UNIVERSIDADE FEDERAL DO OESTE DO PARÁ

INSTITUTO DE ENGENHARIA E GEOCIÊNCIAS

SISTEMAS DE INFORMAÇÃO

LUCAS SÁ NASCIMENTO

ARQUITETURA ORIENTADA A EVENTOS: MAPEAMENTO SISTEMÁTICO E ESTUDO DE CASO DE APLICAÇÃO PRÁTICA

SANTARÉM - PA

2024

LUCAS SÁ NASCIMENTO

ARQUITETURA ORIENTADA A EVENTOS: MAPEAMENTO SISTEMÁTICO E ESTUDO DE CASO DE APLICAÇÃO PRÁTICA

Trabalho de Conclusão de Curso apresentado ao programa da Computação, para obtenção do grau de Bacharel em Sistemas de Informação; Universidade Federal Do Oeste Do Pará, Instituto de Engenharia e Geociências.
Orientador(a): Rennan José Maia da Silva.

SANTARÉM - PA

2024

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO OESTE DO PARÁ
COORD. DO CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ATA Nº 1 / 2025 - CBSI (11.01.09.14)

Nº do Protocolo: 23204.002248/2025-81

Santarém-PA, 12 de fevereiro de 2025.

ATA DE DEFESA

Ao décimo sétimo dia do mês de outubro de 2024, o discente LUCAS SÁ NASCIMENTO, regularmente matriculado no curso de Bacharelado em Sistemas de Informação da Universidade Federal do Oeste do Pará, defendeu o Trabalho de Conclusão de Curso (TCC) intitulado: ARQUITETURA ORIENTADA A EVENTOS: MAPEAMENTO SISTEMÁTICO DA LITERATURA, como requisito para a obtenção do título de Bacharel em Sistemas de Informação. Após a defesa pública, a Banca Examinadora, constituída pelos professores Me. Rennan José Maia da Silva (Presidente e Orientador); Me. Bruno Almeida Silva (Membro interno); e Me. Roberto Pereira do Nascimento (Membro interno) reuniu-se para avaliar e atribuir a aprovação com nota média de 8,1 do TCC.

(Assinado digitalmente em 17/02/2025 08:22)
BRUNO ALMEIDA DA SILVA
IEG (11.01.09)
Matrícula: ###870#6

(Assinado digitalmente em 17/02/2025 16:14)
RENNAN JOSE MAIA DA SILVA
IEG (11.01.09)
Matrícula: ###044#4

(Assinado digitalmente em 24/02/2025 16:39)
ROBERTO PEREIRA DO NASCIMENTO
IEG (11.01.09)
Matrícula: ###612#7

Visualize o documento original em https://sipac.ufopa.edu.br/public/documentos/index.jsp informando seu número: **1**, ano: **2025**, tipo: **ATA**, data de emissão: **12/02/2025** e o código de verificação: **d4e5c75948**

AGRADECIMENTO

Em primeiro lugar, gostaria de agradecer meus pais, Luis Da Silva Nascimento e Joana Sá Nascimento por todo o suporte durante toda minha vida. Os dois, fizeram "o pouco se transformar em muito" e deram-me, apesar de toda dificuldade, todo o alicerce possível. Ademais, gostaria de agradecer à Universidade Federal Do Oeste do Pará, juntamente com todo corpo docente, por fornecer uma infraestrutura de qualidade e um ambiente inclusivo, diverso e propício à aprendizagem. Agradeço também a minha companheira Julielly Bembe, por todo apoio e parceria nesses últimos anos. Sou grato também por toda orientação e suporte prestado pelo professor Rennan Jose Maia da Silva durante esses anos na academia. Por último, agradeço aos meus avós, José França e Maria Ivonete. Eles foram essenciais na minha construção como indivíduo!

RESUMO

De acordo com estudos recentes, a arquitetura baseada em eventos (EDA) vem sendo amplamente adotada pela indústria de software nos últimos anos, um fenômeno potencializado pela crescente utilização da arquitetura de microsserviços no desenvolvimento de software. Nesse contexto, a EDA é utilizada como um mecanismo de comunicação entre os componentes de sistemas, baseada na emissão de eventos. Um exemplo disso é a empresa Spotify, que adota essa abordagem para a entrega de eventos. Diante disso, com o objetivo de explorar os cenários de aplicabilidade da EDA, bem como as ferramentas empregadas em sua implementação, o presente estudo realiza um mapeamento sistemático da literatura, estruturado em três etapas: planejamento, condução e relatório. Para responder às questões de pesquisa, foram analisados 200 trabalhos, identificando que a arquitetura pode ser aplicada em cenários que demandam processamento em tempo real, sistemas que exigem escalabilidade ou modularidade, integração de sistemas distribuídos e Internet das Coisas (IoT), além do gerenciamento de fluxos de trabalho complexos, simulações e modelagem em ambientes dinâmicos. Adicionalmente, o estudo aponta que as três ferramentas mais recorrentes na implementação da EDA são, respectivamente, Apache Kafka, RabbitMQ e Mosquitto. Como contribuição prática, o trabalho inclui um projeto *open-source* contendo tutoriais que demonstram exemplos de uso dessas ferramentas na aplicação da EDA, com o intuito de auxiliar estudantes da disciplina de sistemas distribuídos na implementação de sistemas utilizando esse tipo de arquitetura.

Palavras-chave: Event-driven architecture; EDA; Arquitetura orientada a eventos; Arquitetura de Software.

ABSTRACT

According to recent studies, event-driven architecture (EDA) has been widely adopted by the software industry in recent years, a phenomenon boosted by the growing use of microservice architecture in software development. In this context, EDA is used as a communication mechanism between system components, based on the emission of events. An example of this is the company Spotify, which uses this approach to deliver events. In view of this, with the aim of exploring the applicability scenarios of EDA, as well as the tools used in its implementation, this study carries out a systematic mapping of the literature, structured in three stages: planning, conducting and reporting. To answer the research questions, 200 papers were analyzed, identifying that the architecture can be applied in scenarios that require real-time processing, systems that require scalability or modularity, integration of distributed systems and the Internet of Things (IoT), as well as management of complex workflows, simulations and modeling in dynamic environments. In addition, the study shows that the three most common tools for implementing EDA are Apache Kafka, RabbitMQ and Mosquitto, respectively. As a practical contribution, the work includes an open-source project containing tutorials that demonstrate examples of the use of these tools in the application of EDA, with the aim of helping students in the field.

Keywords: Event-driven architecture; EDA; Arquitetura orientada a eventos; Arquitetura de Software.

SUMÁRIO

TCC

Arquitetura Orientada a Eventos: Mapeamento Sistemático e Estudo de Caso de Aplicação Prática

Lucas Sá Nascimento[1] & Rennan José Maia da Silva[2]

[1] Bacharelado em Sistemas de Informação/ UFOPA; lucassnascimento20@gmail.com
[2] Programa de Computação / UFOPA; rennan.silva@ufopa.edu.br

Resumo: De acordo com estudos recentes, a arquitetura baseada em eventos (EDA) vem sendo amplamente adotada pela indústria de software nos últimos anos, um fenômeno potencializado pela crescente utilização da arquitetura de microsserviços no desenvolvimento de software. Nesse contexto, a EDA é utilizada como um mecanismo de comunicação entre os componentes de sistemas, baseada na emissão de eventos. Um exemplo disso é a empresa Spotify, que adota essa abordagem para a entrega de eventos. Diante disso, com o objetivo de explorar os cenários de aplicabilidade da EDA, bem como as ferramentas empregadas em sua implementação, o presente estudo realiza um mapeamento sistemático da literatura, estruturado em três etapas: planejamento, condução e relatório. Para responder às questões de pesquisa, foram analisados 200 trabalhos, identificando que a arquitetura pode ser aplicada em cenários que demandam processamento em tempo real, sistemas que exigem escalabilidade ou modularidade, integração de sistemas distribuídos e Internet das Coisas (IoT), além do gerenciamento de fluxos de trabalho complexos, simulações e modelagem em ambientes dinâmicos. Adicionalmente, o estudo aponta que as três ferramentas mais recorrentes na implementação da EDA são, respectivamente, Apache Kafka, RabbitMQ e Mosquitto. Como contribuição prática, o trabalho inclui um projeto *open-source* contendo tutoriais que demonstram exemplos de uso dessas ferramentas na aplicação da EDA, com o intuito de auxiliar estudantes da disciplina de sistemas distribuídos na implementação de sistemas utilizando esse tipo de arquitetura.

Palavras-chave: Event-driven architecture; EDA; Arquitetura orientada a eventos; Arquitetura de Software.

1. Introdução

Nos últimos anos, a arquitetura de microsserviços (MSA) caminha em passos largos em direção à aceitação por parte da indústria de software, uma vez que ela faz frente a arquitetura orientada a serviços (SOA), pois baseia-se na fragmentação de sistemas que anteriormente possuíam grandes proporções, com inúmeras regras de negócio acopladas [77]. Nesse sentido, segundo Lercher et al. [77], a MSA substituiu os modelos compartilhados de serviços por modelos de domínio independentes, os quais utilizam para comunicação entre microsserviços interfaces de programação de aplicações (APIs) e *brokers* de mensagens. À vista disso, há trabalhos que permitem observar a adoção e utilização dessas arquitetura por *bigtechs* como: Amazon, Netflix e ThoughtWorks [1].

Para Oliveira et al. [2], o desenvolvimento de aplicações empresariais tem ocorrido em ambientes totalmente instáveis, de forma que sua normalidade contempla-se em meios com complexidade elevada e mudanças aceleradas. Nesse sentido, observa-se âmbitos de desenvolvimento cada vez mais instáveis e voláteis, onde exige-se flexibilidade elevada e rápida adaptabilidade, visto a ciclos de desenvolvimentos e prazos de entrega menores [5], onde, faz-se necessário adotar como prática de desenvolvimento a modularização de sistemas. Logo, aplicações com baixo nível de acoplamento, visando diminuir o custo de manutenção, tem sido a busca da indústria visto aos cenários desafiadores [7].

Na opinião de Lazzari et al. [3], por esses e outros motivos, equipes de desenvolvimento de software buscam implantar estilos arquitetônicos, *patterns* e princípios de *software design* a fim de manter a estabilidade, seja em processo de desenvolvimento, seja em processo de manutenção. Segundo ele, a indústria de *software* tem adotado amplamente o uso de eventos como alternativa ao desenvolvimento modular em arquiteturas, como por exemplo, a empresa Spotify que introduz a entrega de eventos de streaming. Nesse sentido, Laigner et al. [4], observou em seu estudo empírico, que a adoção de uma arquitetura orientada a eventos, trouxe melhorias na manutenção de um sistema, que há anos passava por manutenção.

