Daniel da Silva Melo

Uma Abordagem para Indexação de Dados no Padrão FHIR em Banco de Dados NoSQL Colunares

Brasil

Março de 2025

Daniel da Silva Melo

Uma Abordagem para Indexação de Dados no Padrão FHIR em Banco de Dados NoSQL Colunares

Monografia

Universidade Federal de Itajubá – UNIFEI

Programa de Pós-Graduação em Ciência e Tecnologia Computação

Orientador: Adler Diniz de Souza

Coorientador: Laercio Augusto Baldochi Junior

Brasil

Março de 2025

"Porque eu sei os planos que tenho para vocês, diz o Senhor.
São planos de bem, e não de mal, para lhes dar o futuro pelo qual anseiam.
Naqueles dias, quando vocês clamarem por mim em oração, eu os ouvirei.
Se me buscarem de todo o coração, me encontrarão.
(Bíblia Sagrada, Jeremias 29:11-13)

Resumo

A escalabilidade do Apache Cassandra o torna uma opção atrativa para ecossistemas de saúde; no entanto, seu modelo de consulta restrito à chave primária impede a flexibilidade com as operações de busca definidas pelo padrão FHIR, como consultas por faixas de valores, buscas textuais com curingas e a combinação de múltiplos atributos. Para superar essa limitação fundamental, este trabalho apresenta e valida a CLiF-Arch (Cassandra Lucene-indexed FHIR Architecture), uma arquitetura de banco de dados que habilita consultas avançadas diretamente no Cassandra. A arquitetura implementa um índice secundário customizado que integra a biblioteca HAPI FHIR, para a análise semântica dos recursos, com o motor de busca Apache Lucene para a indexação dos parâmetros. Experimentos com 10 milhões de registros demonstraram que a CLiF-Arch suporta as buscas exigidas pelo FHIR diretamente via CQL. Essa flexibilidade, contudo, acarreta um custo de desempenho: o processo síncrono de análise e indexação elevou a latência média de inserção de aproximadamente 1 ms para a faixa de 3 a 4 ms. Os resultados demonstram que a CLiF-Arch é uma solução funcional que estabelece um compromisso explícito entre latência de escrita e flexibilidade de consulta, um requisito indispensável para a adoção do Cassandra em ambientes que necessitam de total conformidade com o padrão FHIR.

Palavras-chave: Bancos de dados distribuídos, motores de busca, indexação, sistemas de informação médica, prontuários eletrônicos, bibliotecas de software, FHIR.

Abstract

Apache Cassandra's scalability makes it an attractive option for healthcare ecosystems; however, its primary key-restricted query model hinders flexibility with FHIR-defined search operations, such as range queries, wildcard text searches, and the combination of multiple attributes. To overcome this fundamental limitation, this work presents and validates CLiF-Arch (Cassandra Lucene-indexed FHIR Architecture), a database architecture that enables advanced queries directly within Cassandra. The architecture implements a custom secondary index that integrates the HAPI FHIR library for semantic analysis of resources with the Apache Lucene search engine for parameter indexing. Experiments with 10 million records demonstrated that CLiF-Arch supports FHIR-required searches directly via CQL. This flexibility, however, comes at a performance cost: the synchronous analysis and indexing process increased the average insertion latency from approximately 1 ms to the 3-4 ms range. The results demonstrate that CLiF-Arch is a functional solution that establishes an explicit compromise between write latency and query flexibility—an indispensable requirement for adopting Cassandra in environments that demand full compliance with the FHIR standard.

Keywords: Distributed databases, Search engines, Indexing, Medical information systems, Electronic medical records, Software libraries, FHIR.

Lista de ilustrações

Figura 1 – Organização dos Recursos do FHIR 18
Figura 2 – Esquema de armazenamento Cassandra 20
Figura 3 – Diagrama de classes da integração entre o *plugin* Stratio e a biblioteca HAPI-FHIR 23
Figura 4 – Estrutura de indexação local e distribuída com o *plugin* Lucene no Cassandra 24
Figura 5 – Fluxo de Consulta e Recuperação de Dados no plugin Lucene 26
Figura 6 – Fluxo de compilação e implantação da arquitetura em um ambiente de cluster 28
Figura 7 – Diagrama de atividades do processo de *benchmark* 30
Figura 8 – Boxplot da latência média por operação para os cenários indexado e desnormalizado 32
Figura 9 – Gráfico de densidade da distribuição de latência por operação 34
Figura 10 – Boxplot comparando a latência do Lucene com e sem a integração do HAPI-FHIR 35

Lista de tabelas

Tabela 1 – Teste t pareado para latência com CLiF-Arch e com a abordagem desnormalizada 33
Tabela 2 – Teste T pareado para comparação entre Lucene com e sem HAPI-FHIR 36

Lista de abreviaturas e siglas

ACID	Atomicidade, Consistência, Isolamento e Durabilidade
ANS	Agência Nacional de Saúde Suplementar
API	Application Programming Interface (Interface de Programação de Aplicações)
BM25	Best Matching 25 (Algoritmo de ranqueamento de relevância em busca)
Bloom filter	Estrutura de dados probabilística para teste de pertencimento a um conjunto
CDSS	Clinical Decision Support System (Sistema de Apoio à Decisão Clínica)
CID	Classificação Internacional de Doenças
CMS	Centers for Medicare & Medicaid Services
CRUD	Create, Read, Update, Delete (Operações básicas de persistência de dados)
CT	Clinical Terms (Componente de Termos Clínicos do SNOMED)
EHR	Electronic Health Record (Registro Eletrônico de Saúde / Prontuário Eletrônico do Paciente)
ESD20-28	Estratégia de Saúde Digital para o Brasil 2020-2028
FHIR	Fast Healthcare Interoperability Resources
GDPR	General Data Protection Regulation (Regulamento Geral de Proteção de Dados)
HAPI-FHIR	HL7 API for FHIR (Biblioteca de código aberto para aplicações FHIR em Java)
HIE	Health Information Exchange (Intercâmbio de Informações de Saúde)
HIPAA	Health Insurance Portability and Accountability Act
HL7	Health Level Seven International
IoMT	Internet of Medical Things (Internet das Coisas Médicas)
IoT	Internet of Things (Internet das Coisas)

JSON	JavaScript Object Notation
LOINC	Logical Observation Identifiers Names and Codes
mHealth	Mobile Health (Saúde Móvel)
OHDSI	Observational Health Data Sciences and Informatics
PCHR	Personally Controlled Health Record (Registro de Saúde Pessoalmente Controlado)
PHR	Personal Health Record (Registro Pessoal de Saúde)
PROADI-SUS	Programa de Apoio ao Desenvolvimento Institucional do SUS
QUORUM	Nível de consistência em sistemas distribuídos como o Apache Cassandra
R	Linguagem de Programação R (Linguagem e ambiente para computação estatística)
RDF	Resource Description Framework
RIM	Reference Information Model (Modelo de Informação de Referência do HL7)
RNDS	Rede Nacional de Dados em Saúde
RWD	Real-World Data (Dados do Mundo Real)
RxNorm	Nomenclatura normalizada para medicamentos clínicos
SQL	Structured Query Language
SSD	Solid State Drive (Unidade de Estado Sólido)
SSTable	Sorted String Table (Tabela de Strings Ordenadas, estrutura de dados do Cassandra)
SUS	Sistema Único de Saúde
TF-IDF	Term Frequency-Inverse Document Frequency
TISS	Troca de Informação da Saúde Suplementar
UHIN	Utah Health Information Network
vCPU	Virtual Central Processing Unit (Unidade Central de Processamento Virtual)
vnode	Virtual Node (Nó Virtual no anel do Apache Cassandra)
XML	eXtensible Markup Language

Sumário

1 INTRODUÇÃO 10

2 MOTIVAÇÃO E PROBLEMA 12
2.1 Questões de Pesquisa e Hipóteses 13

3 REVISÃO BIBLIOGRÁFICA 15
3.1 O Padrão FHIR e Sua Adoção 16
3.2 Tecnologias de Banco de Dados para FHIR 18
3.3 Apache Cassandra e Apache Lucene 19

4 A ARQUITETURA CLIF-ARCH 22
4.1 Arquitetura e Componentes 22
4.2 Fluxos de Operação 25
4.2.1 Fluxo de Indexação 25
4.2.2 Fluxo de Consulta 25

5 METODOLOGIA EXPERIMENTAL 27
5.1 Delineamento Experimental 27
5.2 Carga de Trabalho e Coleta de Dados 29

6 ANÁLISE DOS RESULTADOS 31
6.1 Análise Comparativa de Desempenho 31
6.2 Análise Isolada dos Componentes 34

7 CONCLUSÃO 37

REFERÊNCIAS 39

1 INTRODUÇÃO

A gestão de grandes volumes de dados tornou-se um desafio central na área da saúde, impulsionada pela integração de fontes heterogêneas como dispositivos da Internet das Coisas Médicas (*Internet of Medical Things* – IoMT), registros eletrônicos de saúde (*Electronic Health Records* – EHRs) e sistemas clínicos (LEMUS-ZúñIGA et al., 2022; DRIDI et al., 2020; ABDELOUAHID et al., 2023). A carência de interoperabilidade entre estes sistemas, tanto dentro de uma mesma instituição quanto entre diferentes entidades de saúde, historicamente resulta em dificuldades na análise de prontuários e em custos financeiros adicionais, como a repetição de exames (ABDELOUAHID et al., 2023). Nesse cenário, a interoperabilidade semântica é um requisito fundamental, o que levou ao desenvolvimento de padrões como o *Fast Healthcare Interoperability Resources* (FHIR) pela *Health Level Seven International* (HL7) (AYAZ et al., 2021). A capacidade de gerenciar esses dados de forma escalável e tolerante a falhas é crítica para a tomada de decisões em tempo real, justificando a adoção de sistemas distribuídos e de padrões abertos como base tecnológica para ecossistemas de saúde, como a Rede Nacional de Dados em Saúde (RNDS) no Brasil (FAMÁ; FARIA; PORTUGAL, 2022; LAKSHMAN; MALIK, 2010).

Contudo, a implementação do padrão FHIR impõe desafios significativos à camada de persistência. Bancos de dados relacionais, embora consistentes, mostram-se limitados em escalabilidade horizontal para o volume, variedade e velocidade dos dados de saúde modernos (KLEPPMANN, 2017; KUMAR, 2024). Como alternativa, o Apache Cassandra[1] oferece uma solução escalável e de alta disponibilidade (LAKSHMAN; MALIK, 2010), mas sua arquitetura revela uma incompatibilidade fundamental: o modelo de consulta, otimizado para acessos por chave primária, é nativamente incapaz de suportar de forma flexível todas as operações de busca exigidas pelo FHIR, como buscas por faixa de valores (*range queries*), buscas textuais com curingas (*wildcard searches*) ou a combinação lógica de múltiplos atributos (CHEBOTKO; KASHLEV; LU, 2015). A abordagem convencional para contornar essa limitação, a desnormalização de dados, torna-se impraticável no contexto do FHIR devido ao vasto número de parâmetros de busca por recurso, o que resultaria em duplicação excessiva de dados e complexidade de manutenção insustentável (CHEBOTKO; KASHLEV; LU, 2015).

É neste cenário de incompatibilidade que este trabalho se insere, propondo e validando a CLiF-Arch (*Cassandra Lucene-indexed FHIR Architecture*), uma arquitetura de banco de dados projetada para habilitar a total conformidade com o padrão FHIR sobre um *cluster* Apache Cassandra. A arquitetura se baseia em um mecanismo de indexação

[1] Disponível em: <https://cassandra.apache.org/>

secundária customizado, que integra o motor de busca Apache Lucene, baseado no conceito de índice invertido (FAMÁ; FARIA; PORTUGAL, 2022), com a biblioteca HAPI-FHIR[2], uma consolidada implementação de código aberto para aplicações FHIR (SHOUMIK et al., 2017). Durante uma operação de escrita, um plugin (*Stratio's Cassandra Lucene Index*[3]) intercepta a transação, utiliza a HAPI-FHIR (em sua versão customizada) para analisar o conteúdo do recurso, extrai os parâmetros de busca relevantes e os indexa em um índice Lucene. O acesso aos dados é realizado por meio de consultas CQL padrão, que delegam a lógica da consulta ao motor Lucene, permitindo que o usuário execute buscas complexas diretamente no Cassandra.

A investigação da viabilidade e do desempenho desta arquitetura é orientada por duas questões de pesquisa centrais. A primeira, de validação funcional, busca determinar se a arquitetura CLiF-Arch é capaz de atender aos requisitos de consulta do FHIR que são nativamente insuportáveis pelo Cassandra. A segunda, de análise de desempenho, visa quantificar o impacto da arquitetura na latência das operações de escrita e de leitura em comparação com uma implementação Cassandra padrão.

