INSTITUTO FEDERAL DE SANTA CATARINA

JENEFFER FARIAS BORA RIBEIRO

Especificação de Sistema Modular para Redução da Evasão e Retenção Acadêmica

São José - SC

março/2025

ESPECIFICAÇÃO DE SISTEMA MODULAR PARA REDUÇÃO DA EVASÃO E RETENÇÃO ACADÊMICA

Monografia apresentada ao Curso de Engenharia de Telecomunicações do campus São José do Instituto Federal de Santa Catarina para a obtenção do diploma de Engenheiro de Telecomunicações.

Orientador: Professor Ederson Torresini, Me

São José - SC

março/2025

JENEFFER FARIAS BORA RIBEIRO

Especificação de Sistema Modular para Redução da Evasão e Retenção Acadêmica

Este trabalho foi julgado adequado para obtenção do título de Engenheiro de Telecomunicações, pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

São José - SC, 21 de março de 2025:

gov.br Documento assinado digitalmente
EDERSON TORRESINI
Data: 01/04/2025 23:10:25-0300
Verifique em https://validar.iti.gov.br

Professor Ederson Torresini, Me
Orientador
Instituto Federal de Santa Catarina

gov.br Documento assinado digitalmente
CLEBER JORGE AMARAL
Data: 02/04/2025 08:02:47-0300
Verifique em https://validar.iti.gov.br

Professor Cleber Jorge Amaral, Dr.
Instituto Federal de Santa Catarina

Fernanda Carolina Dias, Pedagoga
Instituto Federal de Santa Catarina

Documento assinado digitalmente

gov.br

FERNANDA CAROLINA DIAS
Data: 03/04/2025 12:42:03-0300
Verifique em https://validar.iti.gov.br

Dedico este trabalho a todas as mulheres — as que vieram antes de mim, as que caminham ao meu lado e as que ainda virão. Que possamos sempre ocupar todos os espaços que desejarmos, pois o lugar de uma mulher é onde ela quiser, inclusive na engenharia.

AGRADECIMENTOS

Em primeiro lugar, dedico este trabalho aos meus primeiros e maiores professores: meus pais, Alcione e João Décio. Vocês são, e sempre serão, meu maior exemplo de amor, dedicação e determinação. Se cheguei até aqui, foi porque tive o apoio incondicional e o amor de vocês em cada passo do caminho.

À minha irmã, Jayne, por sua força e coragem inspiradoras. Sua determinação incansável sempre foi um exemplo para mim, mostrando que nenhum desafio é grande demais quando se tem garra e perseverança.

Ao meu esposo, Arthur Machado, meu grande incentivador, que esteve sempre ao meu lado, me motivando e apoiando em cada decisão, celebrando comigo as conquistas e me fortalecendo nos desafios.

Faltam palavras para expressar a imensa gratidão que sinto por cada um de vocês quatro. Este breve registro é apenas uma pequena homenagem de amor, por todo apoio que vocês sempre me proporcionaram.

Agradeço ao meu orientador, Éderson, não apenas por aceitar estar ao meu lado nesta última etapa do curso, mas por ter sido um dos grandes mentores com quem tive o privilégio de compartilhar boa parte da minha jornada no IFSC. “Se eu vi mais longe, foi por estar sobre ombros de gigantes.”

Expresso também minha gratidão a todo o corpo docente e aos técnicos administrativos do IFSC, especialmente aos do campus São José, pelo apoio e dedicação ao longo dessa caminhada.

Por fim, mas não menos importante, agradeço de coração a todos os colegas e amigos que fiz durante meus anos de estudos no IFSC. Não citarei nomes para evitar cometer a injustiça de esquecer alguém, mas meu reconhecimento sincero vai a cada um de vocês que estiveram ao meu lado. Ninguém se forma sozinho, e se cheguei até aqui, foi porque pude contar com pessoas incríveis que tornaram essa jornada inesquecível.

É preciso ter esperança, mas ter esperança do verbo esperançar;
porque tem gente que tem esperança do verbo esperar.
E esperança do verbo esperar não é esperança, é espera.
Esperançar é se levantar, esperançar é ir atrás,
esperançar é construir, esperançar é não desistir!
Esperançar é levar adiante, esperançar é juntar-se com outros para fazer de outro modo.
Paulo Freire

RESUMO

A missão do Instituto Federal de Santa Catarina (IFSC) é promover a inclusão e formar cidadãos por meio da educação profissional, científica e tecnológica, gerando, difundindo e aplicando conhecimento e inovação para contribuir com o desenvolvimento socioeconômico e cultural. No entanto, a evasão e a retenção de alunos comprometem essa missão, dificultando a formação de profissionais. Para apoiar o principal objetivo da instituição, é essencial adotar métodos que incentivem a conclusão da formação dos estudantes. Este trabalho tem como objetivo especificar a implementação de um sistema composto por serviços modulares, projetados para coletar, processar e disponibilizar dados acadêmicos de forma estruturada. Esses serviços viabilizam a extração de informações sobre os alunos, permitindo seu tratamento e distribuição para interfaces que possam alertar os responsáveis pelo processo educacional sobre possíveis casos de evasão ou retenção. A especificação abrange a definição de *pipelines* de dados, algoritmos de classificação e serviços de troca de mensagens, proporcionando uma abordagem proativa para o acompanhamento dos estudantes e a tomada de decisões estratégicas ao longo de sua trajetória acadêmica.

Palavras-chave: Evasão. Retenção. Monitoramento educacional. Serviços modulares.

ABSTRACT

The mission of IFSC is to promote inclusion and educate citizens through professional, scientific, and technological education, generating, disseminating, and applying knowledge and innovation to contribute to socioeconomic and cultural development. However, student dropout and retention hinder this mission, making it difficult to train professionals. To support the institution's primary goal, it is essential to adopt methods that encourage students to complete their education. This work aims to specify the implementation of a system composed of modular services designed to collect, process, and provide academic data in a structured manner. These services enable the extraction of student-related information, allowing its processing and distribution to interfaces that can alert those responsible for the educational process about potential cases of dropout or retention. The specification includes defining data *pipelines*, classification algorithms, and messaging services, providing a proactive approach to student monitoring and strategic decision-making throughout their academic journey.

Keywords: Dropout. Retention. Educational monitoring. Modular services.

LISTA DE ABREVIATURAS E SIGLAS

ACID Atomicidade, Consistência, Isolamento e Durabilidade.

ACL *Access Control List*.

API *Application Programming Interface*.

API Gateway *Application Programming Interface Gateway*.

API RESTful Application Programming Interface Representational State Transfer.

CAA Coeficiente de Aproveitamento Acadêmico.

CF Constituição Federal.

CRM *Customer Relationship Management*.

EC2 *Elastic Compute Cloud*.

ELT *Extract, Load, Transform*l.

ERP *Enterprise Resource Planning*.

ETL *Extract, Transform, Load*l.

IA Inteligência Artifical.

IaaS *Infrastructure as a Service*.

IdP *Identity Provider*.

IEA Índice de Eficiência Acadêmica.

IECH Índice de Eficiência em Carga Horária.

IEPL Índice de Eficiência em Períodos Letivos.

IFSC Instituto Federal de Santa Catarina.

IoT *Internet of Things*.

IVS Índice de Vulnerabilidade Social.

JSON *JavaScript Object Notation*.

JWT JSON Web Tokens.

KNN *K-Nearest Neighbors*.

LDB Lei de Diretrizes e Bases.

LGPD Lei Geral de Proteção de Dados.

MC Média de Conclusão.

MCN Média de Conclusão Normalizada.

MQTT Message Queuing Telemetry Transport.

NoSQL *Not Only SQL*.

OAuth *Open Authorization*.

OIDC *OpenID Connect*.

PaaS *Platform as a Service*.

PROEN Pró-Reitoria de Ensino.

Pub/Sub Publicação e Assinatura.

QoS Quality of Service.

SaaS *Software as a Service*.

SGBD Sistema de Gerenciamento de Banco de Dados.

SIGAA Sistema Integrado de Gestão de Atividades Acadêmicas.

SISSA Sistema Integrado de Suporte ao Sucesso Acadêmico.

SQL *Structured Query Language*.

SVM *Support Vector Machines*.

TI Tecnologia da Informação.

SUMÁRIO

1 INTRODUÇÃO

A Constituição Federal (CF) de 1988, em seu artigo 6º, estabelece que a educação é um direito social. Como responsabilidade do Estado e da família, o direito à educação deve se materializar promovendo o desenvolvimento integral do indivíduo, preparando-o para a cidadania ativa e qualificando-o para o mercado de trabalho (BRASIL, 1988). O direito à educação é fundamental para o respeito à dignidade humana. Este princípio é reconhecido em diversos documentos, tratados e acordos, tanto nacionais quanto internacionais, dos quais o Brasil é signatário. Além disso, é reforçado pela legislação brasileira que trata da educação, especialmente na Lei de Diretrizes e Bases (LDB).

Entretanto, reconhecer a educação como um direito fundamental não é suficiente. É essencial implementar e fornecer ações que garantam esse direito de forma efetiva. Situações em que um aluno está em potencial risco de evasão ou retenção acadêmica comprometem diretamente esse direito, exigindo intervenções proativas para assegurar a continuidade e o sucesso do percurso educacional do aluno (MEC, 2014). Neste contexto, a identificação precoce de alunos em risco e a implementação de medidas preventivas são fundamentais para garantir o sucesso acadêmico e a conclusão dos cursos (UFES, 2018).

Contudo, é necessário reconhecer que podem haver dificuldades, por parte da instituição e dos interessados no processo pedagógico, para identificar e acompanhar alunos em risco de evasão ou retenção escolar. Muitas vezes, as informações relevantes sobre o desempenho e comportamento acadêmico dos alunos estão dispersas em diferentes sistemas e não são facilmente acessíveis.

No âmbito do IFSC, por exemplo, é utilizado o SIGAA como sistema de gestão acadêmica. Ele concentra dados relevantes como histórico escolar, frequência, notas e índices de desempenho. No entanto, apesar de reunir essas informações, o sistema não possui mecanismos automatizados de sinalização ou alerta que permitam a identificação proativa de alunos em situação de risco de evasão ou retenção. Dessa forma, o acompanhamento ainda depende da iniciativa individual dos profissionais da educação e da consulta manual aos dados disponíveis.

Adicionalmente, a comunicação entre os envolvidos no processo educacional pode ser fragmentada, dificultando a implementação de intervenções adequadas e oportunas.

É necessário, portanto, um sistema que integre essas informações e facilite a comunicação entre os professores, coordenadores de curso e equipe pedagógica. Neste sentido, o uso da tecnologia pode ser uma aliada na educação, oferecendo recursos acessíveis e disponíveis por meio de navegadores e diversos aplicativos. O uso de aplicativos *web*, por exemplo, é um padrão comum e amplamente acessível para facilitar a comunicação e a

gestão educacional.

Com o uso de sistemas de notificação, como *chatbots*, os interessados no processo educacional podem ser informados sobre a situação de possível evasão ou retenção do aluno. Os *chatbots* são amplamente utilizados em páginas *web* e ferramentas de *chat*, como *WhatsApp*, *Telegram* e *Google Chat*, para facilitar a interação com os usuários e fornecer informações de maneira rápida e eficiente. Quando combinados com tecnologias de aprendizado de máquina e algoritmos de detecção de padrões, esses serviços podem se tornar uma ferramenta útil para identificar alunos em potencial situação de evasão ou retenção acadêmica.

Diante do exposto, este trabalho propõe a especificação de um sistema composto por serviços modulares, projetados para coletar, processar e disponibilizar dados acadêmicos. A solução baseia-se na coleta, processamento e disponibilização desses dados para sistemas de Inteligência Artifical (IA), a fim de identificar padrões de risco, além de oferecer interfaces que permitam a integração com serviços de notificação, facilitando a comunicação direta com os envolvidos. Com isso, busca-se proporcionar uma abordagem proativa no enfrentamento desses desafios.

Dessa forma, esses profissionais podem tomar medidas preventivas e necessárias para evitar a evasão ou retenção dos alunos, garantindo uma intervenção adequada e o acompanhamento necessário para o sucesso acadêmico dos estudantes.

1.1 Objetivo geral

Especificar um sistema modular para monitoramento e notificação de alunos em potencial risco de evasão ou retenção acadêmica, permitindo que informações relevantes sejam organizadas e disponibilizadas para os responsáveis pedagógicos e acadêmicos, de modo a apoiar ações preventivas e tomadas de decisão.

1.1.1 Objetivos específicos

- Definir os requisitos necessários para o funcionamento do sistema, considerando as necessidades institucionais e fluxos de informação acadêmica.
- Estruturar a arquitetura modular do sistema, estabelecendo os principais blocos funcionais e sua integração para garantir um fluxo de dados.

1.2 Metodologia

Este trabalho foi desenvolvido em três etapas principais. Na primeira etapa, foi realizada uma revisão bibliográfica para embasar a especificação do sistema, consolidando

conceitos fundamentais sobre armazenamento, processamento e disponibilização de dados.

Na segunda etapa, foi realizada a especificação do sistema, abrangendo a definição dos componentes funcionais, a arquitetura modular, o fluxo dos dados e os mecanismos de integração entre os serviços. Essa etapa detalha como as informações são extraídas, processadas e disponibilizadas para os interessados.

Por fim, na terceira etapa, foram discutidos os desafios da proposta e sugestões para trabalhos futuros.

1.3 Organização do Texto

O texto é organizado da seguinte forma: no Capítulo 1, são apresentados os motivadores e objetivos deste trabalho. O Capítulo 2 contém a revisão bibliográfica, onde são descritos conceitos importantes para a compreensão e especificação do sistema proposto.

No Capítulo 3, é realizada a especificação do sistema, detalhando sua arquitetura modular, os componentes funcionais e os fluxos de processamento e disponibilização dos dados acadêmicos. Por fim, o Capítulo 4 apresenta a síntese dos resultados obtidos, discutindo os desafios enfrentados na especificação e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo fornecer a base técnica e teórica necessária para atingir os objetivos gerais e específicos deste trabalho. O propósito é embasar o desenvolvimento da solução para a problemática identificada e discutida no Capítulo 1, que é a proposta de uma solução para a identificação de alunos em potencial risco de evasão e retenção.

2.1 Computação em Nuvem

A computação em nuvem pode ser definida como um conjunto de serviços e/ou recursos computacionais são fornecidos através da *internet* (nuvem). O termo "nuvem" em computação surgiu devido à sua representação em diagramas de rede. No início dos anos 1990, cientistas da computação usavam o símbolo de uma nuvem para representar a infraestrutura de servidores e armazenamento que compunha "a rede" (WORLD ECONOMIC FORUM, 2015). O termo ganhou ampla aceitação com a introdução de serviços como o *Elastic Compute Cloud* (EC2) da *Amazon Web Services* em 2006, que permitia às empresas alugar servidores virtuais. A A ideia era de que esses recursos estavam disponíveis sob demanda pela internet, semelhante a utilidades como eletricidade, enfatizando a abstração da infraestrutura que os suporta (ISLAM et al., 2023). Essa abstração e facilidade de acesso impulsionaram a adoção da computação em nuvem, oferecendo benefícios como escalabilidade, eficiência de custos e inovação. Ao utilizar serviços de nuvem, as organizações podem focar em suas atividades principais sem se preocupar com a manutenção de *hardware* físico e podem rapidamente se adaptar às demandas em mudança, escalando recursos conforme necessário.

2.1.1 Modelos de Implantação

A classificação da nuvem pode ser dividida em quatro tipos: nuvem pública, nuvem privada, nuvem híbrida e nuvem comunitária. A escolha da forma de implementação da nuvem depende das necessidades específicas da empresa ou instituição que utilizará os serviços. Para determinar o tipo de nuvem a ser utilizado, são considerados fatores como custo, segurança e elasticidade, sendo esta a capacidade da infraestrutura de ajustar automaticamente seus recursos conforme a demanda. Os modelos de implementação da nuvem são:

- **Nuvem Pública**: esse é considerado o modelo mais comum para computação em nuvem. Nele, a infraestrutura é operada por terceiros, ou seja, pertence a um

provedor externo que armazena e mantém os dados dos usuários. A categoria é baseada no modelo de pagamento por uso (*pay-per-use*), em que a empresa, ou instituição, contratante paga apenas pelos recursos utilizados (QI NETWORK, 2024).

- **Nuvem Privada**: a característica deste tipo de serviço é que os recursos são fornecidos exclusivamente para um grupo de usuários de uma instituição. Neste modelo a infraestrutura é própria da instituição (VERAS, 2012).

- **Nuvem Híbrida**: neste modelo, temos uma composição no uso de nuvens públicas e privadas por uma mesma instituição ou empresa. Isso pode estar atrelado a regras de *compliance*, que podem exigir o armazenamento de dados locais e sigilosos em uma nuvem privada, permitindo a transferência desses dados entre ambas as nuvens. Dessa forma, é possível aproveitar as vantagens de aplicações e portabilidade de dados oferecidas pela combinação das nuvens públicas e privadas (VERAS, 2012).