Nesse sentido, Rahmatulloh et al. [33], sinaliza que a *event-driven architecture* (EDA) está sendo amplamente implantada em diversos domínios, uma vez que ela contribui diretamente para construção de sistemas distribuídos proporcionando alta flexibilidade e concorrência. Dessa forma, considerando a ausência de estudos na literatura que investiguem os contextos de aplicabilidade da arquitetura orientada a eventos e as ferramentas empregadas para esse fim, o presente trabalho realiza um mapeamento sistemático fundamentado no protocolo proposto por Kitchenham et al. [79]. Para isso, foram analisados duzentos estudos provenientes de três bases de dados distintas, com o objetivo de mapear os contextos relacionados à aplicação da EDA. Ademais, busca-se identificar e mapear as ferramentas utilizadas na implementação dessa arquitetura e a elaboração de um projeto *open-source* contendo exemplos de implementações a fim de auxiliar a introdução ao tema.

Os produtos obtidos neste trabalho são valiosos tanto para pesquisadores quanto para desenvolvedores e arquitetos de software interessados em explorar, compreender e aplicar os conceitos da arquitetura orientada a eventos. Ao mapear os cenários e contextos de aplicação da EDA, o presente trabalho justifica-se plenamente pela sua relevância prática, impacto acadêmico e contribuição significativa para a formação de profissionais capacitados a desenvolver e gerenciar sistemas distribuídos modernos e eficientes.

A seguir, objetivando introduzir conceitos essenciais sobre a *event-driven architecture*, será apresentada uma breve descrição, baseada na literatura, dos conceitos que permeiam a EDA.

1.1. Event-driven Architecture EDA

Arquitetura orientada a eventos, *Event-driven architecture* (EDA), ou arquitetura baseada em eventos é um *pattern* de comum uso em sistemas distribuídos, principalmente entre microsserviços, para comunicação de forma assíncrona e desacoplada [5][33]. A EDA possibilita que distintos serviços ou componentes de softwares possam enviar e receber eventos de forma independente [33]. Para Richards et al. [8], pode ser utilizado por aplicações grandes, cujo objetivo é fornecer alto grau de escalabilidade, no entanto, também pode ser aplicada em aplicações de menor porte.

Segundo Trabelsi et al. [31], em EDA, um evento pode representar duas categorias: profissionais ou técnicos. Eventos técnicos apontam eventos de infraestrutura ou de ambiente, por sua vez, segundo o autor, todos os outros tipos de eventos são considerados de negócio pois estão diretamente associados ao domínio. Esses eventos são utilizados para transmitir informações, mudanças de estados e até mesmo condições significativas entre diferentes componentes ou sistemas [34] [69].

Para auxiliar na compreensão, na Figura 1 é apresentada uma modelagem proposta para um sistema fictício cujo objetivo é organizar e disponibilizar os resultados de jogos de futebol. A arquitetura adotada baseia-se no padrão orientado a eventos, o qual utiliza eventos para promover a comunicação assíncrona entre diferentes componentes de software. O processo tem início a partir da atualização de uma partida, realizada através de um agente por meio de aplicação web, que emite eventos para um barramento, informando a ocorrência de uma atualização na partida em questão. O barramento, ou canal de eventos, por sua vez, armazena e notifica seus consumidores, informando que há um novo evento disponível para processamento. O evento, então, é consumido por diferentes componentes que realizam ações conforme suas respectivas regras de domínio. No exemplo da Figura 1, há dois consumidores responsáveis, respectivamente, por: Informar os usuários sobre acontecimentos relacionados à competição; Realizar cálculos estatísticos sobre a partida, com o intuito de disponibilizá-los em uma plataforma web.

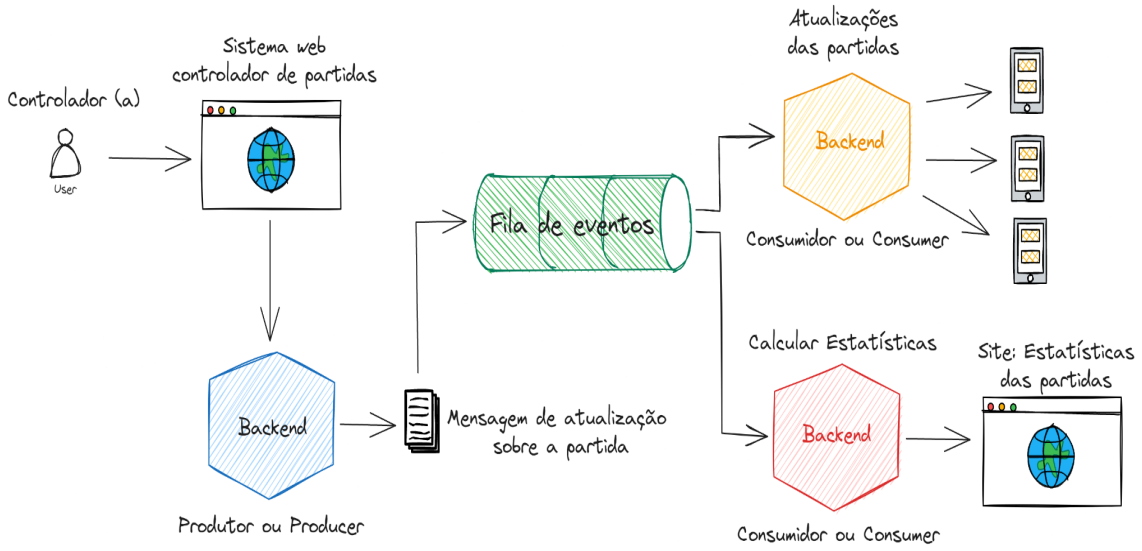

Figura 1. Arquitetura para sistema de atualização em tempo real dos resultados de partidas de futebol, baseado em EDA. Fonte: Elaborado pelo autor.

1.2. Componentes Principais da EDA

A EDA possui componentes fundamentais para seu funcionamento, para Richards et al. [8], a arquitetura é composta por componentes desacoplados, com objetivo de receber e processar eventos de forma assíncrona. Na literatura, observa-se duas principais topologias quando fala-se sobre *event-driven architecture*, *Mediator* e *Broker* [8][69].

O *Mediator* é comumente utilizado com responsabilidades de orquestração de eventos, para isso utiliza-se de quatro componentes: *event queues*, *event channels*, *event mediator* e *event processor.*

> "O fluxo do evento começa com um cliente enviando um evento para uma fila de eventos, que é usada para transportar o evento até o *mediator* de eventos. O *mediator* de eventos recebe o evento inicial e o orquestra enviando eventos assíncronos adicionais para os canais de eventos a fim de executar cada etapa do processo. Processadores de eventos, que monitoram os canais de eventos, recebem o evento do *mediator* de eventos e executam a lógica de negócios específica para processar o evento." [8].

A seguir, a figura 2 apresenta a topologia do componente *Mediator.*

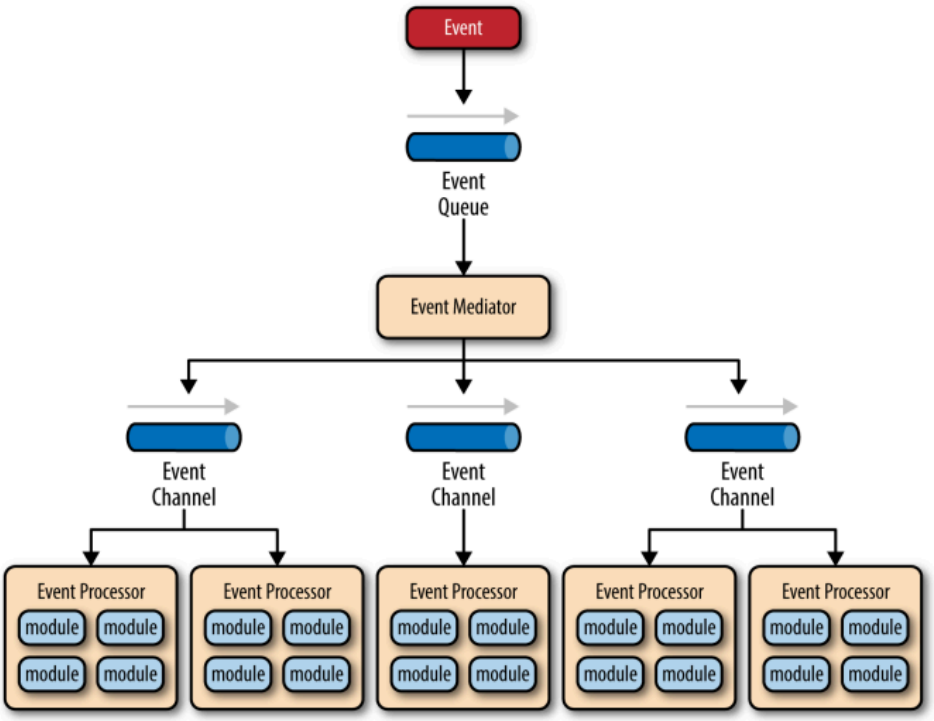

Figura 2. Topologia do componente *Mediator*. Fonte: Figura apresentada no livro "Software Architecture Patterns: Understanding Common Architecture Patterns and When to Use Them" de 2015 por Mark Richards.

O *Broker*, por sua vez, é utilizado em casos que não necessita-se de um mediador realizando orquestração, ou seja, o fluxo de eventos é distribuído a partir dos próprios componentes de processamento do sistema [8][69].

> "A topologia de *broker* difere da topologia do *mediator* porque não há um mediador central de eventos; em vez disso, o fluxo de mensagens é distribuído entre os componentes processadores de eventos de maneira encadeada por meio de um *broker* de mensagens leve (por exemplo, ActiveMQ, HornetQ, etc.). Essa topologia é útil quando você tem um fluxo de processamento de eventos relativamente simples e não deseja, ou não precisa de orquestração central de eventos" [8].

A seguir, a figura 3, apresenta a topologia do componente *Broker*.

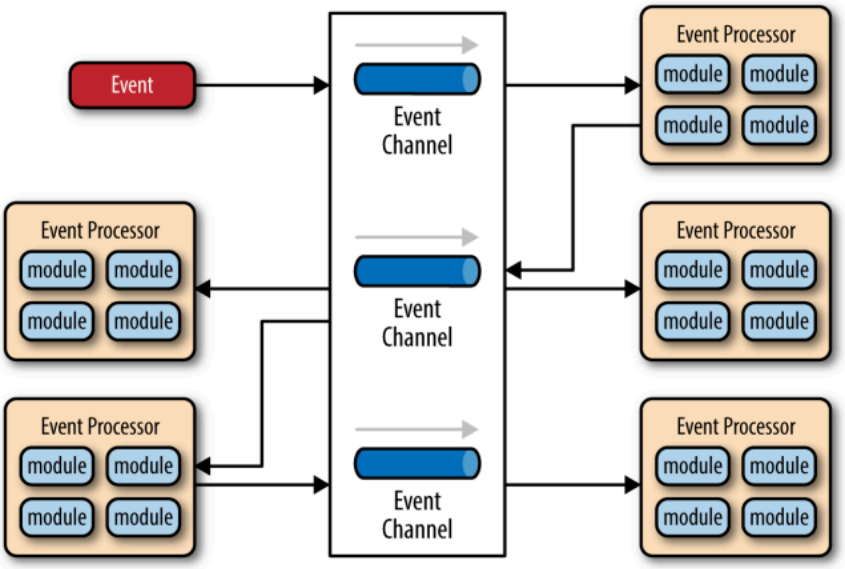

Figura 3. Topologia do componente *Broker*. Fonte: Figura apresentada no livro "Software Architecture Patterns: Understanding Common Architecture Patterns and When to Use Them" de 2015 por Mark Richards.