Portanto, esta pesquisa visa demonstrar o potencial da CLiF-Arch como uma arquitetura de banco de dados viável para a adoção do Apache Cassandra em ecossistemas de saúde que demandam total conformidade com o padrão FHIR, estabelecendo um precedente para a integração de tecnologias complementares a fim de superar desafios específicos de armazenamento e consulta de dados.

O trabalho está organizado da seguinte forma: o Capítulo 2 aprofunda a motivação e o problema de pesquisa que guiam o estudo. O Capítulo 3 apresenta a fundamentação teórica sobre o padrão FHIR e as tecnologias de banco de dados envolvidas. O Capítulo 4 detalha a arquitetura CLiF-Arch, descrevendo sua implementação e fluxos de operação. O Capítulo 5 descreve a metodologia experimental, enquanto o Capítulo 6 apresenta e discute os resultados da avaliação de desempenho, validando as hipóteses de pesquisa. Finalmente, o Capítulo 7 sumariza as conclusões do trabalho, destaca suas principais contribuições e aponta direções para pesquisas futuras.

[2] Disponível em: <https://hapifhir.io/>
[3] Disponível em: <https://github.com/Stratio/cassandra-lucene-index>

2 MOTIVAÇÃO E PROBLEMA

A transição para a saúde digital impulsionou a geração de um volume de dados sem precedentes, exigindo infraestruturas tecnológicas que garantam não apenas o armazenamento, mas também a disponibilidade e a escalabilidade dos sistemas de informação (KUMAR, 2024). Em cenários de grande escala, como os da Rede Nacional de Dados em Saúde (RNDS) (BRASIL, 2024) no Brasil ou de grandes redes hospitalares, a capacidade de lidar com milhões de registros de pacientes, dados de sensores da *Internet of Medical Things* (IoMT) e transações clínicas em tempo real é um requisito indispensável (RUBí; GONDIM, 2019; BRAUNSTEIN, 2018). Nesse contexto, o Apache Cassandra emerge como uma tecnologia de banco de dados particularmente promissora. Sua arquitetura distribuída, sem ponto único de falha e otimizada para altas cargas de escrita, oferece a robustez e a escalabilidade horizontal necessárias para suportar ecossistemas de saúde modernos, que demandam resiliência e desempenho contínuo (LAKSHMAN; MALIK, 2010).

A escolha do Apache Cassandra é, portanto, motivada por suas vantagens arquitetônicas intrínsecas para lidar com *big data* (CHEBOTKO; KASHLEV; LU, 2015). Sistemas de saúde que operam em escala nacional ou que integram dados de milhares de dispositivos simultaneamente não podem depender de arquiteturas que exijam paradas para manutenção ou que apresentem gargalos em um servidor central. O modelo *peer-to-peer* do Cassandra, onde todos os nós são equivalentes, garante que a falha de um ou mais servidores não comprometa a disponibilidade do sistema como um todo (LAKSHMAN; MALIK, 2010; CARPENTER; HEWITT, 2020). Além disso, sua capacidade de escalar linearmente, simplesmente adicionando novos nós ao *cluster*, responde diretamente à necessidade de crescimento contínuo do volume de dados em saúde (LAKSHMAN; MALIK, 2010). Essas características o tornam ideal para armazenar dados de séries temporais, como os gerados por dispositivos de monitoramento de pacientes (WANG et al., 2020), e para suportar aplicações analíticas que processam grandes conjuntos de dados para extrair informações clínicas úteis (KUMAR, 2024).

Contudo, a adoção do Cassandra em conjunto com padrões de interoperabilidade como o FHIR revela um conflito fundamental. O padrão FHIR foi projetado com uma API de busca rica e flexível, permitindo que os sistemas recuperem dados com base em uma vasta gama de atributos clínicos e administrativos (AYAZ et al., 2021; BRAUNSTEIN, 2018). De fato, o FHIR define mais de 150 tipos de recursos (como Paciente, Observação, etc.), cada um com dezenas de parâmetros de busca, criando um universo de combinações de consulta que um sistema de persistência precisa suportar (AYAZ et al., 2021). É comum, por exemplo, que uma aplicação precise buscar todos os pacientes do sexo masculino, com

idade entre 40 e 50 anos, diagnosticados com uma determinada condição. Esse tipo de consulta, que combina múltiplos filtros e aplica condições de faixa de valores, é trivial em bancos de dados relacionais, mas representa um desafio significativo para o Cassandra (CARPENTER; HEWITT, 2020).

A performance do Cassandra está intrinsecamente ligada ao seu modelo de dados, que exige que as consultas especifiquem uma chave de partição para localizar os dados de forma eficiente (CHEBOTKO; KASHLEV; LU, 2015). Consultas que não atendem a esse requisito forçam uma varredura completa de todos os nós do *cluster*, um processo proibitivamente lento e que não escala, tornando-o inviável para aplicações em tempo real (CHEBOTKO; KASHLEV; LU, 2015; CARPENTER; HEWITT, 2020). Os índices secundários nativos do Cassandra, embora úteis para consultas simples em colunas de baixa cardinalidade, não são otimizados para as operações complexas, como buscas textuais com curingas (*wildcard searches*) ou a combinação de múltiplos atributos, que são essenciais para o FHIR. A solução padrão para este problema seria a desnormalização, ou seja, a criação de múltiplas tabelas, cada uma modelada para um padrão de consulta específico. No entanto, a quantidade de parâmetros de busca definidos pelo FHIR — o recurso *Patient* sozinho possui mais de 25 (AYAZ et al., 2021) — tornaria essa abordagem insustentável, levando a uma explosão no consumo de armazenamento e a uma complexidade de manutenção que anularia os benefícios de usar o Cassandra. Fica evidente, portanto, que existe uma lacuna tecnológica: há a necessidade de um mecanismo que possa conciliar a escalabilidade do Cassandra com a flexibilidade de consulta exigida pelo FHIR, sem recorrer a uma desnormalização impraticável.

2.1 Questões de Pesquisa e Hipóteses

Diante do problema exposto, a investigação deste trabalho é guiada por uma questão de pesquisa central, que busca explorar uma solução para a lacuna identificada:

> Motores de busca avançados podem flexibilizar o uso do Apache Cassandra para servidores FHIR?

Esta questão propõe que a integração de uma tecnologia especializada em indexação e busca, como o Apache Lucene, pode ser a chave para superar as limitações do Cassandra. Para validar esta proposição, a questão principal se desdobra em duas questões específicas, que abordam a funcionalidade e a viabilidade da abordagem. A primeira é uma questão de compatibilidade funcional: a arquitetura proposta é capaz de executar as consultas complexas definidas pelo padrão FHIR? Essa verificação é o pré-requisito para qualquer análise posterior. A segunda é uma questão de análise de desempenho: qual o custo, em termos de latência, de se introduzir essa camada adicional de indexação? A resposta a essa

questão é determinante para a viabilidade prática da solução em cenários de produção. Para o escopo deste trabalho, que utiliza um protótipo em ambiente local, estabeleceu-se um limiar de 50 ms para a latência, acima do qual o desempenho é considerado inadequado. Este valor foi determinado com base em medições de operações de linha de base realizadas durante a instanciação e testes iniciais do *cluster* Cassandra.

Para responder a essas questões de forma empírica e objetiva, foram formuladas as seguintes hipóteses de pesquisa. Para a questão de compatibilidade funcional, que será avaliada por meio de testes de passa/falha, as hipóteses são:

- **$\mathbf{H0_{func}}$:** A arquitetura CLiF-Arch, que integra o Apache Lucene, falha em executar um ou mais dos tipos de consulta complexos especificados pelo padrão FHIR, demonstrando ser funcionalmente incompatível.
- **$\mathbf{H1_{func}}$:** A arquitetura CLiF-Arch executa com sucesso as consultas complexas especificadas pelo padrão FHIR, demonstrando ser funcionalmente compatível.

Para a questão de análise de desempenho, que será avaliada por meio de testes de significância estatística, as hipóteses são:

- **$\mathbf{H0_{write}}$:** A latência média das operações de escrita (criação, atualização, exclusão) na arquitetura CLiF-Arch não é estatisticamente superior à latência de um modelo de dados Cassandra nativo e desnormalizado.
- **$\mathbf{H1_{write}}$:** A latência média das operações de escrita na arquitetura CLiF-Arch é estatisticamente superior à latência de um modelo de dados Cassandra nativo e desnormalizado.
- **$\mathbf{H0_{read}}$:** A latência média das operações de leitura complexa, habilitadas pela CLiF-Arch, não é estatisticamente diferente da latência de operações de leitura simples (por chave primária) em um modelo Cassandra nativo.
- **$\mathbf{H1_{read}}$:** A latência média das operações de leitura complexa, habilitadas pela CLiF-Arch, é estatisticamente diferente da latência de operações de leitura simples em um modelo Cassandra nativo.

A validação ou refutação dessas hipóteses através da experimentação, detalhada nos capítulos subsequentes, fornecerá as evidências necessárias para determinar se a integração de um motor de busca avançado constitui, de fato, uma solução viável e eficaz para flexibilizar o uso do Apache Cassandra em servidores FHIR, respondendo assim à questão de pesquisa principal deste trabalho.

3 REVISÃO BIBLIOGRÁFICA

A informática em saúde tem evoluído ao longo de décadas, impulsionada pela necessidade de melhorar a prestação de cuidados por meio da adoção de tecnologias digitais (BRAUNSTEIN, 2018). Os primeiros registros eletrônicos de saúde (EHR, do inglês *Electronic Health Records*) e os sistemas de apoio à decisão clínica (CDSS) foram concebidos para agilizar o gerenciamento de dados e auxiliar os médicos na tomada de decisões (ABDELOUAHID et al., 2023). Contudo, a implementação inicial dessas tecnologias enfrentou desafios significativos, como a falta de protocolos padronizados e uma interoperabilidade limitada entre sistemas distintos (DRIDI et al., 2020). Essa carência de interoperabilidade, tanto intra quanto inter-entidades, historicamente gerou problemas na análise de prontuários e custos financeiros adicionais, como a repetição de exames (ABDELOUAHID et al., 2023). Apesar dessas dificuldades, essas inovações estabeleceram as bases para o desenvolvimento da tecnologia em saúde como a conhecemos hoje.

Com o tempo, a necessidade crescente de compartilhar dados entre diferentes prestadores e organizações levou ao surgimento das iniciativas de intercâmbio de informações de saúde (HIE, do inglês *Health Information Exchange*). Essas iniciativas visavam reduzir redundâncias nos cuidados, melhorar a segurança dos pacientes e possibilitar melhores desfechos clínicos, permitindo o acesso a dados clínicos independentemente do sistema de origem (LEMUS-ZúñIGA et al., 2022; RUBí; GONDIM, 2019). No entanto, a ausência de normas técnicas consistentes e a complexidade dos fluxos de trabalho nos serviços de saúde continuaram sendo barreiras significativas para uma troca de dados verdadeiramente contínua e eficiente (MOREIRA et al., 2020). A interoperabilidade emergiu, assim, como um desafio central na aplicação prática da informática em saúde, uma vez que a integração de dados de fontes diversas era dificultada pela falta de protocolos padronizados (FAMÁ; FARIA; PORTUGAL, 2022).

Esforços para superar essas limitações resultaram no desenvolvimento do padrão HL7 FHIR, que fornece uma estrutura modular e baseada em APIs para a troca de dados, mostrando-se particularmente eficaz para lidar com as limitações de sistemas legados (BENDER; SARTIPI, 2013; BRAUNSTEIN, 2018). O FHIR foi concebido para superar as dificuldades de padrões anteriores, como a variabilidade de implementação do HL7 v2 e a complexidade do HL7 v3 (AYAZ et al., 2021). A interoperabilidade semântica, que assegura que os dados compartilhados sejam não apenas transmitidos, mas também compreendidos uniformemente, tornou-se um elemento essencial para o sucesso das iniciativas de HIE (MOREIRA et al., 2020; DRIDI et al., 2020). O padrão FHIR representa um avanço significativo nessa direção, organizando informações em recursos modulares que

representam conceitos clínicos essenciais, como pacientes, consultas e medicamentos (AYAZ et al., 2021). Essa modularidade permite que desenvolvedores acessem e compartilhem informações específicas sem exigir a integração total do sistema (SREEJITH; SENTHIL, 2023). Além disso, o FHIR foi projetado para funcionar com padrões de codificação existentes, como CID, SNOMED CT e LOINC, garantindo compatibilidade e simplificando sua implementação (BRAUNSTEIN, 2018; WALINJKAR, 2018; ABDELOUAHID et al., 2023).