- **Nuvem Comunitária**: o ambiente de nuvem é provisionado para uso exclusivo por uma determinada comunidade de consumidores que compartilham interesses comuns, como requisitos de segurança e monitoramento (VERAS, 2012).

2.1.2 Modelos de Serviços

Os modelos de serviço em computação em nuvem representam diferentes formas de oferecer recursos de Tecnologia da Informação (TI) através da internet, onde cada modelo fornece um nível distinto de abstração e gerenciamento para os usuários finais. Eles são projetados para atender a várias necessidades de negócios, permitindo que as organizações aproveitem a flexibilidade, escalabilidade e eficiência da nuvem sem a necessidade de manter infraestrutura física própria.

Estes modelos podem ser divididos em três tipos de serviços: *Platform as a Service* (PaaS), *Software as a Service* (SaaS) e *Infrastructure as a Service* (IaaS). Cada um desses modelos oferece diferentes níveis de controle, flexibilidade e gerenciamento.

- **IaaS**: provedores de infraestrutura como serviço oferecem recursos virtualizados, como processamento, memória, armazenamento e rede, permitindo que os usuários acessem e configurem a infraestrutura conforme suas necessidades. Esse modelo possibilita a contratação exclusiva da infraestrutura virtual, eliminando a necessidade de investimento em *hardware* físico próprio e oferecendo maior flexibilidade e controle sobre o ambiente de execução das aplicações (RASHID; CHATURVEDI, 2019).

- **PaaS**: no modelo de plataforma como serviço (PaaS), um provedor externo oferece um ambiente completo para desenvolvimento e implantação de aplicações. Com o

PaaS, os usuários – em geral, desenvolvedores de *software* – podem criar, executar e gerenciar suas aplicações sem se preocupar com a configuração ou manutenção da infraestrutura. Essa solução é projetada para dar suporte a todo o ciclo de vida de um aplicativo *web*, incluindo desenvolvimento, testes, implantação, gerenciamento e atualizações (MICROSOFT AZURE, 2024).

- **SaaS**: neste modelo de serviço, os provedores gerenciam e mantêm o *software* de aplicação, o sistema operacional e outros recursos. Aplicações SaaS eliminam a necessidade de instalar e manter o *software* localmente, oferecendo vantagens como a escalabilidade e a eficiência proporcionadas pelo *multitenancy* — arquitetura na qual uma única instância de software atende a vários usuários (RASHID; CHATURVEDI, 2019).

2.1.3 Benefícios e Desafios

A escolha de um modelo de serviço ou de implantação em computação em nuvem deve ser baseada em uma análise dos benefícios e desafios associados. O usuário, ou organização, é responsável por avaliar se a relação entre benefícios e riscos é aceitável para adotar os modelos de serviços e implantação discutidos nas seções anteriores (SAJID; RAZA, 2013).

Ao considerar os principais benefícios da computação em nuvem,Thomas (2009) destaca pontos como a redução nos custos com *hardware*, processamento, armazenamento, largura de banda e *software*. Esses custos são sob demanda, ou seja, o usuário paga apenas pelo que consome. Além disso, pode-se citar o aumento da escalabilidade e confiabilidade, pois ao hospedar o serviço em servidores na nuvem, são proporcionados benefícios como *backup*, redução de latência, tolerância a falhas e suporte a picos de demanda.

Ao discutirmos os desafios da computação em nuvem, um dos principais é a segurança das informações. Anteriormente, esses dados eram armazenados em data centers das próprias organizações e em computadores pessoais. No contexto da computação em nuvem, os usuários desconhecem a localização exata de seus dados e a origem dos dados armazenados junto aos seus. Portanto, a proteção da privacidade dos usuários e a integridade das informações são essenciais para os provedores de infraestrutura e serviços (KAUFMAN, 2009). Outro fator desafiador na computação em nuvem é a disponibilidade, pois os usuários esperam que as aplicações estejam sempre acessíveis e operacionais, especialmente nos momentos de maior necessidade (SUN MICROSYSTEMS, INC., 2009).

Ao optar por soluções de computação em nuvem, é importante avaliar fatores como segurança, privacidade, disponibilidade e custo-benefício, entre outros. Esses aspectos ajudam a garantir que a adoção dessas tecnologias atenda às necessidades e expectativas de organizações e usuários.

2.2 Microsserviços

Os microsserviços ganharam ampla popularidade nos últimos anos, impulsionados pela adoção de práticas DevOps e tecnologias de contêineres como Kubernetes e Docker. Empresas como Netflix, Amazon e LinkedIn têm utilizado essa arquitetura de *software* em suas *stacks* de desenvolvimento, aproveitando suas vantagens para melhorar a escalabilidade, flexibilidade e eficiência operacional (VIGGIATO et al., 2018).

Em contraste, a arquitetura monolítica, tradicionalmente adotada por muitas organizações (THATIKONDA; MUDUNURI, 2024), caracteriza-se pela integração de todos os componentes de um sistema em um único bloco de programa executável. Nesse modelo, a aplicação é desenvolvida e implantada como uma única unidade, o que pode facilitar o desenvolvimento inicial e a gestão centralizada. No entanto, conforme a aplicação cresce em complexidade e tamanho, surgem desafios significativos, como dificuldades para escalar partes específicas do sistema, maior tempo de implantação e manutenção complexa (SHADRACK, 2023).

Nesta seção, discutiremos as características das arquiteturas de microsserviços e como essa abordagem se diferencia das demais, especialmente das arquiteturas monolíticas.

2.2.1 Arquitetura de Microsserviços

Microsserviços podem ser definidos como uma arquitetura de *software* onde cada função é implementada como um serviço independente. O objetivo dos microsserviços é dividir a aplicação em pequenos conjuntos de outras aplicações de modo que a falha de uma dessas pequenas aplicações não implique uma falha total do sistema como um todo. Uma aplicação baseada em microsserviços possui componentes independentes, mas com um alto grau de acoplamento; ou seja, esses componentes, embora independentes, trabalham juntos para entregar um serviço (CONCEIÇÃO; PINTO, 2021).

Os microsserviços disponibilizam suas funcionalidades internamente e para sistemas web através de uma arquitetura cliente-servidor (Application Programming Interface Representational State Transfer (API RESTful)) ou uma arquitetura orientada a eventos (Publicação e Assinatura (Pub/Sub)). Essa abordagem os caracteriza como um sistema distribuído. Microsserviços utilizam unidades autônomas que são coordenadas em uma infraestrutura distribuída por meio de tecnologias de contêineres leves, como Docker, o que facilita a escalabilidade e a gestão descentralizada.

A adoção dessa arquitetura também está associada a práticas ágeis, como DevOps, que reduzem o tempo entre a implementação de mudanças e a sua liberação em produção. Essa independência entre os microsserviços permite que cada um seja desenvolvido, implantado e escalado separadamente, otimizando a autonomia e a substituibilidade dos serviços, embora também apresente desafios significativos, como a complexidade de gerenciamento

e a necessidade de monitoramento contínuo (HASSAN; ALI; BAHSOON, 2017).

De acordo com IBM (2024), embora a discussão sobre microsserviços geralmente foque em definições e características arquitetônicas, seu valor real é melhor compreendido pelos benefícios corporativos e de negócios, como:

- Atualizações de código mais fáceis, permitindo a inclusão de novos recursos ou funcionalidades sem alterar todo o aplicativo.
- Uso de diferentes *stacks* e linguagens de programação para componentes distintos.
- Possibilidade de ajustar a escala dos componentes de forma independente, reduzindo o desperdício e os custos associados ao ajuste de escala de aplicativos inteiros, especialmente quando apenas um recurso está sobrecarregado.

2.2.2 Autenticação e Autorização em Microsserviços

A segurança em arquiteturas baseadas em microsserviços é essencial para garantir que apenas usuários e sistemas autorizados possam acessar dados e funcionalidades sensíveis. Neste contexto, dois conceitos fundamentais são amplamente empregados:

- **Autenticação:** Processo que verifica a identidade de um usuário ou sistema, normalmente realizado por meio de credenciais como *login* e senha, *tokens* de autenticação (como JSON Web Tokens (JWT)) ou certificados digitais (KOVALOV, 2024).
- **Autorização:** Define quais ações e recursos um usuário autenticado pode acessar, garantindo que ele possua as permissões necessárias para interagir com o sistema (LAMARI et al., 2024).

Diferentemente das arquiteturas monolíticas, onde a autenticação e autorização costumam ser centralizadas (NEWMAN, S., 2019), em microsserviços essas responsabilidades são distribuídas entre diferentes componentes do sistema. Embora essa abordagem ofereça maior flexibilidade e escalabilidade, também aumenta a complexidade da gestão de credenciais, permissões e segurança. Assim, diferentes estratégias foram desenvolvidas para lidar com esse desafio, destacando-se:

- **Autenticação baseada em Tokens:** Um JWT é gerado no momento da autenticação e anexado a todas as requisições subsequentes, permitindo que os serviços validem a identidade do usuário sem a necessidade de consultas contínuas a um servidor central. Essa abordagem reduz a latência e melhora a escalabilidade do sistema, mas exige medidas rigorosas para evitar ataques como a reutilização de tokens expirados (KOVALOV, 2024).

- **Proxies de Autenticação (*Application Programming Interface Gateway* (API Gateway)):** Uma alternativa para descentralizar a autenticação sem comprometer a segurança é o uso de um API Gateway. Esse componente atua como intermediário entre os clientes e os microsserviços internos, sendo responsável por validar credenciais, gerenciar logs de acesso e encaminhar apenas requisições autenticadas aos serviços solicitados (OJHA, 2024). Essa solução facilita a administração da segurança, mas pode se tornar um ponto único de falha se não for bem projetada.

- ***Open Authorization* (OAuth) 2.0 e *OpenID Connect* (OIDC)**: Os protocolos OAuth 2.0 e OIDC são amplamente utilizados para delegação de autenticação e autorização em sistemas distribuídos. O OAuth 2.0 permite conceder permissões a terceiros sem expor credenciais do usuário, enquanto o OIDC adiciona uma camada de autenticação sobre o OAuth, permitindo a gestão unificada de identidades em múltiplos serviços. Essa abordagem é comum em ambientes corporativos e plataformas SaaS, garantindo um acesso mais seguro e centralizado (LAMARI et al., 2024).

A escolha da abordagem de autenticação e autorização depende das necessidades específicas do sistema, levando em conta fatores como desempenho, escalabilidade e nível de segurança exigido. Enquanto a autenticação distribuída proporciona maior autonomia aos serviços, também impõe desafios, como a sincronização de credenciais, mitigação de ataques de repetição e proteção contra vazamento de identidade.

2.2.3 Comparação com Arquitetura Monolítica

A escolha entre arquiteturas monolíticas e de microsserviços depende de diversos fatores, incluindo a complexidade do sistema, a necessidade de escalabilidade e os recursos disponíveis para manutenção. Arquiteturas monolíticas são caracterizadas pela integração de todos os componentes em uma única aplicação, o que simplifica o desenvolvimento inicial e a implantação. No entanto, essa abordagem pode levar a problemas de escalabilidade e manutenção à medida que o sistema cresce, já que qualquer alteração ou atualização requer a recompilação de toda a aplicação (GOS; ZABIEROWSKI, 2020). Além disso, a interdependência dos componentes pode resultar em uma maior vulnerabilidade a falhas, onde um erro em um módulo pode afetar toda a aplicação.

Por outro lado, a arquitetura de microsserviços divide a aplicação em serviços menores e independentes, cada um responsável por uma funcionalidade específica. Essa modularidade permite uma maior flexibilidade e facilidade de manutenção, pois diferentes equipes podem trabalhar em serviços separados, atualizando e escalando-os independentemente (NEWMAN, Sam, 2015). No entanto, essa abordagem introduz uma complexidade adicional na gestão e comunicação entre serviços, além de exigir uma infraestrutura robusta

para orquestração e, especialmente, para o monitoramento dos microsserviços, incluindo auditoria e telemetria.

Essas arquiteturas também podem ser implementadas em diferentes modelos de desenvolvimento de *software*, como *waterfall* ou ágil. No caso dos microsserviços, é possível ainda integrar serviços PaaS e SaaS para otimizar partes da solução, facilitando a manutenção e a escalabilidade. Dessa forma, a escolha da arquitetura deve considerar a natureza do sistema, a capacidade da equipe de desenvolvimento e as demandas futuras, ajustando-se ao contexto específico de cada projeto.

2.3 Banco de Dados

De acordo com Oracle (2024a), um banco de dados pode ser definido como uma coleção organizada de informações ou dados estruturados, no caso de sistemas de bancos de dados relacionais. Os bancos de dados são componentes essenciais para a persistência de dados em sistemas de informação e desempenham um papel crítico em praticamente todas as áreas da computação, incluindo negócios, engenharia, medicina, educação, entre outras. Segundo Elmasri e Navathe (2005), um banco de dados é uma coleção onde os dados são fatos que podem ser gravados e que possuem um significado implícito. Esses dados fornecem a base para a informação processada e interpretada, permitindo a tomada de decisões.

Um banco de dados possui fontes específicas das quais os dados são derivados e diferentes níveis de interação com eventos do mundo real. Essas características garantem que o banco de dados não apenas armazene dados, mas também os organize e os torne acessíveis e úteis para seus usuários, refletindo as mudanças e necessidades do mundo que ele representa.

2.3.1 Banco de Dados Relacionais

Os bancos de dados relacionais organizam informações em tabelas estruturadas com linhas e colunas, onde cada linha representa um registro específico, e cada coluna descreve os atributos relacionados a esse registro (AMAZON WEB SERVICES, 2023a). Caracterizam-se por exigir um esquema rígido, com a necessidade de definir previamente a estrutura das tabelas e seus relacionamentos. Um exemplo disso é o uso de chaves primárias e estrangeiras para manter a integridade referencial dos dados (LIMA, 2024).

Além disso, esses bancos de dados seguem as premissas de conformidade Atomicidade, Consistência, Isolamento e Durabilidade (ACID), um conjunto de propriedades que garante que as transações sejam processadas de maneira confiável. A atomicidade assegura que uma transação seja inteiramente completada ou revertida em caso de falha. A consistência mantém a transição do banco de dados entre estados válidos. O isolamento

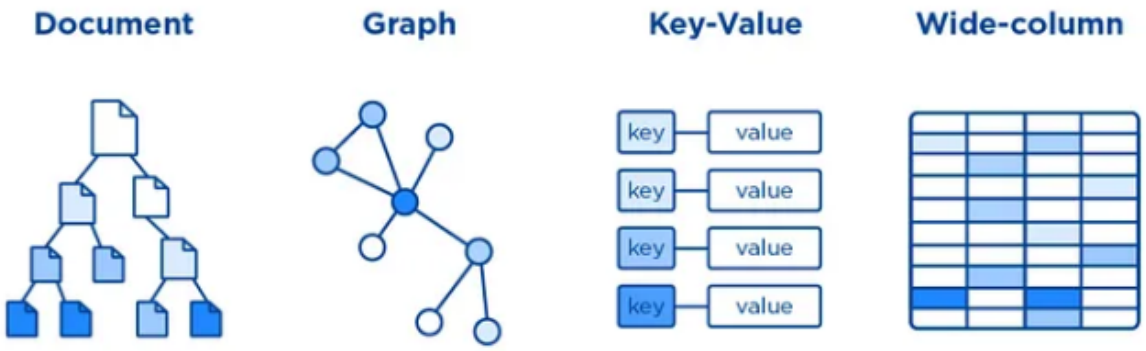
impede a interferência entre transações simultâneas. Por fim, a durabilidade garante que as mudanças sejam permanentes, mesmo após falhas no sistema (COLIN, 2023).

Outra característica é a capacidade de realizar consultas complexas, onde o uso de *Structured Query Language* (SQL) permite combinar dados de múltiplas tabelas, facilitando a manipulação e recuperação de informações de forma declarativa.

2.3.2 Banco de Dados Não Relacionais

Bancos de dados não relacionais, ou *Not Only SQL* (NoSQL), oferecem uma maneira mais flexível de armazenar dados, utilizando formatos não tabulares que podem ser categorizados em banco de dados orientado a documentos (*documents stores*), banco de dados orientado a chave-valor (*key-value stores)*, banco de dados orientado a colunas (*wide-columns stores*) ou banco de dados orientado a grafos (*graphs stores*) (GOOGLE CLOUD, 2024). A Figura 1 ilustra um exemplo de como esses bancos organizam seus dados.

Figura 1 – Categorias de bancos não relacionais



Fonte:Linhares (2024).

Esses bancos de dados são projetados para lidar com grandes volumes de dados não estruturados, proporcionando escalabilidade e desempenho em aplicações distribuídas. Segundo Lourenço et al. (2015), diferentemente dos bancos de dados relacionais, que priorizam o suporte a transações ACID, os bancos de dados *Not Only SQL* (NoSQL) sacrificam pelo menos um desses atributos para oferecer maior disponibilidade e escalabilidade, além de permitir acessos simultâneos mais eficientes.

