Trabalho Final
Documento de arquitetura de software

# 1. Introdução

### 1.1 Finalidade

Este documento possui como objetivo definir os aspectos da Arquitetura do software de transferência de pacientes do trabalho final da disciplina de padrões arquiteturais e é direcionado aos stakeholders do software a ser desenvolvido, tais como: Gerentes do Projeto, Clientes e equipe técnica, possuindo grande foco para os Desenvolvedores e a Equipe de implantação.

### 1.2 Escopo

Este documento se baseia no documento de requisitos do software de transferência de pacientes do trabalho final da disciplina de padrões arquiteturais para definir os atributos de qualidade a serem priorizados, bem como, os estilos arquiteturais que favorecem tais atributos e as representações das visões arquiteturais e seus subprodutos.

### 1.3 Definições, Acrônimos e Abreviações

AAS: Artefato de Arquitetura de Software.
RAS: Requisito de Arquitetura de Software.
Id.: Identificador.
Software: Conjunto de documentações, guias, metodologias, processos, códigos e ferramentas para a solução de um problema.
Sistema: Conjunto de pessoas, softwares, hardwares e outros sistemas para a solução de um problema.
Stakeholder: Indivíduo, grupo ou organização que possua interesse no Sistema.
Visão Arquitetural: Produto resultante da interpretação de um Stakeholder do sistema.
Ponto de Vista Arquitetural: Produto resultante da execução de uma Visão Arquitetural.
Arquitetura de Software: Forma como os componentes são agrupados com o objetivo de construir um software ou sistema.
Trade-Off: Cada arquitetura de software possui seus atributos de qualidade que são favorecidos e desfavorecidos, o trade-off consiste em ter a consciência dessas características para escolher uma arquitetura que favorece os atributos de qualidade priorizados.
Atributos de qualidade: São atributos que impactam diretamente na concepção de um software, são definidos conforme a ISO-IEEE 9126.
Cerne: Funcionalidade-chave / núcleo do software.
Sistema Operacional: Software responsável por gerenciar e abstrair a interação entre o usuário e o hardware ou aplicações externas e o hardware.
UML: Sigla para Linguagem de Modelagem Unificada.
HTML: Sigla para Linguagem de Marcação de Hipertexto.
HTTP: Sigla para Protocolo de Transferência de Hipertexto.
Json: Sigla para Notação de Objetos Javascript.

REST: Sigla para Representational State Transfer.
DAO: Sigla para Data Access Object.
Nó-físico: Termo para representar um componente físico de modo geral, como um navegador ou um banco de dados, por exemplo.

### 1.4 Referências

| Id | Nome do Artefato |
|---|---|
| AAS_1 | Especificacao_Trabalho_Final_PAS_2022_2 |
| AAS_2 | ISO-IEEE 9126 |
| AAS_3 | ISO-IEEE 42010 |
| AAS_4 | Slides Ministrados em Sala |
| AAS_5 | 4+1 View |

### 1.5 Visão Geral

Os próximos tópicos descrevem quais serão os requisitos e restrições utilizados para definir a arquitetura a ser implementada, bem como, quais atributos de qualidades serão priorizados e o porquê da escolha. Quais os padrões arquiteturais serão utilizados conforme os atributos de qualidade selecionados e como funcionará o trade-off entre esses padrões arquiteturais, bem como o porquê da escolha dos padrões arquiteturais. Quais e como as visões arquiteturais serão detalhadas e quais os pontos de vista da arquitetura serão utilizados para descrever as visões.