Estudos práticos demonstram o impacto positivo das iniciativas de HIE na prestação de serviços de saúde. A Rede de Informação de Saúde de Utah (UHIN), por exemplo, mostrou como os HIEs regionais podem reduzir duplicações e melhorar a coordenação do cuidado, enquanto a *CommonWell Health Alliance* utiliza o FHIR para promover o compartilhamento nacional de dados entre diferentes prestadores (BRAUNSTEIN, 2018). Paralelamente, o crescimento do volume e da complexidade das informações de saúde tornou a privacidade e a segurança preocupações ainda mais críticas (FAMÁ; FARIA; PORTUGAL, 2022). Regulamentações como a HIPAA buscam garantir que os dados dos pacientes sejam protegidos, embora o equilíbrio entre privacidade e acesso eficiente aos dados permaneça um desafio constante (YU et al., 2021). Tecnologias emergentes, como o *blockchain* assistido por contratos inteligentes, também têm sido exploradas para aumentar a confiança e a segurança no HIE, oferecendo um registro de transações descentralizado e imutável que promete melhorar a confiabilidade e a autenticação nos sistemas de saúde (SREEJITH; SENTHIL, 2023).

3.1 O Padrão FHIR e Sua Adoção

Antes do FHIR, padrões como HL7 V2 e V3 moldaram o cenário da interoperabilidade em saúde. Embora eficazes em seus contextos, esses padrões enfrentaram limitações de complexidade técnica e flexibilidade. O HL7 V3, por exemplo, introduziu uma abordagem estruturada por meio do Modelo de Informação de Referência (RIM), mas sua alta complexidade dificultou a implementação prática (YU et al., 2021; AYAZ et al., 2021; BRAUNSTEIN, 2018). Essas dificuldades evidenciaram a necessidade de soluções mais intuitivas, abrindo caminho para o FHIR, que surgiu como uma resposta direta às demandas contemporâneas por soluções em tempo real, escaláveis e de fácil desenvolvimento (ZAMPOGNARO; PARAGLIOLA; FALANGA, 2021; BENDER; SARTIPI, 2013). A filosofia central do FHIR baseia-se na simplicidade e modularidade, organizando os dados em recursos que representam conceitos clínicos ou administrativos, como Paciente, Observação e Medicação, permitindo sua combinação para atender a diferentes casos de uso (MOREIRA et al., 2020; AYAZ et al., 2021; WALINJKAR, 2018).

Para o desenvolvimento deste trabalho, outros padrões robustos como OpenEHR e

ISO 13606 foram considerados. O OpenEHR oferece uma especificação abrangente para a construção de prontuários eletrônicos flexíveis, enquanto o padrão ISO 13606 foi projetado para a troca de extratos de prontuários com significado completo, tornando-o ideal para projetos complexos de interoperabilidade (PEDRERA-JIMéNEZ et al., 2023). A escolha pelo HL7 FHIR, no entanto, foi uma decisão pragmática baseada em seu alinhamento com os objetivos deste projeto. O FHIR utiliza padrões web modernos, como APIs RESTful, para reduzir a complexidade da implementação sem sacrificar a integridade da informação, o que facilita o desenvolvimento e a integração rápidos, sendo a abordagem mais adequada para os requisitos deste estudo (BENDER; SARTIPI, 2013; PEDRERA-JIMéNEZ et al., 2023; AYAZ et al., 2021).

Cada recurso no FHIR é descrito por atributos e metadados, podendo se referir a outros recursos para criar um modelo rico de dados clínicos, representados em formatos como JSON, XML ou RDF (RUBí; GONDIM, 2019; HONG; MORRIS; SEO, 2017; WALINJKAR, 2018). A organização modular do padrão é ilustrada em um *framework* hierárquico, como o apresentado na Figura 1, que facilita a navegação e a governança dos recursos, divididos em camadas que vão desde a fundação técnica até a contextualização para casos de uso específicos (BRAUNSTEIN, 2018). Essa flexibilidade é ampliada pelo mecanismo de *profiling*, que permite restringir ou estender os recursos base para atender a necessidades específicas de um país ou caso de uso, como adicionar um campo para a certidão de nascimento de um paciente ou exigir um identificador nacional (POINTNER; PRASCHL; KRAUSS, 2023; PEDRERA-JIMéNEZ et al., 2023). Essa abordagem agnóstica ao esquema possibilita a coexistência de múltiplos padrões arquiteturais dentro de uma única solução (ABDELOUAHID et al., 2023; DRIDI et al., 2020).

A aplicação do FHIR tem impulsionado inovações em diversas frentes. Plataformas de saúde pessoal (PHR) utilizam o FHIR para criar ecossistemas interconectados, permitindo que pacientes gerenciem seus dados de saúde provenientes de hospitais e dispositivos de IoT em um repositório centralizado (HONG; MORRIS; SEO, 2017). Um exemplo notável é o projeto da *Georgia Tech*, que utilizou APIs baseadas no padrão para agregar dados de múltiplos sistemas de EHR de forma coesa, permitindo a aplicação de modelos analíticos para identificar condições críticas em pacientes (BRAUNSTEIN, 2018). O padrão também desempenha um papel central na evolução dos registros de saúde pessoais (PHRs), promovendo maior transparência e engajamento dos pacientes. Iniciativas como a API *Blue Button* 2.0 nos Estados Unidos exemplificam essa abordagem, garantindo interoperabilidade no compartilhamento de dados entre pacientes e aplicativos (BRAUNSTEIN, 2018). No Brasil, a Agência Nacional de Saúde Suplementar (ANS) desenvolve um projeto, no âmbito do PROADI-SUS, que adapta o padrão de Troca de Informação da Saúde Suplementar (TISS) para o FHIR, visando integrar informações e apoiar a gestão do Sistema Único de Saúde (SUS) (BRASIL, 2024).

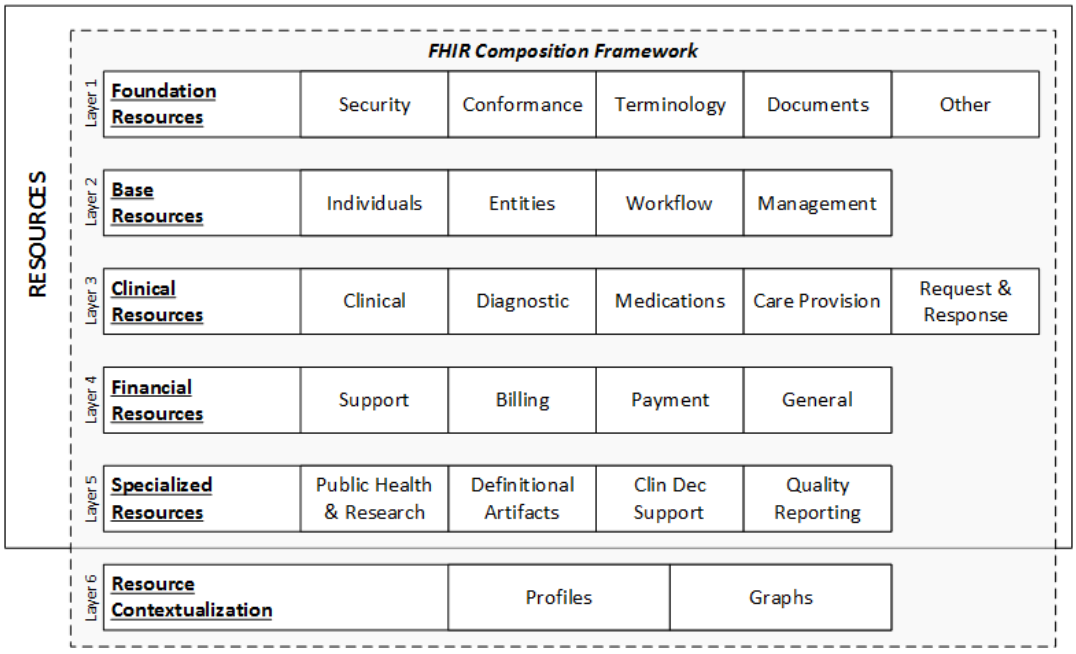

Figura 1 – Organização dos Recursos do FHIR

Tecnologias emergentes também se beneficiam da integração com o FHIR. Dispositivos vestíveis e sistemas da *Internet of Medical Things* (IoMT) utilizam o padrão para harmonizar fluxos de dados em tempo real (FAMÁ; FARIA; PORTUGAL, 2022; LEMUS-ZúñIGA et al., 2022), enquanto as tecnologias de saúde móvel (mHealth) redefinem os cuidados centrados no paciente (YU et al., 2021; WALINJKAR, 2018). No contexto da saúde pública, o FHIR facilita a coleta de dados para monitoramento epidemiológico e análises sobre determinantes sociais da saúde (ABDELOUAHID et al., 2023). Por fim, o padrão também transforma os sistemas de apoio à decisão clínica (CDS), simplificando sua integração com EHRs para fornecer informações úteis no ponto de atendimento (SREEJITH; SENTHIL, 2023). A adaptabilidade do FHIR continua a ser um elemento essencial na transição para um modelo de saúde mais colaborativo e centrado no paciente (AYAZ et al., 2021; RUBí; GONDIM, 2019).

3.2 Tecnologias de Banco de Dados para FHIR

A implementação do padrão FHIR em sistemas de grande escala exige uma escolha cuidadosa da tecnologia de banco de dados. A literatura científica revela um cenário onde tanto bancos de dados relacionais quanto *NoSQL* são utilizados, cada um com suas vantagens e desvantagens. Sistemas de gerenciamento de bancos de dados relacionais (SGBDRs) como PostgreSQL, Oracle e MySQL continuam a ser uma escolha convencional e bem suportada, especialmente em implementações de servidores que utilizam o HAPI

FHIR (KIRILOV, 2024; KIRILOV, 2025; SHOUMIK et al., 2017). Embora ofereçam consistência e integridade transacional, os SGBDRs enfrentam desafios de escalabilidade e flexibilidade para lidar com a heterogeneidade dos dados de saúde modernos, como notas clínicas não estruturadas e dados genômicos (KUMAR, 2024).

Com o crescimento do volume e da variedade dos dados, os bancos de dados *NoSQL* surgiram como uma alternativa flexível e escalável (KUMAR, 2024). Dentro do ecossistema *NoSQL*, os armazenamentos de documentos como o MongoDB são a categoria mais prevalente para implementações FHIR, devido à sua capacidade de armazenar recursos diretamente em formato JSON sem um esquema predefinido (SREEJITH; SENTHIL, 2023; KUMAR, 2024; SHOUMIK et al., 2017). Para casos de uso que envolvem dados altamente interconectados, os bancos de dados em grafos como o Neo4j são eficazes para gerenciar os múltiplos relacionamentos entre entidades clínicas, como o fluxo de pacientes e a transmissão de doenças (POINTNER; PRASCHL; KRAUSS, 2023; KUMAR, 2024). Já os armazenamentos de chave-valor, como Redis e DynamoDB, são ideais para processos em tempo real que exigem recuperação de dados com baixa latência, como em sistemas de monitoramento de pacientes (KUMAR, 2024).

Embora menos comuns que os bancos de dados de documentos ou grafos, os armazenamentos de famílias de colunas como o Apache Cassandra possuem um papel importante e bem documentado na literatura, especialmente para o gerenciamento e análise de grandes volumes de dados de saúde (KUMAR, 2024). A arquitetura do Cassandra é destacada por sua adequação para análises em tempo real em cenários de *big data*, com escalabilidade horizontal entre múltiplos centros de dados (KUMAR, 2024; CHEBOTKO; KASHLEV; LU, 2015). Sua aplicação é particularmente relevante no domínio da análise de saúde, onde pode suportar modelos preditivos e avaliação da saúde da população. No entanto, seu modelo de consistência eventual pode ser uma desvantagem em ambientes clínicos que exigem precisão imediata dos dados (KUMAR, 2024). A aplicabilidade do Cassandra para análises em larga escala fornece uma forte justificativa para sua seleção neste estudo, já que a flexibilização de seu modelo direcionado à consulta, que está sendo proposto pela arquitetura Clif-Arch, pode facilitar sua implementação e aumentar o escopo de usos possíveis.

3.3 Apache Cassandra e Apache Lucene

O Apache Cassandra é um sistema de banco de dados distribuído projetado para arquiteturas descentralizadas, o que lhe confere alta disponibilidade e tolerância a falhas (LAKSHMAN; MALIK, 2010). Sua abordagem *peer-to-peer*, onde todos os nós são equivalentes, o torna ideal para lidar com grandes volumes de dados em *clusters* sem pontos únicos de falha (LAKSHMAN; MALIK, 2010; CARPENTER; HEWITT, 2020). Os

dados são organizados em *Keyspaces*, que contêm tabelas e definem o fator de replicação, essencial para a resiliência do sistema. As tabelas, por sua vez, são divididas em partições e distribuídas entre os nós com base em uma chave de partição, conforme ilustrado na Figura 2. A chave primária pode incluir colunas de agrupamento (*clustering*) que ordenam os dados dentro de cada partição, otimizando a recuperação de dados ordenados (CHEBOTKO; KASHLEV; LU, 2015; CARPENTER; HEWITT, 2020).