Diante das demandas de escalabilidade e flexibilidade nas aplicações modernas, os bancos de dados NoSQL surgem como uma alternativa aos modelos relacionais tradicionais. Cada tipo —seja chave-valor, documentos, colunas ou grafos— é projetado para resolver problemas específicos, permitindo que as organizações escolham a solução mais adequada conforme suas necessidades. Essa diversidade de abordagens, combinada com a capacidade de escalar horizontalmente e otimizar o desempenho em grandes volumes de dados, torna

os bancos NoSQL uma alternativa considerável para o desenvolvimento de sistemas distribuídos e de alta disponibilidade.

2.4 *Pipeline* de Dados

Um *pipeline* de dados é um sistema organizado que coleta dados brutos da origem, processa, armazena e compartilha de forma automatizada, visando fornecer dados limpos e consistentes para aplicativos e usuários finais. Ele é essencial em arquiteturas de processamento de dados, permitindo a ingestão, transformação, armazenamento e análise eficiente e escalável de grandes volumes de dados (OLEGHE; SALONITIS, 2020).

Um *pipeline* de dados é composto por uma série de etapas ou processos que os dados percorrem desde sua origem até o destino final. Essas etapas incluem coleta, limpeza, transformação, armazenamento e análise. O objetivo principal de um *pipeline* de dados é automatizar o fluxo de informações, garantindo que estejam disponíveis, sejam precisas e utilizáveis para diversas aplicações, como análise de negócios, aprendizado de máquina e visualização de dados (AMAZON WEB SERVICES, 2024).

Nesta seção, serão exploradas as etapas que envolvem a implementação de um *pipeline* de dados, as tecnologias utilizadas e técnicas de processamento de dados, como *Extract, Transform, Load*l (ETL) e *Extract, Load, Transform*l (ELT).

2.4.1 Componentes Principais de um *Pipeline* de Dados

Os componentes de um *pipeline* de dados são responsáveis por conduzir os dados desde sua origem até a extração de valor, que pode ser feita por meio de análises e relatórios. Nesta seção, será detalhado o papel de cada componente com exemplos aplicados.

- **Fonte de Dados**: As fontes de dados alimentam o *pipeline* e podem ser internas, como sistemas transacionais SQL, NoSQL, API de terceiros, sistemas de sensores IoT e dados de um CRM ou ERP, que gerenciam grandes volumes de transações em tempo real (HOFFER; PRESCOTT; MCFADDEN, 2007). Já as fontes externas incluem dados de parceiros comerciais, governos ou mercados, frequentemente relacionados a concorrência e demografia de clientes (RANJAN, 2009).

- **Ingestão de Dados**: A ingestão de dados refere-se à coleta de dados a partir das fontes mencionadas anteriormente. Ela pode ser realizada de duas maneiras: por lotes (*batch*), onde os dados são processados em intervalos definidos, como diariamente ou semanalmente, ou de forma contínua (*streaming*), em que os dados são processados em tempo real na medida em que são gerados (DATACAMP, 2023). A escolha entre esses métodos depende das necessidades do sistema e do volume de

dados a ser processado. A ingestão em lotes é geralmente mais simples e adequada para aplicações que não exigem respostas imediatas, enquanto a ingestão contínua é essencial para cenários que demandam *insights* em tempo real, como a detecção de fraudes ou análise de fluxos de tráfego em redes (IBM, 2023b).

- **Processamento de Dados**: Esta é uma das etapas mais críticas do *pipeline*, onde os dados coletados são avaliados para garantir sua qualidade. Nessa fase, os dados brutos passam por um processo de análise para verificar sua "validade", já que podem conter erros, estar incompletos ou incluir campos inválidos. Assim, ocorre a limpeza, transformação e enriquecimento dos dados, assegurando que eles estejam corretos e prontos para serem usados nas próximas etapas do *pipeline* (NAEEM, 2024).

- **Armazenamento de Dados**: Os dados transformados e prontos para uso são armazenados em bancos de dados ou sistemas de arquivos distribuídos, sendo esta uma das etapas mais críticas do *pipeline* de dados, pois não só fornece a infraestrutura necessária para a análise posterior, mas também atua como um repositório confiável para manter dados brutos e transformados (REIS; NAKAOKA; NETO, 2023).
  O tipo de armazenamento depende da estrutura dos dados e das necessidades de acesso. Por exemplo, dados estruturados podem ser armazenados em um banco de dados relacional, enquanto dados não estruturados, como imagens ou arquivos, podem ser armazenados em soluções de armazenamento de objetos.

- **Análise de Dados**: Uma vez armazenados, os dados passam por um processo de análise que visa gerar *insights* a partir das informações coletadas e compará-los com grandes conjuntos de dados já existentes (DATABRICKS, 2024). Esse processo permite identificar novas tendências e padrões a partir dos dados, identificando as relações entre variáveis em grandes volumes de dados, possibilitando a criação de modelos que representem processos do mundo real. Cientistas de dados e analistas utilizam o repositório centralizado, aplicando diversas técnicas e ferramentas, como SQL avançado, aprendizado de máquina e métodos estatísticos, para extrair informações úteis, como padrões, relações, tendências e anomalias (DOMO, 2024).

2.4.2 Técnicas de Processamento de Dados em Pipelines

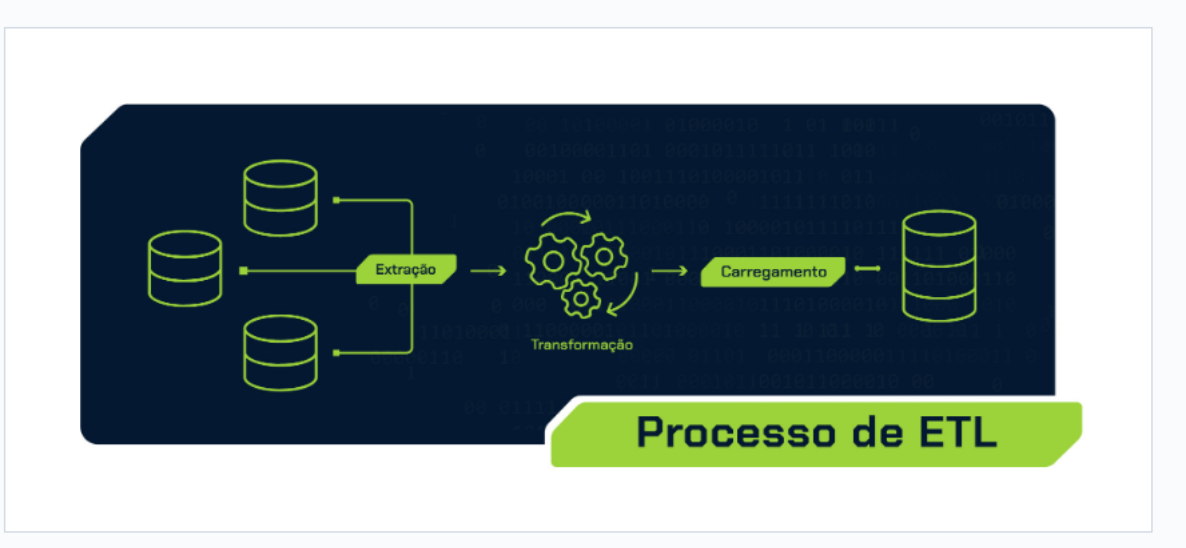
O processamento de dados em *pipelines* envolve a aplicação de técnicas específicas que organizam e preparam os dados para seu uso final. Duas abordagens comuns nesse contexto são o *Extract, Transform, Load*l (ETL) e o *Extract, Load, Transform*l (ELT), que diferem na sequência e no momento em que as transformações são aplicadas aos dados ao longo do fluxo. Essas técnicas são essenciais para garantir que os dados brutos coletados sejam adequadamente estruturados e estejam prontos para análise e outras aplicações.

2.4.2.1 ETL (*Extract, Transform, Load*)

No modelo ETL, representado na Figura 2, os dados seguem um fluxo tradicional de análise de dados, começando pela extração de dados de diversas fontes, transformados conforme necessário e, então, carregados em um sistema de destino, como um *data warehouse* (CURSOS PM3, 2024). Esse método é especialmente adequado para cenários onde as transformações de dados precisam ser realizadas antes do carregamento, garantindo assim a consistência e a qualidade dos dados processados.

De acordo com Rambabu, Althati e Selvaraj (2023), as vantagens dessa abordagem incluem um maior controle sobre a qualidade dos dados, tornando-a uma opção para situações em que transformações complexas são necessárias. No entanto, uma das desvantagens é a introdução de latência, já que as transformações devem ser concluídas antes do carregamento, o que pode aumentar o tempo total de processamento.

Figura 2 – Processo ETL



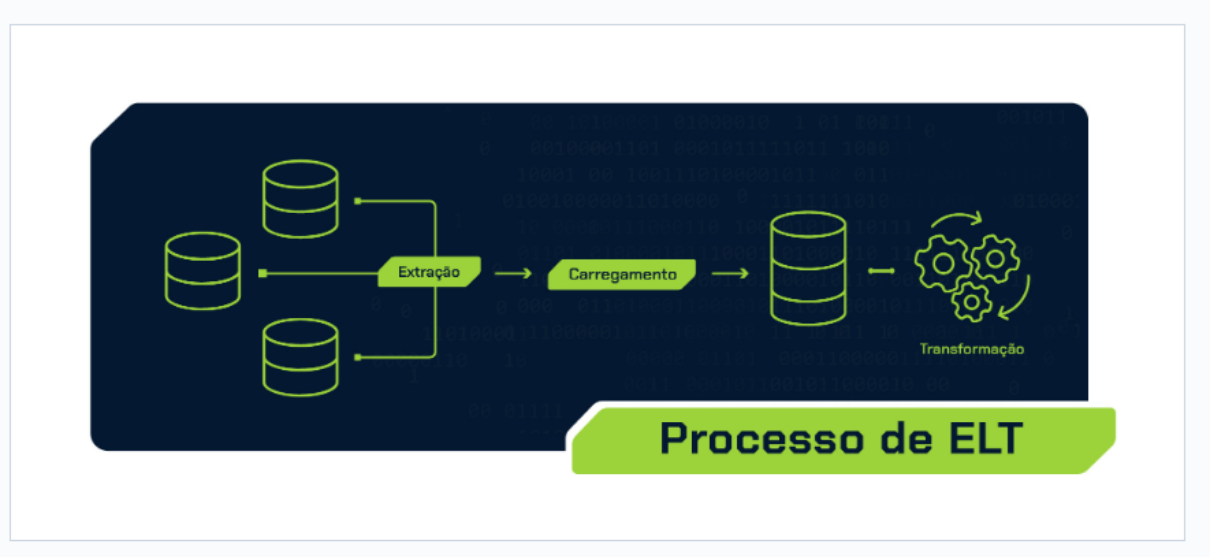
Fonte:Calanca (2023).

2.4.2.2 ELT (*Extract, Load, Transform*)

De acordo com Rastelli (2024), o ELT é uma evolução do processo ETL, que modifica a sequência tradicional das etapas para melhorar o desempenho no processamento de grandes volumes de dados. No modelo ELT (Figura 3), os dados são extraídos de suas fontes e carregados diretamente no destino, como um *data lake* ou *data warehouse*, sem que transformações sejam realizadas previamente. Essas transformações ocorrem posteriormente, conforme a demanda, dentro do ambiente de armazenamento, proporcionando uma maior flexibilidade e agilidade no processamento e manipulação das informações (AMAZON WEB SERVICES, INC., 2024).

A abordagem ETL é especialmente eficaz em cenários que envolvem grandes quantidades de dados não estruturados ou semiestruturados, permitindo a escalabilidade necessária para lidar com esses volumes de dados em tempo real. No entanto, para que isso seja possível, é necessário dispor de uma infraestrutura robusta que suporte o processamento dinâmico das transformações diretamente no ambiente de armazenamento (RAMBABU; ALTHATI; SELVARAJ, 2023).

Figura 3 – Processo ELT



Fonte:Calanca (2023).

2.5 Algoritmos de classificação

A classificação é uma técnica na análise de dados usada para organizar informações em categorias com base em padrões identificados. Os algoritmos de classificação utilizam dados históricos para categorizar novas informações (MATEUS, 2023).

Segundo Finnstats (2022), os algoritmos de *machine learning* podem ser classificados em três tipos principais:

- **Aprendizado supervisionado**: usa dados rotulados para treinar o modelo, permitindo que ele faça previsões com novos dados.

- **Aprendizado não supervisionado**: o modelo trabalha sem dados rotulados, aprendendo a identificar padrões por conta própria, sendo conhecido como algoritmo de *clustering*.

- **Aprendizado por reforço**: baseado em tentativa e erro, onde o modelo melhora com base nos próprios acertos e falhas.

Os algoritmos de *machine learning* podem ser agrupados em duas categorias: classificadores estatísticos e classificadores estruturais.

Classificadores estatísticos utilizam modelos probabilísticos para fazer predições. Esses algoritmos funcionam estimando a probabilidade de um dado pertencer a uma determinada classe com base em informações estatísticas. Entre os classificadores estatísticos estão o Naive Bayes e o *Support Vector Machines* (SVM).

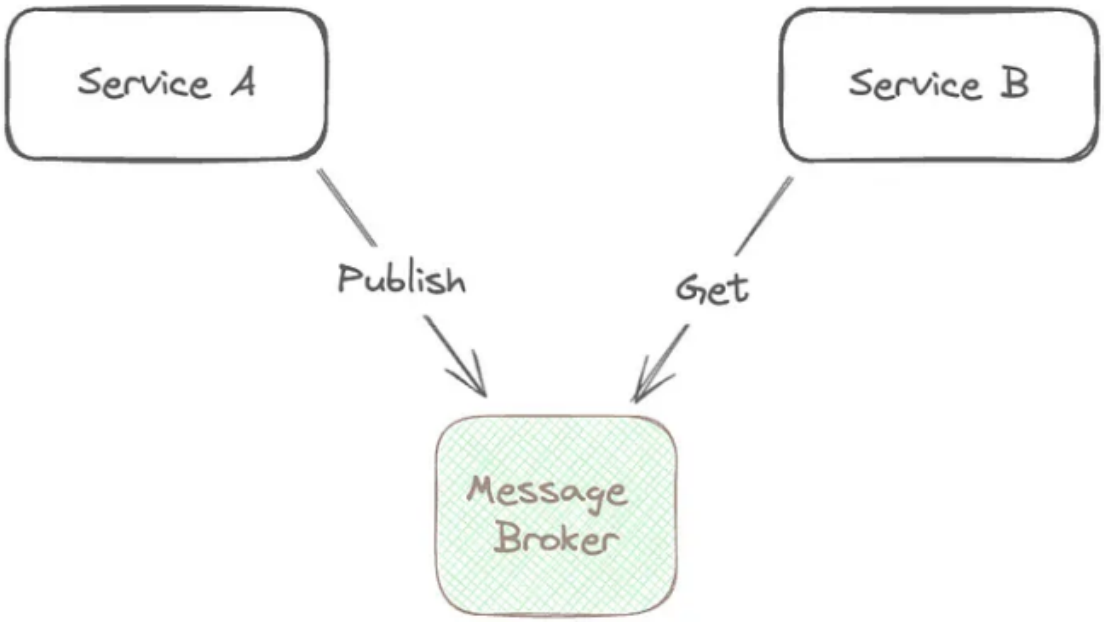
Classificadores estruturais não baseiam suas decisões em probabilidades, mas sim em regras ou estruturas explícitas. Eles frequentemente usam árvores de decisão, redes neurais, ou distâncias entre pontos no espaço de dados para classificar novas amostras. Esses algoritmos funcionam aprendendo uma representação explícita da relação entre os atributos dos dados e as classes. Entre os classificadores estruturais, destacam-se as Árvores de Decisão, o *K-Nearest Neighbors* (KNN) e o *Random Forest* (LENZ, 2017).

A distinção entre essas abordagens permite que diferentes algoritmos sejam aplicados conforme a natureza dos dados e os objetivos da análise. Em contextos onde o volume e a diversidade de dados são elevados, a escolha do método de classificação adequado pode impactar diretamente na qualidade das previsões e na capacidade de extrair informações relevantes. Ao compreender as características de cada técnica — seja estatística ou estrutural —, é possível selecionar o algoritmo mais eficaz para lidar com os desafios específicos de classificação, otimizando a tomada de decisões e a precisão dos resultados.

2.6 Serviços de troca de mensagens

De acordo com Moschetta (2024), os serviços de troca de mensagens são sistemas que facilitam a comunicação entre diferentes componentes de *software*, permitindo a transmissão de informações de forma assíncrona. Esses serviços são essenciais em arquiteturas distribuídas, nas quais a comunicação direta e síncrona entre componentes pode se tornar ineficiente ou impraticável. Em um modelo de troca de mensagens assíncronas, o sistema emissor envia a mensagem independentemente da disponibilidade imediata do sistema receptor. Um exemplo típico é a notificação sobre uma alteração de estado ou evento crítico: a comunicação assíncrona permite que a notificação seja armazenada e entregue posteriormente, assegurando que o sistema de destino (denominado como *Service* B, conforme ilustrado na Figura 4) receba a informação, mesmo que esteja temporariamente indisponível no momento do envio.

Figura 4 – Filas: Assegurando Entrega em Comunicação Assíncrona



Fonte:Moschetta (2024).