# 2. Contexto da Arquitetura

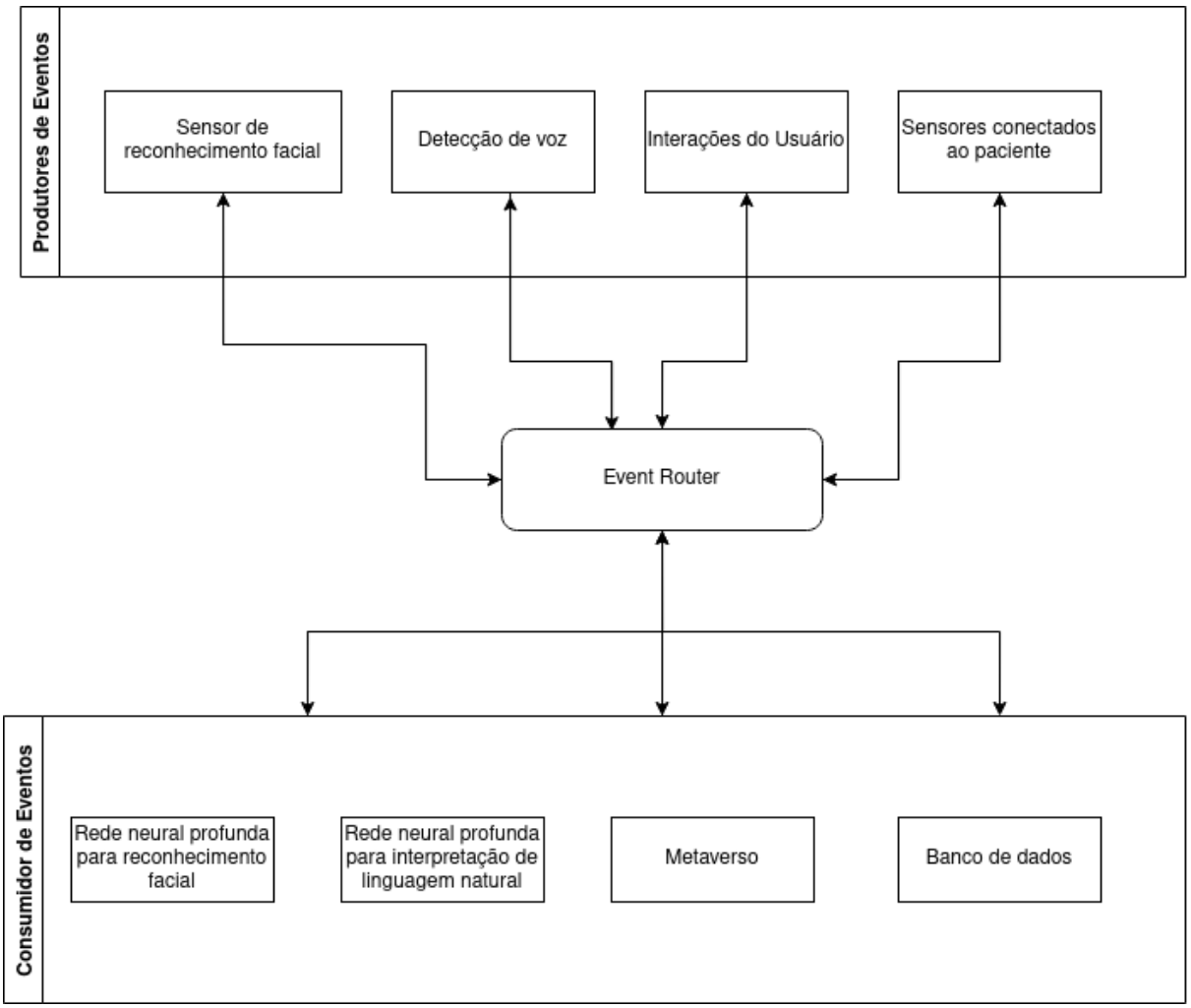
2.1 Funcionalidades e Restrições Arquiteturais

| Id | Tipo | Id. do Documento de Requisitos SUFG |
|---|---|---|
| RAS_1 | Requisitos Não-Funcionais | Deve utilizar WEB 3.0 |
| RAS_2 | Requisitos Não-Funcionais | A transferência de informações deve ser feita de forma segura |
| RAS_3 | Requisitos Não-Funcionais | O sistema deve estar sempre disponível |
| RAS_4 | Requisitos Não-Funcionais | Deve utilizar metaverso |
| RAS_5 | Requisitos Não-Funcionais | Deve utilizar Inteligência |

| | | |
|---|---|---|
| | | Artificial |
| RAS_6 | Requisitos Não-Funcionais | realidade aumentada (AR) e virtual (VR) |
| RAS_7 | Requisitos Funcionais | Deve ser capaz de reconhecer a face do paciente |
| RAS_8 | Requisitos Funcionais | Deve ser capaz de conceber um diagnóstico a partir da fala do paciente acerca dos sintomas. |
| RAS_9 | Requisitos Funcionais | Deve ser capaz de realizar a transferência de dados do paciente |
| RAS_10 | Requisitos Funcionais | Deve ser capaz de criar um gêmeo virtual no metaverso |
| RAS_11 | Requisitos Funcionais | Deve ser capaz de monitorar os dados do avatar a fim de identificar dados do paciente. |