1.2.1. Produtores de Eventos

Os produtores de eventos são sistemas ou componentes de software responsáveis por identificar mudanças de estados ou comportamentos e realizar a emissão desses eventos através do canal utilizado. Um único produtor, ou *producer*, comumente encontrado assim na literatura, pode possuir diversos consumidores, os quais irão efetuar ações baseadas nos eventos publicados [78].

1.2.2. Consumidores de Eventos

Os consumidores ou *consumers* de eventos são componentes ou serviços autônomos, carregados por lógica de negócio que processam eventos publicados pelo seus respectivos produtores. Esses componentes podem realizar inúmeras atividades desencadeadas a partir de um único evento recebido, dependendo exclusivamente de sua regra de negócio [8]. O objetivo desse componente é manter-se isolado visto aos seus produtores, uma vez que assim, é possível gerenciar cada um dos componentes isoladamente [6].

1.2.3. Canais de Eventos

Os canais de eventos são os meios pelos quais o *Mediator* transmite eventos de forma assíncrona dos produtores para os consumidores, ou diferentes agentes do sistema. Eles podem ser implementados usando tecnologias como filas de mensagens, tópicos de publicação ou assinatura, ou *message brokers* dedicados [8][12][31][38].

Uma vez apresentados os conceitos e componentes elementares da arquitetura orientada a eventos, esse trabalho também desenvolve um projeto[1] *Open-source* disponibilizado no github[2]. O repositório público, apresenta tutoriais introdutórios, os quais utilizam a linguagem Golang[3] para implementar as ferramentas Apache Kafka[4] e RabbitMQ[5]. Estes tutoriais visam auxiliar discentes integrantes da disciplina de sistemas distribuídos, ministrada na Universidade Federal do Oeste do Pará, pesquisadores e desenvolvedores iniciantes interessados no contexto comunicação entre serviços.

1.3. Objetivos

1.3.1. Objetivo Geral

O presente estudo tem como objetivo realizar um mapeamento sistemático em volta da arquitetura orientada a eventos, compreender os contextos apropriados para sua aplicação e introduzir as ferramentas encontradas que auxiliam a aplicação da EDA. Ademais, o estudo apresenta um estudo de caso juntamente com um repositório *open-source* contendo implementações de algumas ferramentas levantadas através do mapeamento. O intuito do material é prestar apoio à introdução da arquitetura de sistemas, aos discentes da disciplina de sistemas distribuídos.

1.3.2. Objetivos Específicos

Compreender os cenários apropriados para aplicação ou utilização de uma arquitetura orientada a eventos.

Descrever as principais ferramentas utilizadas na aplicação de arquitetura orientada a eventos.

Construir um estudo de caso e elaborar de material de apoio contendo exemplos de implementações utilizando ferramentas usadas para a estabelecer comunicação em EDA, destinado aos discentes da disciplina de sistemas distribuídos da Universidade Federal do Oeste do Pará com o objetivo de facilitar a introdução ao tema.

1.3. Justificativa

De acordo com Lazzari et al. [3], a arquitetura orientada a eventos está sendo incorporada pela indústria. Seu trabalho aponta que empresas têm adotado a abordagem de eventos em suas arquiteturas, sua aplicação facilita o processo de

---

[1] https://github.com/lucassanascimento/tcc-event-driven-architecture
[2] https://github.com
[3] https://go.dev/
[4] https://kafka.apache.org/
[5] https://www.rabbitmq.com/

divisão do software em módulos e ajuda na evolução e manutenção das aplicações [3].

Num mundo em que dados estão sendo gerados em ritmos elevadíssimos nos últimos anos, fator que tem se intensificado com uso dos smartphones e das redes sociais [4], Laigner et al. [4], aponta que o uso de arquitetura orientada a eventos, juntamente com a arquitetura de microserviços, apresenta-se como paradigma atraente para o desenvolvimento de software.

O crescimento do volume de dados processados não é exclusividade das aplicações web. Também é possível visualizar tal crescimento no contexto de internet das coisas e inteligência artificial [5]. No estudo de Khriji et al. [5], é apresentado que o número de dispositivos IoT está atualmente no auge das expectativas infladas, prevendo um aumento de 21% entre 2016 e 2022 para cerca de 18 bilhões. Segundo o mesmo estudo, os dispositivos IoT com conexões celulares são projetados para atingir 1,5 bilhão em 2022, ou cerca de 70% de uma ampla categoria de IoT. Khriji et al. [5], também apresenta em seu trabalho que o processamento de eventos em tempo real através de *streaming*, é capaz de considerar restrições de tempo, afim de entregar resultados precisos em baixa latência. Processos esses que são possíveis uma vez que aplicado padrões como: *event-driven*, *publish/subscribe systems* e *event-stream processing* [5].

Portanto, uma vez que ratifica-se a importância que circunda o tema, arquitetura orientada a eventos, esse estudo oferece um mapeamento sistemático com o objetivo de auxiliar profissionais e pesquisadores na compreensão de possíveis cenários para seleção e aplicação da arquitetura baseada em eventos bem como as ferramentas utilizadas para tal aplicação. Para isso, foram definidas as questões de pesquisa: (QP1) quando implantar uma arquitetura orientada a eventos?; e (QP2) quais as principais ferramentas utilizadas em arquiteturas orientadas a eventos? Dessa forma, visualiza-se que a exploração dos conceitos fundamentais contribuirá para melhor compreensão do funcionamento desses instrumentos a fim de facilitar a comunicação e colaboração em projetos que utilizem arquitetura orientada a eventos.

1.4. Organização do Trabalho

Este trabalho está organizado em seis capítulos, os quais, cada um deles apresenta ao leitor informações pertinentes à compreensão do mesmo. Na primeira sessão, nomeada de introdução, apresenta-se um breve contexto sobre a EDA e introduz ao leitor conceitos essenciais para o entendimento da arquitetura baseada em eventos. Nessa sessão, também são apresentados os objetivos gerais e específicos do estudo. Na segunda seção, serão apresentados os trabalhos relacionados que contribuíram diretamente e indiretamente para a realização deste trabalho. Logo, será apresentada uma sequência de estudos que guiarão o leitor a trabalhos que partilham o tema presente neste estudo. Na terceira seção é feita a

descrição do processo metodológico utilizado neste estudo, bem como todo o protocolo a qual define o mapeamento sistemático. Na Seção 4, são apresentados os resultados obtidos a partir da implementação do protocolo apresentado na seção 3. Na seção 5, é apresentado um estudo de caso contendo implementações de algumas ferramentas obtidas através do mapeamento realizado. Por fim, na seção 6 são apresentadas as considerações finais do trabalho, assim como os possíveis trabalhos futuros. Ademais, há também a seção de referências bibliográficas da literatura.

2. Trabalhos Relacionados

A fim de auxiliar no entendimento do trabalho, foram buscados estudos cujo tema permeia arquitetura orientada a eventos, bem como mapeamentos e revisões sistemáticas da literatura. A seguir, serão apresentados os trabalhos que contribuíram para a construção do presente estudo.

Comumente, alguns pesquisadores têm explorado a arquitetura orientada a eventos com o objetivo de entendê-la e utilizá-la. Nesse sentido, Trabelsi et al. [31] apresenta um estudo exploratório com objetivo de discutir a *event-driven architecture* e seus elementos bem como compreender possíveis lacunas entre a academia e a indústria. Para isso, o autor indaga em seu estudo quais são os elementos da EDA de acordo com a academia e se esses elementos podem ser encontrados na indústria. Para tal, o autor realiza uma revisão da literatura focada em estudos que discutem a EDA em contextos de software e arquitetura, excluindo estudos que abordam EDA em contextos não relacionados a software, como circuitos elétricos, e estudos que, embora mencionem a EDA, não descrevem seus padrões arquitetônicos. Desse modo, oito estudos que atendem a esses critérios e fornecem uma descrição dos elementos e interações da EDA são selecionados. De modo a responder às questões de pesquisa, Trabelsi et al. [31] analisa três plataformas industriais amplamente utilizadas no mercado: Apache Kafka, AWS EventBridge[6] e GCP Eventarc[7]. Ele investiga como cada plataforma implementa os elementos da EDA definidos na literatura e identifica as divergências e lacunas entre a teoria e a prática industrial. Dessa forma, constata-se que não há um entendimento comum em volta da definição da arquitetura, bem como um método disseminado de como deve-se conceber tal. O autor também vislumbra dois principais cenários de aplicação: processamento de dados em tempo real e comunicação reativa entre componentes de software. No entanto, Trabelsi et al. [31] explicita em seu trabalho que há limitação em relação à quantidade de estudos selecionados para extração dos elementos da EDA, uma vez que um mapeamento sistemático não foi realizado, fato que implica diretamente na busca de trabalhos.

Roda et al. [80], por sua vez, realiza um mapeamento sistemático para analisar como a arquitetura de software é usada em sistemas sensíveis ao contexto. O

[6] https://aws.amazon.com/pt/eventbridge/
[7] https://cloud.google.com/eventarc/docs?hl=pt-br

trabalho discute a importância de ambientes inteligentes em sistemas de software, onde a interação do usuário ocorre de maneira transparente. Nesse contexto, sistemas sensíveis ao contexto (*context-aware systems*) se destacam por oferecer flexibilidade, adaptabilidade e a capacidade de agir autonomamente em nome dos usuários. Desse modo, o mapeamento tem como objetivo revisar como os conceitos de arquitetura de *software* são aplicados no desenvolvimento de sistemas sensíveis ao contexto, e como essas arquiteturas lidam com a adaptação ao contexto, considerando as dimensões de usuário, plataforma e ambiente. Para isso, o estudo determina quatro questões de pesquisa. Desta maneira, o estudo identificou que, embora existam diversos conceitos de Arquitetura de Software aplicados em *context-aware systems*, como designs, frameworks e padrões, não há uma abordagem padronizada clara para sua arquitetura. Ademais, as arquiteturas existentes consideram as dimensões de usuário, plataforma e ambiente, mas de maneira ainda fragmentada, revelando lacunas significativas na integração dessas dimensões. Além disso, a maturidade das propostas e das avaliações de adaptações contextuais é limitada, com poucas avaliações empíricas robustas, destacando a necessidade de mais pesquisas e desenvolvimento na área para alcançar uma maior padronização e qualidade nos sistemas sensíveis ao contexto.