2.6.1 Serviços de troca de mensagens na comunicação entre Microsserviços

Na arquitetura de microsserviços, os serviços de troca de mensagens desempenham um papel importante na comunicação entre os diversos microsserviços que compõem uma aplicação. Sistemas de mensagens, baseados no paradigma de notificação de eventos, emergem como uma alternativa para mediar a comunicação entre microsserviços. Esses sistemas operam de forma independente do *software* principal e permitem a entrega e recepção de mensagens entre serviços distribuídos, sem que esses serviços precisem se conhecer diretamente (FUNDAÇÃO DE SOFTWARE APACHE, 2017). Sistemas de mensagens são normalmente divididos em dois modelos:

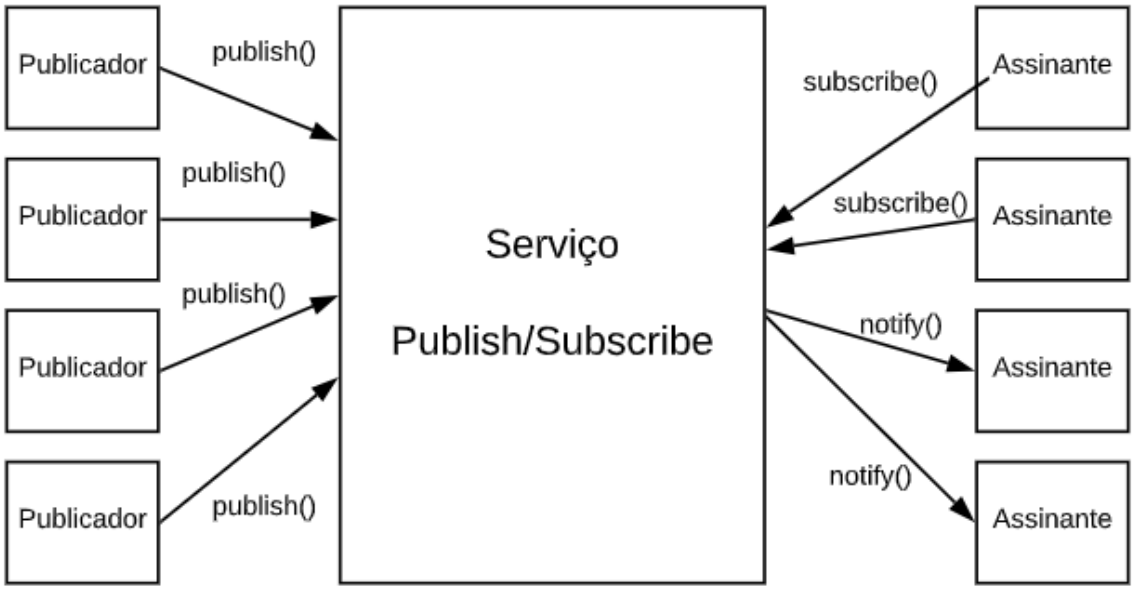
- **Filas de Mensagens (*Message Queues*)**: Conforme ilustrado na Figura 5, neste modelo, as mensagens (ou eventos) ficam armazenadas na fila até serem processadas e então removidas. Cada mensagem é consumida uma única vez, sempre por um único consumidor e possibilita dividir o processamento de dados entre vários consumidores, evitando a duplicidade de leitura. De acordo com Amazon Web Services (2023b), as filas de mensagens servem para desacoplar processos mais complexos, armazenar tarefas em *buffers* ou lotes, e ajudar a distribuir de maneira uniforme picos de carga de trabalho.

Figura 5 – Arquitetura de Filas de Mensagens: Comunicação Assíncrona entre Produtor e Consumidor

Fonte:Amazon Web Services (2023b).

- **Publicação/Assinatura (*Publish/Subscribe*)**: No modelo Pub/Sub, as mensagens são chamadas de eventos. Os componentes dessa arquitetura incluem publicadores e assinantes: os publicadores geram eventos e os enviam para o serviço de Pub/Sub, que geralmente opera em um servidor separado. Já os assinantes precisam assinar previamente os eventos dos quais desejam receber notificações (VALENTE, 2020). Assim que um evento é publicado, todos os assinantes interessados são notificados, como ilustrado na Figura 6.

Figura 6 – Arquitetura *Publish/Subscribe*: Fluxo de Comunicação entre Publicadores e Assinantes



Fonte:Valente (2020).

No contexto de uma aplicação baseada em eventos, os sistemas de mensagens que adotam o modelo de Publicação/Assinatura representam a opção mais adequada (FERNANDES, 2018).

2.7 *Chatbots*

Segundo Oracle (2024b), um *chatbot* é um programa que simula conversas humanas, seja por texto ou voz, permitindo interações com dispositivos digitais de forma semelhante a uma conversa com uma pessoa real. Nesse contexto, o usuário pode emitir um comando ao *chatbot*, que, por sua vez, pode utilizar-se de mecanismos de processamento de linguagem natural para interpretar a solicitação e localizar a informação desejada em uma base de conhecimento previamente criada (MELO; PESSOA; PASCHOAL, 2023).

Esses sistemas podem variar em complexidade, desde modelos simples que respondem a perguntas básicas com frases curtas, até assistentes digitais avançados, que aprendem e se adaptam para oferecer interações cada vez mais personalizadas conforme recebem mais dados.

Existem diferentes tipos de *chatbots*, classificados principalmente em dois grupos: *chatbots* **baseados em regras** e *chatbots* **baseados em inteligência artificial.**

Chatbots baseados em regras utilizam lógica condicional para construir fluxos de conversa automatizados, atuando como uma versão interativa de perguntas frequentes. O designer de conversas programa combinações específicas de perguntas e respostas para que o *chatbot* entenda a entrada do usuário e responda de maneira precisa, mas limitada a essas interações predefinidas (IBM, 2023a). Esses *chatbots* funcionam por meio da detecção de palavras-chave, o que facilita seu treinamento e permite que lidem bem com perguntas já esperadas. No entanto, eles enfrentam dificuldades ao receber consultas complexas ou inesperadas, pois suas respostas são totalmente dependentes do conteúdo programado previamente, o que restringe sua capacidade de adaptação a novos contextos (THORAT; JADHAV, 2020).

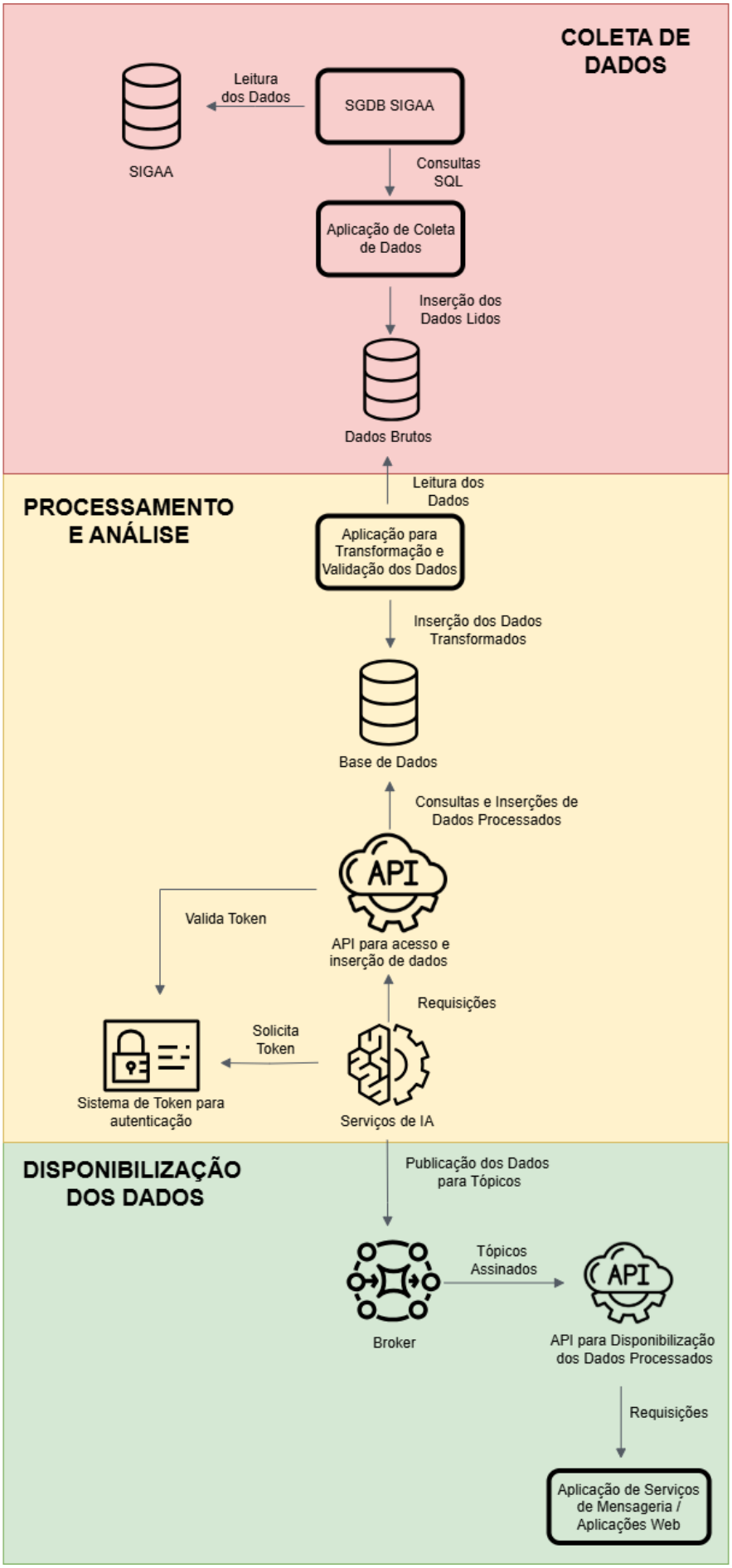
Chatbots baseados em IA combinam técnicas de aprendizado de máquina, inteligência artificial e processamento de linguagem natural para compreender o usuário, superando limitações dos *chatbots* baseados apenas em regras. Esses *chatbots* são capazes de entender a linguagem humana e possuem um fluxo de conversa pré-definido, o que permite resolver as solicitações de forma mais flexível, mesmo quando as perguntas dos usuários são formuladas de maneiras diferentes. Com o uso de IA e aprendizado de máquina, esses *chatbots* podem adaptar suas respostas e aprimorar suas interações com o tempo (MACIEL et al., 2021).

3 DESENVOLVIMENTO

Neste capítulo, serão detalhadas as etapas que compõem a especificação dos serviços modulares para o sistema proposto, abrangendo desde a coleta de dados até sua disponibilização para integração com serviços de troca de mensagens e aplicações web. A Figura 7 ilustra esse fluxo, que se organiza em três fases principais: coleta de dados, processamento e análise, e disponibilização dos dados.

Ao longo desta seção, serão apresentadas as interfaces e os serviços que compõem cada uma dessas fases, detalhando suas funcionalidades, interações e os dados que manipulam. A especificação dessas interfaces busca garantir a modularidade e a escalabilidade do sistema.

Figura 7 – Arquitetura de Coleta, Processamento e Disponibilização de Dados

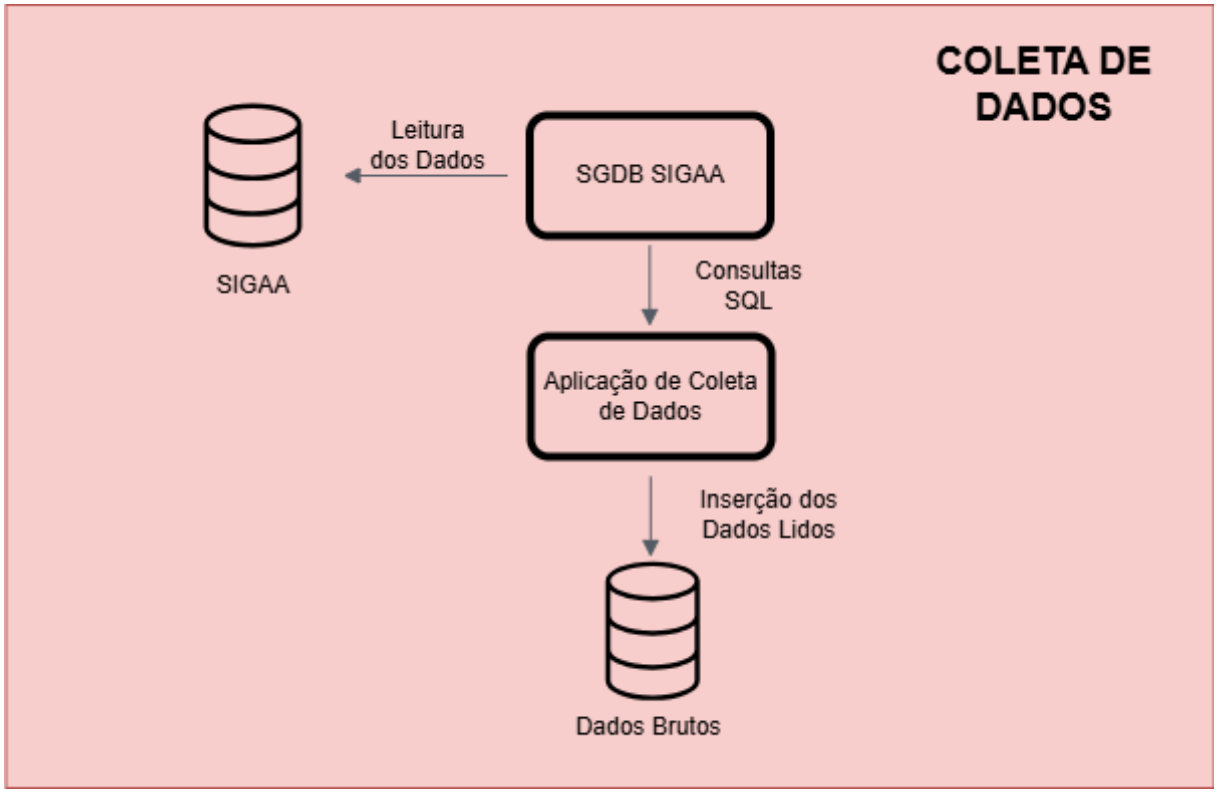


Fonte: A própria autora.

3.1 Coleta de Dados

A coleta de dados consiste na extração das informações necessárias para alimentar o sistema proposto. Nessa etapa, os dados são obtidos a partir do SIGAA e armazenados inicialmente em um repositório de dados brutos, sem modificações. Esse processo garante que as informações estejam disponíveis para as fases seguintes, onde serão transformadas e analisadas. A Figura 8 ilustra esse fluxo, destacando as principais etapas envolvidas na coleta. A seguir, são apresentadas as fontes dos dados, os mecanismos de extração e o armazenamento utilizado nesta etapa.

Figura 8 – Estrutura da Etapa de Coleta de Dados



Fonte: A própria autora.

3.1.1 Fonte dos Dados

A coleta de dados no sistema proposto tem como principal fonte o SIGAA, utilizado pelo IFSC para gerenciar informações acadêmicas. Esse sistema centraliza registros sobre os alunos, como matrículas, notas, frequência e histórico escolar, fornecendo informações que podem ser acessadas para monitoramento do desempenho acadêmico e identificação de possíveis casos de evasão e retenção.

Embora estudos sobre evasão apontem diversos fatores externos, como aspectos socioeconômicos e motivacionais (COIMBRA; SILVA; COSTA, 2021), o presente trabalho se restringe às informações disponíveis diretamente no sistema acadêmico institucional.

Segundo Silva Garcia, Lara e Antunes (2021), mesmo com essa limitação, os dados estruturados do SIGAA já permitem identificar padrões relevantes de risco de evasão e retenção.

Para efeito de especificação, assume-se que a plataforma acadêmica utiliza um Sistema de Gerenciamento de Banco de Dados (SGBD), permitindo o acesso estruturado às informações disponíveis no sistema. Os principais dados que podem ser extraídos incluem:

- **Histórico acadêmico**: disciplinas cursadas, status de aprovação/reprovação, carga horária concluída e pendente. Essas informações são consolidadas ao final de cada semestre.
- **Notas e frequência lançadas ao longo do semestre**: valores atualizados continuamente pelos professores, permitindo o monitoramento em tempo real do desempenho do aluno.
- **Status da matrícula**: registros de trancamentos, tempo total no curso e períodos de afastamento, indicando possíveis padrões de retenção.

3.1.2 Indicadores Acadêmicos Disponíveis no SIGAA

Além dos registros de notas nas disciplinas e frequência, o SIGAA disponibiliza indicadores acadêmicos calculados automaticamente, os quais podem ser utilizados como insumos para análise de risco de evasão. Esses índices sintetizam o desempenho do aluno ao longo do curso e permitem avaliações mais diretas de sua progressão acadêmica.