Grande parte dos RAS citados na tabela acima são referentes á requisitos definidas pelo documento de especificação do trabalho final da disciplina. Os RAS serão os responsáveis por guiar as decisões sobre quais estilos arquiteturais serão adequados para favorecer os atributos de qualidade priorizados.
O RAS_1 deixa explícito que o contexto da aplicação é Web 3.0 , enquanto que os outros RAS sendo de contextos  diferentes deixam claro a existência de componentes, um cliente e vários servidores que se comunicam entre si, trocam informações e são responsáveis por aspectos diferentes do sistema. Eles direcionam para a utilização do estilo arquitetural de microsserviços e fica claro, também, a necessidade da utilização de uma arquitetura orientada a eventos, pois a aplicação usa eventos para acionamento e comunicação entre serviços desacoplados e é comum em aplicações modernas criadas com microsserviços. Um evento é uma mudança ou uma atualização no estado, como a fala do paciente sobre seus sintomas, o diagnóstico do médico, uma ação dentro do metaverso que deve refletir no paciente. Eventos podem conter o estado (os sintomas, o diagnóstico ou os dados do paciente) ou podem ser identificadores (um medicamento foi aplicado ao paciente no metaverso o avatar reagiu de determinada maneira, por exemplo).

Conforme definido pelos tópicos anteriores,  a arquitetura do software a ser desenvolvido será uma arquitetura de microsserviços orientada a eventos  e prioriza os atributos de qualidade: segurança, manutenibilidade e portabilidade. Uma representação básica da arquitetura de eventos:




As tecnologias que poderiam  ser utilizadas:

- **Kafka**: para processamento de fluxo, pipelines de dados em tempo real e integração de dados em escala e  dados em tempo real por exemplo, para lidar com os eventos envolvendo os  gêmeos virtuais, o processamento de dados do paciente, o processamento de dados de voz e o processamento de dados de reconhecimento facial.
- **Docker:** para modularizar as aplicações e facilitar a escalabilidade.
- **Kubernetes:** para a orquestração de containers.
- **Apache Spark**: para executar engenharia de dados, ciência de dados e aprendizado de máquina em máquinas ou clusters de nó único tanto dentro do contexto do bot da web 3.0 quanto para redes neurais profundas e interpretação de linguagem natural.
- **TensorFlow:** Criação e treinamento de redes neurais profundas para processamento de linguagem natural.
- **Firebase:** Banco de dados não relacional que pode ser utilizado para armazenar dados.
- **AWS S3:** Serviço de armazenamento de arquivos da amazon, para armazenamento de arquivos referentes ao paciente, como diagnósticos, imagens e vídeos.

- **Postgres:** Banco de dados relacional para garantir a consistência dos dados e armazenar dados.
- **TerraVR:** Ferramenta para desenvolvimento de  aplicações em realidade virtual.

# 3. Implementação

## 3.1 Frontend e Backend

Na implementação do projeto foram feitas as aplicações frontend e backend, desenvolvendo as interfaces em que o médico e ou os enfermeiros iriam interagir para fazer o controle dos pacientes. Nela é possível cadastrar pacientes, colocar suas informações pessoais básicas e gerenciar medicamentos receitados para o paciente cadastrado. Além disso, foi desenvolvido um modelo de banco de dados para armazenar todas essas informações e seus relacionamentos. Podemos ver a interface da aplicação nas imagens a seguir:

