mercado, e finalizou 2023 com mais de 25 empresas no mercado e obtendo um índice de concentração considerado moderado de acordo com as definições do Banco Central.

A entrada de novas credenciadoras implicou em alterações nas estratégias comerciais e de negócio das credenciadoras tradicionais do setor, como por exemplo o fim de alugueis de equipamentos, que passaram vender seus equipamentos aos lojistas para processarem suas transações, ou até mesmo oferecer tais equipamentos sem custo (Brasil (s.d.)). Essas decisões são tomadas com base no contexto, premissas e suposições, guiados através do objetivo da própria decisão. Enquanto o contexto e suposições dizem respeito a aspectos fora do controle da credenciadora, as premissas e conhecimentos sobre a empresa são dependentes de seus próprios dados corporativos.(Diván (2017)).

Para se obter o conhecimento corporativo sobre a empresa (como os principais clientes, periodos com mais vendas, etc), é necessário processar os dados dados internos dessa empresa, ou seja, definir o conjunto de regras a serem aplicadas em cada etapa de transformação do dado. Para o processamento, fatores como o volume de dados a ser processado e a velocidade com que esses dados são gerados são relevantes.

Younas (2019) explica que big data se refere ao volume massivo de dados diversos que são gerados em larga escala e processados em uma alta velocidade. Desta maneira, sistemas de big data contemplam inicialmente três novas caracteristicas, conhecidas como 3V (volume, variedade e velocidade). O volume diz respeito a quantidade massiva de dados gerados, a variedade se refere aos diferentes tipos de dados que podem ser usados em conjunto, e a velocidade contempla o quão veloz os dados são gerados, processados e movidos entre os sistemas.

Diante dos cenários, considerando o avanço do setor nos últimos anos e a contínua expansão do mercado de credenciadoras, percebemos que eventos externos tendem a impli-

car cada vez mais em alterações nas estratégicas internas e nas orientações para tomadas de decisões. Dado esta volatilidade, torna-se interessante para as empresas obterem insights rápidos e dinamicos sobre seus dados, para que possam concentrar seus esforços em relacionar as análises com o contexto e as suposições externas.

1.1 Problema

O problema de acompanhamento de dados de maneira rápida e dinâmica para a obtenção de insights no contexto de uma credenciadora pode ser reduzido a outros problemas da mesma natureza, como por exemplo, extrair indicadores pré definidos para acompanhamento de campanhas de marketing, ou acompanhar a adesão de determinados perfis de grupos a um determinado produto.

Os desafios de big data são detectar, antecipar e predizer informações com a granularidade mais refinada possível. O problema é que esses sistemas geralmente atuam em processamentos de lotes (batchs), e que são capazes de prover diversos insights sobre o histórico dos dados, mas não são capazes de identificar e tratar os dados do presente a medida que os eventos sejam recebidos em realtime (Soumaya et al. (2017)), limitando a capacidade de resposta às necessidades do negócio.

Conforme sistemas de big data evoluem, a integração com novas fontes de dados apresenta desafios, como a variação de formatos de dados, terminologias distintas e estruturas dos dados de novos sistemas (Rozony et al. (2024)). Além disso, para reduzir o tempo de publicação de um produto, o desenvolvimento é focado mais em execuções do que em requisitos de engenharia, e desta forma, são priorizados os requisitos funcionais, enquanto os requisititos não funcionais são ignorados ou adicionados futuramente no ciclo de desenvolvimento (Sachdeva e Chung (2017)).

Esses problemas são resolvidos com a uma solução comum: projetar um sistema de big data planejado para atender e se adaptar a atributos de qualidades, como se integrar com outros sistemas, processar seus dados de maneira ágil e garantir a disponibilidade dos serviços.

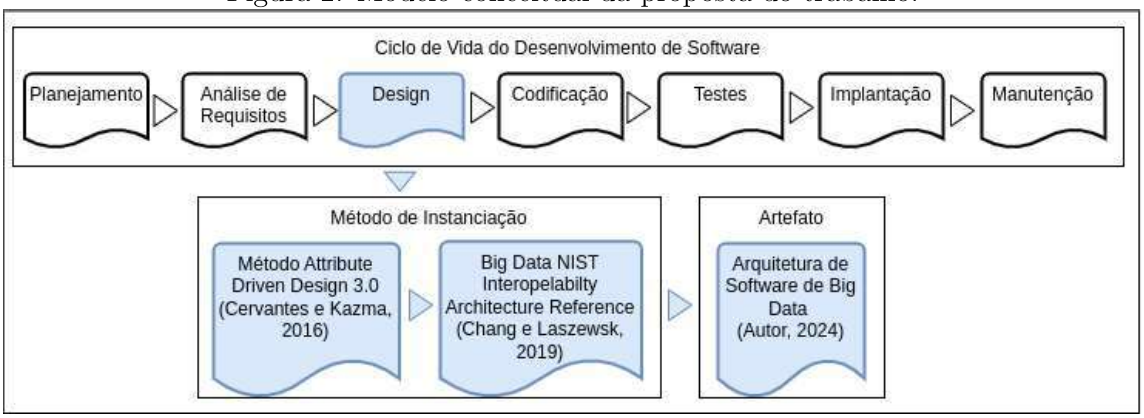
Dessa forma, o problema a ser tratado nesta monografia é como projetar sistemas de big data que atendam a atributos de qualidade essenciais, como desempenho, escalabilidade e extensibilidade, que são requisitos de negócio críticos para tomadas de decisão e eficaz no cenário corporativo atual.

1.2 Objetivo

Este trabalho tem como objetivo geral projetar uma arquitetura de software de big data para uma credenciadora, que acompanhe e forneça indicadores pré-definidos em near realtime. A arquitetura deve ser modular e eficaz com o tratamento de requisitos não funcionais críticos, como escalabilidade, performance e extensibilidade.

O Design Science Research (DSR) (Wieringa (2014)) contribui para analisar e elucidar o contexto e o problema, sendo a primeira fase do nosso projeto de pesquisa. Este esforço temporário auxilia na identificação dos produtos gerados a partir desta pesquisa. O método Attribute Driven Design 3.0 (ADD 3.0) (Cervantes e Kazman (2016)) orienta a definição dos drivers da arquitetura do software, bem como sugere estratégias de interações para atingir o objetivo do projeto. Por fim, o NIST Big Data Reference Architecture (NBDRA) (Chang, Boyd e Levin (2019)) auxilia na instanciação da arquitetura de software de big data, apoiando com os componentes necessarios para a elaboração, construção e execução do projeto. Portanto, o resultado deste trabalho é uma arquitetura de software de big data construída a partir da aplicação do método ADD 3.0 e instanciada a partir do NBDRA. A Figura 2 apresenta o modelo conceitual da proposta deste trabalho.

Figura 2: Modelo conceitual da proposta do trabalho.



Fonte: o Autor

Iremos, portanto, trabalhar na etapa de Design do ciclo de vida do desenvolvimento do software. Nesta etapa, utilizamos a estrutura proposta pelo ADD 3.0 (Cervantes e Kazman (2016)) para identificação e definição dos drivers de arquitetura do projeto, e assim, levantamos os casos de uso, cenários de atributos, restrições e preocupações arquitetônicas, com o intuito de definir os direcionamentos da arquitetura de design. Em seguida, definimos e instanciamos a arquitetura de referência conforme o modelo de arquitetura proposto por Chang, Boyd e Levin (2019).

Sendo assim, a proposta de arquitetura tem como escopo entregar métricas pré-definidas (contador/quantidade, somatório, valor máximo, valor mínimo, valor médio, desvio padrão médio e moda) em tempo próximo de near-realtime, sendo implementado utilizando tecnologias opensource e capaz de conectar drivers para diferentes fontes de dados. Desta maneira, os objetivos especificos deste trabalho são:

- Aplicar método que ensine como produzir um design de arquitetura: estudar metodologias que direcionem na elaboração e construção de um design de arquitetura.
- Identificar arquitetura de referência que se enquadre no contexto: analisar arquiteturas de referência que satisfaçam os drivers arquiteturais do projeto
- Indicar metodologia que se adapte à necessidade de rapida mudança do negócio: escolher metodologia de desenvolvimento que contemple iterações rápidas e dinamicas para se adequar as necessidades de mudança de negócio

1.3 Justificativa

O volume e variedade crescente dos dados gerados pelas empresas têm contribuído para adoção de arquiteturas de big data escaláveis e performáticas. Porém, projetos dessas arquiteturas apresentam desafios significativos, como a modularidade, que embora seja um princípio fundamental da engenharia de software, pode ser desafiador em sistemas de big data devido a heterogeneidade dos dados e a necessidade de processamento em larga escala. Bass, Clements e Kazman (2021) diz que uma arquitetura é o primeiro passo na criação de um software que pode endereçar os requisitos de qualidade, sendo o mapeamento das funcionalidades frente às estruturas de software o fator determinante das qualidades suportadas pela arquitetura.

Reis (2019) apresenta uma arquitetura para Big Data as a Service em nuvem privada. Em sua proposta é usado e estendido alguns modelos como Big Data as a Service, NIST Interoperability Framework, Cloud Computing e Arquitetura em Camadas. As analises e resultados foram baseados em quatro métricas: a) conformidade com a definição de Big Data as a Service; b) adequação ao ambiente em nuvem privada; c) conformidade com arquiteturas de referencia consolidadas; d) performance do sistema. Com base nesses pontos, os resultados comprovaram que a proposta é adequada para utilização em ambiente corporativo, pois além de se obter desempenho comparável aos datacenters tradicionais, a arquitetura apresenta redução de custo e operação automatizada.

Mcheick, Qi e Charara (2011) utiliza o método Attribute Driven Design (ADD) para prover um modelo de design para um conector em uma arquitetura distribuida baseado em MVC (Model-View-Controller). Seu estudo contempla os passos definidos pelo método, em uma única interação, resultando em um modelo de design para um design de conectores, exemplificado em um caso de uso baseado na arquitetura distribuida MVC. O modelo permitiu o projetar e desenvolver facilmente os conectores e vincular de forma coerentes as camadas de Visualização e Modelo, além de atender aos requisitos funcionais e de qualidade dos conectores. Dessa forma, podemos entender que o ADD contribuí para a projeção de desenvolvimento em camadas que viabilizem requisitos e qualidades.

Este trabalho, portanto, visa contribuir para abordagem que combina as práticas de engenharia de software com técnicas de arquitetura de sistemas distribuídos a fim de desenvolver aplicações de big data modulares e escaláveis que garantam atendam determinados requisitos não funcionais como performance, extensibilidade e escalabilidade. Para analisarmos os resultados, avaliaremos se o design da arquitetura do software projetado atende aos requisitos não funcionais estabelecidos através dos drivers de arquitetura.

1.4 Metodologia de Pesquisa

Para o desenvolvimento deste trabalho, foi selecionado a metodologia Design Science Research (DSR), que propõe quatro principais fases para analisar o problema e o contexto, além de resolver os problemas de design e responder as questões de conhecimento. Este método de pesquisa busca uma solução de um problema do mundo real através do desenvolvimento de um artefato a ser projetado sob um contexto, segundo Wieringa (2014). As fases do DSR estão apresentadas na Figura 3.

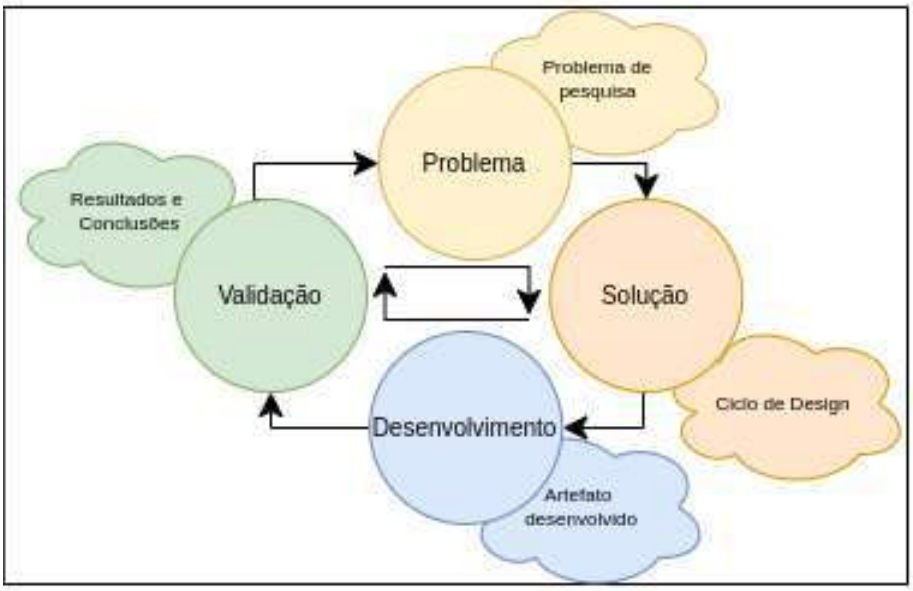
Para Wieringa (2014), as fases do projeto de pesquisa são:

- Problema: é investigado o problema de fato. Ou seja, pode ser identificado os stakeholders, a diferença do projeto em relação a outros projetos, os objetivos dos envolvidos, dentre outros.