Os principais índices disponíveis no sistema são:

- Média de Conclusão (MC): média ponderada das notas obtidas nos componentes curriculares.
- Média de Conclusão Normalizada (MCN): padronização da MC considerando a média e o desvio-padrão do curso.
- Índice de Eficiência em Carga Horária (IECH): relação entre a carga horária cursada e a carga horária utilizada.
- Índice de Eficiência em Períodos Letivos (IEPL): comparação entre carga horária acumulada e carga horária esperada.
- Índice de Eficiência Acadêmica (IEA): produto entre MC, IECH e IEPL.
- Coeficiente de Aproveitamento Acadêmico (CAA): média ponderada do rendimento escolar do aluno.

Esses índices podem auxiliar na definição de critérios para classificar alunos em diferentes perfis de risco. Por exemplo, a redução progressiva do CAA pode indicar queda de desempenho (evasão), enquanto um IEA abaixo da média do curso pode sugerir dificuldades na progressão acadêmica (retenção).

A escolha de utilizar os dados já estruturados e disponibilizados pelo SIGAA garante que o sistema proposto esteja alinhado à realidade da instituição e possa ser implementado sem necessidade de coleta de informações externas. Dessa forma, as análises e notificações geradas pelo sistema terão como base dados objetivos e atualizados regularmente.

3.1.3 Aplicação de Coleta de Dados

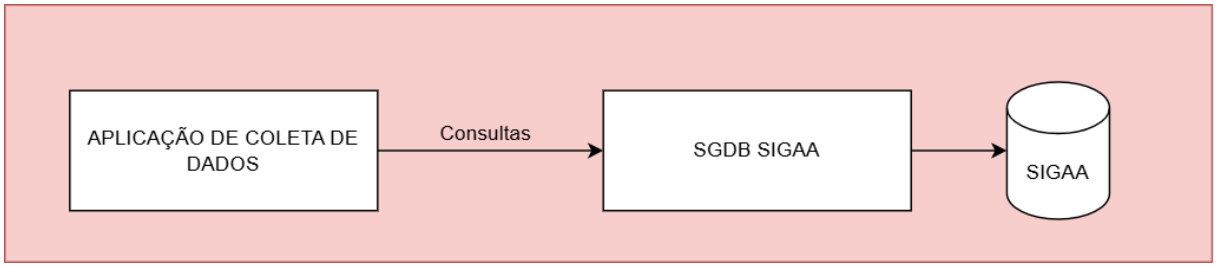
A aplicação de coleta de dados é um serviço responsável por acessar periodicamente o SIGAA para extrair as informações acadêmicas e armazená-las no repositório de dados brutos. A coleta pode ser feita de diferentes formas, dependendo da disponibilidade de acesso ao SIGAA:

- Integração via *Application Programming Interface* (API): Caso o SIGAA ofereça uma API, o sistema pode realizar consultas periódicas para obter os dados necessários.
- Acesso direto ao banco de dados: Em alguns casos, a extração pode ser feita diretamente do Sistema de Gerenciamento de Banco de Dados (SGBD), desde que as permissões adequadas sejam concedidas.

Neste caso, assume-se um acesso direto ao banco de dados por meio do SGBD, realizado por uma aplicação desenvolvida exclusivamente para essa finalidade. A aplicação pode ser executada diariamente, pois coletas frequentes distribuem a carga de processamento ao longo do tempo. Além disso, atualizações diárias minimizam o risco de inconsistências nos dados e aumentam a precisão das análises (RAMEZANI; IRANMANESH; NAEIM et al., 2025).

A Figura 9 ilustra um exemplo de como essa integração entre a aplicação de coleta e o SGBD poderia ser realizada.

Figura 9 – Integração da Aplicação de Coleta com o Banco de Dados SIGAA



Fonte: A própria autora.

3.1.4 Armazenamento de Dados Brutos

Após a extração das informações do SIGAA, os dados coletados são armazenados antes de serem processados e analisados. Nesta etapa, a validação imediata não é necessária, pois o foco é preservar os dados em seu formato original para futuras transformações. Essa abordagem é fundamental em sistemas que lidam com fluxos contínuos e grandes volumes de dados (RANI, 2025).

Dada a natureza variada das informações, opta-se pelo uso de bancos de dados NoSQL, que oferecem flexibilidade ao armazenar dados sem um formato predefinido. Isso os torna ideais para cenários onde os registros podem variar ou o esquema ainda não está definido (ORTIZ, 2024).

O uso de NoSQL se justifica por três fatores principais:

- **Flexibilidade:** Permite armazenar diferentes tipos de registros sem adaptações prévias, o que é essencial para sistemas distribuídos (RANI, 2025).
- **Escalabilidade**: Bancos como *MongoDB* e *DynamoDB* suportam o crescimento do volume de dados sem comprometer o desempenho. Isso é muito relevante para para aplicações que dependem da rápida ingestão e processamento de dados (KATSAROS, 2024).
- **Eficiência:** Como não há necessidade de normalização imediata, os registros podem ser armazenados rapidamente, garantindo disponibilidade para análise posterior (KATSAROS, 2024).

3.1.5 Estruturação e Armazenamento dos Dados

A aplicação de coleta de dados estabelece uma conexão direta com o SGBD do SIGAA para extrair as informações acadêmicas relevantes. Após a extração, os dados coletados são estruturados pela aplicação de coleta em objetos *JavaScript Object Notation* (JSON), um formato amplamente utilizado devido à sua flexibilidade e compatibilidade com diversos sistemas e linguagens de programação (ALMEIDA, 2024). Esse formato facilita tanto o armazenamento quanto a manipulação subsequente, permitindo uma organização clara das informações. Um exemplo de estrutura JSON para os dados coletados pode ser visto no código 3.1:

Código 3.1 – Estrutura JSON dos Dados Coletados

```json
{
  "aluno": {
    "matricula": "2025123456",
    "nome": "João da Silva"
  },
  "desempenho_academico": [
    {
      "disciplina": "Matemática",
      "nota": 8.5,
      "frequencia": 95
    },
    {
      "disciplina": "Física",
      "nota": 7.0,
      "frequencia": 88
    }
  ],
  "historico_academico": {
    "disciplinas_cursadas": 20,
    "disciplinas_aprovadas": 18,
    "disciplinas_reprovadas": 2,
    "carga_horaria_concluida": 1600,
    "carga_horaria_pendente": 400
  },
  "status_matricula": {
    "trancamentos": 1,
    "afastamentos": 0,
    "tempo_no_curso": "3 anos"
  },
  "indices_academicos": {
    "CAA": 7.8,
    "IEA": 0.85
  }
}
```

Após a extração das informações do SIGAA, os dados coletados são armazenados para posterior processamento e análise.

Para o armazenamento inicial dos dados, opta-se pelo uso de um banco de dados NoSQL orientado a documentos, como o *MongoDB*. Esse tipo de banco armazena nativamente os dados em JSON, proporcionando flexibilidade e escalabilidade para lidar com

diferentes tipos de informações acadêmicas extraídas do SIGAA. Diferente dos bancos de dados relacionais, que exigem esquemas rígidos, essa abordagem permite a ingestão eficiente de grandes volumes de dados sem necessidade de conversões complexas, reduzindo o tempo de processamento e melhorando o desempenho geral do sistema (OKO-ODION, 2024).

3.2 Processamento e Análise

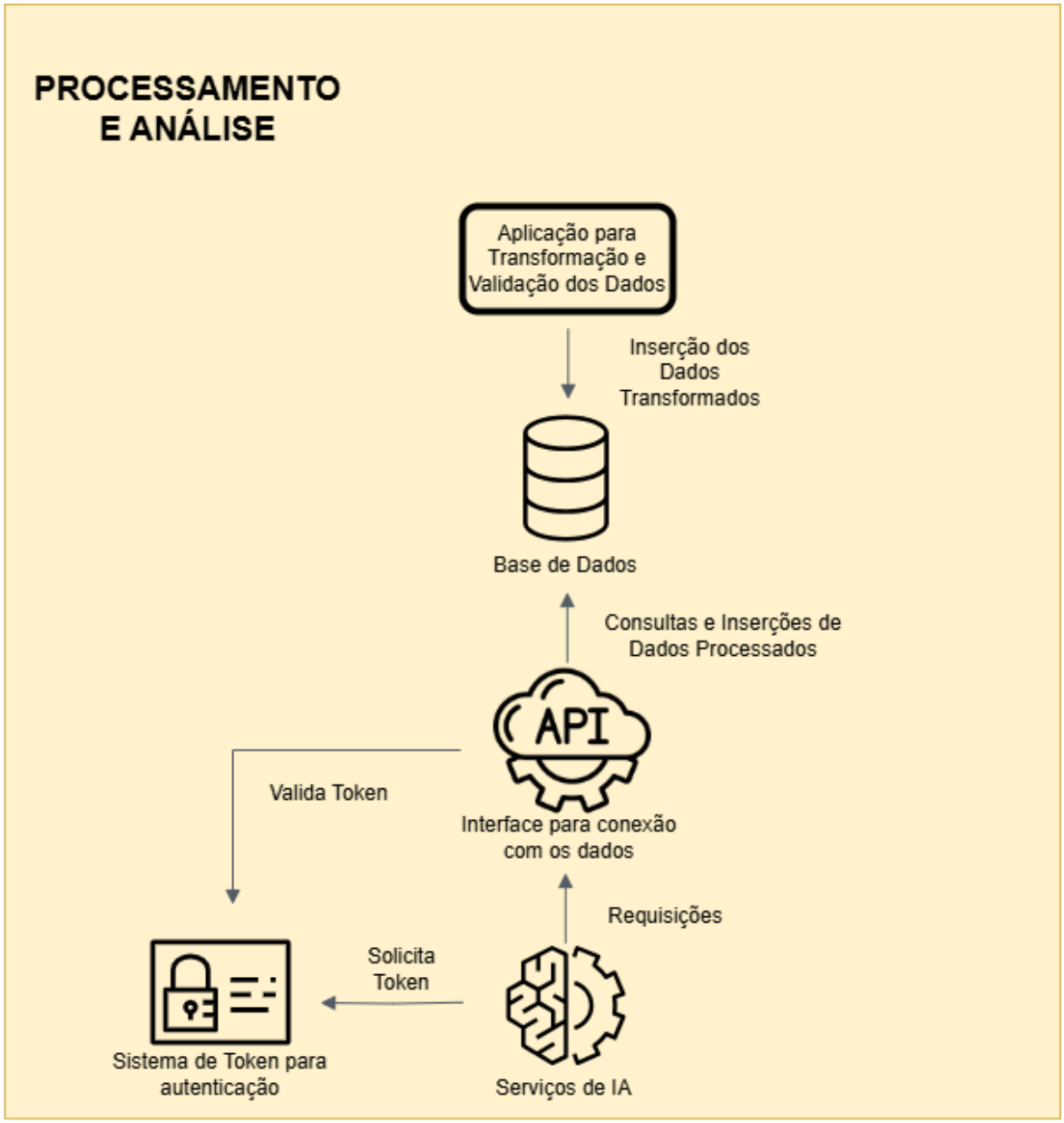
Após a coleta e o armazenamento dos dados brutos, descritos na Seção 3.1, é necessário processá-los para garantir sua integridade e disponibilidade para análises e aplicações futuras. Essa etapa inclui a transformação e validação dos registros, o armazenamento em um banco NoSQL e a disponibilização dos dados por meio de interfaces, permitindo seu processamento e análise por outras aplicações, como serviços de inteligência artificial (IA).

O Fluxo de Processamento e Análise, ilustrado na Figura 10, pode ser dividido em cinco módulos principais:

1. **Transformação e Validação dos Dados** – Os registros extraídos passam por um processo de limpeza e padronização, garantindo que estejam completos, consistentes e formatados corretamente antes do armazenamento.
2. **Armazenamento na Base de Dados** – Os dados transformados são armazenados em um banco de dados NoSQL orientado a chave-valor, permitindo buscas rápidas e suportando múltiplas requisições simultâneas.
3. **Autenticação via *Token*** – Um sistema de autenticação baseado em *tokens* garante que apenas aplicações autorizadas possam consultar ou inserir informações na base.
4. **Interface para Conexão com Sistemas de IA** – Permite que aplicações externas acessem os dados processados.
5. **Serviços de IA e Disponibilização dos Dados** – As informações são consumidas por serviços de IA, que realizam análises e disponibilizam os resultados para aplicações externas.

Nas próximas seções, cada um desses módulos será detalhado, explicando seu papel e como interagem dentro do sistema.

Figura 10 – Fluxo de Processamento e Análise de Dados



Fonte: A própria autora.

3.2.1 Transformação e Validação dos Dados

Os dados coletados do SIGAA podem conter informações inconsistentes, incompletas ou em formatos diferentes dos esperados. Para garantir que os registros estejam prontos para consulta e análise, esta etapa aplica um conjunto de regras de transformação e validação, incluindo:

- **Verificação de campos obrigatórios:** Garante que todas as informações essenciais estejam presentes.
- **Normalização de formatos:** Ajusta datas, códigos de disciplinas e nomes para um padrão unificado.
- **Tratamento de valores ausentes:** Estratégias como interpolação ou exclusão de registros inválidos são aplicadas conforme necessário.
- **Remoção de registros duplicados:** Evita a existência de múltiplas entradas para o mesmo dado.

Após a validação, os dados são inseridos na Base de Dados, garantindo que apenas informações corretamente formatadas e verificadas sejam armazenadas.

3.2.2 Armazenamento na Base de Dados

Os dados transformados são armazenados em um banco de dados NoSQL orientado a chave-valor, devido sua eficiência na busca, flexibilidade no armazenamento e escalabilidade (ABRAMOVA; BERNARDINO; FURTADO, 2014). Esse modelo permite consultas rápidas, um fator essencial para atender às demandas de serviços externos que acessam essas informações com frequência (BADGUJAR, 2023). Além disso, a estrutura do banco possibilita o armazenamento de registros no formato JSON, garantindo compatibilidade com diferentes aplicações e facilitando a integração com sistemas que consomem esses dados (GUNAWAN; DARMAWAN, 2021).

O papel deste banco de dados na arquitetura do sistema vai além do simples armazenamento dos registros transformados. Ele também persiste os resultados gerados por aplicações e serviços de inteligência artificial previstos na arquitetura, permitindo que informações processadas sejam reutilizadas em futuras consultas ou sirvam como base para novos processamentos de dados. Com isso, o banco se torna um componente central, assegurando a continuidade e eficiência no uso das informações disponíveis.

Outro aspecto relevante é que as inserções realizadas no banco não ocorrem de forma arbitrária, mas sim estruturada, seguindo um fluxo definido pelas aplicações de IA. Isso garante que os dados processados possam ser acessados por outros serviços e utilizados

como um repositório para futuras análises e decisões. Dessa forma, o banco de dados não apenas suporta consultas de informações validadas, mas também viabiliza a reutilização estratégica dos dados já analisados.

3.2.3 Sistema de Autenticação e Controle de Acesso

A segurança no acesso aos dados é um aspecto muito importante a se considerar, especialmente devido à natureza sensível das informações armazenadas. Como o objetivo fim do sistema proposto é fornecer um sistema capaz de auxiliar na redução da evasão e retenção acadêmica, a anonimização completa dos dados não é uma opção viável. Isso ocorre porque a proposta envolve a sinalização desses casos às partes interessadas, como coordenadores de curso, professores e equipes pedagógicas, que podem intervir para mitigar os problemas identificados.

Diante desse cenário, a implementação de um sistema de autenticação e autorização é fundamental para garantir um controle seguro sobre o acesso às informações. Esse mecanismo permite que apenas usuários autorizados consultem os dados, prevenindo acessos indevidos e protegendo a privacidade dos alunos.

A autorização pode ser realizada por um provedor de identidade (*Identity Provider* (IdP)) externo à solução proposta, encarregado de validar as credenciais dos usuários. Além disso, os controles de autorização, incluindo os mecanismos de *Access Control List* (ACL), podem ser obtidos a partir do SIGAA, garantindo a conformidade com as permissões institucionais existentes.

Embora a autenticação seja realizada por um IdP externo, a autorização – ou seja, a definição de quais dados cada usuário pode acessar – é gerenciada internamente pelo sistema. Dessa forma, é possível associar as credenciais autenticadas a um conjunto de permissões específicas, determinando, por exemplo, se um usuário pode apenas ler os dados ou também modificá-los. Além disso, pode haver granularidade no controle de acesso, restringindo as permissões com base em critérios como campus, nível de ensino e perfil do usuário.

Entre as abordagens possíveis para a transmissão segura das credenciais, pode-se utilizar *tokens* de acesso, como JWT, que encapsulam as permissões do usuário e são assinados digitalmente para garantir autenticidade. Esses *tokens* podem ser enviados junto às requisições aos microsserviços, permitindo que cada serviço valide o acesso de forma independente, sem necessidade de contato direto com o glsidp.