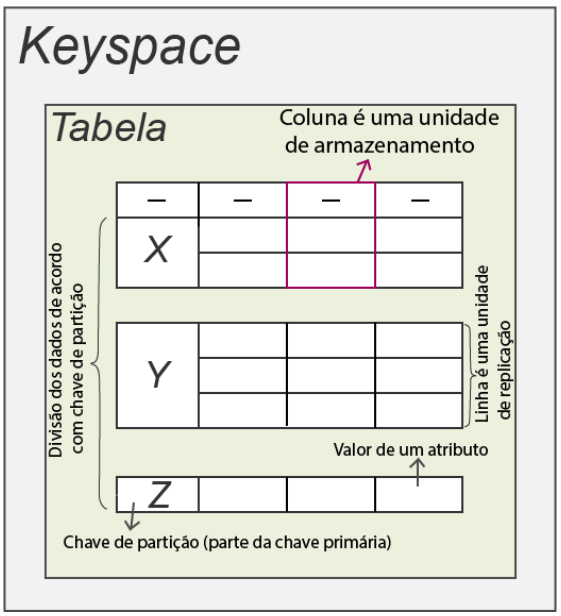

Figura 2 – Esquema de armazenamento Cassandra

A eficiência do Cassandra depende fundamentalmente de consultas que utilizam a chave de partição. Buscas que não o fazem resultam em varreduras lentas e ineficientes (CHEBOTKO; KASHLEV; LU, 2015; CARPENTER; HEWITT, 2020). No contexto do FHIR, com seus múltiplos parâmetros de busca por recurso, torna-se inviável modelar todas as consultas possíveis usando chaves de partição distintas. Os mecanismos nativos do Cassandra, como índices secundários e visões materializadas, impõem dificuldades de gerenciamento e perdas de desempenho quando aplicados a cenários de consulta tão complexos (CHEBOTKO; KASHLEV; LU, 2015; CARPENTER; HEWITT, 2020; ALAPATI, 2018). A alternativa, a desnormalização de dados, leva a uma duplicação insustentável de dados e a uma complexidade de manutenção proibitiva (CARPENTER; HEWITT, 2020; ALAPATI, 2018). Para mitigar esses desafios, a integração com ferramentas de indexação externas, como o Apache Lucene, surge como uma abordagem mais adequada.

O Apache Lucene é uma biblioteca de busca de texto completo, de código aberto

e escrita em Java, que serve de base para plataformas como Elasticsearch[1] e Solr[2]. Sua arquitetura é centrada no conceito de índice invertido, uma estrutura de dados que permite buscas rápidas em grandes volumes de texto (SHARMA, 2020). O Lucene suporta uma vasta gama de consultas, desde buscas simples por palavras-chave até operações complexas, como consultas booleanas, buscas com caracteres curinga e consultas por intervalo. Sua capacidade de classificar resultados com base na relevância, utilizando algoritmos como TF-IDF ou BM25, e sua flexibilidade para indexar formatos de dados semiestruturados como JSON, o tornam ideal para integração com sistemas de armazenamento que carecem de capacidades de busca avançada (SHARMA, 2020; LUCENE, 2024).

A integração do Lucene com o Cassandra é uma abordagem tecnológica conhecida, geralmente viabilizada por meio de *plugins* que incorporam seu mecanismo de indexação ao banco de dados. Nessa configuração, os dados gravados no Cassandra são simultaneamente indexados pelo Lucene, permitindo que consultas complexas sejam executadas diretamente sobre os dados, combinando assim a escalabilidade do Cassandra com o poder de busca do Lucene (SHARMA, 2020). Essa capacidade de lidar com diversos tipos de consultas e se integrar a sistemas distribuídos demonstra a versatilidade do Lucene na resolução de desafios complexos de dados, justificando sua escolha como componente central da arquitetura proposta neste trabalho.

[1] Disponível em: <https://www.elastic.co/>
[2] Disponível em: <https://solr.apache.org/>

4 A ARQUITETURA CLIF-ARCH

A solução para a incompatibilidade entre o Apache Cassandra e os requisitos de busca do padrão FHIR reside na criação de uma arquitetura que estenda as capacidades nativas do banco de dados. Este estudo propõe e valida a CLiF-Arch (Cassandra Lucene-*indexed* FHIR *Architecture*), uma arquitetura que habilita a conformidade completa com o FHIR ao integrar um motor de busca avançado diretamente na camada de persistência. O objetivo não é avaliar um modelo de dados incomum, mas sim validar uma arquitetura de banco de dados que capacite o Cassandra, com suas vantagens únicas de escalabilidade e disponibilidade (LAKSHMAN; MALIK, 2010), a suportar de forma eficiente as complexas demandas de consulta do padrão FHIR, superando suas limitações conhecidas por meio de uma abordagem específica.

A contribuição técnica deste trabalho consistiu na integração e customização do código-fonte do plugin de código aberto *Stratio's Cassandra Lucene Index*, adicionando a ele a biblioteca HAPI-FHIR. O processo de desenvolvimento seguiu etapas bem definidas: primeiramente, o repositório do plugin foi clonado e a biblioteca HAPI-FHIR, uma implementação Java de referência para o padrão FHIR (SHOUMIK et al., 2017; KIRILOV; DUGAS, 2024), foi adicionada como uma dependência do projeto. Em seguida, foram realizadas modificações no código Java do plugin para que, ao interceptar uma operação de escrita, ele utilizasse o HAPI-FHIR para analisar o recurso FHIR (em formato JSON) e extrair seletivamente os parâmetros de busca a serem indexados. Por fim, o projeto foi compilado em um arquivo JAR e implantado na instalação do Cassandra, resultando em uma imagem Docker[1] que encapsula a solução pronta para uso.

4.1 Arquitetura e Componentes

A solução adota uma abordagem modular, encapsulando a nova lógica de manipulação de dados FHIR e minimizando as alterações no núcleo do sistema existente. Arquiteturas modulares e fracamente acopladas são uma prática recomendada para simplificar o desenvolvimento, a manutenção e a implantação de serviços independentes, especialmente em sistemas complexos (SHOUMIK et al., 2017). A arquitetura original do plugin Stratio, projetada com base em padrões de projeto como *Strategy* e *Factory*, facilitou sua extensão. A Figura 3 ilustra o diagrama de classes da integração, que foi dividida em três pacotes lógicos principais: o código existente do Stratio (em azul), a nova camada de integração FHIR (em verde) e as dependências de bibliotecas externas (em laranja).

[1] Disponível em: <https://hub.docker.com/r/danielsmelo/cassandra-fhir>

A nova funcionalidade é encapsulada em um pacote dedicado, que se acopla à estrutura existente de forma coesa. Conforme ilustrado na Figura 3, os pontos de entrada para esta nova camada são as classes *FhirMapper* e *FhirMapperBuilder*. O *FhirMapperBuilder* atua como a fábrica que, ao identificar a configuração *type*: "fhir"no esquema do índice, instancia o *FhirMapper*. Este, por sua vez, é a implementação concreta da interface *Mapper*, sendo especializado em processar os dados FHIR. O *FhirMapper* delega o trabalho de análise ao *SearchParamExtractorHelper*, o principal orquestrador da lógica FHIR. Este componente utiliza a biblioteca HAPI-FHIR para decodificar o JSON do recurso em um objeto estruturado, uma decisão estratégica que evita a recriação de uma complexa lógica de interpretação. Para a extração dos dados, o Helper utiliza o padrão *Factory* por meio da *SearchParamExtractorFactory*, que fornece o extrator adequado (*SearchParamExtractor*) para cada tipo de parâmetro de busca (como *token, string, date*), garantindo que a lógica seja isolada e de fácil manutenção. Os valores extraídos são então encapsulados em Objetos de Transferência de Dados (DTOs), representados pela classe *AbstractSearchParam* e suas subclasses, que finalmente geram os campos a serem escritos no índice Lucene.

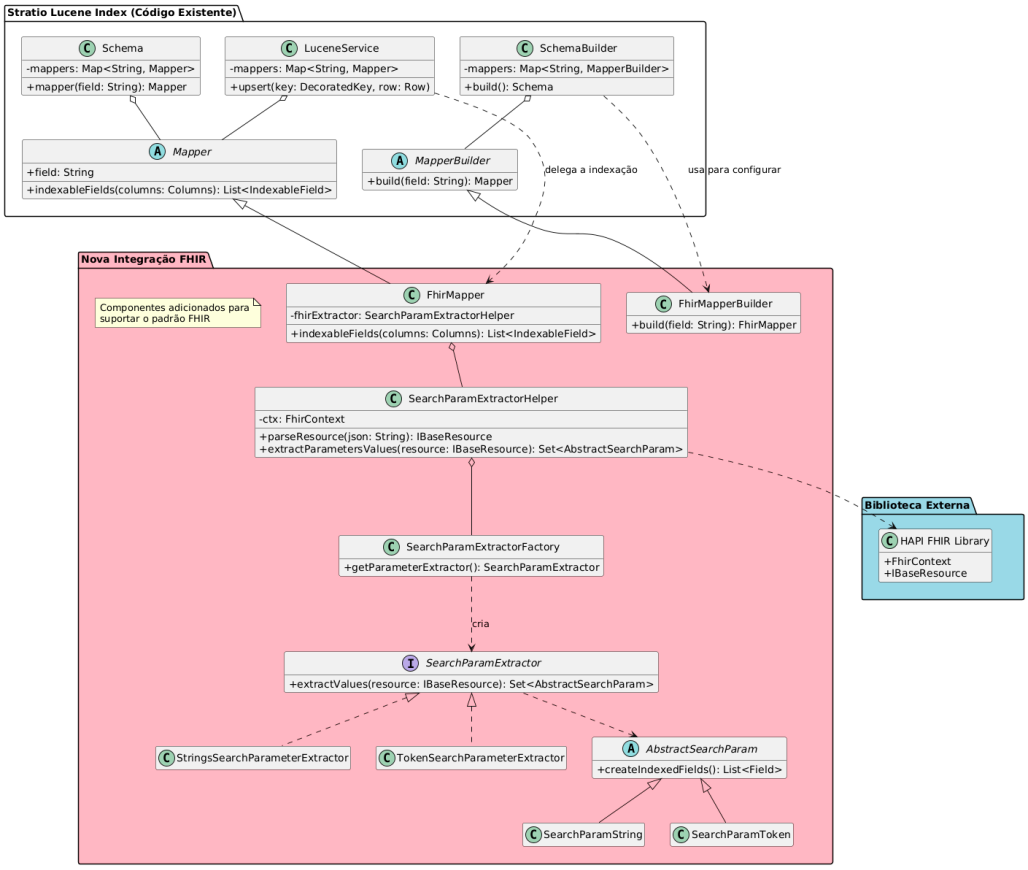

Figura 3 – Diagrama de classes da integração entre o *plugin* Stratio e a biblioteca HAPI-FHIR

A estrutura de banco de dados resultante é ilustrada na Figura 4. A arquitetura utiliza uma única "Tabela FHIR"no Cassandra para armazenar todos os recursos de um mesmo tipo, cada um como um documento JSON. A estratégia de replicação do Cassandra garante a distribuição de cópias dos dados (réplicas) entre os nós do *cluster* (LAKSHMAN; MALIK, 2010). O plugin de índice customizado atua em cada nó que contém uma réplica; quando um recurso é escrito, o plugin cria e mantém um índice Lucene local para os dados daquele nó específico. Dessa forma, a indexação acompanha a distribuição e a resiliência dos dados, garantindo que as capacidades de busca estejam disponíveis de forma descentralizada e escalável (LUCENE, 2024).