- Solução: é especificado os requisitos. Por exemplo, além das suposições em relação ao contexto, há a validação em relação ao que esta especificado e como contribui para os objetivos dos stakeholders.

- Desenvolvimento: é desenvolvido o artefato. Nesta etapa, o projeto de pesquisa é executado com base no problema e na solução, e se obtem como resultado o produto

Figura 3: Fases do Projeto de Pesquisa do DSR.



Autor: Wieringa (2014)

gerado a partir da pesquisa.

- Validação: é analisado os efeitos do artefato no contexto. Sendo assim, nesta etapa validamos se os efeitos do artefato frente ao contexto da pesquisa, e analisamos se os efeitos satisfazem os requisitos,

Sendo assim, investigamos o problema de pesquisa, identificando seus atores e suas necessidades. Em seguida especificamos os requisitos que atendam as necessidades dos stakeholders, e seguimos para o desenvolvimento do artefato, cujo resultado é o design de arquitetura de software, validado na última etapa do método conforme análise de satisfação dos requisitos.

2 BIG DATA E ARQUITETURA DE REFERÊNCIA

Neste capítulo, é apresentado a fundamentação teórica desta monografia, ou seja, a revisão das pesquisas e discussões de outros autores sobre o tema deste trabalho. A seção 2.1 contextualiza o que é big data e sistemas de big data, seguido pela seção 2.2 que explica o NIST Big Data Reference Architecture (NBDRA). A seção 2.3 conceitua o que é design de software e por fim a seção 2.4 introduz o Attribute Driven Design 3.0.

2.1 Big Data e Sistemas de Big Data

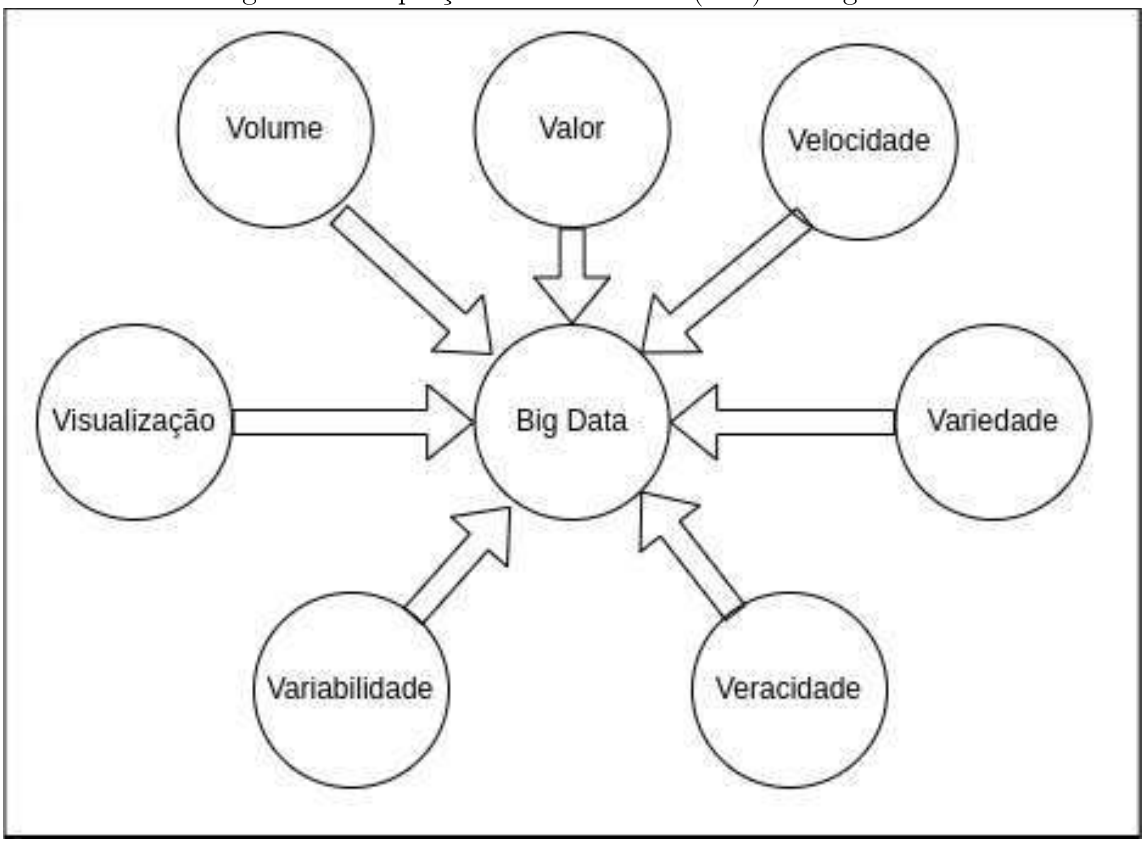
A literatura atual possuí várias definições para big data. Conjuntos de dados em larga escala são a combinação de conjuntos de dados grandes, complexos, diversos e heterogeneos, gerados a partir de várias origens, como transações comerciais, redes sociais, sensores, fluxo de cliques, etc. Uma abordagem comparativa para definir big data é: conjuntos de dados em larga escala que estão além das capacidades de gerenciamento de dados convencional, e com métodos analíticos para capturar, armazenar, acessar, gerenciar, compartilhar, processar, analisar e visualizar estes dados em tempo aceitável (Rao et al. (2019)).

A geração de dados em larga escala causa tres grandes desafios: volume, velocidade e variedade, também conhecidos como modelo 3V para big data. Futuramente, o modelo se estendeu ao modelo 5V, adicionando as caracteristicas de veracidade e valor, e alguns autores contemplam o modelo 7V, com as características variabilidade e visualização (Rao et al. (2019)). A Figura 4 apresenta uma adaptação aos 7Vs do Big Data.

De acordo com Rao et al. (2019), as dimensões das caracteristicas de big data podem ser descritas como:

- **Volume** se refere a magnitude dos conjuntos de dados em larga escala, ou seja, o tamanho dos tipos distintos de dados adquiridos de diversas fontes de dados
- **Velocidade** é a taxa com que o dado é recebido e refinado para fins analíticos. Redes sociais como Facebook e Youtube geram fluxo de dados continuos através de fotos e videos compartilhados

Figura 4: Adaptação às 7 dimensões (7Vs) do Big Data



Fonte: Adaptado de Rao et al. (2019)

- **Variedade** são os tipos distintos de representação de dados, como estruturados, semi-estruturados e não estruturados, como dados tabulares ou relacionais (estruturados), vídeos ou músicas (não estruturados) e formatos como XML, JSON e HTML (semi-estruturados)

- **Veracidade** se refere aos ruídos, anormalidades e vieses dos dados, como inputs incorretos e processos aleartórios

- **Valor** se refere a uma dimensão na persectiva do negócio, como ganhos operacionais, reduções de custos, etc.

- **Variabilidade** representa as mudanças dinamicas nas taxas de fluxo de dados, ou seja, o rápido crescimento da velocidade de dados gera caminhos imprevisiveis, e gerenciar essas variações é um desafio

- **Visualização** que representa a visualização de insighs intuitivos escondidos em estruturas de dados, que são contribuidores na tomada de decisão estratégica para geração de novos valores de negocio.

Quantos aos requisitos destes tipos de sistemas, podem ser difividos em requisitos funcionais e requisitos não funcionais (NFR). Os requisitos funcionais dizem respeito às funcionalidades do sistema, enquanto Rahman e Reza (2020) aborda requisitos não funcionais como métricas importantes no design de software visto que definem sintaxe, semantica, restrições e protocolos para cada atividade do software. Em caso do sistema ser grande e complexo, como sistemas de big data, então NFRs especificos de domínio e arquitetonicamente significativos (como performance, conformidade, confiabilidade, segurança e usabilidade) devem ser identificados antes de projeto o produto real.

2.2 NIST Big Data Reference Architecture

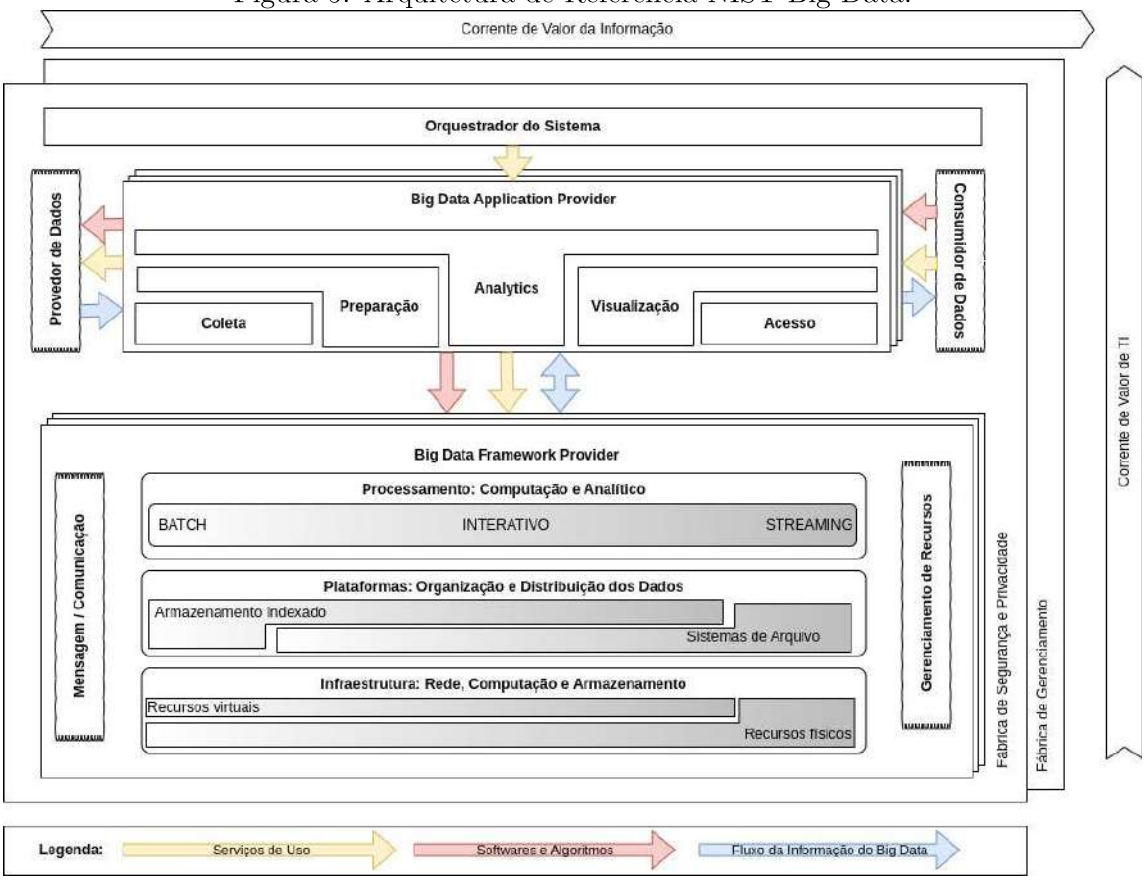
O NIST Big Data Reference Architecture (NBDRA) é uma arquitetura de referência para Big Data que provê uma linguagem em comum para vários stakeholders e melhora o entendimento dos diversos componentes, processos e sistemas de Big Data em um contexto de modelo conceitual de Big Data, de acordo com Chang e Laszewski (2019).

O NBDRA é dividido em cinco componentes principais, onde cada componente representa diferentes funções técnicas dentro do sistema de big data. Para Chang e Laszewski (2019), os componentes funcionais são: orquestrador do sistema, provedor de dados, consumidor de dados, provedor de aplicação de big data e provedor de framework de big data. A arquitetura contempla duas outras atribuições, denominadas fábrica de segurança e privacidade e fábrica de gerenciamento, que se integram com todos os componentes da arquitetura. A Figura 5 sumariza os requisitos para a interface dos componentes do NBDRA.

O autor define o Orquestrador do Sistema como responsável por definir a propriedade geral, a governança, as funções políticas e monitoramento do cumprimento dos requisitos do sistema de Big Data. Algumas atividades para esse componente incluem: a) propriedade do negócio, ou seja, definir stakeholders e atividades dos componentes funcionais); b) governança, isto é, definir as políticas e processos de governaça geral do sistema conforme definição dos stakeholders); c) arquitetura do sistema, sendo assim, definir os requisitos gerais e diretrizes técnicas que deverão estar contemplados na arquitetura do sistema; d) ciência de dados, atividades no geral com diversas definições de requisitos que geralmente incluem algoritmos individuais, mineiração de dados ou até mesmo aplicações e; e) segurança e privacidade;

No que diz respeito ao Big Data Application Provider, este é dividido em cinco sub-

Figura 5: Arquitetura de Referência NIST Big Data.



Fonte: Chang e Laszewski (2019)

componentes, que são a coleta, a preparação, o analytics, a visualização e o acesso. As definições e responsabilidades de cada etapa está descrito na Tabela 1 a seguir.