3.2.4 Autenticação Baseada em Tokens

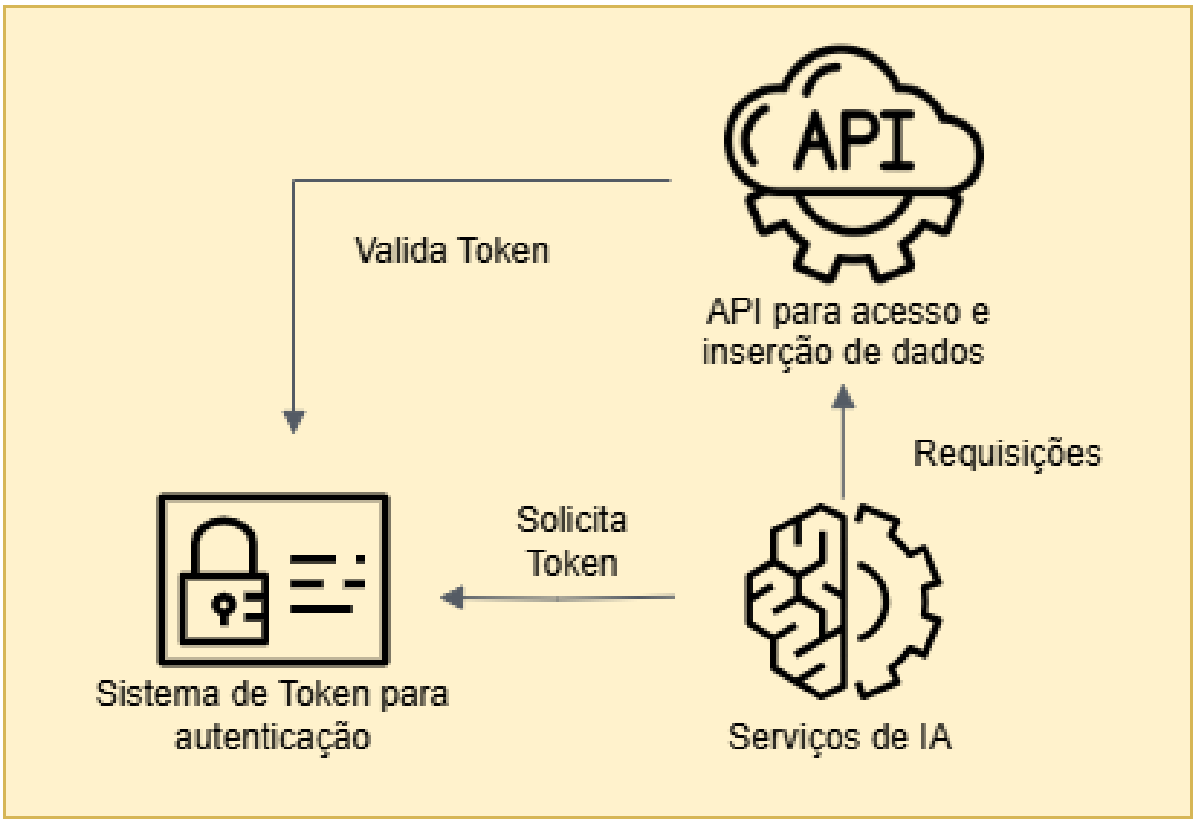
O sistema de autenticação via JWT é integrado ao fluxo de processamento e análise de dados conforme a Figura 11, garantindo que apenas serviços e usuários autorizados

possam acessar as informações.

O fluxo de autenticação ocorre da seguinte maneira:

1. O usuário ou serviço externo solicita um *token* ao sistema de autenticação, enviando suas credenciais.
2. O servidor valida as credenciais e, se forem corretas, retorna um JWT contendo as permissões do usuário.
3. O cliente inclui esse *token* em todas as requisições aos serviços protegidos.
4. Cada serviço valida o *token* antes de processar a solicitação, garantindo que o acesso seja autorizado.

Figura 11 – Fluxo de Autenticação e Validação de *Token*



Fonte: A própria autora.

Esse fluxo garante que todas as interações no sistema sejam seguras, rápidas e escaláveis, protegendo as informações acadêmicas contra acessos não autorizados.

3.2.5 Interface para Conexão com Sistemas de IA

A comunicação entre a base de dados processada e os serviços de inteligência artificial (IA) ocorre por meio de uma API RESTful, garantindo um acesso seguro e

estruturado aos dados. Essa interface é responsável por disponibilizar as informações acadêmicas validadas e permitir a persistência de novos dados analisados.

Os *endpoints* fornecidos possibilitam tanto a recuperação quanto a atualização das informações na base de dados. Dessa forma, os serviços de IA podem consumir os dados acadêmicos validados e, posteriormente, armazenar novos resultados, garantindo um fluxo contínuo de análise e atualização.

Para assegurar a segurança no acesso às informações, todas as requisições enviadas à API RESTful devem ser autenticadas via *tokens* de acesso, conforme definido na Seção 3.2.4. Cada requisição à interface deve conter um *token* válido, garantindo que apenas serviços autenticados possam consumir ou modificar os dados.

O exemplo do Código 3.2 demonstra uma requisição segura para a API RESTful, solicitando os indicadores acadêmicos de um aluno específico.

Código 3.2 – Exemplo de Requisição GET à API

```
GET /dados/aluno/2025123456 HTTP/1.1
Host: api.sistema.edu
Authorization: Bearer eyJhbGciOiJIUzI1...
```

Se a requisição for bem-sucedida, a API RESTful retorna os dados acadêmicos do aluno no formato JSON, conforme ilustrado no Código 3.3.

Código 3.3 – Exemplo de Resposta da API para Consulta de Dados

```
{
  "matricula": "2025123456",
  "nome": "João da Silva",
  "curso": "Técnico em Edificações",
  "historico_academico": {
    "disciplinas_cursadas": 20,
    "disciplinas_aprovadas": 18,
    "disciplinas_reprovadas": 2,
    "carga_horaria_concluida": 1600,
    "carga_horaria_pendente": 400
  },
  "status_matricula": "Ativo",
  "indices_academicos": {
    "CAA": 7.8,
    "IEA": 0.85
  }
}
```

Além do acesso às informações, a API RESTful permite que os serviços de IA enviem dados processados para armazenamento. O Código 3.4 apresenta um exemplo de requisição para inserir a classificação de um aluno em relação ao risco de evasão.

Código 3.4 – Exemplo de Requisição POST para Inserção de Dados

```
POST /dados/classificacao HTTP/1.1
Host: api.sistema.edu
Authorization: Bearer eyJhbGciOiJIUzI1...
Content-Type: application/json

{
  "matricula": "2025123456",
  "classificacao_risco": "Alto",
  "probabilidade": 0.85,
  "motivos": ["Baixo rendimento", "Frequência reduzida"]
}
```

Neste exemplo, um serviço de IA classifica um aluno como tendo um alto risco de evasão, registrando a informação no sistema. Essa capacidade de persistência garante que os resultados das análises sejam armazenados e disponibilizados para consultas futuras, permitindo que as partes interessadas possam agir de maneira proativa.

Por fim, a disponibilização dos dados por meio de uma interface via API RESTful visa proporcionar maior flexibilidade no acesso às informações, permitindo que os serviços de IA consultem a Base de Dados conforme a necessidade, de maneira dinâmica.

3.2.6 Mecanismo de Auditoria

Para melhorar a rastreabilidade e o controle dos acessos à API RESTful que faz a interface entre os serviços de IA e a Base de Dados, pode-se implementar um mecanismo de auditoria que registre as requisições feitas pelos serviços consumidores. Esse mecanismo permite monitorar o uso da API, identificar acessos indevidos e analisar padrões de consulta, contribuindo para a gestão da segurança.

Uma alternativa para viabilizar esse monitoramento sem adicionar uma nova camada de desenvolvimento é o uso de soluções já estabelecidas, como o *OpenTelemetry*. Essa ferramenta possibilita a coleta e exportação de métricas e *logs* de acesso, registrando informações relevantes sobre as requisições, como a aplicação requisitante, o usuário autenticado, a ação realizada e o status da operação.

O Código 3.5 apresenta um exemplo de estrutura de registro de auditoria, onde são armazenadas informações como *timestamp*, aplicação responsável pela requisição, usuário autenticado e o *endpoint* acessado.

Código 3.5 – Exemplo de Registro de Auditoria na API

```
{
  "timestamp": "2024-03-07T14:35:00Z",
  "aplicacao": "ServicoDeIA-01",
  "usuario": "coordenador_123",
  "acao": "GET /api/indicadores-academicos",
  "status": "Sucesso"
}
```

A implementação desse tipo de monitoramento possibilita um acompanhamento mais detalhado do uso dos dados, auxiliando na identificação de padrões de acesso e contribuindo para a segurança do sistema.

3.2.7 Serviços de IA

Os Serviços de Inteligência Artificial (IA) previstos no sistema possuem um papel fundamental na análise dos dados acadêmicos. No entanto, como o objetivo deste trabalho é a especificação de um sistema modular, não há uma funcionalidade única e rígida prevista para esses serviços. Em vez disso, o sistema foi projetado para permitir que múltiplos serviços de IA possam processar os dados disponíveis, aplicando diferentes abordagens analíticas para atender a necessidades específicas.

Dessa forma, cada serviço de IA pode ser especializado em analisar determinados aspectos dos dados acadêmicos, considerando diferentes fatores de risco e métricas. Entre as possíveis abordagens que podem ser adotadas, pode-se citar como exemplos:

- **Análise de vulnerabilidade socioeconômica:** Um serviço de IA pode avaliar o Índice de Vulnerabilidade Social (IVS) dos alunos e correlacioná-lo com taxas de evasão observadas em históricos anteriores.
- **Correlação entre desempenho acadêmico e plano de ensino:** Outra abordagem possível é a criação de um serviço de IA que avalia a evolução do aluno dentro de cada disciplina, comparando seu desempenho com os objetivos de aprendizado definidos pelos professores nos planos de ensino.
- **Detecção de padrões de retenção e atraso na conclusão do curso:** Um serviço pode analisar trajetórias acadêmicas para identificar padrões que indiquem retenção em disciplinas críticas ou um possível prolongamento excessivo do tempo de curso.

Cada um desses serviços pode operar de forma independente, consumindo os dados disponibilizados pelo sistema e gerando resultados específicos que serão utilizados para

aprimorar as estratégias de acompanhamento acadêmico.

Os resultados das análises realizadas pelos serviços de IA podem seguir dois caminhos principais:

- **Armazenamento na Base de Dados:** Os dados processados podem ser persistidos na base de dados do sistema, garantindo que estejam disponíveis para consultas futuras e aprimoramento contínuo das análises.
- **Publicação em um *Broker* de Mensagens:** Além do armazenamento, os dados analisados são encaminhados para um *broker* de mensagens, permitindo que outras aplicações consumam as informações processadas em tempo real. Dessa forma, os resultados podem ser disponibilizados para as partes interessadas, como coordenadores, professores e equipes pedagógicas.

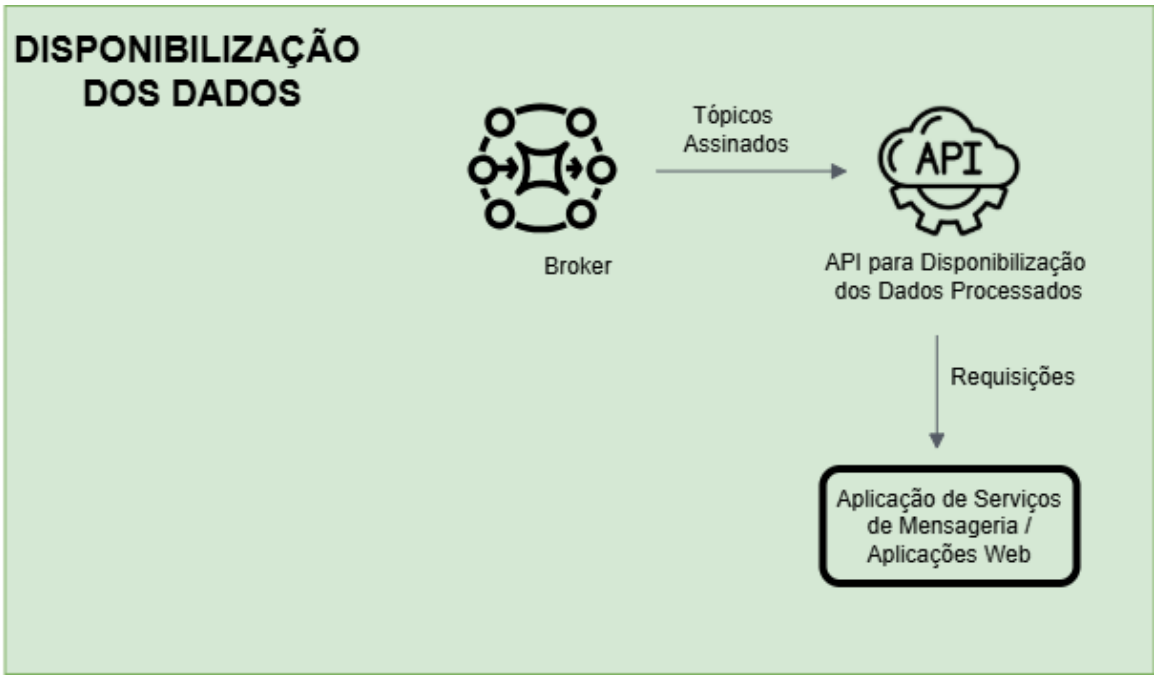
O mecanismo de disponibilização dos dados processados e sua distribuição para os interessados será detalhado na próxima seção.

3.3 Disponibilização dos Dados

A disponibilização dos dados processados ocorre por meio de um modelo de comunicação assíncrono baseado em publicadores e assinantes (pub/sub), utilizando um *broker* de mensagens. Nesse modelo, os serviços de IA atuam como produtores, publicando as informações analisadas nos tópicos do *broker*. Os consumidores podem se inscrever nesses tópicos para receber os dados em tempo real.

A Figura 12 ilustra esse fluxo, no qual o *broker* centraliza a comunicação entre os serviços produtores e consumidores de mensagens. A aplicação responsável pela disponibilização dos dados atua como um dos consumidores, assegurando que as informações possam ser acessadas por diferentes aplicações. Esse serviço pode ser estruturado como um *cluster* de microsserviços, implementado por meio de APIs, onde cada um se inscreve nos tópicos de seu interesse. Dessa forma, os dados são disponibilizados para aplicações web ou serviços de mensageria, permitindo a integração com diferentes interfaces e canais de comunicação.

Figura 12 – Fluxo de Disponibilização dos Dados



Fonte: A própria autora.

3.3.1 Modelo de Publicação e Assinatura

A comunicação entre os serviços de IA e as interfaces consumidoras de dados segue o modelo de publicação e assinatura (Pub/Sub) e pode ser implementada por meio de protocolos de troca de mensagens, como o Message Queuing Telemetry Transport (MQTT). Esse protocolo se destaca por sua eficiência em sistemas distribuídos, possibilitando a transmissão assíncrona de mensagens e assegurando a entrega dos dados a múltiplos consumidores simultaneamente (PETHIG, 2024).

Neste modelo, os serviços de IA atuam como produtores de mensagens, publicando os resultados das análises nos tópicos gerenciados pelo *broker*. As aplicações que precisam consumir essas informações atuam como assinantes, inscrevendo-se nos tópicos relevantes para receber notificações sempre que novas mensagens forem publicadas.

O *broker* centraliza esse fluxo de comunicação, garantindo que cada mensagem seja entregue corretamente aos consumidores interessados. Diferente de um modelo tradicional de comunicação baseada em requisições diretas, o modelo Pub/Sub elimina a necessidade de que os produtores conheçam diretamente os consumidores, proporcionando maior flexibilidade e escalabilidade ao sistema (JARVA, 2024).

3.3.2 Publicação de Mensagens no *Broker*

Cada serviço de IA, ao concluir uma análise, publica suas descobertas em um tópico específico no *broker*. As mensagens devem ser serializáveis, utilizando formatos como

JSON ou equivalentes, garantindo que qualquer aplicação inscrita possa interpretá-las corretamente.

O Código 3.6 apresenta um exemplo de publicação de uma mensagem contendo a classificação de risco de evasão de um aluno:

Código 3.6 – Exemplo de Publicação MQTT para Classificação de Risco

```
Tópico: sistema/analise/classificacao

{
  "matricula": "2025123456",
  "classificacao_risco": "Alto",
  "probabilidade": 0.85,
  "motivos": ["Baixo rendimento", "Frequência reduzida"]
}
```

Neste exemplo, um serviço de IA publica uma mensagem em determinado tópico, `sistema/analise/classificacao`, indicando que um aluno foi classificado com alto risco de evasão. Todas as aplicações inscritas nesse tópico receberão essa mensagem imediatamente.

3.3.3 Inscrição e Consumo de Mensagens

As aplicações interessadas nos dados publicados pelos serviços de IA precisam se inscrever nos tópicos relevantes (para a sua aplicação) do *broker* para receber as mensagens. Essas aplicações podem incluir sistemas de alerta para coordenadores acadêmicos, serviços de mensageria ou interfaces *web*.

A inscrição no tópico pode ser feita de forma contínua ou sob demanda, dependendo do comportamento esperado para cada serviço consumidor.

Assim que uma nova mensagem é publicada no tópico, todos os consumidores inscritos recebem os dados. Para garantir a entrega confiável, o sistema pode adotar um Quality of Service (QoS) 2, assegurando que cada mensagem seja entregue, pelo menos, uma vez a cada assinante, independentemente de instabilidades. Dessa forma, mesmo que um consumidor esteja temporariamente *offline*, as mensagens permanecem armazenadas no *broker* até que sejam consumidas com sucesso.