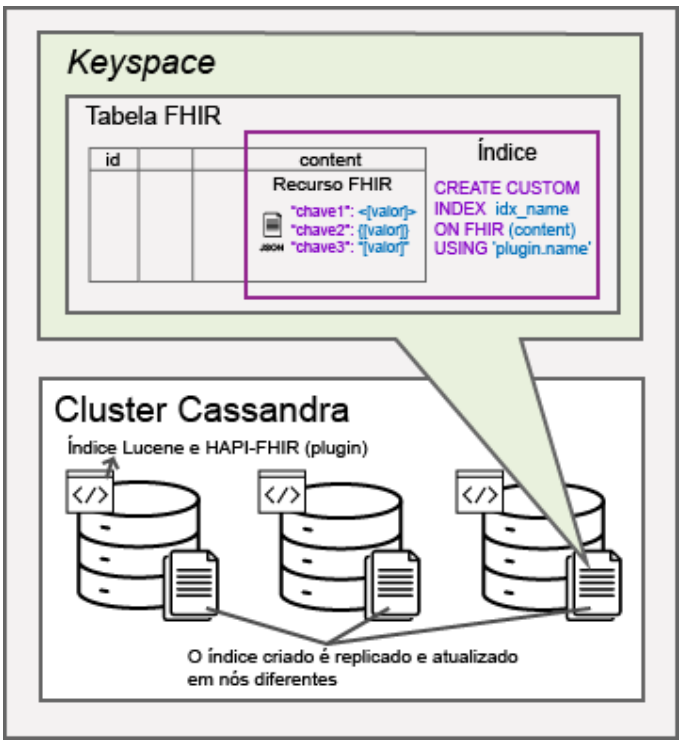

Figura 4 – Estrutura de indexação local e distribuída com o *plugin* Lucene no Cassandra

A biblioteca HAPI-FHIR desempenha um papel fundamental na análise e extração de informações dos recursos, permitindo a indexação seletiva. A especificação FHIR disponibiliza um conjunto de parâmetros de pesquisa padrão, mas também permite a customização através do recurso SearchParameter (AYAZ et al., 2021; BRAUNSTEIN, 2018). Na CLiF-Arch, quando um novo SearchParameter é definido, o HAPI-FHIR o reconhece e passa a extrair e indexar os valores correspondentes nos recursos subsequentes. É importante notar que dados preexistentes não são reindexados automaticamente, sendo necessário um processo de reindexação para garantir a consistência das buscas em todo o histórico de dados.

4.2 Fluxos de Operação

Para compreender o funcionamento da arquitetura na prática, é útil detalhar os dois fluxos de operação principais: o de indexação, que ocorre durante a escrita de dados, e o de consulta, que ocorre durante a leitura.

4.2.1 Fluxo de Indexação

O processo de indexação de um recurso FHIR é síncrono e ocorre em uma sequência de etapas bem definidas toda vez que um dado é escrito ou atualizado na tabela monitorada pelo Lucene:

1. O *LuceneService* do *plugin* Stratio detecta uma operação de escrita em uma coluna configurada com o *FhirMapper*.
2. O *FhirMapper* é invocado e delega o processamento do documento JSON ao *SearchParamExtractorHelper*.
3. O *Helper* utiliza a biblioteca HAPI-FHIR para converter o JSON em um objeto *IBaseResource* estruturado e validado.
4. Para cada parâmetro de busca a ser indexado (conforme definido no esquema do índice), o *Helper* requisita o *SearchParamExtractor* apropriado à *SearchParamExtractorFactory*.
5. O *SearchParamExtractor* correspondente extrai o valor do objeto *IBaseResource* e o encapsula em uma instância de *AbstractSearchParam*.
6. A instância de *AbstractSearchParam* gera os campos (*Fields*) Lucene correspondentes, formatados adequadamente para busca.
7. Esses campos são retornados ao *LuceneService*, que finaliza o processo ao escrevê-los no índice Lucene local daquele nó.

4.2.2 Fluxo de Consulta

A Figura 5 ilustra o fluxo de uma consulta na arquitetura CLiF-Arch, que permite ao usuário final interagir com o sistema utilizando a sintaxe de busca do FHIR, abstraindo a complexidade da camada de indexação. O processo ocorre em três etapas sequenciais:

1. **Tradução da Consulta:** A consulta de busca FHIR, com seus parâmetros e modificadores, é traduzida para a sintaxe de consulta equivalente do Lucene. Essa consulta é então incorporada a uma instrução da *Cassandra Query Language* (CQL), utilizando a sintaxe expr() do *plugin* na cláusula *WHERE*.

2. **Busca Indexada e Recuperação de Dados:** A instrução CQL é submetida ao *cluster* Cassandra. O nó coordenador distribui a consulta aos nós relevantes. Em cada nó, o *plugin* Lucene utiliza seu índice local para identificar as linhas que correspondem às condições da consulta. Com base nessas correspondências, o Cassandra recupera os recursos FHIR completos das linhas identificadas e os retorna ao nó coordenador.

3. **Agregação e Resposta:** O nó coordenador coleta os recursos recuperados de todos os nós consultados e os consolida em um único conjunto de resultados. Finalmente, este conjunto é tipicamente formatado como um recurso FHIR do tipo *Bundle* e retornado à aplicação cliente que originou a requisição.

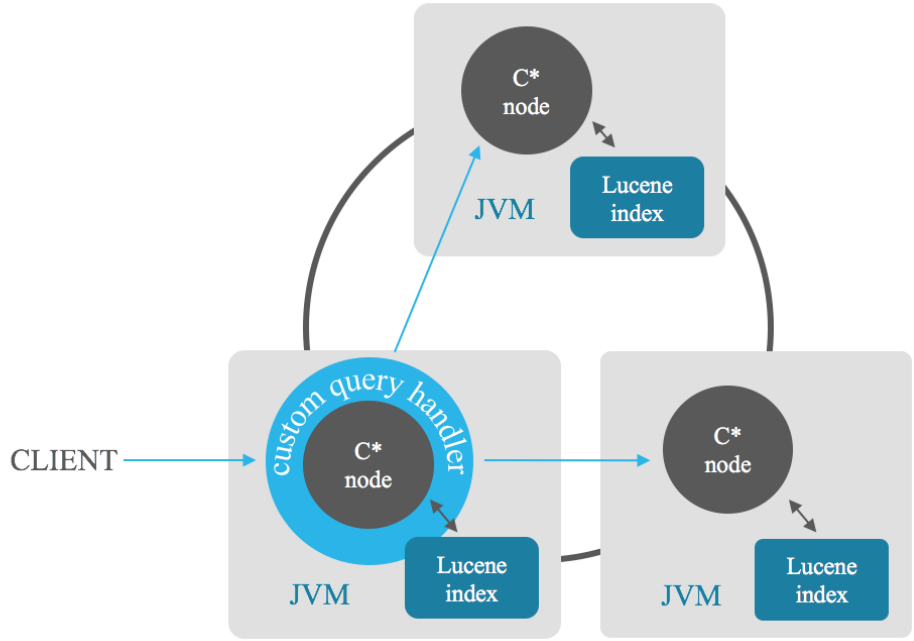

Figura 5 – Fluxo de Consulta e Recuperação de Dados no plugin Lucene

Fonte: <https://github.com/Stratio/cassandra-lucene-index>

A análise do fluxo de consulta elucida o mecanismo pelo qual a arquitetura CLiF-Arch harmoniza tecnologias distintas para superar uma limitação fundamental de consulta. Ao instituir uma camada de abstração que traduz a sintaxe de busca do padrão FHIR para o motor Lucene, a arquitetura permite uma interação padronizada e interoperável, ocultando da aplicação cliente a complexidade da indexação subjacente. Esse processo explora a busca distribuída e paralela dos índices Lucene locais em cada nó, ao mesmo tempo que se beneficia da resiliência e da capacidade de agregação de dados do Cassandra. O ciclo se completa com a entrega dos resultados em recursos FHIR, garantindo a total aderência ao padrão e, assim, validando a eficácia da abordagem.

5 METODOLOGIA EXPERIMENTAL

Para validar as hipóteses de pesquisa de forma rigorosa, foi desenhado um plano experimental baseado na comparação de dois cenários arquitetônicos distintos, submetidos a uma bateria de testes de carga. O objetivo desta metodologia é quantificar as diferenças de desempenho entre a arquitetura proposta e uma abordagem convencional, fornecendo dados empíricos para avaliar a viabilidade da solução.

5.1 Delineamento Experimental

A avaliação quantitativa da arquitetura proposta foi realizada por meio de um processo de *benchmark* rigoroso, que comparou o desempenho de dois ambientes de banco de dados distintos sob uma carga de trabalho realista de operações CRUD (*Create, Read, Update, Delete*). O primeiro ambiente, denominado cenário indexado, corresponde à própria arquitetura CLiF-Arch, com uma única tabela por recurso e o índice Lucene customizado. O segundo, o cenário desnormalizado, serve como grupo de controle (*baseline*), representando a abordagem convencional para otimizar consultas no Cassandra nativo, onde múltiplas tabelas duplicadas são criadas, cada uma com uma chave primária diferente para atender a um padrão de consulta específico. A comparação direta entre esses dois cenários é fundamental para isolar e medir os benefícios de flexibilidade e os custos de desempenho da CLiF-Arch.

A implementação e a validação da arquitetura exigiram um processo de compilação (*build*) e implantação (*deploy*) robusto e reprodutível. Conforme ilustrado na Figura 6, um *pipeline* automatizado foi adotado, utilizando tecnologias de contêineres para garantir a consistência do ambiente. O processo iniciou-se com a compilação do código-fonte do *plugin*, que integra o *Stratio Lucene Index* e a biblioteca HAPI-FHIR, em um artefato JAR. Subsequentemente, uma imagem Docker customizada foi criada, incorporando este *plugin* a uma imagem base do Apache Cassandra. Finalmente, a partir desta imagem customizada, foi instanciado um *cluster* de três nós, com um deles atuando como nó de contato inicial (*seed node*) para a descoberta da topologia.

Para garantir a consistência e a reprodutibilidade dos testes, todas as instâncias do *cluster* foram configuradas com recursos computacionais idênticos: 8 vCPUs, 16 GB de RAM e 20 GB de armazenamento SSD, todos hospedados na mesma máquina física para eliminar a variabilidade de latência de rede. O conjunto de dados utilizado foi derivado do Synthea[1], uma plataforma de código aberto que gera dados de saúde sintéticos compatíveis

[1] Disponível em: <https://synthetichealth.github.io/synthea/>

com o padrão FHIR. Para assegurar uma análise robusta em um cenário de grande volume, o conjunto de dados original foi expandido de 50.000 para aproximadamente 10 milhões de registros, preservando a integridade estrutural e a variabilidade dos dados. Os recursos FHIR foram armazenados no Cassandra em formato JSON, utilizando identificadores únicos como chaves primárias para otimizar a distribuição dos dados entre os nós do *cluster*.

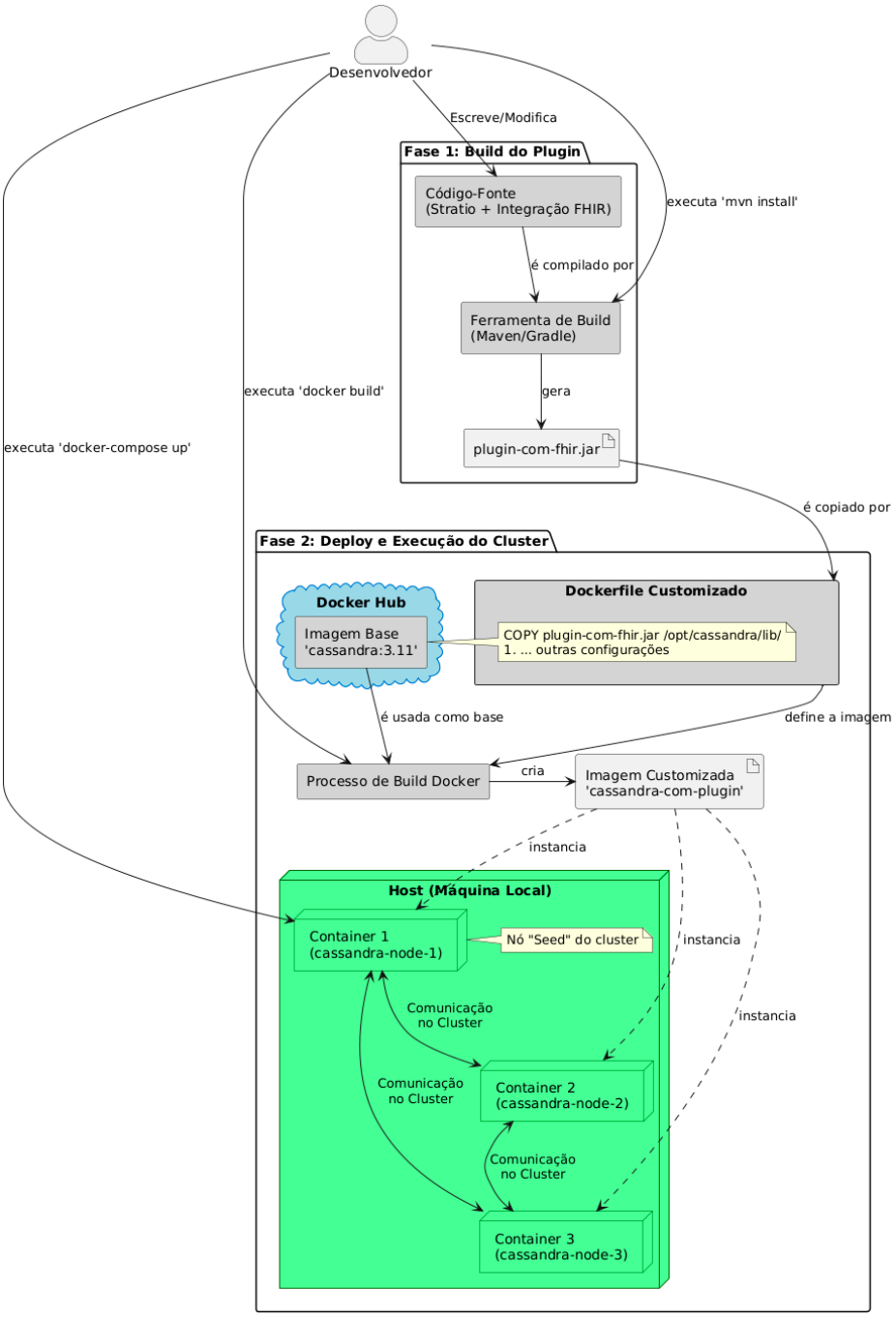

Figura 6 – Fluxo de compilação e implantação da arquitetura em um ambiente de cluster