O Big Data Framework Provider, segundo Chang, Boyd e Levin (2019), é responsável por oferecer suporte ao gerenciamento e comunicações entre seus subcomponentes (Processamento, Plataforma e Infraestrutura) e os demais componentes do sistema. O componente de Comunicação oferece mecanismos que suportam a comunicação entre as camadas de Aplicação e de Framework enquanto que o componente de Gerenciamento aloca os recursos para as atividades. Esta camada inclui também os subcompoentes de Infraestrutura (que oferece as funções computação, armazenamento e rede necessárias), Plataforma (que gerencia e distribui os dados em um sistema de big data) e Processamento (que descreve como o dado será processado no suporte da aplicação do sistema de big data)

As atividades para a fábrica de gerenciamento, no que diz respeito ao gerenciamento dos sitemas, pode se concentrar em atividades como: a) configuração de responsabilidade e rastreabilidade do dado; b) gerenciamento e automatização no uso dos recursos do

Tabela 1: Responsabilidades do componente Big Data Application Provider.

Componente	**Definições e Responsabilidades**
Coleta	A coleta lida com a(s) interface(s) do provedor de dados. Sua principal função é receber os dados a serem processados. Também é nesta camada que os metadados iniciais são criados.
Preparação	A preparação é onde as primeiras transformações acontecem, como a validação de dados, limpeza, padronizações, etc. Esta camada também pode agregar dados de diferentes provedores de dados.
Analytics	Esta etapa implementa as técnicas para extração de conhecimento dos dados, incluindo a codificação de baixo nível da regra de negócio do sistema de big data. Pode ser dividida em multiplos módulos para executarem parte especifica da regra de negócio.
Visualização	A visualização prepara os elementos processados para apresentação ao consumidor de dados. O objetivo deste módulo é formatar e apresentar o dado de uma maneira que otimize o entendimento e conhecimento.
Acesso	O acesso lida com a(s) interface(s) do consumidor de dados. Sua principal foco é comunicar e interagir com os consumidores. Esta camada também confirma que os metadados descritivos e administrativos e seus esquemas são capturados e mantidos para acesso pelo consumidor de dados a medida que os dados são trafegados.

Fonte: Chang, Boyd e Levin (2019)

ambiente de big data; c) monitorar atividades para mitigação no suporte operacional e; d) gestão automatizada de pacotes para suporte de deploys e configurações. No que diz respeito a gestão do ciclo de vida do big data, este verifica se o dado está sendo trabalhado corretamente pelos demais componentes do NBDRA, e para isso, precisa abordar assuntos como gestão de politicas, metadados, acessibilidade e manutenção, além de recuperação de dados (Chang, Boyd e Levin (2019)).

Quanto as atividades para a fábrica de segurança e privacidade, essas fornecem a gestão dos acessos do sistema de big data e seus serviços. Este componente tem três principais responsabilidades associadas: a) autenticação, para garantir que quem ou o que está acessando algo é quem diz ser; b) autorização, para garantir que o usuário ou processo tenha direitos para acessar os recursos e serviços e; c) auditoria, garantir o armazenamento de eventos que ocorrem no sistema para suportar analises futuras (Chang, Boyd e Levin (2019)).

Por fim, Chang e Laszewski (2019) coloca que, além do NBDRA ser organizado em torno das cinco principais atribuições, suas multiplas subatribuições são alinhadas em torno da corrente de valor de informação e de tecnologia. O valor da informação é criado com a coleta dos dados, integração, analises e aplicação dos resultados, enquanto que o

valor da tecnologia é criado através da rede, infraestrutura, plataformas, ferramentas de aplicação e outros serviços de TI. A intersecção entre ambos valores se encontra na camada de Big Data Application Provider, indicando que a análise de dados e implementação geram valor para os stakeholders em ambas as visões (informação e tecnologia).

2.3 Design de Software

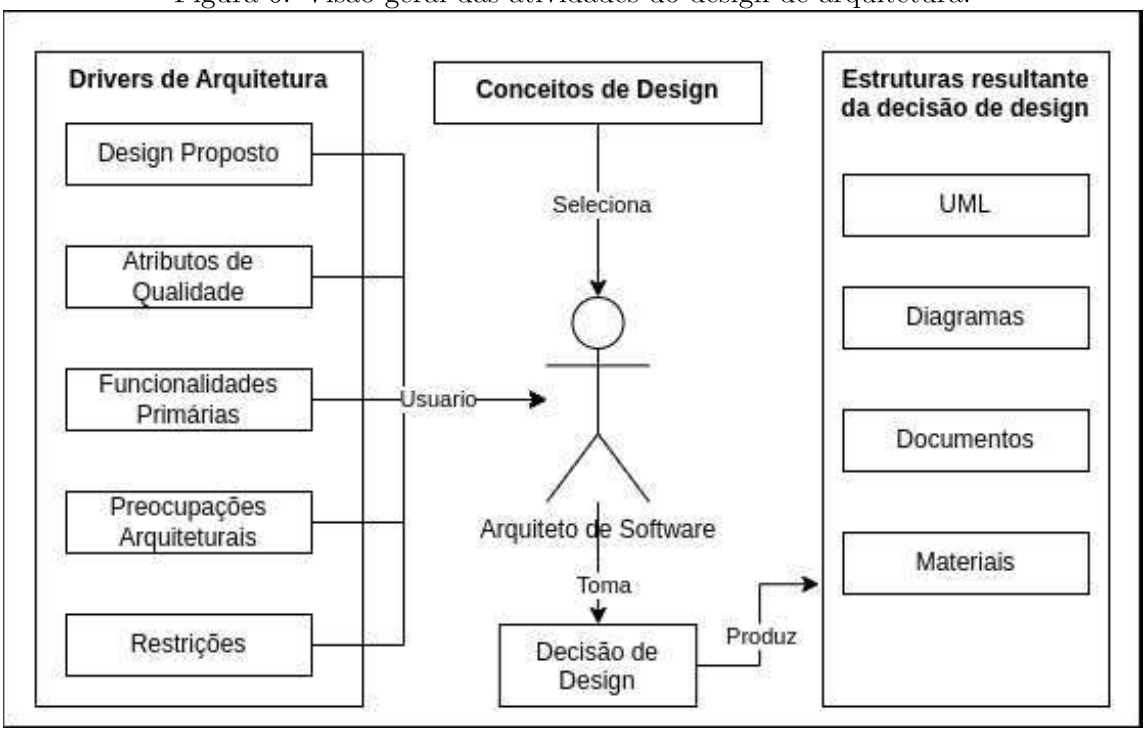
Para Cervantes e Kazman (2016), o design é um processo em que o resultado é a criação do proprio design, ou seja, das descrições do que se deseja construir como versão final. Projetar este design significa tomar decisões que atinjam os objetivos considerando habilidades e materiais disponíveis e que satisfaçam os requisitos e restrições. Portanto, o resultado do processo de design pode ser uma coisa, um substantivo, ou um artefato que será eventualmente implementado. No que diz respeito a design de arquitetura de sistemas de softwares, a definição se mantem, ou seja, as decisões são tomadas para transformar os drivers de arquitetura em estruturas, que são utilizadas para guiar o projeto. Os drivers de arquitetura contemplam a proposta do design, requisitos, restrições e preocupações arquiteturais, e neste sentido, são os orientadores das tomadas de decisão durante o processo de design (Cervantes e Kazman (2016)).

O design de arquitetura é o passo essencial para atingir o produto e, consequentemente, os objetivos do projeto, que podem ser técnicos (como atingir baixos indices de latência em um vídeo-game) ou não-técnicos (como buscar entrar em um novo mercado). Dado o objetivo, a escolha de arquiteturas de referências particulares podem prover bons fundamentos para atingí-los (Cervantes e Kazman (2016)). Por exemplo, a escolha da arquitetura de referência ETL pode contribuir para resolver um problema de processamento de grandes volumes de dados durante o processamento batch, porém pode não contribuir para extração de indicadores em realtime considerando restrições de custo.

A Figura 6 apresenta a visão geral das atividades do design de arquitetura segundo Cervantes e Kazman (2016).

Desta forma, o arquiteto de software, com base nos drivers de arquitetura levantados a partir das necessidades dos usuários, seleciona conceitos de design que satisfaçam os drivers mencionados, decide o melhor candidato dos conceitos apresentados e produz as estruturas que documentam as decisões tomadas.

Figura 6: Visão geral das atividades do design de arquitetura.



Fonte: Cervantes e Kazman (2016)

2.4 Attribute Driven Design 3.0

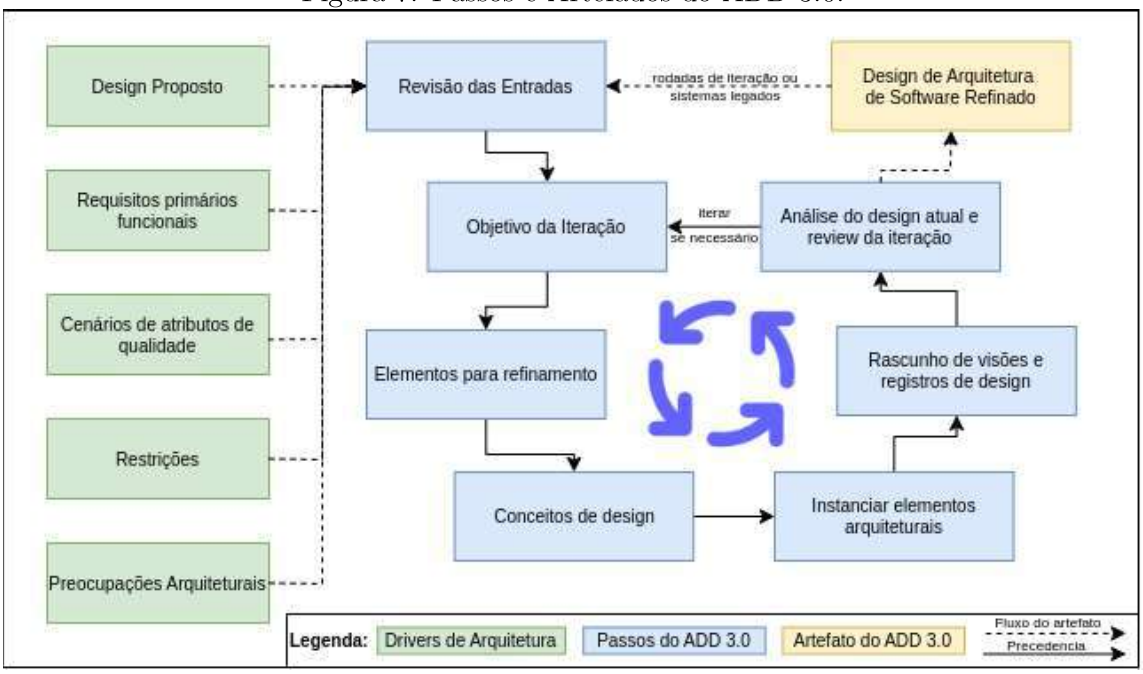
Nesta seção, apresentaremos o Attribute Driven Design 3.0, um método de Design de Arquitetura que provê um guia do que é necessário para se produzir um design adequado, conforme Cervantes e Kazman (2016).

O principal diferencial deste método em comparação com outros semelhantes é a sua capacidade de elaborar um guia detalhado, passo a passo, de como as tarefas devem ser performadas dentro de uma interação de design. O autor propõe uma interação de design como sendo um grupo de decisões de projeto através das quais um subconjunto dos drivers é transformado em estruturas. Para além do diferencial, o ADD foi o primeiro método proposto com foco especialmente em atributos de qualidade e sua obtenção através da seleção de diferemtes tipos de estruturas. Outra importante contribuição é o reconhecimento da importância da analise e documentação durante todo processo de design Cervantes e Kazman (2016).

A Figura 7 apresenta os artefatos do método, bem como o passo a passo a ser executado. Sendo assim, abordaremos as etapas de definição dos drivers, o passo a passo em cada etapa do processo, e o resultado do trabalho, ou seja, o design da arquitetura de

software.

Figura 7: Passos e Artefados do ADD 3.0.



Fonte: Cervantes e Kazman (2016)

Para Cervantes e Kazman (2016) as etapas de definição dos drivers são:

- Design proposto: nesta etapa se identifica a motivação do desenvolvimento do sistema, esclarecendo os objetivos a serem atingidos.
- Requisitos primários funcionais: definidas como funcionalidades críticas para obter os objetivos de negócio que motivaram o desenvolvimento do sistema.
- Cenários de Atributos de qualidade: são pequenas descrições de como o sistema deve responder à determinadas ações. São construidos a partir de atributos de qualidade, que são propriedades mensuráveis ou testáveis do sistema, definidas a partir dos requisitos funcionais e não funcionais, e usadas para indicar o quanto o sistema satisfaz as necessidades dos stakeholders.
- Restrições: uma restrição é uma decisão sobre a qual o arquiteto tem pouco ou nenhum controle, sendo assim, podem ser tecnologias mandatórias, sistemas que necessitam ser integrados ou interoperabilizados, leis que devem ser cumpridas, habilidades e disponibilidades dos desenvolvedores, prazos não negociáveis, compatibilidades com versões mais antigas de sistemas, e semelhantes

- Preocupações Arquiteturais: são aspectos adicionais que necessitam ser considerados como parte da arquitetura de design, mas não são expressas como requisitos tradicionais, como alocação de times, autenticação/autorização, logs, configurações, gestão de dependências, e semelhantes.