A partir desse ponto, cada aplicação pode processar a informação conforme sua necessidade, seja exibindo um alerta em uma interface *web*, enviando notificações via *e-mail* ou persistindo os dados em um banco de armazenamento local.

3.3.4 Entrega e Persistência dos Dados

Dependendo do tipo de consumidor, as mensagens podem ser tratadas de diferentes maneiras:

- **Consumidores que apenas repassam informações:** Alguns serviços podem atuar apenas como intermediários, recebendo as mensagens do *broker* e encaminhando-as diretamente para uma interface final.
- **Consumidores que armazenam dados para consulta posterior:** Em alguns casos, pode ser necessário armazenar temporariamente as informações antes de encaminhá-las. Isso pode ocorrer, por exemplo, quando os dados precisam ser cruzados com outras fontes ou quando o sistema permite consultas retrospectivas sobre alunos classificados com risco de evasão.

Essa abordagem garante flexibilidade na forma como os dados são tratados, permitindo que diferentes aplicações utilizem as mensagens publicadas de acordo com suas necessidades.

3.3.5 Encaminhamento das Mensagens para os Consumidores

Após serem publicadas no *broker*, as mensagens são encaminhadas para os serviços consumidores, que podem realizar diferentes ações dependendo do propósito da aplicação. Algumas possibilidades incluem:

- Notificações em tempo real para professores e coordenadores via serviços de mensageria como *Google Chat* e *e-mail*.
- Atualização de *dashboards* acadêmicos, permitindo a visualização centralizada dos dados processados.
- Ação automatizada em sistemas acadêmicos internos, podendo sugerir medidas preventivas para alunos em situação de risco.

3.3.6 Interfaces Consumidoras dos Dados

Os dados processados e publicados no *broker* precisam ser entregues aos usuários finais por meio de interfaces apropriadas. Esse processo não ocorre de forma direta, mas através de serviços consumidores que atuam como intermediários entre o *broker* e as aplicações que apresentarão os dados aos interessados.

Uma vez recebidas as mensagens, estas podem ser imediatamente encaminhadas para as interfaces finais ou passar por uma etapa intermediária de processamento. Dependendo da necessidade da aplicação, os dados podem ser enriquecidos com informações

adicionais, armazenados temporariamente em uma base local ou reformatados antes de serem disponibilizados.

A API, ou APIs, de disponibilização dos dados desempenha um papel central nesse processo, atuando como o ponto de entrada para diferentes tipos de interfaces consumidoras. Entre as principais formas de entrega da informação, destaca-se o uso de aplicações *web*, que permitem consultas detalhadas pelos coordenadores e equipes pedagógicas, e serviços de mensageria, que possibilitam notificações automáticas sobre eventos relevantes.

3.3.7 Exemplo de Apresentação dos Dados no Google Chat

Uma das formas de disponibilizar os dados analisados aos interessados é através de serviços de mensageria, como o *Google Chat*. Esse tipo de integração possibilita que coordenadores, professores e equipes pedagógicas recebam alertas automatizados sempre que um aluno for identificado com risco de evasão.

A seguir, o Código 3.7 apresenta um exemplo de *payload* estruturado para envio de notificações ao *Google Chat*. Essa mensagem contém informações sobre o aluno identificado, incluindo seu nome, curso e classificação de risco.

Código 3.7 – Exemplo de Notificação via Google Chat

```
POST https://chat.googleapis.com/v1/spaces/AAAAA/messages

{
  "cards": [
    {
      "header": {
        "title": "Alerta de Risco de Evasão",
        "subtitle": "Aluno identificado com alto risco",
        "imageUrl": "https://example.com/alert.png"
      },
      "sections": [
        {
          "widgets": [
            {
              "keyValue": {
                "topLabel": "Nome do Aluno",
                "content": "João da Silva"
              }
            },
            {
              "keyValue": {
                "topLabel": "Curso",
```

```
                "content": "Técnico em Edificações"
              }
            },
            {
              "keyValue": {
                "topLabel": "Classificação de Risco",
                "content": "Alto (85%)",
                "contentColor": "RED"
              }
            },
            {
              "buttons": [
                {
                  "textButton": {
                    "text": "Ver detalhes",
                    "onClick": {
                      "openLink": {
                        "url": "https://sistema.edu/aluno/2025123456"
                      }
                    }
                  }
                }
              ]
            }
          ]
        }
      ]
}
```

Neste exemplo, um serviço consumidor recebe a notificação de risco de evasão, estrutura a mensagem conforme o padrão aceito pelo *Google Chat* e a publica no canal apropriado. Esse processo garante que os responsáveis possam acessar as informações rapidamente e tomar as medidas necessárias para intervir na situação do aluno.

3.3.8 Personalização de Experiências dos Usuários (Chatbot)

Na camada de aplicação *web* ou serviços de mensageria, o tratamento dos dados pode ser personalizado de acordo com as necessidades dos usuários. Um exemplo dessa personalização é a utilização de *chatbots*, que permite uma interação mais dinâmica e

direcionada entre o sistema e os usuários interessados, como coordenadores de curso, professores e equipes pedagógicas.

Por meio de um *chatbot*, os usuários podem definir suas preferências em relação às notificações, selecionando turmas ou alunos específicos sobre os quais desejam receber alertas. Além disso, o *chatbot* pode oferecer funcionalidades interativas, permitindo que os usuários sinalizem determinadas situações que possam impactar a análise do sistema. Por exemplo, se um aluno for identificado como em risco de evasão devido à frequência insuficiente, mas estiver afastado por motivo de atestado médico, o usuário pode informar essa condição diretamente pelo *chatbot*, evitando notificações desnecessárias e refinando a qualidade das informações disponíveis para o acompanhamento acadêmico.

Essa abordagem melhora a experiência do usuário ao possibilitar uma comunicação mais eficiente e adaptável, além de reduzir a sobrecarga de informações irrelevantes. Através do *backend* do *chatbot*, as interações registradas podem ser processadas, garantindo que as notificações sejam ajustadas conforme o contexto específico de cada aluno ou curso.

4 CONCLUSÃO

Este trabalho apresentou a especificação de um sistema modular voltado à mitigação da evasão e retenção acadêmica. A proposta estrutura um modelo que parte da extração de dados acadêmicos, com base no SIGAA, e avança pelo processamento e disponibilização dessas informações para que possam ser consumidas por serviços de inteligência artificial. Esses serviços, por sua vez, permitem que os dados analisados sejam disponibilizados para aplicações voltadas às partes interessadas, como professores, coordenadores de curso e equipes pedagógicas, promovendo uma atuação mais proativa e estratégica.

A modularidade foi um dos pilares centrais da proposta, permitindo que diferentes serviços sejam integrados sem a necessidade de reestruturação da solução como um todo. Com uma arquitetura orientada a microsserviços, o sistema favorece a incorporação de novos componentes, como algoritmos de análise ou mecanismos de comunicação com os usuários, garantindo flexibilidade e adaptabilidade. Além disso, a adoção de um modelo de publicação e assinatura viabiliza a distribuição dos dados processados para aplicações externas, que podem apresentar essas informações de maneira personalizada e acessível.

Complementando os dados extraídos do SIGAA, a modularidade da arquitetura também possibilita a integração de outras fontes institucionais, como os dados de perfil de ingresso dos alunos, informações socioeconômicas ou registros de acompanhamento pedagógico. Ao combinar essas diferentes camadas de dados, o sistema amplia sua capacidade analítica, oferecendo uma compreensão mais abrangente dos fatores que impactam a permanência estudantil. Dessa forma, a proposta não se limita a uma base de dados específica, mas se estabelece como uma solução flexível e expansível, capaz de atender diferentes contextos institucionais e demandas de acompanhamento acadêmico.

Iniciativas semelhantes à proposta deste trabalho já vêm sendo adotadas por instituições de ensino, como é o caso do Sistema Integrado de Suporte ao Sucesso Acadêmico (SISSA), desenvolvido pela Universidade Federal de Goiás. Embora também tenha como objetivo a identificação de alunos em risco de evasão, o SISSA é uma aplicação centralizada voltada à análise preditiva, com foco na geração de relatórios a partir de dados institucionais. O diferencial da proposta apresentada neste trabalho está na concepção de uma arquitetura distribuída e modular, que possibilita maior flexibilidade e reuso de serviços. Essa abordagem permite a integração com diferentes fontes de dados e a personalização da entrega das informações, respeitando as especificidades de cada instituição e dos públicos envolvidos.

O principal desafio identificado na especificação do sistema reside na complexidade inerente à sua arquitetura distribuída. A integração entre os diferentes componentes –

desde a coleta dos dados até sua disponibilização para consumo – requer uma coordenação cuidadosa entre os serviços, assegurando que os dados sejam processados corretamente e estejam disponíveis para as aplicações que os utilizarão. No entanto, a utilização de padrões consolidados, como API RESTful para acesso aos dados e mecanismos de *token* para autenticação, contribui para a viabilidade da implementação e segurança das informações acadêmicas.

Além dos desafios técnicos, um aspecto crítico da proposta está na não anonimização dos dados acadêmicos, fator essencial para que a solução cumpra seu papel de fornecer informações relevantes às partes interessadas. Essa característica exige um rigoroso controle de acesso às informações sensíveis dos alunos, garantindo conformidade com a Lei Geral de Proteção de Dados (LGPD). Para mitigar riscos relacionados à privacidade e uso indevido das informações, recomenda-se a criação de comissões de ética e governança de dados, responsáveis por definir diretrizes institucionais para a coleta, processamento e compartilhamento das informações. Essas diretrizes devem alinhar-se aos princípios da LGPD, assegurando que a utilização dos dados acadêmicos esteja fundamentada em bases legais e respeite os direitos dos alunos.

Como trabalhos futuros, sugere-se a implementação prática do sistema proposto em um ambiente acadêmico real, a fim de validar sua efetividade na organização e disponibilização das informações acadêmicas. Além disso, recomenda-se a exploração de serviços já existentes no mercado para compor a solução, reduzindo a necessidade de desenvolvimento integral da aplicação e permitindo maior flexibilidade na sua composição. A integração com ferramentas institucionais, como acesso a outras bases de dados disponibilizadas pela Pró-Reitoria de Ensino (PROEN), pode ampliar o escopo do sistema, fornecendo informações estratégicas para a análise de evasão e retenção. Além disso, a incorporação de *feedbacks* diretos de alunos e professores pode agregar uma dimensão qualitativa às análises, tornando o monitoramento acadêmico mais abrangente.

Este estudo reforça a importância da tecnologia na gestão acadêmica, demonstrando que soluções baseadas em análise de dados podem fornecer suporte estruturado para a tomada de decisões. A modularidade da proposta assegura que futuras implementações possam evoluir conforme novas demandas institucionais surgirem, garantindo que a solução se mantenha relevante e adaptável ao contexto educacional.

REFERÊNCIAS

ABRAMOVA, Veronika; BERNARDINO, Jorge; FURTADO, Pedro. Experimental Evaluation of NoSQL Databases. **International Journal of Database Management Systems**, Academy e Industry Research Collaboration Center (AIRCC), v. 6, n. 3, p. 01–16, jun. 2014. ISSN 0975-5705. DOI: <10.5121/ijdms.2014.6301>. Disponível em: <<http://dx.doi.org/10.5121/ijdms.2014.6301>>. Citado na p. 50.

ALMEIDA, Vitor. **JSON e sua Relevância em Bancos de Dados NoSQL**. Acesso em: 16 fev. 2025. maio 2024. Disponível em: <<https://blog.grancursosonline.com.br/json-e-sua-relevancia-em-bancos-de-dados-nosql/>>. Citado na p. 46.

AMAZON WEB SERVICES. **Banco de Dados Relacional**. [S.l.: s.n.], 2023. Acesso em: 20 ago. 2024. Disponível em: <<https://aws.amazon.com/pt/relational-database/>>. Citado na p. 29.

_____. **Filas de mensagens**. [S.l.: s.n.], 2023. Acesso em: 29 de outubro de 2024. Disponível em: <<https://aws.amazon.com/pt/message-queue/>>. Citado na p. 37.

_____. **O que é um pipeline de dados?** [S.l.: s.n.], 2024. Acesso em: 1 set. 2024. Disponível em: <<https://aws.amazon.com/pt/what-is/data-pipeline/>>. Citado na p. 31.

AMAZON WEB SERVICES, INC. **The Difference Between ETL and ELT**. [S.l.: s.n.], 2024. Acesso em: 20 out. 2024. Disponível em: <<https://aws.amazon.com/pt/compare/the-difference-between-etl-and-elt/>>. Citado na p. 34.

BADGUJAR, P. Optimizing Data Storage and Retrieval in NoSQL Databases Strategies for Scalability. **Journal of Technological Innovations**, 2023. Disponível em: <<http://jtipublishing.com/jti/article/download/65/269>>. Citado na p. 50.

BRASIL, República Federativa do. **Constituição da República Federativa do Brasil de 1988**. [S.l.: s.n.], 1988. <https://www2.senado.leg.br/bdsf/bitstream/handle/id/518231/CF88_Livro_EC91_2016.pdf>. Acesso em: 25 jul. 2024. Citado na p. 19.

CALANCA, Paulo. **O que é um pipeline de dados?** [S.l.: s.n.], jul. 2023. Acesso em: 20 de outubro de 2024. Disponível em: <<https://www.alura.com.br/artigos/o-que-pipeline-dados?srsltid=AfmBOorCRt61MUXAJOEip1ZaS5Nx6v3ueHwMsu-3AzJyi64W6mPM8T_s>>. Citado nas pp. 33, 34.

COIMBRA, Camila Lima; SILVA, Leonardo Barbosa e; COSTA, Natália Cristina Dreossi. A evasão na educação superior: definições e trajetórias. **Educação e Pesquisa**, v. 47, 2021. ISSN 1678-4634. DOI: <10.1590/S1678-4634202147228764>. Disponível em: <<https://www.scielo.br/j/ep/a/WRKk9JVNBnJJsnNyNkFfJQj/>>. Citado na p. 43.

COLIN, S. ACID compliance in SQL databases: Best practices for developing robust applications. **Colin SQL Blog**, 2023. Acesso em: 20 ago. 2024. Disponível em: <<https://colinchsql.github.io/2023-10-31/15-08-57-054718-acid-compliance-in-sql-databases-best-practices-for-developing-robust-applications/>>. Citado na p. 30.

CONCEIÇÃO, Melissa Tidori da; PINTO, Giuliano Scombatti. Arquitetura de Microsserviços: Microservice Architecture. **Revista Interface Tecnológica**, Faculdade de Tecnologia de Taquaritinga, Taquaritinga, São Paulo, Brasil, p. 54–56, dez. 2021. DOI: <10.31510/infa.v18i2.1186>. Citado na p. 26.

CURSOS PM3. **Pipeline de dados: o que é, importância, tipos e componentes**. [S.l.: s.n.], 2024. Acesso em: 20 out. 2024. Disponível em: <<https://www.cursospm3.com.br/blog/pipeline-de-dados/#:~:text=Componentes%20de%20um%20data%20pipeline,Armazenamento%20de%20dados>>. Acesso em: 19 abr. 2024. Citado na p. 33.

DATABRICKS. **What is a Data Pipeline?** [S.l.: s.n.], 2024. Acesso em: 20 out. 2024. Disponível em: <<https://www.databricks.com/glossary/data-pipelines>>. Citado na p. 32.

DATACAMP. Batch vs Stream Processing: When to Use Each and Why It Matters. **DataCamp Blog**, 2023. Acesso em: 13 de outubro de 2024. Disponível em: <<https://www.datacamp.com/blog/batch-vs-stream-processing>>. Citado na p. 31.

DOMO. **The 5 Key Components of a Data Pipeline**. [S.l.: s.n.], 2024. Acesso em: 20 out. 2024. Disponível em: <<https://www.domo.com/learn/article/the-5-key-components-of-a-data-pipeline>>. Citado na p. 32.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados**. São Paulo: Pearson Addison Wesley, 2005. P. 4–5. Revisor técnico: Luis Ricardo de Figueiredo. Citado na p. 29.

FERNANDES, João Pedro Monteiro. **Estratégias para Coreografia de Microsserviços**. Toledo, PR, 2018. Citado na p. 38.

FINNSTATS. **Machine Learning Algorithms Top 5**. [S.l.]: Finnstats, 19 fev. 2022. Acesso em: 20 out. 2024. Disponível em: <<https://finnstats.com/2022/02/19/machine-learning-algorithms/>>. Citado na p. 35.

FUNDAÇÃO DE SOFTWARE APACHE. **Apache Kafka: Introduction**. [S.l.: s.n.], 2017. Acesso em: 28 de outubro de 2024. Disponível em: <<https://kafka.apache.org/intro>>. Citado na p. 37.

GOOGLE CLOUD. **O que é NoSQL?** [S.l.: s.n.], 2024. Acesso em: 20 ago. 2024. Disponível em: <<https://cloud.google.com/discover/what-is-nosql?hl=pt-BR>>. Citado na p. 30.