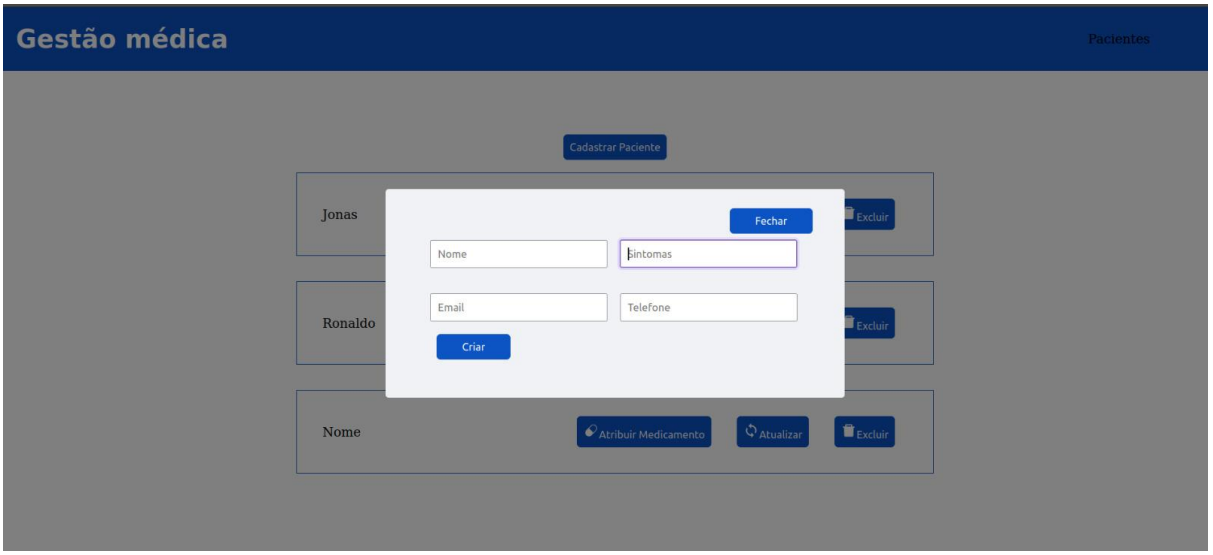



Imagem 1: Cadastro de paciente

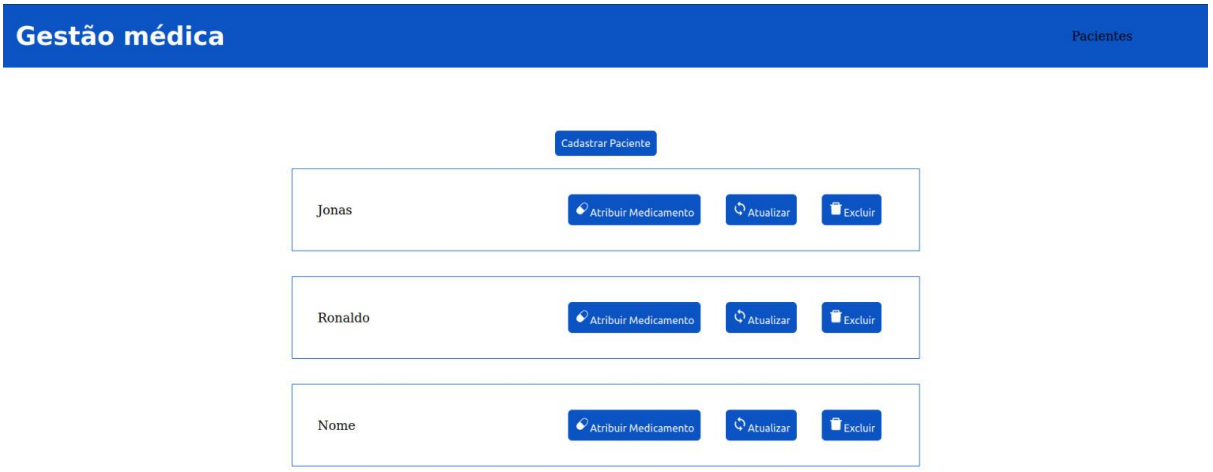



Imagem 2: Listagem de pacientes

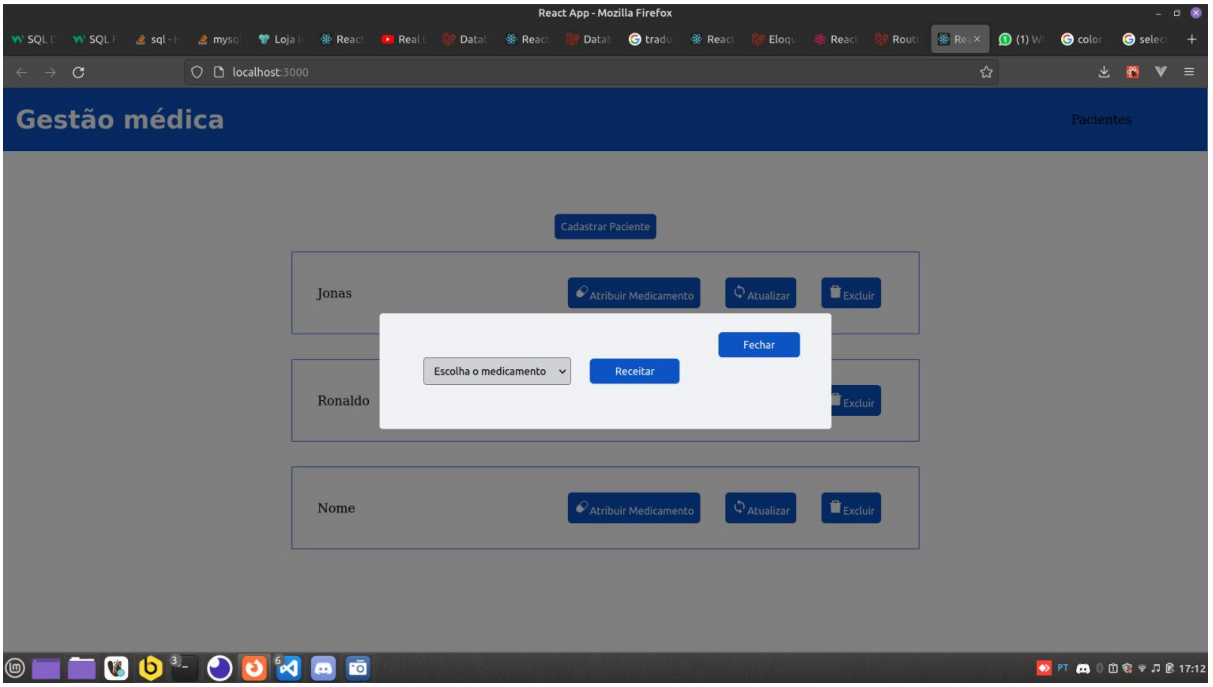



Imagem 3: Receita de medicamentos para um determinado paciente

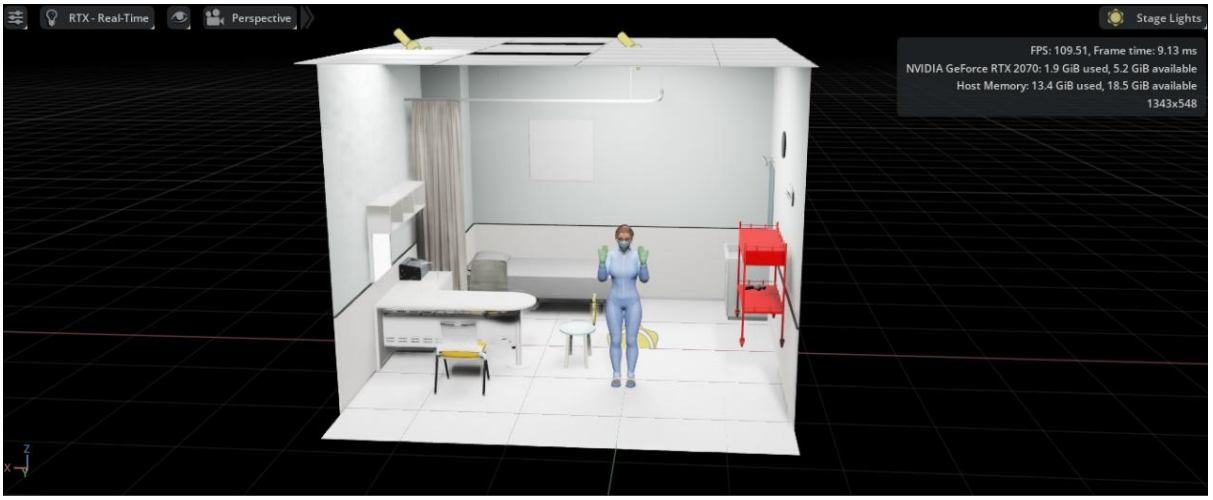

Imagem 4: Alteração dos dados cadastrais de um paciente

3.2 Implementação do metaverso

Para o desenvolvimento inicial do metaverso foi feita a construção de um modelo 3D de um consultório, em que este consultório poderia ser a simulação de um ambiente real, para monitorar os eventos que estão acontecendo por meio de câmeras e sensores no mundo real. Com isso, poderiam também ser implementadas tecnologias disponibilizadas pela própria Nvidia para verificação de equipamentos médicos para cumprimento dos requisitos de higiene, por exemplo. Além disso, com o modelo 3D construído, utilizaremos uma engine como Unity ou Unreal para criar realmente o universo interativo, em que poderiam ser colocados os controles de realidade virtual e aumentada. Já para a parte do envio de dados

em tempo real, seria utilizada a tecnologia SignalR, que cria conexões em WebSocket e permite o envio desses dados para os dispositivos conectados.