Com os drivers de arquitetura definidos, os requisitos para iniciar o ciclo do método estão cumpridos, portanto, a Tabela 2 abaixo detalha as principais atividades de cada etapa do método

Tabela 2: Ciclo de iteração proposto pelo método Attribute Driven Design.

Descrição	**Atividades**
1: Revisão das Entradas	Garantir que os drivers definidos e elucidados estão disponíveis e corretos.
2: Objetivo da Iteração	Estabelecer o objetivo da iteração através da seleção de drivers.
3: Elementos para refinamento	Escolher um ou mais elementos do sistemas para refinar. Esses refinamentos podem significar decomposição em elementos de maior granularidade, combinação de elementos em um de menor granularidade, ou melhoria de elementos previamente identificados
4: Conceitos de design	Escolher um entre conceitos de design que satisfaçam os drivers selecionados. Requer a identificação de alternativas entre os conceitos de design que podem ser usados para atingir o objetivo da iteração
5: Instanciar elementos arquiteturais	Instanciar os elementos arquiteturais, alocar responsabilidades e definir interaces. Além de instanciar os elementos, os elmentos também precisam estar conectados
6: Rascunhar visões e registros de design	Esboçar visualizações e registrar decisões de design. É necessário garantir que as visualizações sejam preservadas. Assim, as atividades de design para a iteração estão finalizadas
7: Analisar design e review de iteração	Analisar o design atual e revisitar o objetivo da iteração e o atingimento da proposta de design. É importante analisar o estado da arquitetura para avaliar a necessidade de novos ciclos de design.

Fonte: Cervantes e Kazman (2016)

O processo pode ser cíclico a partir da revisão das entradas para cada driver, porém é um desafio quando consideramos restrições como tempo e recursos, portanto, é importante endereçar os drivers com altas prioridades, Idealmente, deve ser garantido que os drivers criticos estão satisfeitos, ou ao menos que o design é bom o suficiente para satisfazê-los (Cervantes e Kazman (2016)).

3 PROPOSTA DO USO DO ADD COM NBDRA COMO ARQUITETURA DE REFERÊNCIA

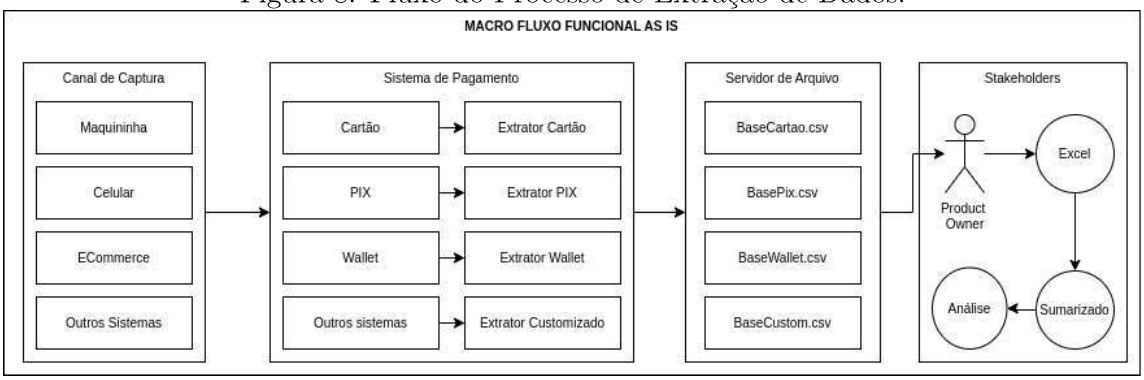
Neste capítulo, será abordado o desenvolvimento deste projeto. Iremos aplicar o Attribute Driven Design com o foco na primeira interação do método, onde é proposto a arquitetura inicial do sistema. Como solução de arquitetura inicial, criaremos uma solução com base na arquitetura de referência NIST Big Data Interoperability Framework. A seção 3.1 descreve o contexto da aplicação, seguido pela seção 3.2 que aborda o método de desenvolvimento para este trabalho. Em seguida na seção 3.3 e por fim discutimos os resultados na seção 3.4

3.1 Contexto da Aplicação

Como caso de uso do nosso contexto, consideraremos o cenário de uma Credenciadora de porte médio, que possui aproxidamente 180 funcionários e 250.000 clientes e um catalogo de 120 produtos. Esta credenciadora provê serviços relacionados a pagamentos a seus clientes, que em sua grande maioria, são estabelecimentos comerciais. Para além de processar pagamentos, esta empresa coleta e processa dados gerados de diversas fontes (como sistemas transacionais, logs de servidor, tráfegso de rede, etc).

Inicialmente, apresentaremos o macro fluxo atual do negócio, com as definições de cada entidade que participa do processo. A Figura 8 apresenta o macro fluxo funcional atual, com algumas sugestões de stakeholders interessados na análise de dados.

Figura 8: Fluxo do Processo de Extração de Dados.



Fonte: o Autor

O Canal de Captura é o sistema responsável por receber o pagamento, ou seja, por qual canal é possível se concretizar o pagamento. Sendo assim, é possível se efetivar vendas através de Maquininhas (comumente chamadas POS ou TEF), de Celular (TapOnPhone), de ECommerce (como sistemas de marketplace), e diversos outros sistemas.

O Sistema de Pagamento é a forma de pagamento, por exemplo, pagamento com Cartão (onde o sistema transacional do cartão se integra com os sistemas das bandeiras), PIX (onde o sistema transacional se integra com o Banco Central), Wallet (onde o sistema transacional se integra com sistemas e bancos de dados internos), entre outros. Cada um desses sistemas podem ter altas e baixas taxas de transmissão em horários distintos, portanto, cada sistema é responsável pela sua própria extração.

Uma vez os dados extraídos, são direcionados a um servidor de arquivo, onde é obtido pelos Stakeholders interessados e pode começar a ser processados conforme os indicadores de cada área.

Com o dado coletado, se torna possível analisá-lo. Um dos candidatos interessados em analisar estes dados é a área de negócios, que precisa acompanhar, por exemplo, o faturamento dos clientes. Para que a análise seja feita, primeiramente os dados coletados precisam ser acessíveis, e entao, passar por um ou mais processos de transformação, até que se obtenha o resultado desejado.

No processo de disponibilização dos dados coletados, é comum que sistemas transacionais, durante os horários de alta taxas de transmissão, priorizem o processamento transacional, e durante horários com taxas de transmissão reduzidas, os sistemas competem entre processar o transacional e extrair os dados ainda não extraídos para enviar às áreas interessadas.

Sendo assim, no melhor dos casos, as áreas recebem os dados com o atraso de um dia. Apesar deste cenário possibilitar alguns acompanhamentos sobre o passado, se torna inviável quando é necessário explorar um contexto mais dinâmico e que necessite de informações do presente, como por exemplo, acompanhar o crescimento de vendas de um produto durante o lançamento da sua primeira programanda anunciada. Neste exemplo, a necessidade é de que, sabendo do horário da propaganda, seja possível acompanhar, em tempo real, as consequencias deste marketing, como aumento na quantidade e volumes transacionados de seus clientes.

Credenciadoras têm, dentre diversos departamentos, as equipes responsáves pelo Desenvolvimento de Produtos, que desenvolvem novas soluções e acompanham indicadores dos Produtos Digitais ofertados pela empresa. Durante o acompanhamento de indicado-

res, é comum as equipes responsáveis pelo gerenciamento destes produtos obterem seus relatórios com atrasos, sendo no cenário mais otimista o atraso diário e no cenário pesimista o atraso mensal. Desta maneira, levantamos alguns potenciais stakeholders que, de alguma maneira, são afetados pelo problema, apresentados na Tabela 3 abaixo.

Tabela 3: Sugestão de stakeholders envolvidos no projeto.

ID	**Stakeholder**	**Motivo**
1	Gestores e Executivos de Produtos	Atualmente, há demora de no mínimo 1 dia para extração de métricas assertivas sobre um produto
2	Parceiros e Estabelecimentos	Parceiros e Clientes podem ter dificuldade de obter suporte a respeito do produto

Fonte: o Autor

3.2 Método de Desenvolvimento

Este trabalho foi desenvolvido através da aplicação do método de Design de Arquitetura ADD 3.0 em conjunto com o NIST Big Data Framework Interoperability, nomeado como ADD-S. Durante o processo de design da arquitetura para novos sistemas, a primeira iteração tem como objetivo a elaboração da estrutura superficial do sistema, bem como a definição de arquiteturas de referência que satisfaçam os drivers e a instanciação da arquitetura selecionada. Para além das discussões - tratadas no detalhamento da respectiva iteração - a arquitetura de referência escolhida foi o NIST Big Data.

O método de desenvolvimento adotado foi o processo de Metodologia Ágil, sendo implementado através do método Scrum. Durante a aplicação do método, identificamos correlações entre os componentes do ADD 3.0 e os artefados do Scrum, apresentadas na Tabela 4 abaixo.

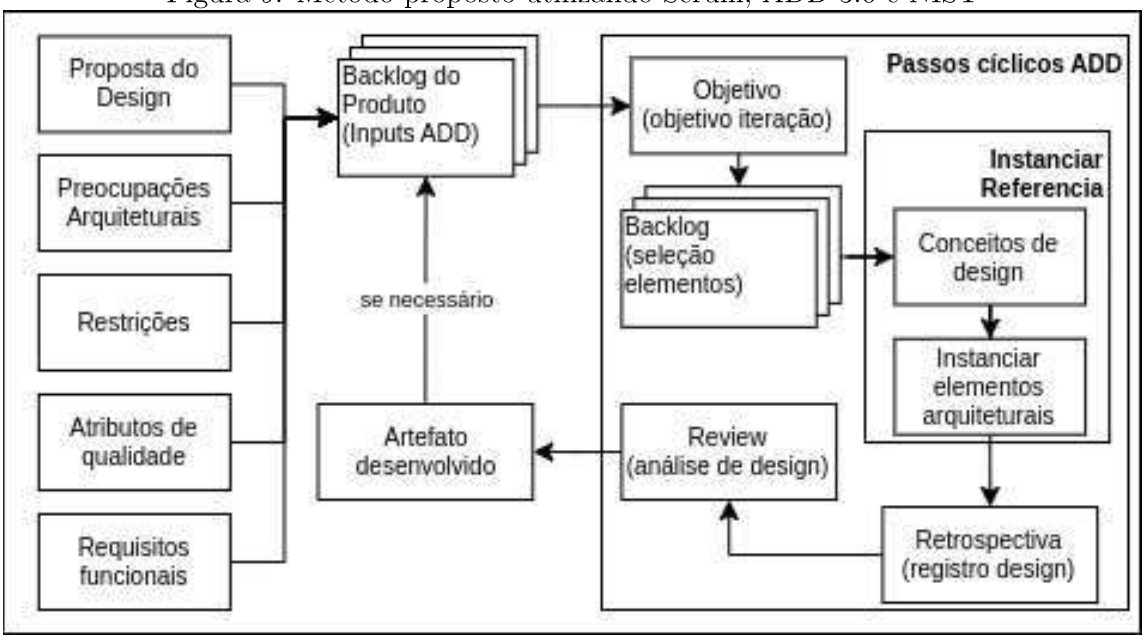
Tabela 4: Correlação proposta entre componentes Scrum e ADD 3.0

Artefato SCRUM	**Correspondência no ADD 3.0**
Backlog do Produto	Revisão das entradas/Drivers Arquiteturais
Histórias de Usuário	Cenários de Atributos de Qualidade
Sprint	Passos 2 à 7 do método ADD
Objetivo da Sprint	Objetivo da Iteração
Backlog da Sprint	Seleção de Elementos para Refinamento
Retrospectiva da Sprint	Rascunho de visões e registros de design
Review da Sprint	Review da Iteração e Análise de Design
Incremento do Produto	Design de Arquitetura de Software Refinado

Fonte: o Autor

A Figura 9 ilusta o método aplicado no desenvolvimento deste trabalho, instanciando o scrum com as devidas correspondências sugeridas.

Figura 9: Método proposto utilizando Scrum, ADD 3.0 e NIST



Fonte: o Autor

O backlog do produto existente no Scrum é definido como as revisões das entradas, elaborados através dos drivers de arquitetura como requisitos funcionais, atributos de qualidade, restrições e preocupações, visando atingir os objetivos de negócio esclarecidos na proposta de design. As histórias de usuário serão construídas de acordo com os cenários de atributos de qualidade, isto pois, uma vez que estes cenários estejam definidos, suas propriedades mensuráveis são muito próximas da descrições de histórias de usuário tradicionais.

A Sprint diz respeito aos passos cíclicos do ADD 3.0. Da mesma maneira que a Sprint busca um incremento do produto ao final de um ciclo do seu processo, ao final de cada interação do método ADD 3.0 espera-se obter um design de arquitetura de software refinado em relação ao design contemplado no inicio do processo. Os demais artefatos, então, são elaborados para cada iteração proposta no método ADD 3.0, sendo assim, o objetivo da sprint é definido no segundo passo da interação através do objetivo da iteração, enquanto o backlog da sprint é selecionado no terceiro passo da iteração a partir da seleção de elementos para refinamento.