GOS, Konrad; ZABIEROWSKI, Wojciech. The Comparison of Microservice and Monolithic Architecture. In: IEEE. 2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH). Lodz University of Technology: [s.n.], 2020. DOI: <10.1109/MEMSTECH49584.2020.9109514>. Citado na p. 28.

GUNAWAN, Rohmat; DARMAWAN, I. Comparison of JSON and XML Data Formats in Document Stored NoSQL Database Replication Processes. **IAEI**, 2021. Disponível em: <<https://www.researchgate.net/profile/Rohmat-Gunawan/publication/354677706_Comparison_of_JSON_and_XML_Data_Formats_in_Document_Stored_NoSql_Database_Replication_Processes/links/6145b72d3c6cb3106977353e/Comparison-of-JSON-and-XML-Data-Formats-in-Document-Stored-NoSql-Database-Replication-Processes.pdf>>. Citado na p. 50.

HASSAN, S.; ALI, N.; BAHSOON, R. Microservice ambients: An architectural meta-modelling approach for microservice granularity. In: IEEE International Conference on Software Architecture (ICSA). [S.l.: s.n.], 2017. P. 1–10. Citado na p. 27.

HOFFER, Jeffrey A; PRESCOTT, Mary B; MCFADDEN, Fred R. **Modern database management**. [S.l.]: Pearson/Prentice Hall Upper Saddle River, NJ, 2007. Citado na p. 31.

IBM. **O que são microsserviços?** [S.l.: s.n.], 2024. <https://www.ibm.com/br-pt/topics/microservices>. Acesso em: 29 jul. 2024. Citado na p. 27.

_____. **Tipos de Chatbots**. [S.l.: s.n.], 2023. Acesso em: 29 de outubro de 2024. Disponível em: <<https://www.ibm.com/br-pt/think/topics/chatbot-types>>. Citado na p. 39.

_____. Your data ingestion strategy is a key factor in data quality. **IBM Blog**, 2023. Acesso em: 13 de outubro de 2024. Disponível em: <<https://www.ibm.com/blog/data-ingestion-strategy>>. Citado na p. 32.

ISLAM, R. et al. The Future of Cloud Computing: Benefits and Challenges. **International Journal of Communications, Network and System Sciences**, v. 16, p. 53–65, 2023. DOI: <10.4236/ijcns.2023.164004>. Citado na p. 23.

JARVA, Aleksi. **OPC UA PUBSUB in Industrial Systems Integration**. Jun. 2024. Master of Science Thesis – Tampere University. Disponível em: <<https://trepo.tuni.fi/bitstream/handle/10024/158659/JarvaAleksi.pdf?sequence=2>>. Citado na p. 58.

KATSAROS, I. Development of a microservices-based cloud platform for real-time satellite data capture and analysis. **Pergamos Library**, 2024. Disponível em: <<https://pergamos.lib.uoa.gr/uoa/dl/object/3420363/file.pdf>>. Citado na p. 46.

KAUFMAN, Lori M. Data Security in the World of Cloud Computing. **IEEE Security and Privacy Magazine**, v. 7, n. 4, p. 61–64, 2009. DOI: <10.1109/MSP.2009.87>. Disponível em: <<https://doi.ieeecomputersociety.org/10.1109/MSP.2009.87>>. Citado na p. 25.

KOVALOV, Y. Authentication and authorization in microservice oriented application design. **Modeling, Control and Information Technologies**, 2024. Citado na p. 27.

LAMARI, S. et al. Leveraging a Microservice Architecture, Access Control and Interoperability Patterns to Manage Privacy-Related User Consents. In: INTERNATIONAL Conference on Service-Oriented Computing. [S.l.]: Springer, 2024. Citado nas pp. 27, 28.

LENZ, Geanderson. **Uma introdução Didática aos Algoritmos de Classificação de Machine Learning**. [S.l.: s.n.], 2017. Acesso em: 20 out. 2024. Disponível em: <<https://medium.com/drafter-ai/uma-introdu%C3%A7%C3%A3o-did%C3%A1tica-aos-algoritmos-de-classifica%C3%A7%C3%A3o-de-machine-learning-460be2d73395>>. Citado na p. 35.

LIMA, Luis Octavio. Modelo Relacional: Conceitos sobre Restrições de Integridade. **Gran Cursos Online**, 2024. Acesso em: 20 ago. 2024. Disponível em: <<https://blog.grancursosonline.com.br/modelo-relacional-conceitos-sobre-restricoes-de-integridade/>>. Citado na p. 29.

LINHARES, Iohana. **Banco de Dados Não Relacionais**. [S.l.: s.n.], 2024. Acesso em: 20 ago. 2024. Disponível em: <<https://medium.com/@iohanalinhares1/banco-de-dados-n%C3%A3o-relacionais-7b79d074a942>>. Citado na p. 30.

LOURENÇO, J. et al. Choosing the right NoSQL database for the job: a quality attribute evaluation. **Journal of Big Data**, Springer Nature B.V., Heidelberg, v. 2, n. 1, p. 1–26, 2015. ISSN 21961115. Disponível em: <<http://search.proquest.com/docview/1987960462/>>. Citado na p. 30.

MACIEL, Victor Vieira et al. Uma Proposta de um Modelo de Sistema Baseado em Chatbot e Inteligência Computacional para Detecção de Crimes de Pedofilia. **Revista de Tecnologia da Informação da Faculdade Lourenço Filho**, Fortaleza, CE, v. 2, n. 2, p. 1, jan. 2021. Citado na p. 39.

MATEUS, Lucas. **Análise de Dados: Explorando Algoritmos de Classificação**. [S.l.]: DIO, 4 dez. 2023. Acesso em: 20 out. 2024. Disponível em: <<https://www.dio.me/articles/analise-de-dados-explorando-algoritmos-de-classificacao>>. Citado na p. 35.

MEC. **Documento Orientador para a Superação da Evasão e Retenção na Rede Federal de Educação Profissional, Científica e Tecnológica**. Brasília, DF: Ministério da Educação, 2014. Acesso em: 26 de outubro de 2024. Disponível em: <<https://avr.ifsp.edu.br/images/pdf/Comissoes_Outros/PermanenciaExito/Documento-Orientador-SETEC.pdf>>. Citado na p. 19.

MELO, Silvana Morita; PESSOA, Eloiziane Barbosa; PASCHOAL, Leo Natan. Uma análise sistemática sobre o uso de chatbots para ensino de computação no Brasil. **CINTED-UFRGS Revista Novas Tecnologias na Educação**, 2023. Acesso em: 29 de outubro de 2024. Disponível em: <<https://orcid.org/0000-0001-5934-2564>>. Citado na p. 39.

MICROSOFT AZURE. **O que é PaaS? Plataforma como serviço**. [S.l.: s.n.], 2024. Acessado em: 31 de outubro de 2024. Disponível em: <<https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-paas>>. Citado na p. 25.

MOSCHETTA, Vitor. **Compreendendo a Comunicação Síncrona e Assíncrona entre Microsserviços**. [S.l.: s.n.], jan. 2024. Acesso em: 28 de outubro de 2024. Disponível em: <<https://medium.com/@vitormoschetta/compreendendo-a-comunica%C3%A7%C3%A3o-s%C3%ADncrona-e-ass%C3%ADncrona-em-desenvolvimento-de-software-809d1f635d1b>>. Citado na p. 36.

NAEEM, Tehreem. Arquitetura de pipeline de dados: tudo o que você precisa saber. **Astera Software Blog**, jan. 2024. Acesso em: 20 out. 2024. Disponível em: <<https://www.astera.com/pt/type/blog/data-pipeline-architecture/>>. Citado na p. 32.

NEWMAN, S. **Building Microservices: Designing Fine-Grained Systems**. 2nd. [S.l.]: O'Reilly Media, 2019. cap. 7, 8. Discussão sobre autenticação centralizada em monólitos e estratégias distribuídas em microsserviços. ISBN 978-1492034025. Citado na p. 27.

NEWMAN, Sam. **Building Microservices: Designing Fine-Grained Systems**. [S.l.]: O'Reilly Media, Inc., 2015. Citado na p. 28.

OJHA, P. R. Securing the digital frontier: Best practices for RESTful API protection in modern web applications. **International Journal of Engineering and Technology Research**, 2024. Citado na p. 28.

OKO-ODION, C. Forecasting Techniques in Predictive Analytics: Leveraging Database Management for Scalability and Real-Time Insights. **ResearchGate**, 2024. Disponível em: <<https://www.researchgate.net/profile/Courage-Oko-Odion/publication/386575569_Forecasting_Techniques_in_Predictive_Analytics_Leveraging_Database_Management_for_Scalability_and_Real-Time_Insights/links/67578bf098f61916184917a7/Forecasting-Techniques-in-Predictive-Analytics-Leveraging-Database-Management-for-Scalability-and-Real-Time-Insights.pdf>>. Citado na p. 48.

OLEGHE, Omogbai; SALONITIS, Konstantinos. A framework for designing data pipelines for manufacturing systems. **Procedia CIRP**, Elsevier BV, v. 93, p. 724–729, 2020. DOI: <10.1016/j.procir.2020.04.016>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S221282712030401X>>. Citado na p. 31.

ORACLE. **O que é um Banco de Dados?** [S.l.: s.n.], 2024. Acesso em: 20 ago. 2024. Disponível em: <<https://www.oracle.com/br/database/what-is-database/>>. Citado na p. 29.

_____. **O que é um chatbot?** [S.l.: s.n.], 2024. Acesso em: 26 de outubro de 2024. Disponível em: <<https://www.oracle.com/br/chatbots/what-is-a-chatbot/>>. Citado na p. 39.

ORTIZ, Isabella. Integrating Advanced Data Handling Approaches in Modern Architectural Designs to Optimize Efficiency and Scalability. **ResearchGate**, 2024. Disponível em: <<https://www.researchgate.net/profile/Isabella-Ortiz-6/publication/386223137_Integrating_Advanced_Data_Handling_Approaches_in_Modern_Architectural_Designs_to_Optimize_Efficiency_and_Scalability_Journal_of_Sustainable_Technologies_and_Infrastructure_Planning/links/67495d443d17281c7de83f26/Integrating-Advanced-Data-Handling-Approaches-in-Modern-Architectural-Designs-to-Optimize-Efficiency-and-Scalability-Journal-of-Sustainable-Technologies-and-Infrastructure-Planning.pdf>>. Citado na p. 46.

PETHIG, Florian. Leveraging the Potential of MQTT in Industrie 4.0. **ResearchGate**, 2024. Disponível em: <<https://www.researchgate.net/profile/Florian-Pethig/publication/380094987_Leveraging_the_Potential_of_MQTT_in_Industrie_40/links/662a5d7d352430415348a989/Leveraging-the-Potential-of-MQTT-in-Industrie-40.pdf>>. Citado na p. 58.

QI NETWORK. **Pública, Privada, Comunidade ou Híbrida? Conheça os modelos de Cloud Computing**. [S.l.: s.n.], 2024. <https://blog.qinetwork.com.br/publica-privada-comunidade-ou-hibrida-conheca-os-modelos-de-cloud-computing/>. Acesso em: 28 jul. 2024. Citado na p. 24.

RAMBABU, Venkatesha Prabhu; ALTHATI, Chandrashekar; SELVARAJ, Amsa. ETL vs. ELT: Optimizing Data Integration for Retail and Insurance Analytics. **Journal of Computational Intelligence and Robotics**, The Science Brigade Publishing Group, v. 3, n. 1, p. 37–65, 2023. Disponível em: <<https://thesciencebrigade.com/jcir/?utm_source=ArticleHeader&utm_medium=PDF>>. Citado nas pp. 33, 34.

RAMEZANI, R.; IRANMANESH, S.; NAEIM, A. et al. Bench to Bedside: AI and Remote Patient Monitoring. **Frontiers in Digital Health**, 2025. DOI: <10.3389/fdgth.2025.1584443>. Disponível em: <<https://www.frontiersin.org/journals/digital-health/articles/10.3389/fdgth.2025.1584443/full>>. Citado na p. 45.

RANI, S. Tools and techniques for real-time data processing: A review. **International Journal of Science and Research Archive**, 2025. Disponível em: <<https://journalijsra.com/sites/default/files/fulltext_pdf/IJSRA-2025-0252.pdf>>. Citado na p. 46.

RANJAN, J. Business intelligence: Concepts, components, techniques and benefits. **Journal of theoretical and applied information technology**, v. 9, n. 1, p. 60–70, 2009. Citado na p. 31.

RASHID, Aaqib; CHATURVEDI, Amit. Cloud Computing Characteristics and Services: A Brief Review. **International Journal of Computer Sciences and Engineering**, v. 7, n. 2, p. 421–426, 2019. DOI: <10.26438/ijcse/v7i2.421426>. Citado nas pp. 24, 25.

RASTELLI, Felipe. **O que são ETL e ELT e quais suas diferenças**. [S.l.]: Coodesh, 2024. Acesso em: 20 out. 2024. Disponível em: <<https://coodesh.com/blog/candidates/o-que-sao-etl-e-elt-e-quais-suas-diferencas/>>. Acesso em: 19 out. 2024. Citado na p. 34.

REIS, Maria Fernanda Moge dos; NAKAOKA, Cassiano; NETO, Geraldo Henrique. Desenvolvimento de uma Pipeline de Dados Utilizando Soluções Open-Source em um Ambiente de Big Data. **Revista Eletrônica de Computação Aplicada (RECA)**, Uni-FACEF, v. 4, n. 1, p. 91–104, 2023. Disponível em: <<https://www.reca.com.br>>. Citado na p. 32.

SAJID, Mohammad; RAZA, Zahid. Cloud Computing: Issues & Challenges. **International Conference on Cloud, Big Data and Trust**, p. 35–41, 2013. Citado na p. 25.

SHADRACK, Bentil. **Monoliths vs Microservices: Breaking Down Software Architectures**. [S.l.: s.n.], 2023. Publicado por Documatic em 5 ago. 2023. Acesso em: 1 set. 2024. Disponível em: <<https://dev.to/documatic/monoliths-vs-microservices-

breaking-down-software-architectures%201keh#:~:
text=Bentil%20Shadrack%20for,de%202023%20%E2%80%A2>>. Citado na p. 26.

SILVA GARCIA, Léo Manoel Lopes da; LARA, Daiany Francisca; ANTUNES, Franciano. Investigação e Análise da Evasão e Seus Fatores Motivacionais no Ensino Superior: um estudo de caso na Universidade do Estado de Mato Grosso. **Avaliação**, v. 26, n. 1, p. 112–136, 2021. ISSN 1414-4077. DOI: <10.1590/S1414-40772021000100007>. Disponível em: <<https://www.scielo.br/j/aval/a/thxzBNWwkN5bHpSH7cFcmFg/>>. Citado na p. 44.

SUN MICROSYSTEMS, INC. **Introduction to Cloud Computing Architecture**. 1. ed. [S.l.], jun. 2009. Citado na p. 25.

THATIKONDA, Vamsi Krishna; MUDUNURI, Hemavantha Rajesh Varma. Microservices vs. Monoliths: Choosing the Right Architecture for Your Project. **International Journal of Software Computing and Testing**, JournalsPub, v. 10, n. 2, p. 31–38, 2024. ISSN 2456-2351. DOI: <10.37628/IJSCT>. Disponível em: <<https://journalspub.com/journal/ijsct>>. Citado na p. 26.

THOMAS, Dave. Cloud Computing – Benefits and Challenges! **Journal of Object Technology**, v. 8, n. 3, p. 37–41, 2009. Disponível em: <<http://www.jot.fm/issues/issue_2009_03/column4/>>. Citado na p. 25.

THORAT, Sandeep A.; JADHAV, Vishakha D. A Review on Implementation Issues of Rule-based Chatbot Systems. In: ICICC. PROCEEDINGS of the International Conference on Innovative Computing and Communication (ICICC 2020). Islampur, Sangli, Maharashtra, India: [s.n.], 2020. Citado na p. 39.

UFES. **Prograd desenvolve ações para evitar retenção e evasão dos estudantes**. Universidade Federal do Espírito Santo (UFES): [s.n.], 2018. Acesso em: 26 de outubro de 2024. Disponível em: <<https://ufes.br/conteudo/prograd-desenvolve-acoes-para-evitar-retencao-e-evasao-dos-estudantes>>. Citado na p. 19.

VALENTE, Marco Tulio. **Engenharia de Software Moderna**. [S.l.]: Independente, 2020. cap. 7, p. 7.5–7.6. Citado na p. 38.

VERAS, M. **Cloud Computing. Nova Arquitetura TI**. Rio de Janeiro: Editora Brasport, 2012. Disponível em: <<https://books.google.com.br/books?id=G4ZIDwAAQBAJ>>. Citado na p. 24.

VIGGIATO, Markos et al. Microservices in Practice: A Survey Study. Dept. de Ciências da Computação, Universidade Federal de Minas Gerais (UFMG), 2018. Citado na p. 26.

WORLD ECONOMIC FORUM. **Why Do We Call It the Cloud?** [S.l.: s.n.], 2015. <https://www.weforum.org/agenda/2015/03/why-do-we-call-it-the-cloud/>. Acesso em: 27 jul. 2024. Citado na p. 23.