5.2 Carga de Trabalho e Coleta de Dados

O fluxo de execução do *benchmark*, ilustrado na Figura 7, foi automatizado para garantir a consistência dos resultados. A metodologia seguiu uma série de etapas iterativas, projetadas para simular uma carga de trabalho realista e medir a latência como principal indicador de desempenho. O processo iniciou-se com a definição de 336 combinações distintas de parâmetros de busca FHIR, representando uma gama variada de cenários de consulta. Para cada uma dessas combinações, um teste de carga foi executado concorrentemente em ambos os ambientes (indexado e desnormalizado).

Dentro de cada iteração, foi simulada uma carga de trabalho intensiva utilizando a ferramenta cassandra-stress. Esta carga consistiu em 10.000 repetições sequenciais de um ciclo de operações CRUD, envolvendo a criação de um novo registro, uma consulta para lê-lo, uma atualização e, finalmente, sua remoção. A decisão por um número elevado de repetições teve como propósito metodológico garantir a estabilidade estatística das medições, minimizando o impacto de flutuações transitórias do ambiente que poderiam distorcer os resultados. A latência de cada uma das 10.000 operações foi registrada para cada ambiente, e ao final do ciclo, a média de latência para aquela combinação de parâmetros foi calculada e armazenada.

A métrica primária de avaliação de desempenho foi definida como a latência média das operações CRUD, medida em milissegundos. A média foi escolhida como medida de tendência central por ser um indicador robusto e universalmente compreendido para sumarizar o desempenho típico em testes de alta repetição, amortecendo significativamente o efeito de anomalias individuais. Durante a execução, foi observado que menos de 0,05% de todas as repetições se comportaram como *outliers*, e seus valores não excederam oito vezes o valor mais frequente, o que reforça a confiabilidade da média como um indicador representativo. A análise comparativa final desses resultados permitiu uma avaliação objetiva da eficiência de cada arquitetura para os diferentes tipos de consulta.

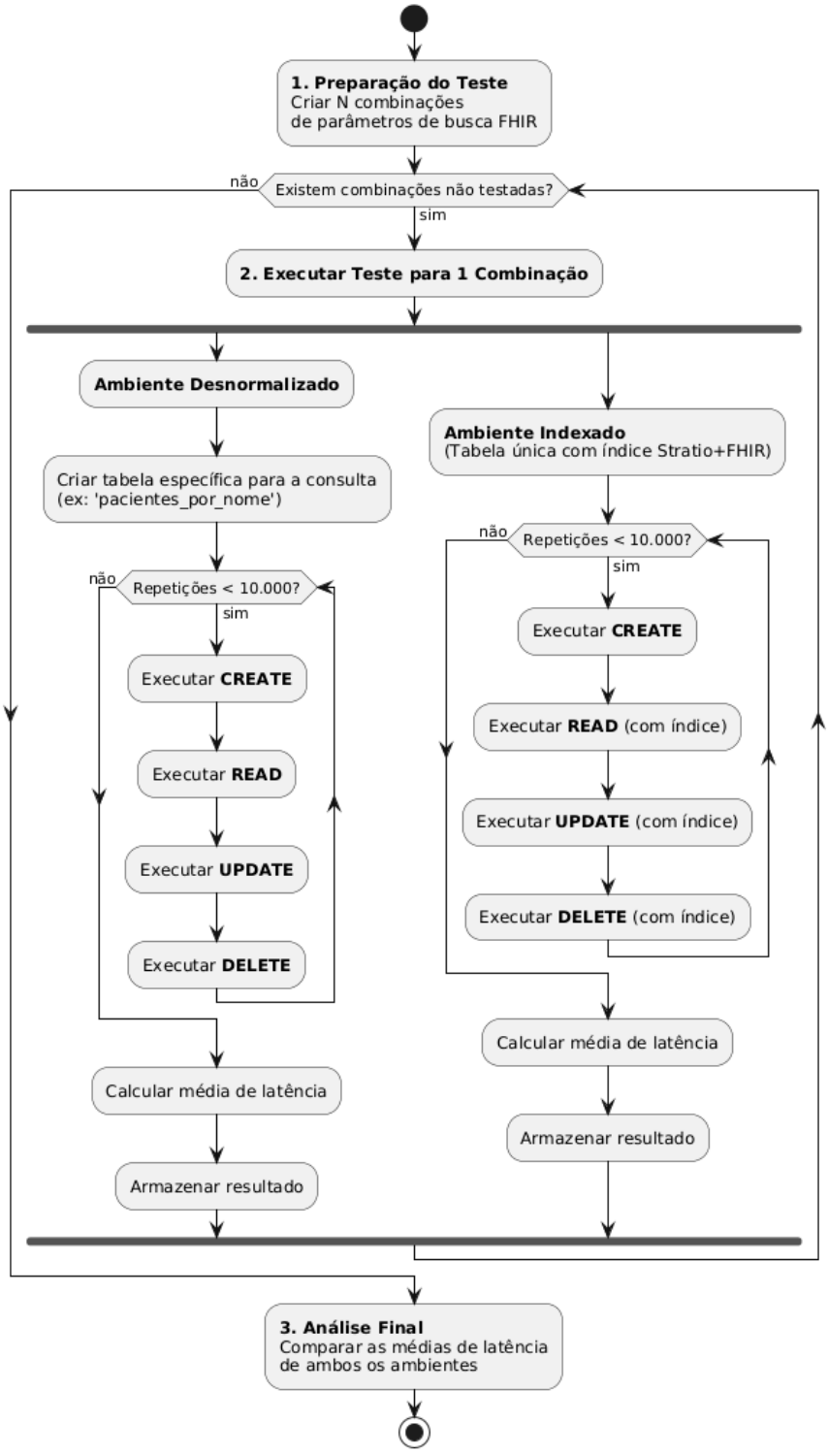

Figura 7 – Diagrama de atividades do processo de *benchmark*

6 ANÁLISE DOS RESULTADOS

Este capítulo apresenta e analisa os resultados obtidos a partir da metodologia experimental. A análise está estruturada para validar sistematicamente cada uma das hipóteses formuladas no capítulo anterior, começando pela validação funcional da arquitetura e prosseguindo para a quantificação de seu desempenho. Ao final, os resultados são discutidos em conjunto para contextualizar as implicações práticas dos achados.

6.1 Análise Comparativa de Desempenho

O primeiro passo da análise foi a validação funcional da arquitetura, que buscou responder à questão sobre a compatibilidade da CLiF-Arch com o padrão FHIR. A metodologia consistiu na execução de uma bateria de testes que simulou operações de busca utilizando a maioria dos tipos de parâmetros (*token, string, date, number*) e prefixos de comparação (eq, gt, le, ap) descritos na documentação oficial do FHIR. Em todos os casos, a arquitetura foi capaz de processar as consultas e retornar os conjuntos de dados corretos, incluindo buscas com múltiplos atributos, consultas por intervalo e buscas textuais com curingas. Este resultado empírico refuta a hipótese nula $H0_{\text{func}}$ e, consequentemente, valida a hipótese alternativa $H1_{\text{func}}$, confirmando que a CLiF-Arch é funcionalmente compatível com os requisitos de busca do padrão.

Uma vez confirmada a funcionalidade, a análise de desempenho quantificou a sobrecarga de latência introduzida pela camada de indexação. Uma visão geral dos resultados é apresentada no *boxplot* da Figura 8, que sumariza a distribuição da latência média para cada operação CRUD nos cenários indexado e desnormalizado. Visualmente, é possível observar duas tendências claras: primeiro, todas as operações no cenário indexado apresentam uma latência mediana superior à de suas contrapartes no cenário desnormalizado; segundo, a operação de "Inserção"no cenário indexado exibe não apenas a maior latência, mas também a maior dispersão de resultados. A latência de inserção, que no cenário desnormalizado se concentra em torno de 1 ms, expande-se para uma faixa entre 3 e 4 ms na CLiF-Arch, evidenciando o custo do processo de indexação síncrona. Este trade-off entre flexibilidade de consulta e latência de escrita é um desafio inerente a muitas arquiteturas de dados de alta performance, onde a manutenção de índices secundários complexos impacta diretamente o tempo de ingestão dos dados (LAKSHMAN; MALIK, 2010; CARPENTER; HEWITT, 2020).

A análise estatística, realizada por meio de um teste t para amostras pareadas (detalhada na Tabela 1), confirma formalmente as observações visuais. Com valores-p

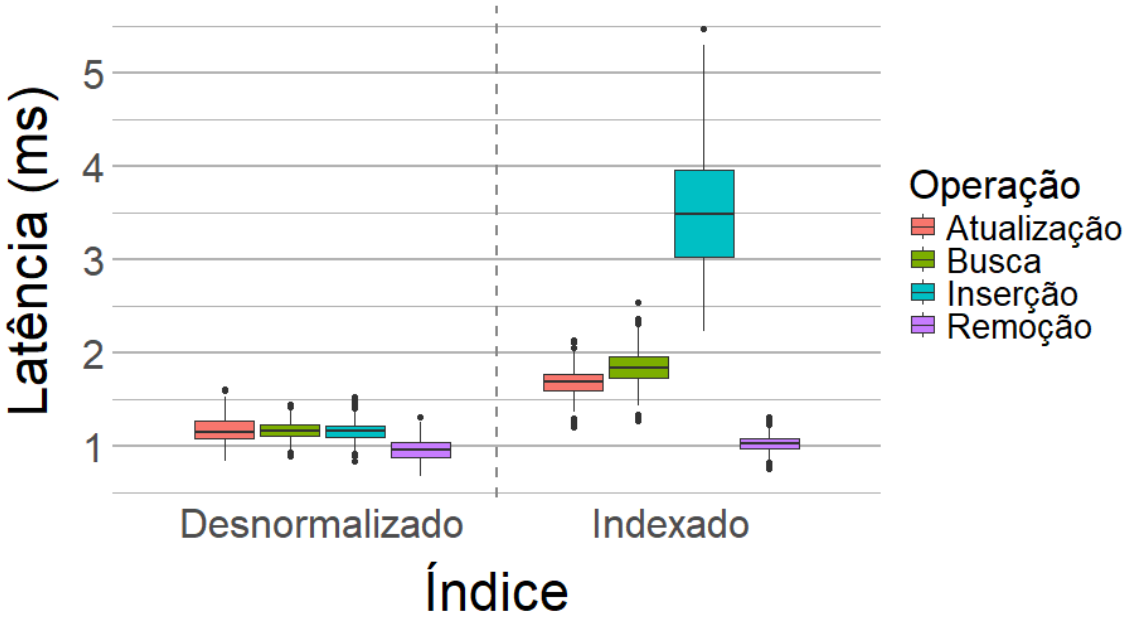

Figura 8 – Boxplot da latência média por operação para os cenários indexado e desnormalizado

drasticamente inferiores ao nível de significância de 0,05 para todas as operações, a análise fornece forte evidência para rejeitar as hipóteses nulas de desempenho. Para as operações de escrita, os resultados permitem rejeitar a $H0_{write}$ e, portanto, validar a $H1_{write}$, confirmando que as operações no cenário indexado são estatisticamente mais lentas. Da mesma forma, a diferença significativa na latência de busca leva à rejeição de $H0_{read}$ e à validação de $H1_{read}$.