O desenvolvimento das tarefas para concluir o backlog da sprint se concentra no quarto passo (escolha de conceitos de design) e quinto passo (instanciar elementos arquiteturais), seguido da retrospectiva da sprint, associada a sexta etapa do método ADD 3.0, onde os rascunhos de visões e registros de design são armazenados. A review da sprint, então, é

atribuida ao sétimo e último passo do método, onde é analisado o design atual e revisitado os objetivos da iteração e atigingimento da proposta de design.

Desta forma, considerando as definições dos drivers de arquitetura apresentadas na seção 3.3, temos os seguintes passos para o método:

- 1º: Definir o conjunto que compõe o backlog do produto
- 2º: Definir o objetivo da iteração/sprint
- 3º: Definir subconjunto do backlog do produto que contemple o objetivo da iteração/sprint
- 4º: Buscar conceitos de design que satisfaçam o backlog da iteração
- 5º: Instanciar elementos da arquitetura e definir responsabilidades
- 6º: Registrar as decisões tomadas durante as etapas anteriores
- 7º: Analisar o resultado de design da arquitetura para verificar que contempla o backlog da iteração
- 8º: Caso o design de arquitetura atual não contemple todo backlog, retomar ao 1º passo.

3.3 Aplicação do Método

Inicialmente, será elucidado os requisitos do sistema, conforme descrição dos problemas abordados, e elaborado os casos de uso primários. Então, definiremos os cenários de atributos de qualidade, as preocupações arquitetônicas e as restrições do sistema. Desta maneira, será definido os drivers de arquitetura, que serão os orientadores das decisões que serão tomadas, buscando evitar retrabalho. Com os Drivers de Arquitetura definidos, daremos sequência à primeira interação proposta pelo método ADD 3.0, que busca estabelecer a estrutura superficial do sistema e sua arquitetura de referência.

3.3.1 Requisitos do sistema

As atividades de elicitação de requisitos foram realizadas com base no contexto e problemas abordado previamente. Os requisitos mais importantes serão apresentados nas próximas subseções, e são compostos por conjuntos de Casos de Uso Primario, Cenários de Atributo de Qualidades, Restrições e Preocupações Arquiteturais, conforme aplicação do Attribute-Design Driven 3.0

3.3.2 Casos de uso primários

Os principais casos de uso primários estão descritos na Tabela 5 abaixo:

Tabela 5: Casos de uso primários

Caso de Uso	**Descrição**
UC-01: Conectar fonte de dados sistemica	COMO **desenvolvedor** QUERO conectar meu sistema transacional ao sistema de big data PARA disponibilizar os dados em near realtime para extração de relatório
UC-02: Fontes já cadastradas	COMO **gerente de sistemas** QUERO obter informações das fontes já cadastradas no sistema PARA listar as tabelas disponíveis para extração de relatório
UC-03: Relatório de fontes selecionadas	COMO **gerente de negócio** QUERO selecionar uma fonte já cadastrada no sistema PARA extrair os indicadores do sistema por data, hora e minuto do evento.
UC-04: Histórico	COMO **product owner** QUERO manter os dados históricos por cinco anos PARA estar adequado ao compliance
UC-05: Métricas do sistema	COMO **executivo de negócio** QUERO selecionar uma fonte de dados alimentada em realtime já cadastrada no sistema PARA calcular os indicadores de quantidade, soma, moda, desvio padrão médio e valores máximos, mínimo e médio
UC-06: Coleta fonte de dados ad-hoc	COMO **product owner** QUERO realizar upload do meu arquivo excel PARA disponibilizar os dados em até um minuto para extração de relatório.
UC-07: Segurança dos dados	COMO **data owner** QUERO permitir o acesso de um usuário a uma tabela PARA garantir que somente usuários autenticados acessem o sistema.

Fonte: o Autor

3.3.3 Restrições

A primeira restrição considera questões financeiras, portanto, devemos priorizar o uso de componentes opensource. A segunda restrição está atribuida ao caso de uso UC-01, que aborda a conexão à novas fontes de dados sistémicas. As restrições associadas estão sumarizadas e apresentadas na Tabela 6 a seguir.

Tabela 6: Restrições

ID	**Restrição**
RES-01	O sistema deve ser elaborado utilizando componentes opensource (por razões financeiras)
RES-02	O sistema deve possibilitar integração com sistemas legados

Fonte: o Autor

3.3.4 Cenário de Atributo de Qualidade

Os cenários de atributos de qualidade mais relevantes estão listados na Tabela 7 abaixo. Para cada cenário de atributo de qualidade, identificamos os seus casos de uso associados

Tabela 7: Cenários de Atributo de Qualidade

ID	**Atributo de Qualidade**	**Cenário**	**Caso de Uso**
QA-1	Performance	O sistema deve armazenar as informações brutas coletadas em near realtime (até 5 segundos).	UC-01,05
QA-2	Performance	O sistema deve armazenar as informações de processamento ad-hoc em até 1 minuto.	UC-01, 06
QA-3	Escalabilidade	O sistema deve armazenar 5 anos de dados brutos (aproximadamente 5GB por dia, 10TB no total).	UC-03
QA-4	Escalabilidade	O sistema deve armazenar dados agregados por minuto (por 12 meses), por hora (por 24 meses) e por dia (por 60 meses).	UC-03
QA-5	Extensibilidade	O sistema deve suportar adição de novas fontes de dados via integração API, sem interferência na coleta de dados.	UC-01
QA-6	Performance	O sistema deve prover relatórios near real time para usuários de negócio, com agregações por minuto e até 1 minuto de latência	UC-01, 03, 06
QA-7	Extensibilidade	O sistema deve prover informação, por meio de API, sobre o catálogo de dados cadastrados como fontes de dados.	UC-01
QA-8	Extensibilidade	O sistema deve realizar operações de CRUD em metadados.	UC-01, 02, 06
QA-9	Segurança	O sistema deve garantir a autenticidade e autorização dos usuários.	UC-09

Fonte: o Autor

3.3.5 Preocupações Arquiteturais

Definimos três preocupações arquiteturais para este sistema que contemplam a construção da arquitetura de uma solução arquitetural, o controle de acesso (especificado na UC-09) e o conhecimento da equipe de desenvolvimento. As preocupações arquiteturais iniciais estão apresentadas na Tabela 8 a seguir.

Tabela 8: Preocupações Arquiteturais Inicias

ID	Preocupação Arquitetural
PRE-01	Estabelecer uma solução arquitetural inicial para um novo sistema
PRE-02	Garantir que os acesso dos dados sejam controlados por sistemas de Autenticação e Autorização
PRE-03	Aproveitar os conhecimentos pessoais e profissionais sobre o ecossistema da Apache para Big Data

Fonte: o Autor

3.3.6 O processo de design

Uma vez com os requisitos definidos e enumerados, seguiremos com a primeira iteração do método Attribute-Driven Design. Este é um sistema desenvolvido do zero, sendo assim, inicialmente revisaremos as entradas do nosso sistema e em seguida aplicaremos as interações propostas pelo método.

3.3.6.1 Passo 1: Revisão das Entradas

O primeiro passo do método consiste na revisão das entradas. Estes estão sumarizados na Tabela 9 a seguir.

Tabela 9: Revisão de Entradas

Categoria	Detalhes
Design proposto	Este é um novo sistema em um domínio relativamente maduro. O desenvolvimento seguirá os padrões adotados no processo Ágil com curtas iterações para coleta rápida de feedback.
Casos de Uso	Os casos de usos designados como primários estão descritos a partir das histórias de usuário UC-01, UC-02, UC-03, UC-04, UC-05 e UC-06
Atributos de Qualidade	Os atributos de qualidade são Performance (QA-1, QA-2 e QA-6), Escalabilidade (QA-3 e QA-4) e Extensibilidade (QA-5, QA-7, QA-8)
Restrições	As restrições estão esclarecidas na descrição de RES-01 e RES-02.
Preocupações Arquiteturais	As preocupações arquiteturais serão consideradas conforme descritas em PRE-01, PRE-02 e PRE-03.

Fonte: o Autor

3.3.6.2 2º Passo: Estabelecendo o objetivo da iteração através da seleção de drivers

No momento, nosso design de arquitetura contempla exclusivamente os drivers de arquitetura, portanto, como primeiro objetivo, definiremos a Estrutura Superficial do Sis-

tema e a Arquitetura de Referência (PRE-01) com base nestes drivers, com o intuito de conseguirmos visualizar o fluxo proposto pela solução e estabelecer uma solução arquitetural inicial para o sistema.

Com o objetivo definido de estabelecer uma solução arquitetural inicial para o sistema, a preocupação da iteração está centrada nas restrições e atributos de qualidade abaixo:

- RES-01, 02: Utilização de componentes opensearch e extensível para integração com sistemas legados
- PRE-01, 02, 03: Estabelecer a arquitetura do sistema, garantindo Autenticidade e Autorização e com aproveitamento do ecossistema da Apache para Big Data
- QA-01, 02, 06: Performance para armazenamento e extração de relatórios
- QA-03, 04: Escalabilidade para armazenamento de grandes volumes de dados
- QA-05, 07, 08: Extensibilidade para integração com sistemas legados

3.3.6.3 3º Passo: Escolher um ou mais elementos do sistema para refinar

Definimos o objetivo da iteração na etapa anterior, e neste caso, identificamos e categorizamos o design da arquitetura como proposto para um novo sistema em domínio maduro. Como o sistema é categorizado como desenvolvido do zero em domínio maduro, algumas preocupações arquiteturais associadas ao design de arquitetura são bem conhecidas, suportadas e bem documentadas, e portanto, podemos aproveitar tais preocupações para atingir o objetivo proposto da iteração. Portanto, uma vez definido a arquitetura inicial base do sistema como artefato a ser entregue ao final da iteração, iremos considerar todo o sistema para ser refinado.

3.3.6.4 4º Passo: Escolher um ou mais conceitos de design que satisfaçam os drivers selecionados

Para a escolha de conceitos de design que satisfaçam os drivers selecionados, trabalharemos com três alternativas, apresentando os pontos vantajosos e desvantajosos de cada uma para justificar a seleção da escolha do melhor conceito de design. As alternativas contemplam analises quanto a Arquitetura ETL, Arquitetura Lambda e Arquitetura NIST. Entendemos que a arquitetura ETL não atende plenamente os requisitos de realtime, enquanto a arquitetura Lambda não satisfaz totalmente os drivers relacionados

a extensibilidade. Por fim selecionamos a arquitetura NIST que contempla os cenários levantados.

Inicialmente estudamos a arquitetura ETL, que consiste em realizar a Extração, Transformação e Carga dos dados. A solução é viável como uma solução inicial, considerando que não haverá alterações no escorpo do projeto, isto pois alguns dos desafios desta arquitetura é a escalabilidade e manutenção, pois a medida que novos conectores são identificados, haverá necessidade de implementar as regras de ETL para cada novo conector. Além disso, a arquitetura é financeiramente custosa e possui deficiencias significativas em relação ao processamento realtime.

As deficiencias da arquitetura no questido de processamento realtime nos direcionou a estudar a arquitetura lambda, que oferece em sua referência instruções de como garantir o processamento realtime e batch das aplicações. Esta solução, apesar de aparentar uma solução de arquitetura viável, traz dois principais pontos de cuidados que resultaram em seu descarte: a) duplicidade de código, visto que a arquitetura propõe camadas apartadas para os processamentos batchs e realtime; b) o esforço aplicado na arquitetura para atender a preocupação PRE-02 deve ser duplicado, uma vez que deve ser aplicado em cada camada proposta pela referência.

Uma vez que descartamos a arquitetura lambda, nos direcionamos a uma arquitetura mais robusta, que atenda plenamente, além do cenário batch/realtime, as restrições e preocupações arquiteturais. Desta forma, avaliamos a arquitetura NIST Big Data Interoperability Framework. Esta referência, assim como a arquitetura lambda, possibilita o processamento realtime e batchs das aplicações. Os principais fatores desta arquitetura que influenciaram positivamente em sua adoção foram: a) a unificação de código, pois diferente da arquitetura lambda não há necessidade de separar o desenvolvimento em duas camadas, contribuindo para os requisitos não funcionais de extensibilidade, e; b) a contemplação nativa de protocolos de segurança e governança de dados, contribuindo para os atributos de qualidade de segurança e escalabilidade e garantindo a cobertura de todos os drivers de arquitetura propostos.