Em outro mapeamento sistemático realizado por Waseem et al. [81], analisa-se aplicações da Arquitetura de Microsserviços (MSA) no contexto do DevOps, uma abordagem que combina práticas de desenvolvimento e operações para acelerar a entrega de software. O estudo realiza um mapeamento sistemático da literatura para identificar, analisar e classificar a pesquisa existente sobre MSA em DevOps. Assim como esse trabalho, o estudo de Waseem et al. [81], realiza o mapeamento sistemático em três etapas: planejamento, coleta e análise de dados, e documentação dos resultados. Foram utilizadas estratégias como pesquisa em bases de dados e técnica de "*snowballing*", que realiza inspeção de referências dos estudos primários para identificar estudos relevantes. A pesquisa foi dividida em duas fases: a busca principal em bases de dados e uma fase complementar usando "*snowballing*". Após essa busca, foram identificados 45 estudos primários relevantes. Os resultados incluem uma classificação dos temas de pesquisa, problemas enfrentados, soluções propostas, desafios, métodos de descrição, padrões de MSA, atributos de qualidade, ferramentas e domínios de aplicação. O estudo revela que há um interesse crescente na adoção de MSA em DevOps, com um aumento significativo no número de publicações entre 2015 e 2018. As principais contribuições são a identificação de temas e subtemas de pesquisa, bem como a discussão dos problemas e soluções encontrados na implementação de arquitetura de Microsserviços em DevOps.

Apesar dos estudos citados anteriormente terem apresentado resultados relevantes ao contexto de arquiteturas de software e até mesmo o trabalho de Trabelsi et al. [31] que aborda *event-driven architecture*, não foi encontrado nenhum mapeamento sistemático cujo aplicação tem como foco na arquitetura orientada a

eventos, mais especialmente no que tange os cenários de sua aplicação bem como suas ferramentas.

3. Procedimentos Metodológicos

O procedimento metodológico selecionado para abordar as questões de pesquisa e fundamentar o estudo foi o mapeamento sistemático da literatura. Essa metodologia permite aos pesquisadores executar levantamentos na literatura em torno de um tema, com objetivo de responder questões de pesquisa evitando processos subjetivos suscetíveis à viés, uma vez que o trabalho deve basear-se em um protocolo previamente definido [9]. Dessa forma, este trabalho adotou o protocolo apresentado por Kitchenham et al. [79], o qual organiza o mapeamento em três etapas principais: planejamento, condução e relatório. Na figura 4, é apresentada a representação gráfica da organização aplicada no mapeamento sistemático do presente estudo.

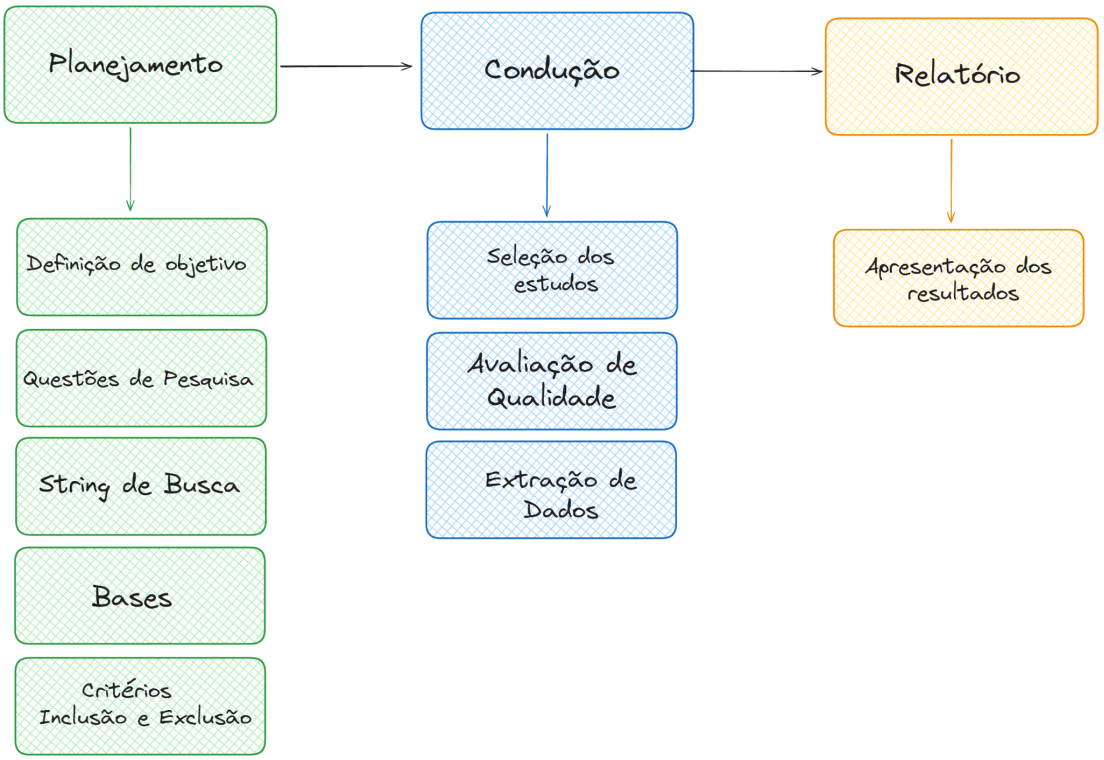

Figura 4. Representação gráfica do protocolo utilizado na condução do mapeamento sistemático do presente estudo. Fonte: Elaborado pelo autor.

3.1. Planejamento

3.1.1. Objetivo e Questões de pesquisa

Atualmente, há diversos trabalhos que disseminam conhecimento sobre a arquitetura orientada a eventos. Nesse modo, com o objetivo entender as necessidades computacionais as quais originam-se a necessidade de implantação

de tal arquitetura, bem como levantar as ferramentas que a possibilita, definimos as seguintes questões de pesquisa (QP):

QP1: Quando implantar uma arquitetura orientada a eventos?

QP2: Quais as principais ferramentas utilizadas em arquiteturas orientadas a eventos?

3.1.2. Definição de Busca e Critérios de Seleção

Uma vez definidas as questões de pesquisa, seguimos para a etapa do protocolo de estruturar a busca por trabalhos que contribuirão para o mapeamento. Dessa forma, para a construção de uma pesquisa sólida, foram realizadas buscas preliminares sobre o tema, com intuito de compreender quais termos seriam utilizados na busca. Esse processo foi realizado de forma interativa e em diversos ciclos, afim de modelar a *string* de busca, conforme orientado por Dermeval et al. [9]. Desse modo, obtivemos a seguinte relação de termos que serão utilizados na busca dos trabalhos.

Tabela 1. Conjunto de termos iniciais considerados nas buscas preliminares para o mapeamento sistemático da literatura.

Termo
Software Design
Software architecture
System Design
event-based architecture
event-driven architecture
event-oriented architecture

Após levantar os conjuntos de termos que seriam utilizados no processo de busca dos estudos juntamente , chega-se a seguinte *string* de busca.

Tabela 2. *String* utilizada na busca dos estudos nas bases selecionadas.

String de busca
("Software Design" OR "Software architecture" OR "System Design") AND ("event-based architecture" OR "event-driven architecture" OR "event-oriented architecture")

Para a condução das buscas destinadas ao mapeamento sistemático, foram selecionadas três bases de conhecimento, a partir das quais os trabalhos seriam recuperados para análise.

Tabela 3. Repositórios utilizados nas buscas dos estudos bem como seus respectivos endereços.

Repositório	URL	Busca
ACM Digital Library[8]	https://dl.acm.org	[[All: "software design"] OR [All: "software architecture"] OR [All: "system design"]] AND [[All: "event-based architecture"] OR [All: "event-driven architecture"] OR [All: "event-oriented architecture"]] AND [E-Publication Date: (01/01/2018 TO *)]
IEEE Digital Library[9]	https://ieeexplore.ieee.org	("Software Design" OR "Software architecture" OR "System Design") AND ("event-based architecture" OR "event-driven architecture" OR "event-oriented architecture") Filters Applied: 2018 - 2024
Science@Direct[10]	https://www.sciencedirect.com	("Software Design" OR "Software architecture" OR "System Design") AND ("event-based architecture" OR "event-driven architecture" OR "event-oriented architecture") Year: 2018-2024

Com intuito de identificar os estudos mais relevantes sobre a utilização da arquitetura orientada a eventos, após a definição das bases de conhecimento, procedeu-se ao estabelecimento dos critérios de inclusão e exclusão. Essa etapa consiste na definição de critérios aos quais todos os trabalhos retornados serão submetidos, orientando a seleção de forma mais direcionada ao campo de estudo da pesquisa [9]. Para Dermeval et al. [9], alguns trabalhos como estudos secundários, artigos resumos, livros, relatórios técnicos e outras formas de literatura cinza são removidos. Portanto, pode-se observar na tabela 4 o critério de inclusão, da mesma forma, na tabela 5, são apresentados os critérios de exclusão definidos para o presente mapeamento sistemático da literatura.

[8] https://dl.acm.org
[9] https://ieeexplore.ieee.org
[10] https://www.sciencedirect.com

Tabela 4. Critérios de inclusão considerados na seleção de estudos obtidos nas bases de dados.

Critério
Estudos que abordam sobre event-driven architecture
Estudos em língua Inglesa
Estudos publicados após 2018
Estudos primários

Tabela 5. Critérios de exclusão considerados na seleção de estudos obtidos nas bases de dados.

Critério
Estudos duplicados
Estudos fora do escopo (event-driven architecture)
Estudos secundários ou Terciários
O estudo é apenas um resumo (abstract)
O estudo não foi escrito em língua inglesa
O estudo não possui resumo (abstract)
Publicação anterior à 2018

3.2. Condução

3.2.1. Busca e Seleção

Posteriormente os critérios de inclusão e exclusão definidos, inicia-se a fase de seleção dos trabalhos retornados do processo de busca. Na primeira etapa, os trabalhos passam por uma análise onde aplica-se os filtros definidos nos critérios de exclusão levando em consideração títulos e *abstracts* dos trabalhos, salvo alguns casos onde há necessidade de consultar também as respectivas conclusões. Uma vez concluída a primeira etapa de seleção, os trabalhos são analisados na íntegra com o objetivo de extrair informações relevantes para responder às questões de pesquisa.

As buscas nas bases de dados iniciaram-se em 1º de agosto de 2024 e concluíram-se em 7 de agosto de 2024. Durante esse processo, aplicaram-se filtros temporais para restringir os resultados a publicações a partir de 2018. Nesse período, as buscas realizadas com a *string* de busca previamente definida, em conjunto com o filtro de ano, resultaram nos trabalhos incluídos neste mapeamento sistemático. Portanto, na Figura 5, é possível visualizar a distribuição dos estudos retornados por base de dados, conforme a string de busca utilizada. Observa-se que 40,5% dos estudos são provenientes da base ACM Digital Library, 53,5% da base ScienceDirect e 6% da base IEEE Digital Library.