É importante ressaltar a robustez da métrica de latência média utilizada. Para cada combinação de parâmetros de busca, cada operação CRUD foi executada 10.000 vezes. Dentro dessas repetições, observou-se que uma fração mínima das operações ($< 0{,}007\%$) apresentou latências atípicas, provavelmente devido a eventos transitórios do sistema, como pausas para coleta de lixo da JVM. Dada a baixa ocorrência e a magnitude controlada desses eventos, seu efeito sobre a média aritmética de 10.000 amostras é desprezível. Isso valida o uso da média como um indicador de desempenho estável e representativo para cada cenário de teste específico.

Os *outliers* observados no *boxplot* da Figura 8, por outro lado, representam não operações individuais, mas sim a latência média de combinações de parâmetros de busca específicas cujo desempenho divergiu significativamente do conjunto. Para ilustrar um caso de uso que resultaria em tal *outlier*, pode-se considerar uma consulta que combine uma

Tabela 1 – Teste t pareado para latência com CLiF-Arch e com a abordagem desnormalizada

Operação	**Valor** (p)	**Estatística** (t)	**Interpretação**
Atualização	$1{,}64 \times 10^{-66}$	28,7	A operação de atualização apresentou latência significativamente maior com o uso do CLiF-Arch em relação à abordagem desnormalizada.
Busca	$5{,}53 \times 10^{-84}$	37,9	A operação de busca apresentou latência significativamente maior com o uso do CLiF-Arch em relação à abordagem desnormalizada.
Inserção	$6{,}81 \times 10^{-95}$	44,7	A operação de inserção apresentou latência significativamente maior com o uso do CLiF-Arch em relação à abordagem desnormalizada.
Remoção	$3{,}35 \times 10^{-7}$	5,32	A operação de remoção apresentou latência significativamente maior com o uso do CLiF-Arch em relação à abordagem desnormalizada.

busca por fragmento de texto em qualquer posição de um nome (*wildcard*), uma busca por aproximação em um endereço com tolerância a erros de digitação (*fuzzy*), e um filtro por intervalo de datas. A execução de tal consulta exige que o mecanismo Lucene realize operações computacionalmente mais custosas, como varreduras de termos e cálculos de distância de edição, resultando em uma latência média consistentemente mais elevada para suas 10.000 repetições.

A heterogeneidade no custo de desempenho das operações de escrita é mais bem visualizada no gráfico de densidade (Figura 9). A curva de "Inserção"para o cenário indexado é notavelmente mais baixa e ampla, indicando uma alta variabilidade nos tempos de resposta. Essa dispersão reflete a complexidade variável dos recursos FHIR: a indexação de um recurso *Observation* simples, com poucos campos, é mais rápida que a de um *Patient* complexo, com múltiplos identificadores, endereços e extensões, que exige uma análise mais intensiva. Portanto, a distribuição de latência para inserções não é uniforme, mas sim dispersa, refletindo a diversidade de complexidade dos dados processados. Essa variabilidade é consistente com a natureza do FHIR, que, ao contrário de modelos de dados rígidos, permite que recursos do mesmo tipo (ex: Paciente) tenham diferentes níveis de detalhe e extensões, impactando o custo de processamento de cada instância individualmente (AYAZ et al., 2021; POINTNER; PRASCHL; KRAUSS, 2023).

A análise de confiabilidade da arquitetura CLiF-Arch foi conduzida como uma etapa integrada e realizada após os testes de latência, aproveitando a própria carga de trabalho do *benchmark* para validar a integridade dos dados. Para cada uma das 3,36 milhões de

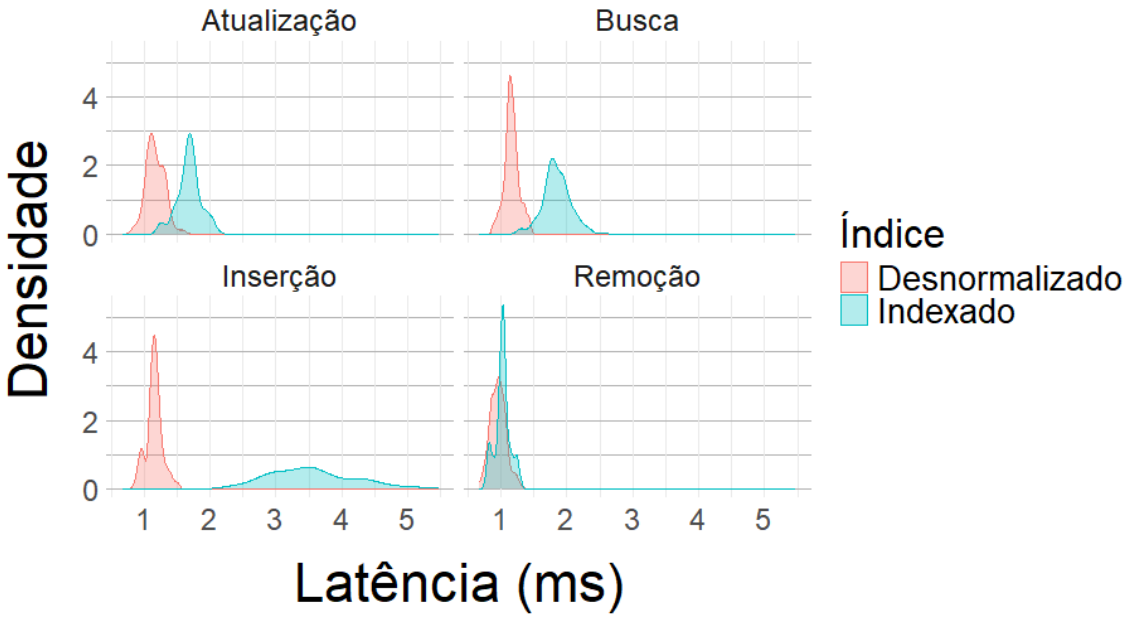

Figura 9 – Gráfico de densidade da distribuição de latência por operação

operações de leitura realizadas durante os testes, o conjunto de resultados retornado pela consulta via CQL foi sistematicamente registrado. Em paralelo, para estabelecer uma base de comparação fidedigna (ou *ground truth*), um *script* automatizado acessava diretamente o conjunto de dados brutos originais para identificar o resultado exato que deveria ser retornado para aquela consulta específica. Finalmente, um segundo *script* de comparação foi executado para confrontar, registro por registro, o resultado obtido da CLiF-Arch com o resultado esperado. Este processo rigoroso, aplicado a todo o volume de consultas, não detectou qualquer tipo de inconsistência, perda ou corrupção de dados, confirmando que 100% das informações foram recuperadas de forma correta e íntegra e atestando a robustez e a confiabilidade da arquitetura proposta.

6.2 Análise Isolada dos Componentes

Para isolar o impacto de cada componente da arquitetura, uma segunda bateria de testes comparou o desempenho do índice Lucene puro (que indexa o documento JSON como pares chave-valor) contra a arquitetura CLiF-Arch completa (que utiliza a biblioteca HAPI-FHIR para uma análise semântica). A Figura 10 apresenta a distribuição de latência para esta comparação. A análise visual do *boxplot* revela que, para a operação de "Inserção", há um claro aumento na latência mediana e na dispersão dos dados quando o HAPI-FHIR

é integrado. Em contrapartida, para a operação de "Busca", as distribuições de latência são visualmente muito semelhantes, sugerindo um impacto mínimo. Curiosamente, para as operações de "Atualização"e "Remoção", a arquitetura completa com HAPI-FHIR apresenta uma latência mediana ligeiramente inferior.

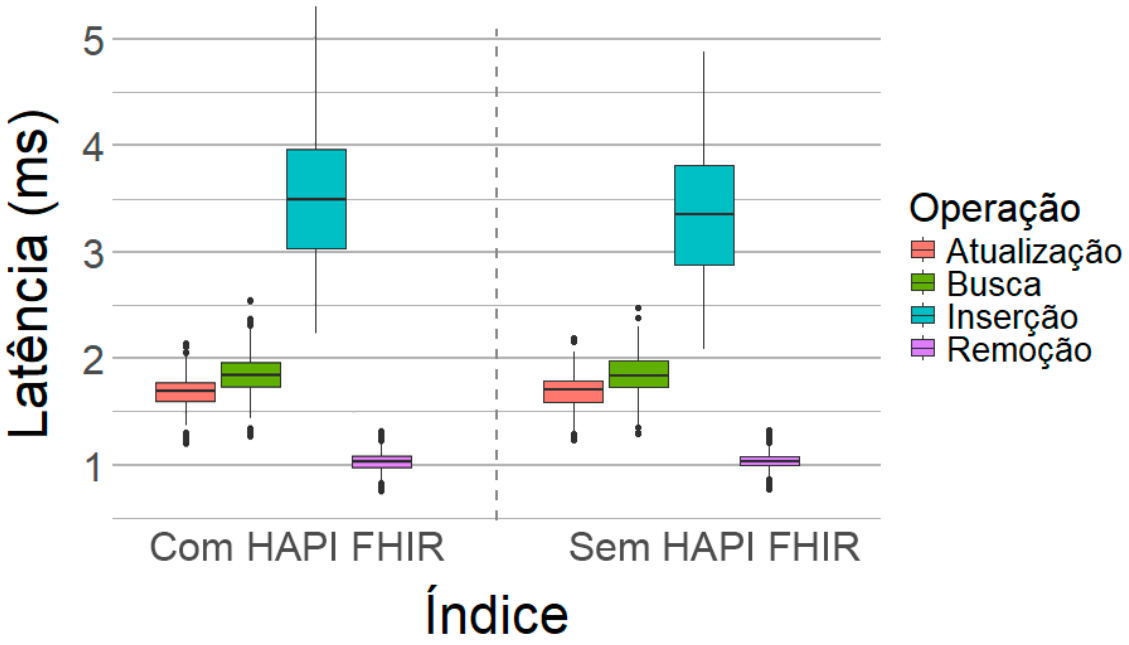

Figura 10 – Boxplot comparando a latência do Lucene com e sem a integração do HAPI-FHIR

A análise estatística, detalhada na Tabela 2, confirma e quantifica essas observações. O valor-p extremamente baixo para a operação de inserção ($6{,}27 \times 10^{30}$) confirma que a utilização do HAPI-FHIR impõe um custo computacional significativo. Isso ocorre porque a biblioteca executa tarefas de validação e análise do recurso para extrair seletivamente os parâmetros de busca, um processo mais intensivo do que a simples indexação de todo o documento. Este achado reforça que a sobrecarga de desempenho na escrita está primariamente associada à camada de processamento semântico, que é fundamental para garantir a conformidade com o padrão FHIR.

Para as operações de busca, o valor-p de 0,146 não é estatisticamente significativo, o que indica que, uma vez que os dados estão indexados, o desempenho da leitura depende quase exclusivamente da eficiência do motor Lucene. Este é um resultado importante, pois demonstra que o custo da análise semântica do HAPI-FHIR é pago no momento da escrita, não penalizando as consultas subsequentes. Para as operações de "Atualização"e "Exclusão", as estatísticas t negativas indicam uma melhora de desempenho com a CLiF-Arch. Uma

Tabela 2 – Teste T pareado para comparação entre Lucene com e sem HAPI-FHIR

Operação	**Valor** (p)	**Estatística** (t)	**Interpretação**
Atualização	$4{,}02 \times 10^{-5}$	-4,22	A operação de atualização apresentou latência significativamente menor com o uso do HAPI-FHIR em relação à versão sem HAPI-FHIR.
Busca	$1{,}46 \times 10^{-1}$	-1,46	A operação de busca apresentou tendência de latência menor com o uso do HAPI-FHIR, porém sem significância estatística.
Inserção	$6{,}27 \times 10^{-30}$	14,0	A operação de inserção apresentou latência significativamente menor na versão sem HAPI-FHIR em relação à versão com HAPI-FHIR.
Remoção	$2{,}97 \times 10^{-3}$	-3,01	A operação de remoção apresentou latência significativamente menor com o uso do HAPI-FHIR em relação à versão sem HAPI-FHIR.

possível explicação é que a análise semântica permite uma gestão mais granular do índice; em vez de reindexar o documento inteiro, a arquitetura pode identificar e modificar apenas os campos específicos que foram alterados, resultando em uma operação mais otimizada.