3.3.6.5 5º Passo: Instanciar os elementos da arquitetura, alocar responsabilidades e definir interfaces

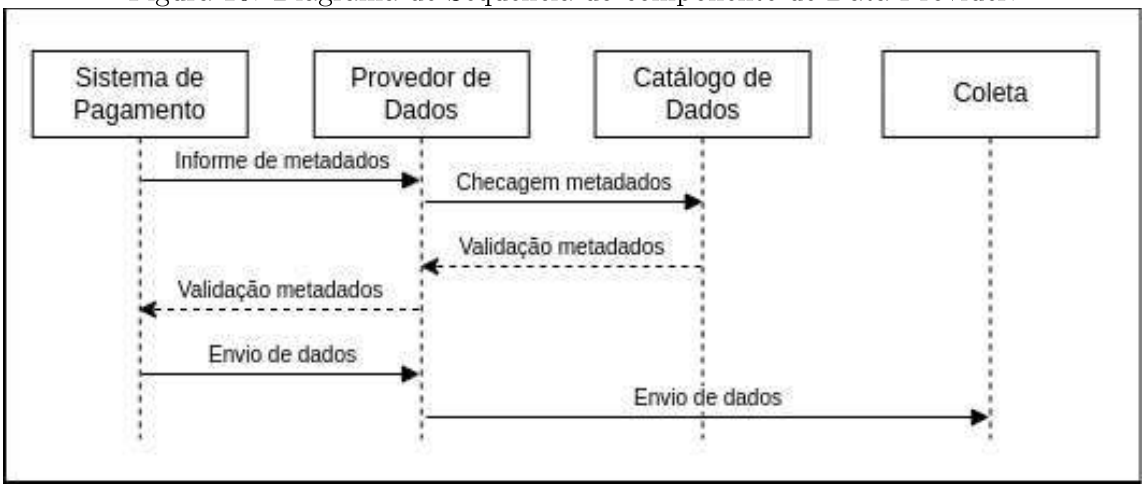
Nesta etapa, definiremos todos os elementos da arquitetura, bem como definiremos as responsabilidades de cada componente e as interfaces de comunicação entre eles. Iremos trabalhar com a instancia dos componentes conforme arquitetura NBDRA com o objetivo

de entregarmos, ao final deste processo, a elaboração e definição da arquitetura do nosso sistema.

Como foco desta monografia é o desenvolvimento de uma aplicação de big data, iremos direcionar os esforços no desenvolvimento da camada Big Data Application Provider proposto pelo NBDRA. Será considerado que as demais camadas já estão instanciadas em uma plataforma de big data, e neste caso, o nosso trabalho será documentar as interfaces de comunicação com essas camadas. Desta forma, iremos definir as responsabilidades e interfaces das etapas de coleta, preparação/cura, análise, visualização e acesso.

Iniciamos, então, pelo componente de Provedor de Dados, cuja principal função é permitir que outros sistemas se integrem com a aplicação de big data para processarem seus dados, contribuindo para o UC-01 e UC-06. Para isso, são definidos duas interfaces de comunicação, uma que diz respeito a gestão de metadados e outra para a coleta de dados. A Figura 10 apresenta o diagrama de sequência para o fluxo de Data Provider/Integration.

Figura 10: Diagrama de Sequencia do componente de Data Provider.



Fonte: o Autor

O Sistema de Pagamento é o sistema gerador de informação, ele é o responsável por disponibilizar os dados transacionais que serão processados pela aplicação de big data. Desta maneira, inicialmente o sistema disponibiliza as informações dos metadados que serão processados (i.e dicionario de dados, donos dos dados, sistemas geradores de informação, nomes de bancos e tabelas, etc) para o Provedor de Dados, que verifica se os metadados estão cadastros no sistema e, caso negativo, realiza o cadastro e retorna as informações cadastradas ao sistema transacional. O sistema transacional, com acesso aos metadados, inicia o processo de envio dos dados, vinculando cada dado ao seu metadado no sistema. O componente provedor de Dados então disponibiliza os dados para a camada

de Coleta.

Para os componentes funcionais desta camada, o Sistema de Pagamento é qualquer sistema que possa se comunicar utilizando o protoclo HTTP, pois a camada Data Provider neste caso diz respeito a um serviço disponibilizado por meio de API Rest, instanciada através da biblioteca Express do framework NodeJS. A camada Catalogo de Dados diz respeito a componentes instanciados pelo ambiente de big data, e neste caso, sua implementação é através do Apache Atlas. Quanto ao sistema de mensageria, responsável pelo fluxo de envio de dados do Data Provider ao Collection, é instanciado a partir do Apache Kafka.

Para o caso de uso dos metadados, temos principalmente quatro informações essenciais, como o nome do banco de dados e da tabela onde os dados estão inseridos, o responsável por este dado (em outras palavras, o dono do dado ou data owner), e um dicionário que representa todos os campos da tabela, com a definição de nome do campo, tipo do dado, categoria e valor padrão. Uma vez que o sistema transacional envie estes dados para o data provider, a camada processa o hashcode deste metadado e realiza a validação perante o sistema de catalogo, e retorna o ID referente ao hashcode processado. O JSON (JavaScript Object Notation) referente a esta interface está definido abaixo.

```
Input: {
    metadata: {
        nomeBancoDeDados : string,
        nomeTabela: string,
        responsavel: UUID,
        dicionario: {
            [nomeCampo: string]: {
                tipoDado: string|integer|double|boolean,
                categoria: "quantitativo"|"qualitativo"|null,
                valorPadrao: string|integer|double|boolean|null,
                required: boolean,
                primaryKey: boolean,
                foreignKey: boolean,
                partitionKey: boolean
            }
        }
    }
```

```
}
Output: { key: string }
```

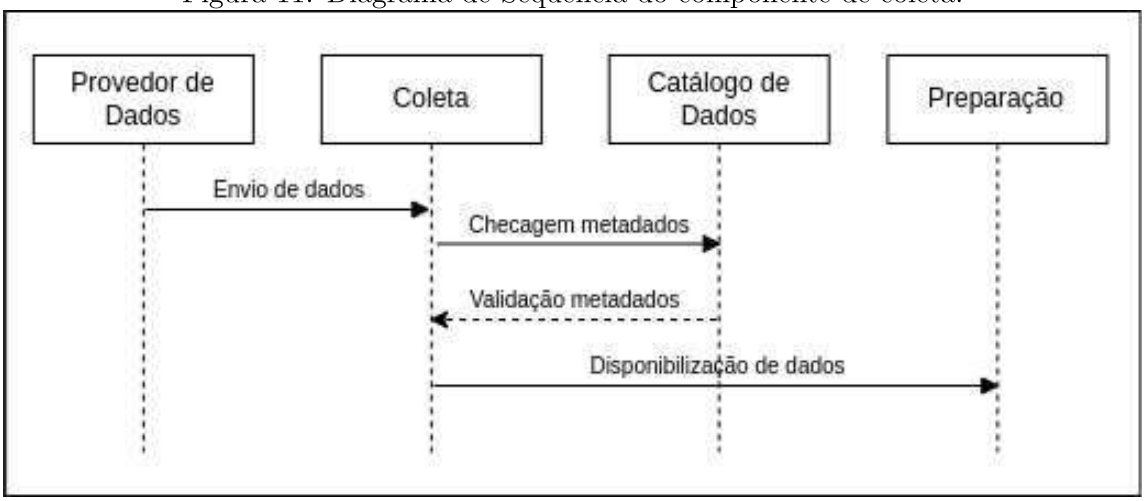
Uma vez que a camada transacional recebe a informação da key referente ao dicionário, esta informação deve ser repassada para o componente de integração em todo envio de novos dados. O envio por parte do sistema transacional se dará por meio de requisições HTTP, enquanto que o envio por parte da camada de integração se dará através de sistemas de mensageria, e deverá implementar a seguinte interface para publicação de seus eventos

```
Input: {
    metadata: {
        key : string,
        timestamp: timestamp
    },
    data: {
        [fieldName: string]: string|double|boolean|null
    }
}
```

No que diz respeito a coleta dos dados, a principal responsabilidade deste componente é validar os metadados e disponibilizar os dados dentro do ambiente da aplicação do big data para a etapa de preparação, contribuindo para os casos de uso referentes à fontes cadastradas (UC-02 e UC-03). Para isso, esta camada recebe dados da camada Provedor de Dados, realiza a validação de metadados e encaminha para a próxima etapa de processamento. A Figura 11 apresenta o diagrama de sequência para o fluxo do componente de coleta.

Inicialmente, o Provedor de Dados disponibiliza as informações dos dados que serão processados, bem como o hashcode dos metadados para validação. Em seguida, a camada de coleta realiza a checagem destes metadados, ou seja, verifica se os metadados estão cadastrados no sistema e os dados trafegados estão de acordo com o dicionário do metadado, e com a validação concluida, repassa os dados e seus metadados para o componente de preparação. Caso a validação não seja satisfeita, o evento é armazenado em uma base distinta para análise futura. Em ambos os casos, não há output na comunicação entre o Provedor de Dados e a Coleta, pois a comunicação é feita de maneira assincrona e de acordo com os protocolos estipulados pelo padrão Producer/Consumer.

Figura 11: Diagrama de Sequência do componente de coleta.



Fonte: o Autor

Quanto aos componentes funcionais desta camada, propomos a utilização de frameworks de processamento de dados em streaming, como o Apache Spark e o Apache Flink. Para a transferência de dados, é proposto a utilização do Apache Kafka como sistema de mensageria.

Em seguida, temos o dado disponível para ser preparado para a etapa de análise. Nesta etapa de curadoria/preparação do dado, o objetivo principal é receber os dados do componente de Coleta, categorizar os dados de acordo com os atributos definidos como qualitativos e quantitativos, de modo que a saída esperada seja os dados ordenados com atributos qualitativos seguidos de atributos quantitativos. Para isso, é definido a interface de entrada e de saída da camada de preparação do dado. A Figura 12 apresenta o diagrama de sequência para o fluxo do componente de preparação,

Figura 12: Diagrama de Sequência do componente de preparação.

Fonte: o Autor

Com os dados recebidos, a camada de preparação realiza a checagem e validação dos metadados, e caso a validação esteja coerente, há a separação entre os dados qualitativos e quantitativos. Uma vez categorizado os dados, é removido os dados que não se enquadram como quantitativos ou qualitativos, bem como também são descartados os dados processados como "nulo"e que não possuem um valor padrão atribuido. Quanto aos componentes funcionais, nesta camada propomos a sugestão de frameworks como Apache Spark e Apache Flink para o processamento de dados, além de Apache Kafka para trafego de mensagem e Apache Atlas como sistema de glossário de dados.

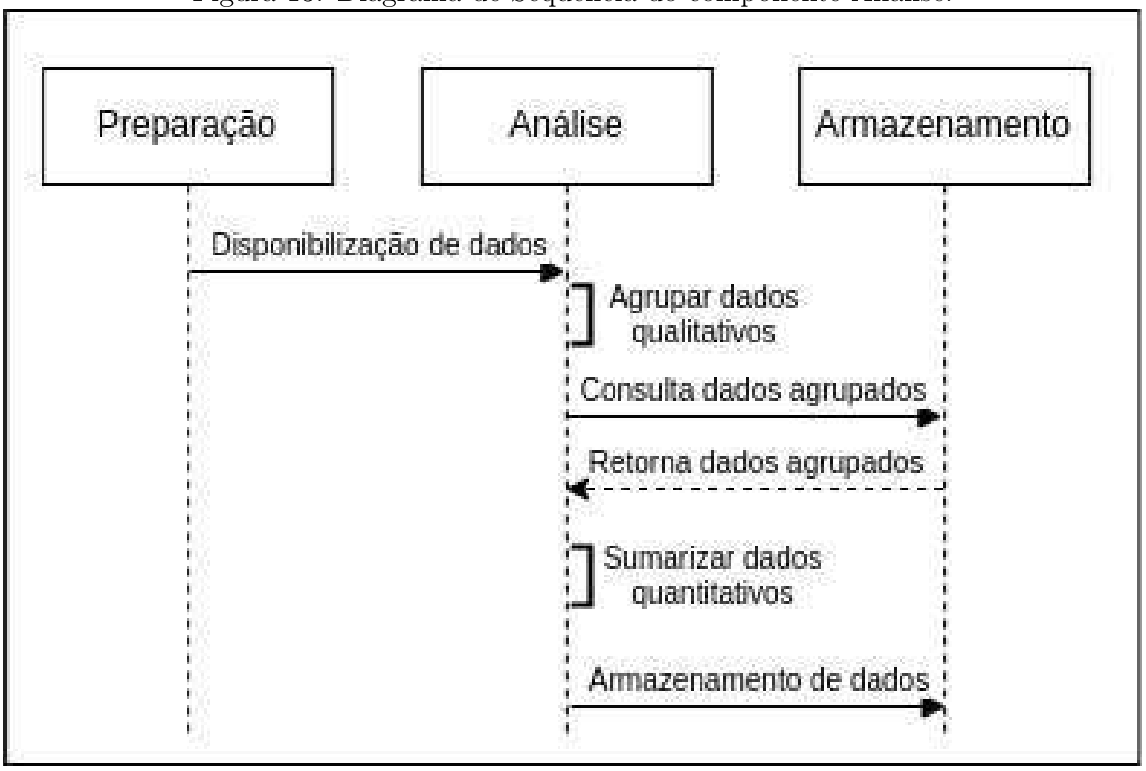
Sendo assim, os dados são submetidos através de sistema de mensageria para o componente Análise. Desta forma, a interface de saída do componentente de preparação representa os dados pré-processados para a próxima fase, e deve ser implementado conforme a interface a seguir.

```
Output: {
    metadata: {
        key : string,
        timestamp: timestamp
    },
    fields: {
        quantity: [string],
        quality: [string]
    },
    data: {
        [fieldName: string]: string|double|boolean
    }
}
```

Com os dados pré processados, são encaminhamos para a camada de Análise. Esta camada tem como principal função realizar o processamento dos dados para que se extraia os indicadores de negócio necessarios, de modo que a saída esperada seja os atributos definidos como qualitativos seguidos das métricas em relação aos atributos definidos como quantitativos. Este componente auxilia na contemplação dos casos de uso relacionados a armazenamento (UC-04) e métricas do sistema (UC-05), pois é nesta camada que o processamento para acompanhamento de métricas baseadas em dados quantitativos e qualitativos acontece. Diferente dos componentes anteriores, neste caso iremos armazenar as informações consolidadas em uma base de dados. A Figura 13 apresenta o diagrama

de sequência para o fluxo do componente de Snálise.