Ademais, na Figura 6, é apresentada a relação entre os estudos recuperados e selecionados por repositório. Na base ScienceDirect, foram recuperados 107 estudos, dos quais 38 foram selecionados. Já na ACM Digital Library, 81 estudos

foram recuperados e 22 selecionados. Por fim, na IEEE Digital Library, 12 estudos foram recuperados e 10 selecionados.

Do mesmo modo, na Figura 7, pode-se observar a distribuição dos estudos recuperados por ano de publicação. Verifica-se que 36 trabalhos foram publicados em 2018, 15 em 2019, 24 em 2020, 45 em 2021, 18 em 2022, 39 em 2023 e 33 em 2024.

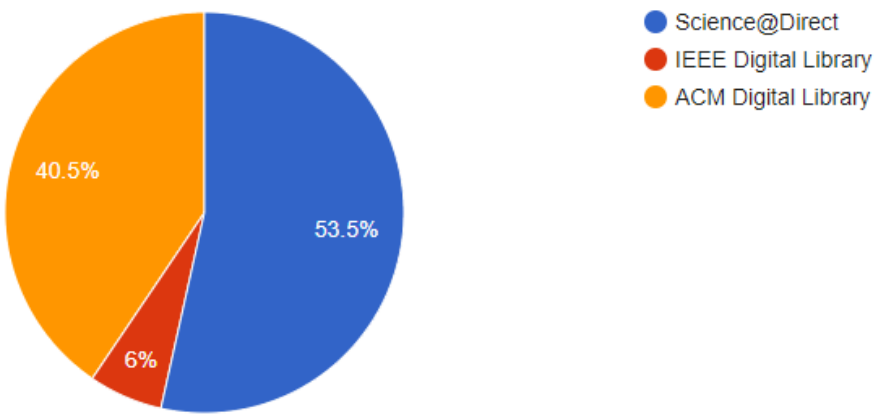

Figura 5. Distribuição por base de estudos retornados através da *string* de busca. Fronte: Concebido através da ferramenta parsifal[11] após conclusão do mapeamento sistemático.

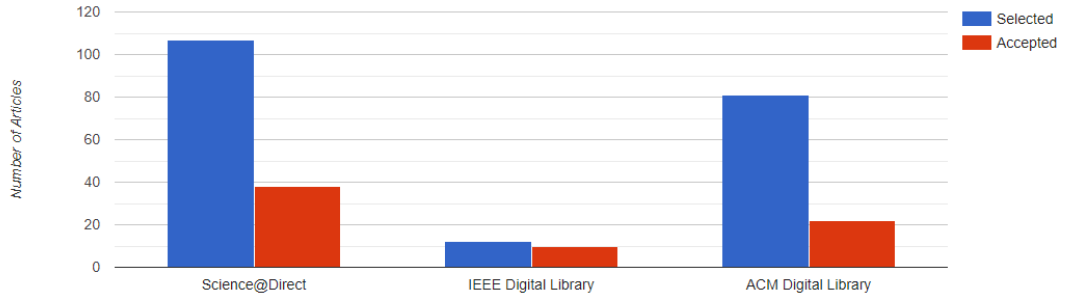

Figura 6. Relação de estudos recuperados e selecionados por repositório, utilizando a string de busca. Fonte: Concebido através da ferramenta parsifal após conclusão do mapeamento sistemático.

[11] https://parsif.al

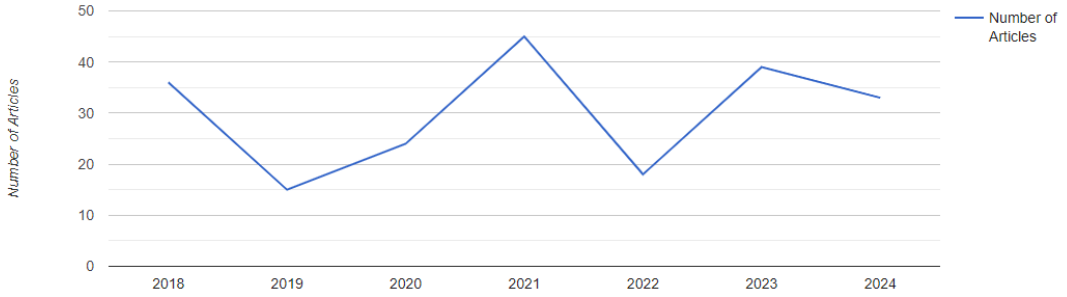

Figura 7. Distribuição de estudos recuperados por ano de publicação. Fonte: Concebido através da ferramenta parsifal após conclusão do mapeamento sistemático.

3.2.2. Avaliação de Qualidade

Nessa etapa, define-se também duas questões de qualidade (QQ) a serem respondidas na análise dos estudos com o objetivo de aumentar a acurácia dos estudos selecionados para o mapeamento sistemático [9]. Para tal, para cada questão levantada, há três possíveis respostas: Sim, Parcialmente e Não.

QQ1: O estudo apresenta caso prático de aplicação de uma arquitetura orientada a eventos?

QQ2: O estudo cita ferramentas utilizadas na aplicação da arquitetura orientada a eventos?

Afim de responder às questões levantadas, todos os artigos selecionados no estudo passam por análise. Portanto, os critérios considerados com finalidade de responder aos questionamentos de qualidade, foram respectivamente:

Tabela 6. Critérios adotados para responder à questão QQ1.

Resposta	**Critério**
Sim	Estudos que apresentaram ou implementam uma arquitetura orientada a eventos
Parcialmente	Estudos que citam ferramentas ou sistemas que utilizam arquitetura orientada a eventos mas não implementam a mesma.
Não	Estudos que não citam e não implementam a arquitetura orientada a eventos.

Tabela 7. Critérios adotados para responder à questão QQ2.

Resposta	**Critério**
Sim	Estudos que citam e/ou utilizam ferramentas para realizar comunicação conforme proposto na arquitetura

Não	Estudos que não citam e não utilizam ferramentas para realizar comunicação conforme proposto na arquitetura

Segundo Dermeval et al. [9], em mapeamentos sistemáticos, mesmos trabalhos que não atendem exclusivamente aos protocolos de avaliação de qualidade devem ser incluídos no estudo, isso pois, esses estudos podem fornecer lacunas na pesquisa. Dessa forma, seguindo a orientação, mesmos estudos que não atendiam nenhum dos itens de qualidade foram incluídos no mapeamento, visto que podem guiar trabalhos futuros e auxiliar na interpretação dos resultados [9].

3.2.3. Extração de Dados

Após a conclusão do protocolo de busca e seleção dos estudos e a definição dos critérios de qualidade, inicia-se a etapa de extração dos dados. Após essas duas etapas de análise, foram examinados o total de 200 trabalhos identificados por meio da *string* de busca, onde 70 deles foram analisados na íntegra, conforme detalhado na seção 3.1.2 desta seção. A Tabela 8 apresenta o resultado dos estudos, refletindo os critérios de inclusão e exclusão definidos previamente e aplicados durante o processo. Observa-se também na tabela 8, que três trabalhos estão descritos na categoria "Não classificado". Esses trabalhos, embora selecionados na primeira análise, que baseia-se em títulos e *abstracts*, não foram incluídos na segunda análise pois não foi possível ter acesso ao trabalho na íntegra.

Tabela 8. Resultado de estudos selecionados para mapeamento e seus respectivos estados.

Status	**Quantidade**
Total	200
Aceitos	70
Recusados	87
Duplicados	40
Não classificado	3

Com intuito de organizar este trabalho, a condução deste mapeamento foi executada com auxílio da ferramenta parsifal[12]. A mesma é uma ferramenta gratuita disponibilizada na web e visa auxiliar pesquisadores na condução de mapeamentos sistemáticos e revisões sistemáticas da literatura, baseando-se no protocolo definido por Kitchenham et al. [79].

[12] https://parsif.al

A fim de coletar informações necessárias, foram definidos os seguintes dados a serem extraídos dos trabalhos relacionados: título; autores; ano de publicação; tipo de publicação; número de citações no google scholar; caso prático de aplicação; e ferramentas utilizadas na aplicação do estudo. Dessa forma, obteve-se a conclusão do preenchimento do formulário de extração descrito no protocolo apresentado no início da seção 3. Na tabela 9, abaixo, são apresentados os estudos selecionados bem como seus respectivos resultados provenientes dos indagamentos a respeito das questões de qualidade definidas na seção 3.2.2 deste trabalho.

Tabela 9. Estudos selecionados para mapeamento sistemático.

ID	Referência	Caso de Aplicação	Ferramenta
EP1	10	O estudo descreve a implementação de um sistema baseado em filas de mensagens, onde componentes como funções sem estado (stateless functions) atuam como consumidores e produtores, processando e enviando populações entre diferentes etapas do algoritmo.	RabbitMQ
EP2	11	O trabalho apresenta a emulação de atributos digitais de dispositivos periféricos, como sensores, onde eventos são gerenciados por uma arquitetura de software orientada a eventos. Essa arquitetura é usada para modularizar e escalar o sistema, bem como para processar comportamentos como leituras/escritas em registradores e interrupções.	-
EP3	12	O estudo descreve a aplicação da EDA em um framework adaptativo para o controle e monitoramento de veículos não tripulados (UVs) em missões complexas, como exemplificado no uso de um Veículo de Superfície Não Tripulado (USV) em um experimento de simulação e operação real.	-
EP4	13	O estudo apresenta um caso prático de aplicação de arquitetura orientada a eventos. A arquitetura é utilizada para gerenciar a comunicação e interação entre módulos em um sistema veicular, especialmente para monitorar a orientação da cabeça do motorista e adaptar a exibição de informações na interface do veículo em tempo real.	Euphoria
EP5	14	O estudo discute a implementação de agentes conversacionais para a Agência de Recursos Hídricos (Water Resources Agency - WRA) e o Departamento de Minas (Bureau of Mines) de Taiwan, ambos afetados por desastres naturais e necessitando de soluções para gerenciar o excesso de informações durante esses eventos. A arquitetura Rich Notification Architecture (RNA) foi utilizada para a entrega de notificações e informações relevantes para as equipes dessas organizações, demonstrando a aplicação prática da arquitetura orientada a eventos na gestão de desastres.	-
EP6	15	O estudo aborda a aplicação de event-driven architecture no contexto da simulação de usinas hidrelétricas. O trabalho discute os desafios da modelagem e simulação de usinas hidrelétricas de grande escala, destacando as limitações das abordagens tradicionais. O trabalho explica como a EDA pode ser integrada com a SOA para superar as dificuldades associadas à orquestração centralizada e ao acoplamento forte de serviços, típicos da SOA. O	NServiceBus, MSMQ e RabbitMQ.