7 CONCLUSÃO

A questão de pesquisa central deste trabalho buscou determinar se a integração de um motor de busca avançado, como o Apache Lucene, poderia flexibilizar o uso do Apache Cassandra para servidores FHIR, superando suas limitações de consulta. Os resultados empíricos obtidos através de uma metodologia de *benchmark* rigorosa permitem responder afirmativamente a esta questão. A arquitetura CLiF-Arch demonstrou ser funcionalmente capaz de executar a vasta gama de consultas complexas especificadas pelo padrão FHIR — incluindo buscas por múltiplos atributos, por intervalo e com curingas — que são nativamente insuportáveis pelo Cassandra. Este achado valida a hipótese de que a integração de tecnologias de indexação especializadas é uma estratégia eficaz para conciliar a escalabilidade de um banco de dados NoSQL colunar com os requisitos de interoperabilidade de padrões de saúde como o FHIR (AYAZ et al., 2021).

A análise de desempenho, por sua vez, revelou o *trade-off* inerente a essa flexibilidade. Conforme validado pelas hipóteses de pesquisa, a introdução da camada de indexação síncrona resultou em um aumento estatisticamente significativo na latência de todas as operações CRUD. O impacto mais notável foi na latência de inserção, que se elevou da faixa de 1 ms para 3 a 4 ms. Este aumento, embora significativo, não atingiu o limiar de 50 ms estabelecido como inadequado para o escopo deste protótipo. Em muitos cenários de saúde, como o armazenamento de dados de IoMT ou a atualização de prontuários, uma latência de poucos milissegundos para a escrita é perfeitamente aceitável, especialmente quando comparada ao benefício de obter uma capacidade de consulta que, de outra forma, seria inviável (FAMÁ; FARIA; PORTUGAL, 2022; RUBí; GONDIM, 2019; CARPENTER; HEWITT, 2020; ALAPATI, 2018). Portanto, a CLiF-Arch representa um compromisso de engenharia explícito e viável: um pequeno custo na performance de escrita é trocado por uma capacidade de leitura imensamente superior e compatível com o FHIR, o que não inviabiliza o uso do Cassandra, mas, pelo contrário, o qualifica para um novo leque de aplicações.

A análise de confiabilidade da arquitetura aprofunda a validação de sua robustez. A obtenção de uma taxa de precisão de 100% em mais de 3,36 milhões de operações de consulta, realizadas comparando os resultados com os dados brutos, atesta que a complexidade introduzida pelo mecanismo de indexação não compromete a integridade dos dados. Este é um requisito não funcional crítico em sistemas de saúde, onde a exatidão das informações é fundamental para a segurança do paciente e para a tomada de decisão clínica (BALDASSANO et al., 2019). A combinação da estratégia de replicação de dados nativa do Cassandra, que garante a durabilidade e a disponibilidade das informações

mesmo em caso de falha de nós (LAKSHMAN; MALIK, 2010; CARPENTER; HEWITT, 2020), com um mecanismo de indexação que se mostrou preciso e confiável, resulta em uma arquitetura que não apenas resolve um desafio de consulta, mas o faz mantendo os mais altos padrões de integridade de dados. A CLiF-Arch, portanto, não é apenas uma solução flexível, mas também uma solução confiável para a persistência de dados FHIR em larga escala.

As principais contribuições deste trabalho podem ser sumarizadas em três frentes. A primeira é arquitetural, com o projeto e a validação da CLiF-Arch, que oferece um modelo específico para a integração de componentes de código aberto a fim de resolver uma lacuna tecnológica conhecida. A segunda contribuição é empírica: a quantificação do impacto de desempenho fornece métricas concretas sobre o *trade-off* entre flexibilidade e latência, oferecendo dados valiosos para que arquitetos de sistema tomem decisões informadas. Por fim, a contribuição técnica materializa-se no protótipo funcional disponibilizado como uma imagem Docker, que serve como prova de conceito e como base para futuras pesquisas na área.

É importante, contudo, reconhecer as limitações deste estudo. A análise de desempenho, embora robusta, foi conduzida em um ambiente simulado em uma única máquina física, e o comportamento em um *cluster* de produção com hardware heterogêneo pode apresentar características distintas. Além disso, o uso de dados sintéticos, embora estruturalmente válidos, pode não capturar todas as nuances de dados clínicos do mundo real. Finalmente, uma análise aprofundada da escalabilidade linear e do *throughput* em *clusters* de grande porte não foi o foco principal, e as afirmações sobre a eficiência em larga escala são baseadas no comportamento esperado da arquitetura.

Com base nos resultados e limitações, diversas direções para trabalhos futuros podem ser exploradas. Uma extensão natural seria a validação da CLiF-Arch em um ambiente clínico real, como em um sistema de EHR hospitalar ou em uma plataforma de monitoramento remoto de pacientes (LEMUS-ZúñIGA et al., 2022; BALDASSANO et al., 2019), para avaliar seu desempenho com dados e cargas de trabalho autênticos. Outra linha de pesquisa seria a integração com tecnologias complementares, como o Apache Spark para processamento distribuído de dados ou o Elasticsearch para aprimorar as capacidades de visualização, em linha com as tendências de análise de *big data* na saúde (RUBí; GONDIM, 2019; KUMAR, 2024). Adicionalmente, a otimização de desempenho, por meio de estratégias como indexação assíncrona ou processamento em lote, representa uma área promissora para mitigar a sobrecarga na latência de escrita, uma abordagem já explorada em sistemas distribuídos para melhorar o *throughput* (LAKSHMAN; MALIK, 2010).

Referências

ABDELOUAHID, R. A. et al. Literature review: Clinical data interoperability models. *Information*, v. 14, n. 7, p. 364, 2023. Citado 5 vezes nas páginas 10, 15, 16, 17 e 18.

ALAPATI, S. R. *Expert Apache Cassandra Administration: Install, configure, optimize, and secure Apache Cassandra databases*. New York: Apress, 2018. ISBN 978-1-4842-3125-8. Citado 2 vezes nas páginas 20 e 37.

AYAZ, M. et al. The fast health interoperability resources (fhir) standard: Systematic literature review of implementations, applications, challenges and opportunities. *JMIR Medical Informatics*, v. 9, p. e21929, 2021. Citado 10 vezes nas páginas 10, 12, 13, 15, 16, 17, 18, 24, 33 e 37.

BALDASSANO, S. N. et al. IRIS: A Modular Platform for Continuous Monitoring and Caretaker Notification in the Intensive Care Unit. *IEEE Journal of Biomedical and Health Informatics*, 2019. Citado 2 vezes nas páginas 37 e 38.

BENDER, D.; SARTIPI, K. HL7 FHIR: An Agile and RESTful approach to healthcare information exchange. In: *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*. [S.l.]: IEEE, 2013. p. 326–331. Citado 3 vezes nas páginas 15, 16 e 17.

BRASIL, G. do. *Guia de Implementação do Projeto de Padronização e Qualificação dos Dados Assistenciais da Saúde Suplementar*. 2024. Accessed: 2024-05-16. Disponível em: <https://fhir-hm.ans.gov.br/>. Citado 2 vezes nas páginas 12 e 17.

BRAUNSTEIN, M. *Health Informatics on FHIR: How HL7's New API is Transforming Healthcare*. Berlin: Springer, 2018. ISBN 978-3-319-93413-6. Citado 5 vezes nas páginas 12, 15, 16, 17 e 24.

CARPENTER, J.; HEWITT, E. *Cassandra: The Definitive Guide, 3rd Edition*. Sebastopol: O'Reilly Media, 2020. ISBN 978-1-098-11516-6. Citado 7 vezes nas páginas 12, 13, 19, 20, 31, 37 e 38.

CHEBOTKO, A.; KASHLEV, A.; LU, S. A big data modeling methodology for apache cassandra. *2015 IEEE International Congress on Big Data*, p. 238–245, 2015. Citado 5 vezes nas páginas 10, 12, 13, 19 e 20.

DRIDI, A. et al. A flexible semantic integration framework for fully-integrated ehr based on fhir standard. *Proceedings of the 12th International Conference on Agents and Artificial Intelligence*, p. 684–691, 2020. Citado 3 vezes nas páginas 10, 15 e 17.

FAMÁ, F.; FARIA, J. N.; PORTUGAL, D. An iot-based interoperable architecture for wireless biomonitoring of patients with sensor patches. *Internet of Things*, v. 19, p. 100547, 2022. Citado 6 vezes nas páginas 10, 11, 15, 16, 18 e 37.

HONG, J.; MORRIS, P.; SEO, J. Interconnected personal health record ecosystem using iot cloud platform and hl7 fhir. *2017 IEEE International Conference on Healthcare Informatics (ICHI)*, p. 362–367, 2017. Citado na página 17.

KIRILOV, N. Capture of real-time data from electronic health records: scenarios and solutions. *mHealth*, v. 10, p. 14, 2024. Citado na página 19.

KIRILOV, N. Scalability of real-time data delivery from electronic health records. *Health and Technology*, v. 15, p. 345–354, 2025. Citado na página 19.

KIRILOV, N.; DUGAS, M. Evaluation of technical approaches for real-time data transfer from electronic health record systems. *Computer Methods and Programs in Biomedicine*, v. 255, p. 108347, 2024. Citado na página 22.

KLEPPMANN, M. *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. Sebastopol: O'Reilly Media, 2017. ISBN 978-1-449-37332-0. Citado na página 10.

KUMAR, S. S. Transforming healthcare data management with nosql: A new era of scalability and efficiency. *International Journal of Science and Research (IJSR)*, v. 13, p. 216–220, 2024. Citado 4 vezes nas páginas 10, 12, 19 e 38.

LAKSHMAN, A.; MALIK, P. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, v. 44, p. 35–40, 2010. Citado 7 vezes nas páginas 10, 12, 19, 22, 24, 31 e 38.

LEMUS-ZúñIGA, L.-G. et al. A proof-of-concept iot system for remote healthcare based on interoperability standards. *Sensors*, v. 22, n. 4, p. 1646, 2022. Citado 4 vezes nas páginas 10, 15, 18 e 38.

LUCENE, A. *Apache Lucene 9.11.1 Documentation*. 2024. Accessed: 2024-05-17. Disponível em: <https://lucene.apache.org/core/9_11_1/index.html>. Citado 2 vezes nas páginas 21 e 24.

MOREIRA, J. et al. Saref4health: Towards iot standard-based ontology-driven cardiac e-health systems. *Applied Ontology*, v. 15, n. 3, p. 385–410, 2020. Citado 2 vezes nas páginas 15 e 16.

PEDRERA-JIMéNEZ, M. et al. Can openehr, iso 13606, and hl7 fhir work together? an agnostic approach for the selection and application of electronic health record standards to the next-generation health data spaces. *Journal of Medical Internet Research*, v. 25, p. e48702, 2023. Citado na página 17.

POINTNER, A.; PRASCHL, C.; KRAUSS, O. Enhancing interoperability of hl7 resources using namespaces in graph databases. In: *12th International Workshop on Innovative Simulation for Healthcare (IWISH 2023)*. [S.l.: s.n.], 2023. p. 001. Citado 3 vezes nas páginas 17, 19 e 33.

RUBí, J. N. S.; GONDIM, P. R. L. Iomt platform for pervasive healthcare data aggregation, processing, and sharing based on onem2m and openehr. *Sensors (Basel, Switzerland)*, v. 19, p. 4283, 2019. Citado 6 vezes nas páginas 12, 15, 17, 18, 37 e 38.

SHARMA, A. *Practical Apache Lucene 8, Uncover the Search Capabilities of Your Application*. New York: Apress, 2020. ISBN 978-1-4842-6344-0. Citado na página 21.

SHOUMIK, F. S. et al. Scalable micro-service based approach to fhir server with golang and no-sql. In: *2017 20th International Conference of Computer and Information Technology (ICCIT)*. [S.l.: s.n.], 2017. Citado 3 vezes nas páginas 11, 19 e 22.

SREEJITH, R.; SENTHIL, S. Smart contract authentication assisted graphmap-based hl7 fhir architecture for interoperable e-healthcare system. *Heliyon*, v. 9, p. e15180, 2023. Citado 3 vezes nas páginas 16, 18 e 19.

WALINJKAR, A. Fhir tools for healthcare interoperability. *Biomed J Sci Tech Res*, v. 9, n. 5, 2018. Citado 3 vezes nas páginas 16, 17 e 18.

WANG, C. et al. Apache iotdb: Time-series database for internet of things. *Proceedings of the VLDB Endowment*, v. 14, n. 2, p. 334–344, 2020. Citado na página 12.

YU, J.-H. et al. Design and implementation of real-time bio signals management system based on hl7 fhir for healthcare services. *International Conference on Platform Technology and Service (PlatCon)*, p. 24–29, 2021. Citado 2 vezes nas páginas 16 e 18.

ZAMPOGNARO, P.; PARAGLIOLA, G.; FALANGA, V. Definition of an fhir-based multiprotocol iot home gateway to support the dynamic plug of new devices within instrumented environments. *Journal of Reliable Intelligent Environments*, v. 7, n. 4, p. 319–331, 2021. Citado na página 16.