Figura 13: Diagrama de Sequência do componente Análise.



Fonte: o Autor

Uma vez recebido os dados pre-processados pela camada de Preparação, estes dados deverão ser agrupados pela lista de atributos inclusos na propriedade quality da interface de entrada. Com os dados agrupados, o componente valida se há registro para esta lista de agrupamentos na base de dados e recupera as consolidações Em caso de não existir registro para a lista de agrupamentos na base, o processo de Armazenamento de dados realiza a criação da tabela, definindo todos os campos qualitativos como chaves primária e as métricas dos campos quantitativos como 0 (zero). Em seguida, para cada atributo incluso na lista de propriedade quantity da interface de entrada, calculamos as seguintes métricas: contador/quantidade, somatório, valor máximo, valor mínimo, valor médio, desvio padrão médio e moda. Na sequência, unificamos as métricas calculadas com as consolidações armazenadas na base de dados, e desta forma, as novas sumarizações são inseridas na base de dados.

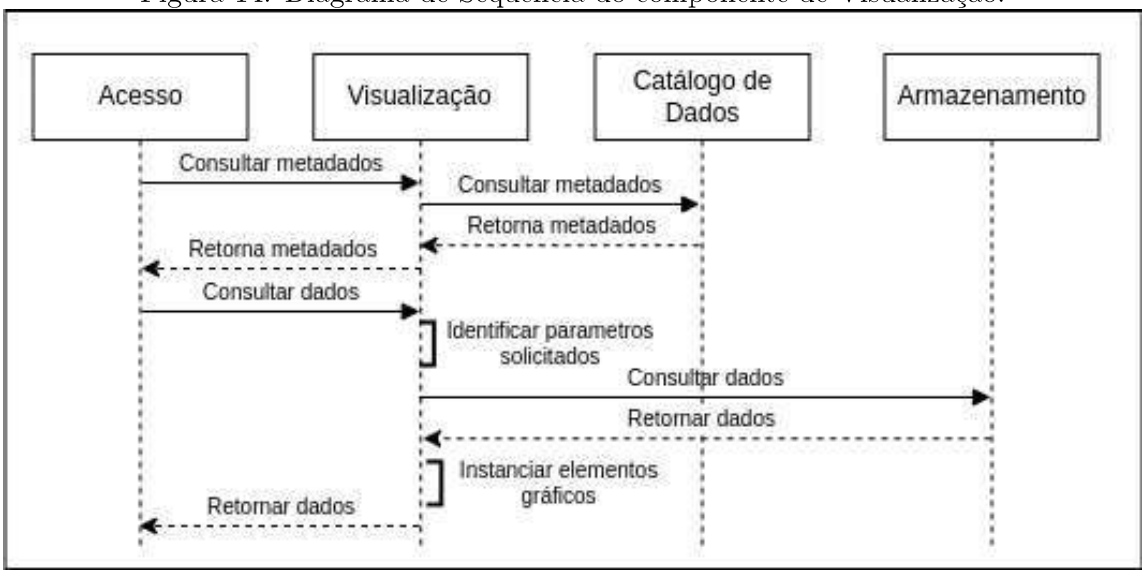
Em relação aos componentes funcionais deste processamento, a transferência de dados da etapa de Preparação para a Análise será através do Apache Kafka. O agrupamento e processamento da camada Análise pode ser executado utilizando Apache Spark ou Apache Flink. Para o Armazenamento, como peça do Big Data Framework Provider, será utilizado

o Apache Hadoop (HDFS), e a instância do banco de dados se dará através do Apache Kudu. A interface de entrada da camada Análise é a definida pela interface de saída do componente de Preparação. Quanto a interface de para o banco de dados, deverá ser implementado seguindo a seguinte definição.

```
Output: {
    metadata: {
        key : string,
        timestamp: timestamp
    },
    data: {
        [qualityField: string]: string|double|integer|boolean,
        [quantityField: string]: double|integer
    }
}
```

Neste ponto, já temos os dados processados e armazenados em nossa base de dados, portanto, seguiremos com a camada de Visualização. O objetivo deste componente é ser capaz apresentar as informações processadas para o componente de Acesso, e desta forma, o componente atende aos casos de uso UC-05. A Figura 14 apresenta o diagrama de sequencia para o fluxo do componente de visualização.

Figura 14: Diagrama de Sequencia do componente de Visualização.



Fonte: o Autor

Com os dados armazenados, o componente de visualização pode obter estes dados

para apresentá-los a camada de acesso. Neste caso, a camada de acesso solicita para a camada de visualização os metadados a serem consultados (como banco de dados e tabela), e a camada de visualização retorna a lista de atributos qualitativos e quantitativos disponíveis para visualização. Desta forma, o componente de acesso repassa ao componente de visualização quais dados devem ser apresentados, e este por sua vez identifica a solicitação, realiza a consulta na base de dados e instancia os elementos gráficos a serem apresentados. Por fim, o componente de visualização retorna ao componente de acesso os dados solicitados.

Para os componentes funcionais desta camada, utilizaremos o protocolo HTTP na requisição de dados e metadados, sendo assim, será desenvolvido uma API instanciada a partir da biblioteca Express do NodeJS. Para a comunicação com a camada Armazenamento, será utilizado drivers de conexão ODBC. Para a visualização do elementos gráficos instanciados, utilizaremos o Apache Superset. A interface de comunicação entre os componentes Acesso e Visualização são divididas entre a interface a respeito dos metadados e a interface a respeito dos dados. Para os metadados, a camada de acesso solicita informações a respeito da base e da tabela que desejam acessar, e a camada de visualização por sua vez retorna a lista de atributos quantitativos e qualitativos da tabela, conforme definição de interface a seguir

```
Input metadados: {
    metadata: {
        database : string,
        table: string
    }
}
Output metadados: {
    metadata: {
        key: string
    },
    data: {
        [qualityField: string]: string|double|integer|boolean,
        [quantityField: string]: double|integer
    }
}
```

Em seguida, a camada de acesso solicita quais atributos devem ser apresentados, com

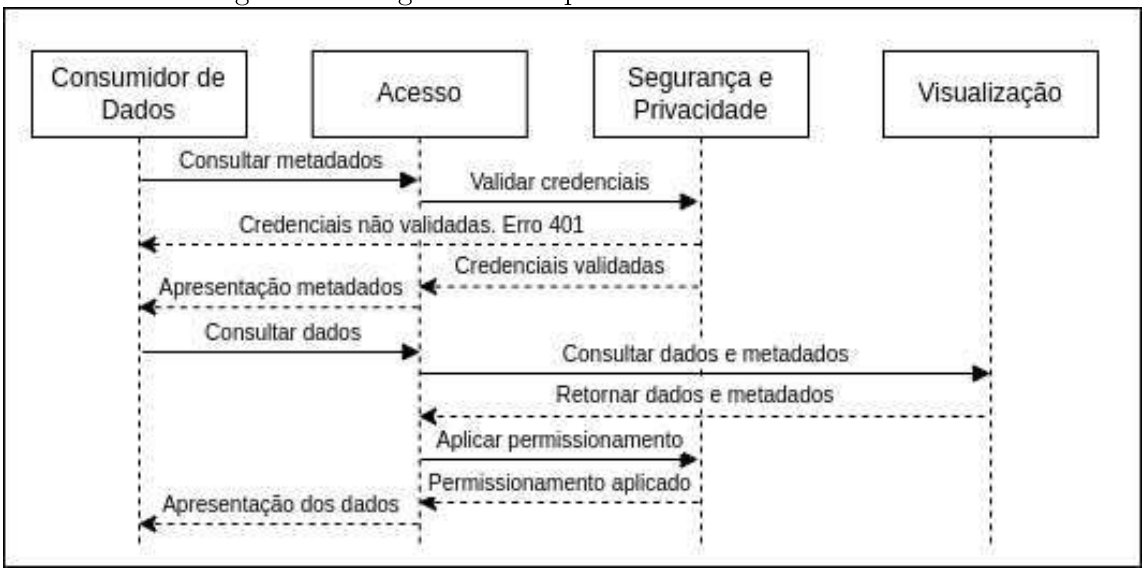
base nos metadados repassados. A camada de visualização então interpreta os metadados, identifica os atributos qualitativos e quantitativos e recupera da base de dados os indicadores solicitados. Com os dados carregados, as informações são instanciadas em elementos gráficos, de modo que cada elemento esteja vinculada a uma consolidação calculada pelo sistema. Finalmente, as representações gráficas são encaminhadas a camada de acesso. A interface definida para a comunicação entre a camada de acesso e visualização para recuperação dos dados está descrita a seguir.

```
Input dados: {
    metadata: {
        key: string
    },
    data: {
        [qualityField: string]: string|double|integer|boolean,
        [quantityField: string]: double|integer
    }
}
```

Por fim, definimos o componente de acesso aos dados. Este componente tem como principal objetivo prover os resultados para o Consumidor de Dados de acordo com as regras de acesso de cada usuário, em conformidade com o caso de uso UC-07. Para isso, esta camada se integra com os compoenntes de segurança e privacidade do ambiente de big data para identificar usuários e permissionamentos, além da integração diretamente com a camada de Visualização para exibição dos dados. A Figura 15 apresenta o diagrama de sequencia atribuido ao componente de Acesso.

Inicialmente o Consumidor de Dados realiza a consulta dps metadados estão disponíveis para serem apresentados. Essa solicitação deve conter as informações de credenciamento do Consumidor de Dados para então ser repassada à camada de Segurança e Privacidade a fim de validar que o acesso é legitimo. Em caso de eventuais problemas de credenciamento, o Consumidor de Dados fica impossibilitado de finalizar a jornada com sucesso. Dado as credenciais validadas, o Consumidor de Dados recebe as informações dos dados disponíveis para consulta, e realiza a solicitação para a camada de acesso, que então recupera as informações da camada de Visualização, e antes de apresentar o resultado ao Consumidor de Dados, aplica as regras de permissionamento e anonimização necessários conforme o perfil de usuário credenciado. Por fim, o componente de acesso apresenta as informações ao Consumidor de Dados.

Figura 15: Diagrama de Sequencia da camada de acesso.



Fonte: o Autor

Quanto aos componentes funcionais desta camada, será instanciado o protocolo HTTP para a aplicação que corresponde ao Data Access, que desta forma, realizará as integrações utilizando API Rest implementado a partir da biblioteca Express da tecnologia NodeJS. A camada de Segurança e Privacidade dizem respeito ao ambiente de Big Data, e estão instanciadas a partir do componente Apache Ranger.

Para a comunicação do Consumidor de Dados com a camada de Acesso, no que diz respeito a consulta de metadados, o componente de acesso recebe em cada requisição a informação do Autorização, que será validada pelos componentes de Segurança e Privacidade. Neste caso, a camada de Acesso apresentará a lista de tabelas e atributos disponíveis em cada tabela, conforme apresentado abaixo.

```
Input metadados: {
    Header.Authorization: string
}

Output metadados: [{
    key: string,
    databaseName: string,
    tableName: string
    qualityField: [string],
    quantityField: [string]
}]
```

O Consumidor de Dados então seleciona qual banco de dados, tabela e atributos quer recuperar e envia ao Acesso. O Acesso então processa a informação e apresenta a informação ao Consumidor de Dados. Para esta integração, o output definido é a visualização de fato, enquanto que o input do componente de acesso está definido a seguir.

```
Input dados: {
    Header.Authorization: string
    metadataKey: string
    qualityField: [string],
    quantityField: [string]
}
```

Desta forma, finalizamos a instância de todos os elementos necessários para a elaboração e criação da arquitetura deste projeto.

3.3.6.6 6º Passo: Rascunhar visões e registrar decisões de design

Uma vez que finalizamos a instância de todos os elementos, bem como as definições de cada processo, temos a elaboração e a criação da arquitetura do projeto definida. Desta maneira, entendemos que o artefato resultante deste processo é a arquitetura de referência instanciada e aplicada ao fluxo funcional atual. Desta maneira, a Figura 16 apresenta a representação da arquitetura com os elementos instanciados, aplicada ao contexto inicial proposto.