		foco está na proposta de um novo método de modelagem, o "Active Perceivable Device-Oriented Modeling" (APDOM), que utiliza características da EDA para melhorar a simulação de usinas hidrelétricas, mas sem citar um caso prático específico onde a EDA foi aplicada.	
EP7	16	-	-
EP8	17	O estudo descreve um caso prático aplicado à Event-Driven Architecture (EDA) no contexto de smart contracts na plataforma Ethereum. A proposta é melhorar a implementação dos contratos inteligentes e do sistema de blockchain através de uma arquitetura orientada por eventos.	-
EP9	18	O estudo apresenta um caso prático de aplicação de uma arquitetura orientada a eventos. Eles demonstram a aplicação dessa arquitetura em dois benchmarks: reconhecimento de dígitos no conjunto de dados MNIST e a fusão sensorial de radar e imagem para a classificação de terras agrícolas. Esses casos práticos são utilizados para demonstrar como a arquitetura proposta pode ser aplicada em cenários reais, focando na eficiência e na redução do consumo de recursos, conforme mencionado no resumo do documento.	-
EP10	19	O estudo apresenta a arquitetura para um sistema de gerenciamento de riscos dinâmico e de segurança rodoviária que utiliza uma arquitetura orientada a eventos como seu modelo de coordenação e comunicação.	-
EP11	20	O trabalho cita a utilização da arquitetura dirigida a eventos no contexto de blockchain para comunicação assíncrona de transações e execução de smart contact.	ActiveMQ, RabbitMQ, Kafka e Pulsar
EP12	21	-	-
EP13	22	-	-
EP14	23	O estudo apresenta a aplicação de uma arquitetura dirigida por eventos na implementação do protocolo Trustful Space-Time Protocol (TSTP) que utiliza técnicas de metaprogramação para implementar uma arquitetura dirigida por eventos que movimenta pacotes armazenados em buffers com metadados, sem cópias desnecessárias e eliminando dependências desnecessárias.	-

EP15	24	A arquitetura orientada a eventos é mencionada na seção que descreve padrões arquitetônicos e de design para RTES. Essa arquitetura é caracterizada por um "processador de eventos" central, que recebe, processa e redireciona os eventos para os assinantes. A configuração é estática, visando aumentar a previsibilidade e assegurar a operação correta em sistemas de tempo real.	-
EP16	25	O trabalho apresenta o desenvolvimento de uma plataforma de sandboxing, chamada Sandboxed Execution Environment (SEE), que utiliza uma arquitetura orientada a eventos para gerenciar e controlar a execução e análise de amostras. Nessa arquitetura, eventos gerados dentro do ambiente de sandbox são tratados por diferentes plugins forenses, que reagem a esses eventos conforme eles ocorrem. Isso demonstra a aplicação real do padrão em um sistema de sandbox para análise de segurança.	-
EP17	26	O trabalho apresenta o uso de Dapr e KEDA em uma arquitetura proposta para microsserviços auto-adaptáveis. Especificamente, Dapr é utilizado para comunicação entre serviços através do padrão de publicação e assinatura (publish and subscribe), e KEDA é usado para escalar horizontalmente com base em gatilhos de eventos.	Azure Service Bus, RabbitMQ, AWS SNS/SQS
EP18	27	O estudo descreve um sistema de pedágio sensível à poluição que adota uma arquitetura microservices orientada a eventos. Nesse sistema, diferentes microsserviços se comunicam por meio de mensagens JSON através de um message broker, utilizando o padrão de mensageria assíncrona publish-subscribe.	-
EP19	28	-	-
EP20	29	O trabalho apresenta um caso prático de aplicação da arquitetura orientada a eventos (event-driven architecture, EDA). Especificamente, ele descreve a proposta, implementação e avaliação da "Event-Driven Data Exchange (EDDE)" como uma substituição ao modelo clássico baseado em frequência para geração de dados em aplicações robóticas. A implementação é apresentada no contexto de sistemas ciberfísicos (CPSs), com ênfase na introspecção e análise de eventos de execução de programas em cobots (robôs colaborativos).	-
EP21	30	O estudo apresenta a arquitetura Euphoria, uma nova proposta de arquitetura com abordagem event-driven para desenvolvimento de software. O trabalho apresenta três cenários de aplicação do Euphoria: interação com displays públicos usando gestos e dispositivos vestíveis, acesso a conteúdo digital ancorado a locais físicos em um ambiente	-

		inteligente, e uma interface de usuário vestível para notificações sociais em um ambiente veicular. Cada cenário ilustra como a arquitetura foi aplicada para implementar interações em ambientes inteligentes.	
EP22	31	-	Apache Kafka, AWS EventBridge, GCP Eventarc
EP23	32	O trabalho apresenta uma aplicação prática da Event-Driven Architecture. Ele descreve como a arquitetura foi implementada em gateways IoT baseados em x86 e ARM para integrar dados de diferentes dispositivos sensores de um sistema de fluxo de materiais. A arquitetura foi utilizada para coletar dados com o objetivo de realizar análises posteriores, como otimização de processos e manutenção.	Mosquitto
EP24	33	O trabalho descreve um sistema de microsserviços utilizado em um processo de negócio de uma loja de tecidos, onde a arquitetura EDA é empregada para gerenciar a comunicação assíncrona entre os serviços principais por meio de um serviço de streaming de eventos, o NATS Streaming.	Apache Kafka, RabbitMQ e NATS Streaming
EP25	34	O estudo apresenta o pattern TriQPAN (Trigger, Query, Process, Action and Notify), no qual utiliza-se de uma arquitetura baseada em eventos. O pattern possui como objetivo garantir que os processos de decisão são registrados para explicar os comportamentos de agentes.	Event Store DB (banco de dados para armazenar eventos)
EP26	35	O estudo apresenta um exemplo chamado "Life Net", que envolve a aplicação de arquiteturas baseadas em microservices, arquitetura orientada a eventos e tecnologia blockchain. Especificamente, o estudo descreve como o sistema Life Net utiliza a arquitetura orientada a eventos para gerenciar emergências médicas para pacientes com Alzheimer. O sistema registra eventos, como pedidos de socorro de pacientes, que são processados por serviços de microservices e armazenados em um log de eventos, permitindo que outros componentes, como os serviços Lifeguard e Lifecare, respondam a esses eventos.	-
EP27	36	O trabalho descreve a implementação de uma arquitetura baseada em eventos para dar suporte ao modelo Pathways de coordenação de cuidados em sistemas de saúde. Essa arquitetura é usada para gerenciar a dinâmica de eventos discretos, como agendamento de	-

		tempo, transições de estado, e entrada/saída de dados dentro do sistema de coordenação de cuidados.	
EP28	37	O estudo descreve a implementação de uma arquitetura baseada em eventos para integrar quatro elementos: a descrição do experimento, o modelo de aprendizado por reforço (RL), o design do jogo e o módulo de coleta de dados. Essa arquitetura permite a criação de experimentos neuropsicológicos gamificados, facilitando a integração e a execução dos experimentos.	-
EP29	38	O trabalho descreve a arquitetura de um Digital Twin (DT) que utiliza conceitos de sistemas distribuídos, incluindo comunicação via canais Pub/Sub, que são fundamentais em arquiteturas orientadas a eventos. Esses elementos indicam uma abordagem que pode ser considerada compatível com EDA	Kafka
EP30	39	O caso prático apresentado no estudo é o uso do GeoRocket para o gerenciamento de grandes conjuntos de dados geoespaciais na nuvem. GeoRocket utiliza uma arquitetura reativa e baseada em eventos, onde componentes independentes (verticles) se comunicam através de um barramento de eventos para realizar importação, indexação e consulta de dados.	Vert.x
EP31	40	O trabalho apresenta um novo sistema I/O distribuído, dinâmico, multicamadas e com consciência da heterogeneidade. O sistema utiliza uma camada de buffer baseada em uma arquitetura orientada a eventos afim de evitar perdas de dados.	-
EP32	41	O estudo apresenta dois casos de uso que utilizam a arquitetura orientada a eventos: Um sistema de vigilância por vídeo que realiza análise inteligente em tempo real de imagens e áudio, gerando alarmes automáticos para melhorar a resposta e a tomada de decisão. Arquitetura baseada em microsserviços e protocolos de mensageria para gerenciar sensores em ambientes inteligentes e conectados à nuvem, permitindo que empresas de serviços e clientes monitorem o consumo de energia e água através de dispositivos sensores, com acesso por meio de uma interface web.	RabbitMQ e Mosquitto
EP33	42	O caso prático apresentado é a implementação de um "Digital Twin" para um escritório da empresa italiana R2M Solution. Esse sistema integra diversos sensores, dados de redes sociais, informações de APIs de terceiros e dados processados da empresa para criar um ambiente virtual imersivo e interativo que representa o espaço físico real.	Apache kafka e Apache Spark

EP34	43	O estudo descreve a utilização de uma arquitetura baseada em eventos para integrar dados em tempo real de sensores IIoT com sistemas de gestão de processos de negócios (WfMS). A arquitetura permite o monitoramento e a ativação de eventos complexos no nível de processos de negócios.	Siddhi
EP35	44	O estudo descreve a aplicação da arquitetura orientada a eventos em ambientes de nuvem baseados em contêineres, como Kubernetes e OpenShift. Especificamente, ele discute como o Kubernetes utiliza controladores para reagir às mudanças de estado dos objetos no API Server através de um mecanismo de notificação chamado "Watch".	-
EP36	45	-	-
EP37	46	O estudo apresenta a implementação de um sistema IoT para detecção de falhas em sensores de pressão de turgor das folhas. Neste sistema, o MQTT é utilizado para a comunicação entre camadas e componentes, permitindo uma arquitetura baseada em eventos para gerenciar dados e detecção de falhas.	Mosquitto
EP38	47	-	-
EP39	48	O estudo apresenta um caso prático de aplicação de arquitetura orientada a eventos. A seção "CLEAR_worker_server" descreve como os serviços do CLEAR utilizam uma arquitetura orientada a eventos, aguardando requisições HTTP POST para entregar dados de entrada e emitir sinais. Eventos específicos acionam processos de trabalho que produzem dados utilizáveis pelo coordenador.	-
EP40	49	-	
EP41	50	O estudo testa os benefícios do sistema de informação AIC (que utiliza uma arquitetura orientada a eventos e modelos) em uma cadeia de suprimentos farmacêutica na França. Ele monitora e interpreta eventos em tempo real para melhorar a resiliência da cadeia de suprimentos, detectando novos riscos e incidentes.	-
EP42	51	-	-
EP43	77	-	RabbitMQ e Apache Kafka