A seguir estão as imagens do modelo 3D do consultório médico criado utilizando o Nvidia Omniverse:







## 4. Desafios desafios e implicações éticas antes de sua implementação ampla e generalizada.

Embora o metaverso ofereça muitas possibilidades na medicina, existem também desafios e implicações éticas que precisam ser considerados antes de sua implementação ampla e generalizada. Aqui estão alguns dos principais desafios e preocupações éticas:

1. **Privacidade de dados**: A coleta, armazenamento e análise de dados médicos em grande escala no metaverso pode levantar preocupações sobre a privacidade dos pacientes. É importante garantir que os dados médicos dos pacientes sejam protegidos e que sua privacidade seja respeitada em todas as fases do processo.
2. **Inclusão**: É importante garantir que o metaverso seja acessível para todas as pessoas, independentemente de sua renda, localização geográfica ou capacidade física. Caso contrário, as desigualdades existentes na sociedade poderiam ser perpetuadas ou agravadas na era do metaverso.
3. **Manipulação de dados**: O metaverso pode ser manipulado para distorcer a realidade e afetar a percepção das pessoas. É importante garantir que a verdade e a integridade dos dados sejam preservadas para evitar a disseminação de informações falsas e enganosas.

4. **Direitos autorais**: O metaverso pode ser usado para criar e compartilhar conteúdo, incluindo obras médicas. É importante garantir que os direitos autorais sejam respeitados e que os criadores de conteúdo recebam o devido reconhecimento e compensação pelo trabalho.
5. **Responsabilidade**: É importante estabelecer clara responsabilidade pelos resultados e possíveis erros no metaverso, especialmente na medicina, onde as consequências de erros podem ser graves.

Esses são apenas alguns dos desafios e preocupações éticas que precisam ser considerados antes da implementação ampla e generalizada do metaverso na medicina. É importante continuar a monitorar essas questões à medida que a tecnologia evolui e a sociedade se adapta à era do metaverso.

# 5. Serviços de mensageria como kafka

Os serviços de mensageria, como o **Apache Kafka**, podem ser úteis para o metaverso, especialmente quando se trata de gerenciar grandes volumes de dados e eventos em tempo real. Algumas formas específicas em que o Apache Kafka pode ser usado no metaverso incluem:

1. **Gerenciamento de dados**: O Apache Kafka pode ser usado para coletar, armazenar e processar dados de sensores e outras fontes em tempo real, como informações sobre as interações dos usuários no metaverso.
2. **Integração de sistemas**: O Apache Kafka pode ser usado para integrar diferentes sistemas no metaverso, incluindo aplicativos, jogos e serviços de nuvem, permitindo que as informações fluam livremente entre eles.
3. **Gerenciamento de eventos**: O Apache Kafka pode ser usado para gerenciar eventos importantes no metaverso, como atualizações de conteúdo, mudanças de estado e interações dos usuários.

No entanto, existem alguns desafios que precisam ser superados ao implementar o Apache Kafka no metaverso. Aqui estão alguns dos principais desafios:

1. **Escalabilidade**: O metaverso pode ter milhões ou até bilhões de usuários gerando grandes volumes de dados em tempo real. É importante garantir que o Apache Kafka seja escalável e possa lidar com esses volumes de dados sem comprometer o desempenho.
2. **Latência**: A latência é crucial no metaverso, pois as interações dos usuários precisam ser processadas em tempo real. É importante garantir que o Apache Kafka possa processar os dados rapidamente e sem atrasos significativos.
3. **Segurança**: A segurança dos dados é crucial no metaverso, especialmente quando se trata de informações sensíveis, como dados médicos. É importante garantir que o Apache Kafka seja seguro e proteja adequadamente os dados.
4. **Integração**: O Apache Kafka precisa ser integrado com outros sistemas no metaverso para que as informações possam ser compartilhadas e processadas de forma eficiente. É importante garantir que essa integração seja suave e sem interrupções.

Em resumo, o Apache Kafka pode ser uma ferramenta valiosa no metaverso, mas também precisa ser implementado corretamente para superar os desafios e garantir o sucesso.