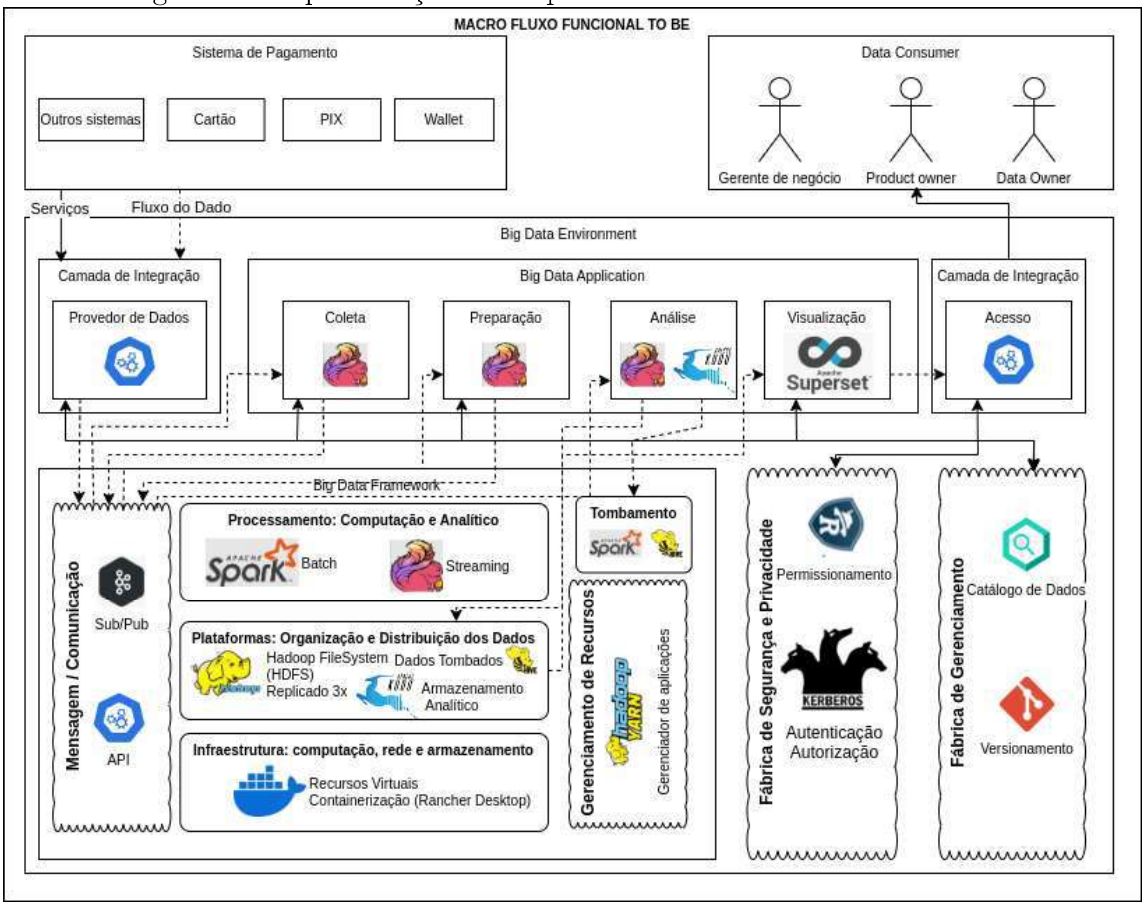
Podemos perceber que, do ponto de vista organizacional, a proposta possibilita a democratização e o acesso às informações, uma vez que os relatórios criados por uma área podem ser compartilhados com outros potenciais stakeholders, contribuindo para uma análise de dados colaborativa.

3.3.6.7 7º Passo: Análise do design atual, review do objetivo da iteração e da proposta de design

Nossa arquitetura, nesta etapa, contempla o objetivo da nossa iteração, cuja definição foi a criação de uma solução arquitetural para o sistema, dedicando os esforços na contemplação dos drivers de arquitetura definidos. Desta maneira, entende-se que a iteração atingiu o seu objetivo principal.

Quanto ao design da arquitetura, ou seja, o trabalho desta monografia, o artefato desenvolvido contempla os objetivos definidos no começo deste projeto. Desta maneira,

Figura 16: Representação da Arquitetura com elementos instanciados



Fonte: o Autor

temos como artefato e produto deste trabalho a definição dos drivers de arquitetura contemplados através do método Attribute Driven Design 3.0, bem como as definições de integração dos elementos que representam o NIST, instanciados com o objetivo de satisfazer os drivers arquiteturais estabelecidos. Por fim, o sistema possibilita a definição das métricas a serem apresentadas com base em métricas pré definidas sistemicamente.

3.4 Resultados

Como artefato resultante da aplicação do método Attribute Driven Design 3.0, obtemos então a representação da arquitetura do software, além de entender os componentes, suas funcionalidades e como se integram. Portanto, para analisarmos os resultados, avaliaremos se o design da arquitetura do software projetado atende aos requisitos não funcionais estabelecidos através dos drivers de arquitetura. A Tabela 10 abaixo sumariza os atributos de qualidade definidos durante a aplicação do método e justifica se o design

da arquitetura do software proposto satisfaz esses atributos.

Tabela 10: Atributos de qualidade e critério de aceite

Atributo de Qualidade	**Caso de Uso**	**Justificativa**
QA-1: Performance	UC-01, 05	A arquitetura é construída utilizando sistemas de Publish/Subscribe em seu componente de comunicação, garantindo tráfego de eventos em tempos inferiores a 5 segundo.
QA-2: Performance	UC-01, 06	O atributo de qualidade não foi contemplado nesta iteração, portanto, deve ser trabalhado em sprints futuras.
QA-3: Escalabilidade	UC-03	A arquitetura contempla um componente para transferir dados do armazenamento analítico para o armazenamento tombado.
QA-4: Escalabilidade	UC-03	O sistema está projetado para injetar o timestamp nos eventos processados, e este dado dado pode ser utilizado como dado qualitativo para extração de indicadores.
QA-5: Extensibilidade	UC-01	A proposta contempla um componente específico para ser responsável exclusivamente pela comunicação com os sistemas aptos a integração
QA-6: Performance	UC-01, 03, 06	A arquitetura contempla peças para armazenamento de dados com desenvolvimento focado em análises rápidas para dados rápidos
QA-7: Extensibilidade	UC-01	A arquitetura proposta prevê um serviço de catálogo de dados que oferece as informações a respeito dos metadados cadastrados no sistema
QA-8: Extensibilidade	UC-01, 02, 06	Parcialmente atendido pois a camada de integração realiza operações de Create e Read no catalogo de dados, enquanto outros componentes realizam somente leitura. Nenhum componente atualiza ou remove metadados.
QA-9: Segurança	UC-09	O design de arquitetura de software contempla serviços a serem integrados com a responsabilidade de prover sistema de autenticação e autorização dos dados.

Fonte: o Autor

4 CONSIDERAÇÕES FINAIS

Neste trabalho projetamos uma arquitetura de software de big data para resolver o problema de acompanhamento de dados de maneira rápida e dinâmica. Para a concepção e elaboração desta arquitetura, foi necessário aplicar métodos que nos ensinam a produzir um design de arquitetura, e nos baseamos no método ADD-S. Além disso, foi necessário estudar arquiteturas de referências distintas e e identificar uma referência que contemple os cenários de atributos de qualidade do sistema.

A aplicação do método ADD-S mostrou que há familiaridade entre os componentes do método ADD e a metodologia ágil Scrum, de maneira que conseguimos instanciar um como o outro. O ADD contribuiu para a definição dos drivers arquiteturais, como os cenários de atributo de qualidade. O NIST como arquitetura de referência instanciada contribuiu na elaboração da camada da aplicação de big data, com as definições das responsabilidades, sugestões de frameworks e design patterns e interfaces de comunicação, além de contemplar os componentes e integrações necessárias com o ecossistema de big data para atingir os requisitos não funcionais especificados através dos drivers.

A vantagem deste método é que foi possível evidenciar um processo iterativo para tratar tanto requisitos funcionais quanto requisitos não funcionais. Como benefícios esperados, a arquitetura proposta auxilia no acompanhamento de métricas baseadas em dados quantitativos e qualitativos, e busca melhorar a compreensão a respeito dos clientes da credenciadora, contribuindo para: a) gestores de negócio e product owners, que terão a possibilidade de acompanhar métricas sobre todos os produtos que possuem responsabilidade e; b) data owners: que terão as ferramentas necessárias para atingir conformidade em relacão a autenticação e autorização dos acessos aos dados.

Nas limitações deste trabalho destacamos a manutenção e evolução do sistema devido ao rápido avanço das tecnologias de big data, pois apesar da modalurização contribuir na troca de tecnologia, pode exigir atualizações que podem ser complexas. Identificamos as oportunidades de aplicação do método em outros casos de uso, como empresa no setor de Marketing para acompanhar o resultado das suas campanhas a medida que as ações de marketing se concretizem, como uma propaganda ou uma exposição. Para trabalhos futuros, pode-se atuar na etapa de Desenvolvimento do Ciclo de Vida do Desenvolvimento de Software

REFERÊNCIAS BIBLIOGRÁFICAS

Bass, Len, Paul Clements e Rick Kazman (2021). *Software Architecture in Practice (SEI Series in Software Engineering) 4th Edition*. Addison-Wesley Professional.

Brasil, Banco Central do (s.d.). *Concentração nos mercados de credenciamento e de emissão de cartões de pagamento - Relatório de Economia Bancária (REB/2023)*. URL: ⟨https://www.bcb.gov.br/content/publicacoes/boxe_relatorio_de_economia_bancaria/reb2023b6p.pdf⟩. (acessado: 18.10.2024).

Cervantes, Humberto e Rick Kazman (2016). *Designing software architectures: a practical approach*. Addison-Wesley Professional.

Chang, Wo e Gregor von Laszewski (2019). *NIST Big Data Interoperability Framework: Volume 8, Reference Architecture Interfaces*. en. DOI: ⟨https://doi.org/10.6028/NIST.SP.1500-9r1⟩.

Chang, Wo L, David Boyd e Orit Levin (2019). *NIST big data interoperability framework: volume 6, reference architecture*. Wo L. Chang, David Boyd, Orit Levin.

Diván, Mario José (2017). "Data-driven decision making". Em: *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, pp. 50–56. DOI: ⟨10.1109/ICTUS.2017.8285973⟩.

Febraban (s.d.). *Transações feitas com Pix crescem 61% no primeiro semestre do ano*. URL: ⟨https://portal.febraban.org.br/noticia/4184/pt-br/⟩. (acessado: 18.10.2024).

Furini, Isabele Chaiben (2020). "Mercado de meios de pagamento no Brasil: visão histórica e tendências globais". Monografia de Graduação. Universidade Federal do Rio Grande do Sul. URL: ⟨https://lume.ufrgs.br/handle/10183/216349⟩.

Lorey, Vilma Ataíde (2008). "Aquisições estratégicas: um estudo sobre o mercado de cartões de crédito". Dissertação de Pós Graduação. Pontifícia Universidade Católica de São Paulo. URL: ⟨https://bdtd.ibict.br/vufind/Record/PUC_SP-1_c29b008a488f67b01f56e2da45f

Mcheick, Hamid, Yan Qi e Hani Charara (2011). "Distributed Software Integration Model Bases on Attribute-Driven Design ADD Method". Em: *International Conference on Software and Data Technologies*. Vol. 2. SCITEPRESS, pp. 53–58.

Rahman, Md Saifur e Hassan Reza (2020). "Systematic mapping study of non-functional requirements in big data system". Em: *2020 IEEE International Conference on Electro Information Technology (EIT)*. IEEE, pp. 025–031.

Rao, T Ramalingeswara et al. (2019). “The big data system, components, tools, and technologies: a survey”. Em: *Knowledge and Information Systems* 60, pp. 1165–1245.

Reis, Marco Antônio de Sousa (2019). “Uma arquitetura de Big Data as a service baseada no modelo de nuvem privada”. Monografia de Mestrado. Universidade de Brasília. URL: ⟨http://www.realp.unb.br/jspui/handle/10482/33759⟩.

Rozony, Farhana Zaman et al. (2024). “A Systematic Review Of Big Data Integration Challenges And Solutions For Heterogeneous Data Sources”. Em: *Academic Journal on Business Administration, Innovation & Sustainability* 4.04, pp. 1–18.

Sachdeva, Vaibhav e Lawrence Chung (2017). “Handling non-functional requirements for big data and IOT projects in Scrum”. Em: *2017 7th International Conference on Cloud Computing, Data Science Engineering - Confluence*, pp. 216–221. DOI: ⟨10.1109/CONFLUENCE.2017.7943152⟩.

Serviços, Associação Brasileira das Empresas de Cartões de Crédito e (s.d.). *Balanço do setor de meios eletronicos de pagamentos - Resultados 2T2024*. URL: ⟨https://api.abecs.org.br/wp-content/uploads/2024/08/Abecs-Apresentacao-2T2024.pdf⟩. (acessado: 04.10.2024).

Soumaya, Ounacer et al. (2017). “Real-time data stream processing challenges and perspectives”. Em: *International Journal of Computer Science Issues (IJCSI)* 14.5, pp. 6–12.

Wieringa, Roel J (2014). “A Road Map of Research Methods”. Em: *Design science methodology for information systems and software engineering*, pp. 215–223.

Younas, Muhammad (2019). “Research challenges of big data”. Em: *Service Oriented Computing and Applications* 13, pp. 105–107.