EP44	6	O estudo realiza um estudo exploratório em volta da arquitetura orientada a eventos e a arquitetura monolítica. Para isso, o trabalho implementou uma plataforma de e-commerce utilizando as duas arquiteturas com o objetivo de avaliar métricas que indicam desempenho.	RabbitMQ e Azure Service Bus
EP45	52	O estudo descreve a implementação da OpenATE (Open-source Automated Transactive Energy Engine), que utiliza uma arquitetura orientada a eventos para controlar simulações e gerenciar mensagens de transação. O sistema permite que cada módulo opere de forma autônoma, abordando problemas de coordenação do tempo simulado, o que é um exemplo claro de como a arquitetura orientada a eventos pode ser aplicada em um cenário de simulação para sistemas de energia transacional (TES).	-
EP46	53	-	Apache Kafka
EP47	54	O estudo menciona a aplicação prática de uma arquitetura orientada a eventos ao descrever o uso do Zeebe, um motor de orquestração de workflows baseado em BPMN para microsserviços e aplicações serverless. O Zeebe usa uma arquitetura orientada a eventos por meio de eventos de mensagem para coordenar essas aplicações.	SQS e SNS
EP48	82	-	-
EP49	55	-	-
EP50	56	O trabalho apresenta um sistema simplificado de e-commerce. Nele, um componente "Buy" envia uma solicitação para o componente "InventoryMgr". A solicitação é gerenciada por um EventBus, e o componente "Buy" não aguarda a resposta imediatamente. Quando a resposta é gerada, ela também é gerenciada pelo EventBus e pode ser recebida por uma instância diferente do componente "Buy". O estado da operação é salvo em um Persistence Store para garantir que a resposta seja corretamente associada à instância que fez a solicitação original.	-
EP51	57	O trabalho descreve a arquitetura da plataforma Elinvar, que utiliza é um event-driven architecture. O trabalho destaca o uso de eventos como uma parte essencial de sua comunicação entre serviços e a implementação de um RPC utilizando Kafka.	Apache Kafka
EP52	58	-	-

EP53	59	O trabalho descreve a aplicação de uma arquitetura baseada em eventos para agentes de caching adaptativos em um sistema distribuído. A arquitetura utiliza padrões de design como *publisher-subscriber* e *observer* para gerenciar fluxos de informações contextuais, realizar decisões de caching e calcular recompensas.	-
EP54	60	-	-
EP55	61	O estudo apresenta utiliza a arquitetura no contexto de smart home para detecção de presença na casa e realizar ações em tempo real.	AMQ e Kafka
EP56	62	O trabalho descreve a aplicação prática de Event-Driven Architecture ao mencionar o uso de mensagens para notificar diferentes serviços sobre a conclusão de tarefas, como na integração entre um sistema de rastreamento de veículos e outros serviços da empresa.	-
EP57	63	O estudo descreve a implementação de uma arquitetura orientada a eventos na linha de montagem de carros de pedal na Scania, Sodertalje. A arquitetura foi utilizada para transformar dados gerados pelas ferramentas de potência em tempo real e publicá-los em um barramento de mensagens (Kafka), permitindo que diferentes sistemas se inscrevessem e consumissem esses dados. A implementação do EDA facilitou a coleta e utilização dos dados do chão de fábrica em tempo real, substituindo ferramentas antigas por modernas e permitindo a comunicação eficiente entre os sistemas.	Kafka
EP58	64	-	-
EP59	65	O estudo apresenta um exemplo prático de sistema de compartilhamento de caronas (Ride-Sharing System - RSS), que é usado para demonstrar como a arquitetura orientada a eventos pode ser aplicada para melhorar aspectos como recuperação de falhas e escalabilidade. O estudo propõe a aplicação de padrões orientados a eventos, como o Event-Sourcing e CQRS, para transformar o design arquitetônico do sistema RSS.	Apache Kafka
EP60	66	-	-
EP61	67	O trabalho descreve a arquitetura DIGICOR, que é baseada em uma abordagem de Event-Driven Service-Oriented Architecture (EDSOA). O foco está na arquitetura e nas capacidades da plataforma para suportar a colaboração na cadeia de suprimentos e a integração de serviços através de eventos.	-

EP63	68	O trabalho cita quatro casos de uso para a arquitetura e executam benchmarks de escalabilidade em ambiente de nuvem. 1. Armazenamento de mensagens ou eventos processadas para consulta posterior. 2. Agregação de eventos em janelas de tempo com objetivo de reduzir a quantidade de eventos consecutivos 3. Agregação de eventos com mesmo atributo de tempo. ex: dia, hora, mês ou ano. 4. Agregação de eventos por grupos de sensores no contexto de IoT	Apache Kafka Streams e Apache Flink
EP64	69	-	-
EP65	70	O trabalho apresenta o VRTrain framework, que segue uma arquitetura baseada em eventos, para o desenvolvimento de aplicações de treinamento em realidade virtual (VR). A arquitetura dirige os comportamentos dos objetos virtuais, as tarefas de treinamento e a coleta de dados, tudo baseado em eventos.	-
EP66	71	O trabalho apresenta um novo módulo para o Kafka-ML com objetivo de treinar modelos de forma federada assíncrona, sendo capaz de combinar recursos de aprendizado federado e fluxo de dados em aplicativos de ML/IA.	Kafka-ML
EP67	72	O trabalho descreve o Triggerflow, uma arquitetura de orquestração baseada em eventos para workflows serverless. O Triggerflow utiliza uma arquitetura de Event-Condition-Action (ECA) e é projetado para lidar com eventos de diversas fontes, executar ações e criar filtros personalizados.	Kafka e RabbitMQ
EP68	73	O estudo discute a programação orientada a eventos como uma alternativa ao modelo thread-per-session. A programação orientada a eventos é usada para gerenciar conjuntos de estados de sessão em resposta a eventos de I/O, e é diretamente suportada por interfaces típicas dos sistemas operacionais	kqueue e epoll
EP68	74	O Trabalho descreve como o WaCoDiS System implementa uma arquitetura orientada a eventos para gerenciar o processamento de dados de observação da terra (Earth Observation). Especificamente, o WaCoDiS utiliza padrões baseados em mensagens assíncronas e o protocolo AMQP para implementar fluxos de dados orientados a eventos, que são acionados por eventos de disponibilidade de dados e outros eventos relevantes, como eventos climáticos extremos.	Apache Kafka e RabbitMQ

EP69	75	O trabalho discute a utilização da arquitetura orientada a eventos no Unreal Engine, onde os componentes comunicam-se usando um modelo de publicação-inscrição (publish-subscribe) para manter a consistência de dados e reduzir o tempo de acesso.	-
EP70	76	-	-

4. Resultados

Após a execução de todas as etapas previstas no protocolo deste mapeamento sistemático, passa-se à fase de análise dos dados, com o objetivo de responder às questões de pesquisa (QP) estabelecidas na Seção 3.1.1 deste trabalho. Nesta seção, serão apresentados os resultados obtidos, bem como os fatos decorrentes da análise, utilizando o processo de mapeamento sistemático.

4.1. QP1: Quando implantar uma arquitetura orientada a eventos?

Entre os estudos selecionados, conforme analisado nos casos de aplicação apresentados na tabela 9, é possível identificar diversos cenários em que os conceitos da arquitetura dirigida a eventos foram implementados ou utilizados. Dessa maneira, constata-se que há uma variedade de contextos nos quais a EDA pode ser aplicada de forma eficaz. A partir da análise dos casos presentes nos estudos selecionados, podem-se observar os seguintes contextos de aplicabilidade.

4.1.1. Processamento em Tempo Real

A EDA é especialmente utilizada em sistemas onde a rapidez na resposta a eventos é crítica. Por exemplo, em sistemas de vigilância por vídeo, como descrito em um dos estudos analisados, a EDA permite a análise em tempo real de imagens e áudio, gerando alarmes automáticos que melhoram a resposta e a tomada de decisão [41]. Esse tipo de arquitetura é ideal para ambientes que demandam processamento contínuo e imediato de grandes volumes de dados, como em sistemas que realizam emissão de eventos em linhas de montagem, sistemas de detecção de presença em casas inteligentes ou na gestão de emergências médicas em tempo real [63][61][35].

4.1.2. Escalabilidade e Modularidade

Outra situação em que a EDA se destaca é em sistemas que necessita-se escalar horizontalmente e modularmente. Em um estudo sobre o uso de Dapr e KEDA para microsserviços auto-adaptáveis, a EDA foi utilizada para facilitar a comunicação entre serviços e escalar de forma dinâmica com base em eventos [26]. Também é possível observar o princípio da modularidade na ferramenta GeoRocket, que utiliza-se da *event-driven architecture* para comunicação entre componentes independentes [39] ou até mesmo no desenvolvimento de e-commerces para processamentos de pagamentos e gestão de inventário, de forma que ambos trabalham de forma paralela, porém sempre consistente [56]. Logo, observa-se que arquitetura permite que cada componente do sistema seja escalado de maneira independente, conforme a demanda, sem a necessidade de reconfigurar todo o sistema, promovendo uma escalabilidade eficiente e em alguns casos de baixo custo.

4.1.3. Integração de Sistemas Distribuídos e IoT

A integração de sistemas distribuídos e a Internet das Coisas (IoT) é outro cenário onde a EDA se mostra altamente eficaz. A arquitetura baseada em eventos é amplamente utilizada para gerenciar a comunicação entre dispositivos IoT, como em sistemas de fluxo de materiais, onde gateways IoT integram dados de diferentes sensores [32] ou realizam a integração entre sensores para criação de ambientes virtuais imersivos e interativos [42]. Esse modelo possibilita a coleta e processamento de dados em tempo real, facilitando a otimização de processos e a manutenção preditiva. Além disso, a EDA é útil na coordenação de eventos em sistemas de controle de veículos não tripulados, onde a interação entre sensores e módulos de controle precisa ser rápida e confiável [12].

4.1.4. Gerenciamento de Fluxos de Trabalho Complexos

A EDA mostra-se eficiente para sistemas que envolvem fluxos de trabalho complexos, como na gestão de cadeias de suprimentos ou na coordenação de cuidados em sistemas de saúde. Em um estudo sobre a implementação de uma EDA em um sistema de coordenação de cuidados, a arquitetura foi utilizada para gerenciar a dinâmica de eventos discretos, como agendamentos e transições de estado, permitindo um controle mais eficiente e flexível dos processos [36]. Outra utilização muito interessante é aplicação em ambientes *cloud* baseados em containers, onde especificamente o Kubernetes utiliza para controle de estados e gestão de mudanças em seus objetos [44]. Portanto, tal aplicabilidade é particularmente relevante em contextos onde a capacidade de reagir a eventos em tempo real é essencial para o sucesso do sistema.

4.1.5. Simulações e Modelagem em Ambientes Dinâmicos

A arquitetura orientada a eventos também se destaca em ambientes de simulação e modelagem que exigem uma coordenação precisa e dinâmica entre diferentes módulos. Por exemplo, na simulação de usinas hidrelétricas, a EDA foi integrada com a Arquitetura Orientada a Serviços (SOA) para superar limitações de orquestração centralizada e acoplamento forte [15]. Outro uso relevante é a utilização da EDA para gerenciamento de risco dinâmico e segurança em rodovias. Ademais, utiliza-se também em aplicações de treinamento em realidade virtual [70] e em treinamentos de modelos de forma federada e assíncrona [71]. Logo, observa-se que a EDA é especialmente útil em simulações complexas que demandam a coordenação de múltiplos agentes e a reação a eventos em tempo real, como a simulação de sistemas de energia transacional [52].

4.2. QP2: Quais as principais ferramentas utilizadas em arquiteturas orientadas a eventos?

Após a análise detalhada dos resultados, constatou-se que, entre os setenta estudos incluídos no mapeamento sistemático, três ferramentas emergiram como as mais citadas no desenvolvimento de arquiteturas orientadas a eventos. As ferramentas que se destacam nesse conjunto de estudos são: Apache Kafka, RabbitMQ e Mosquitto. Apache Kafka é citado em quatorze circunstâncias entre os

estudos analisados. Já RabbitMQ possui dez ocorrências entre os trabalhos. Mosquitto, por sua vez, é referenciado em três trabalhos. A seguir, a figura 8 apresenta as ferramentas referenciadas ou utilizadas nos estudos selecionados no mapeamento sistemático. A imagem apresenta a representação gráfica referente às ferramentas utilizadas agrupadas por ano de apresentação do estudo. Os dados são apresentados ordenados inicialmente por contagem de citações e posteriormente por ordem alfanumérica.

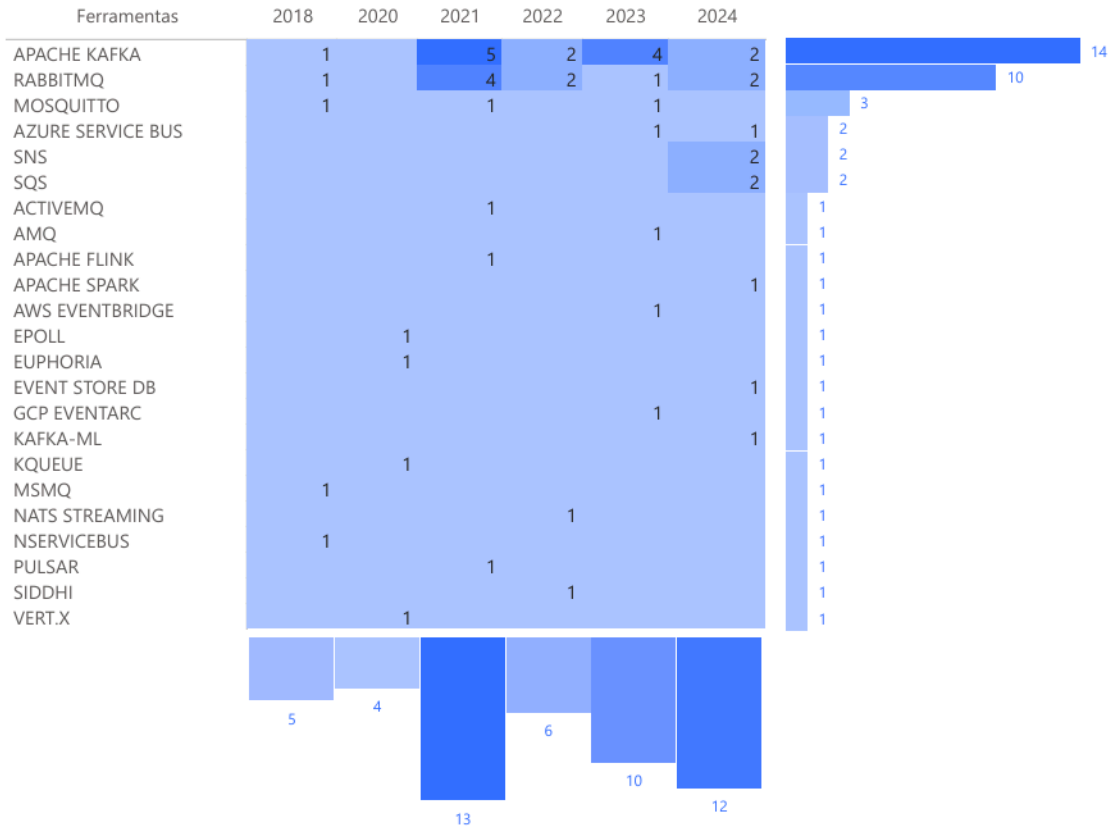

Figura 8. Ferramentas referenciadas ou utilizadas nos estudos selecionados no mapeamento sistemático. Elaborado pelo autor.

Além disso, é possível observar também a presença, pelo menos uma referência ou a utilização, de outras ferramentas tais como: Activemq, AMQ, Apache Flink, Apache Spark, Aws Event Bridge, Azure Service Bus, Epoll, Euphoria, Event Store DB, GCP Eventarc, Kafka-Ml, Kqueue, MSMQ, Nats Streaming, Nservicebus, Pulsar, Siddhi, Sns, Sqs e Vert.X. Ademais, através da Figura 8, observa-se que o ano de 2021 registrou o maior número de publicações sobre o tema, totalizando 13 trabalhos. De maneira similar, verifica-se que, até o momento da realização deste estudo, no ano de 2024, já foram publicados 12 trabalhos relacionados ao tema, o que reforça a relevância da arquitetura para a literatura.

5. Estudo de Caso

Com objetivo didático, o presente estudo apresenta a implementação de componentes da arquitetura apresentada na seção 1.1. O sistema possui como objetivo realizar atualização de partidas em tempo real emitidos a partir da ação de um agente controlador. Esse estudo de caso se limita à implementação de maneira simplificada dos componentes apresentados na figura 9 que estão em destaque na cor verde, uma vez que o objetivo é apenas introduzir conceitos de EDA aos discentes oriundos da disciplina de Sistemas Distribuídos. Logo, destaca-se que cada item da arquitetura pode possuir em seu domínio regras complexas em caso de uma implementação para um ambiente de produção, no entanto, o foco deste estudo é apenas apresentar, de forma prévia, conceitos da EDA em prática através de exemplos, dessa maneira, reitera-se que os exemplos apresentados aqui limitam-se às implementações simplificadas.

O projeto foi implementado utilizando a linguagem Golang e faz uso da ferramenta Docker[13] para virtualização das ferramentas Kafka e RabbitMQ. O mesmo apresenta implementações dos componentes: Producer (1), Fila de eventos (2), Backend consumidor de eventos (3) e Backend consumidor de eventos (4). O projeto foi desenvolvido de modo que utilizam ambas as ferramentas supracitadas e seu código está disponibilizado na íntegra de maneira *open-source* no repositório tcc-event-driven-architecture.

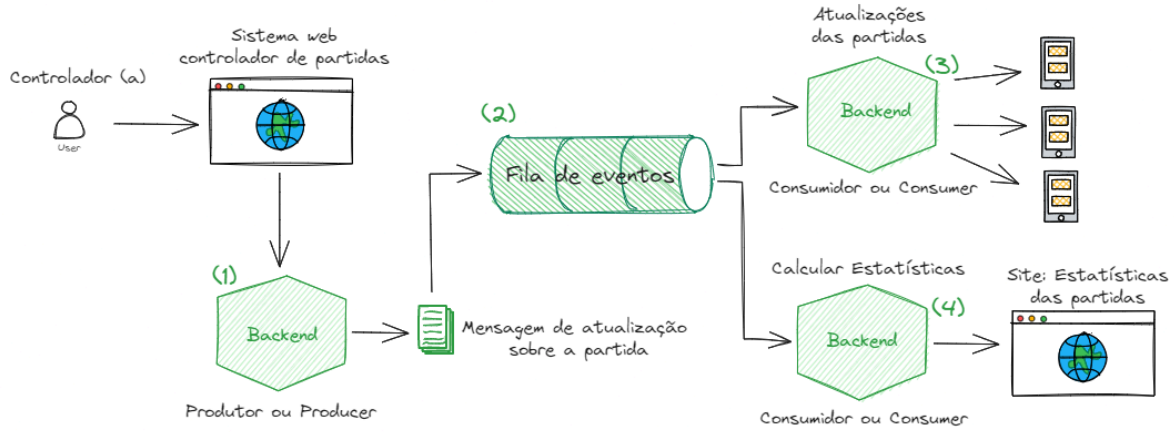

Figura 9. Arquitetura proposta para sistema de atualização de partidas em tempo real proposta para o estudo de caso. Fonte: Elaborado pelo autor.

5.1. Produtor

O componente Producer (1) é responsável por emitir os eventos a partir da interação com o agente controlador. Como neste estudo será implementado apenas alguns elementos da arquitetura, e de maneira simplificada, o produtor atuará emitindo eventos a cada 5 segundos, simulando ações de atualização emitidas pelo controlador.

[13] https://www.docker.com/

5.1.1. Kafka

Na figura 10, é apresentada a função invocada a cada 5 segundos, cujo a finalidade é realizar o envio dos eventos. O exemplo em questão, utiliza a ferramenta Kafka como canal de eventos e realiza a tentativa de envio do evento recebido para o barramento também recebido como parâmetro.

```
11  func Producer(ctx context.Context, logger *slog.Logger, event Event, broker, topic string) error {
12          w := kafka.NewWriter(kafka.WriterConfig{
13                  Brokers:  []string{broker},
14                  Topic:    topic,
15                  Balancer: &kafka.LeastBytes{},
16          })
17          defer w.Close()
18
19          eventBytes, e := json.Marshal(event)
20          if e != nil {
21                  logger.Error("failure to marshal event", slog.String("error", e.Error()))
22                  return e
23          }
24          err := w.WriteMessages(ctx,
25                  kafka.Message{
26                          Key:   []byte("football-example"),
27                          Value: eventBytes,
28                  },
29          )
30          if err != nil {
31                  logger.Error("could not write message:", slog.String("error", err.Error()))
32                  return err
33          }
34          return nil
35  }
```

Figura 10. Trecho do código responsável pela emissão dos eventos ao produtor, utilizando Kafka como componente (2) da arquitetura proposta. Fonte: Projeto elaborado pelo autor disponível em repositório *open-source*.

5.1.2. RabbitMQ

Na figura 11, apresenta-se uma função produtora, codificada em Golang, cuja finalidade é realizar o envio dos eventos. A mesma também é invocada a cada 5 segundos com finalidade de publicar eventos de domício no barramento de eventos e utiliza a ferramenta RabbitMQ para propagação de eventos. A função *Producer* recebe como parâmetro um canal definido pelo cliente da ferramenta e um evento, e portanto, tenta enviar este último no canal.

```go
func Producer(ctx context.Context, logger *slog.Logger, ch *amqp.Channel, event Event) error {
	body, err := json.Marshal(event)
	if err != nil {
		logger.Error("Failed to marshal event", slog.String("error", err.Error()))
		return err
	}

	// Publish the message on the "events" exchange
	err = ch.Publish(
		"events", // exchange
		"",       // routing key
		false,    // mandatory
		false,    // immediate
		amqp.Publishing{
			ContentType: "application/json",
			Body:        body,
		})
	if err != nil {
		logger.Error("Failed to publish message", slog.String("error", err.Error()))
		return err
	}
	return nil
}
```

Figura 11. Função que descreve o comportamento do *producer*, escrita em Golang e faz uso da ferramenta RabbitMQ. Fonte: Projeto elaborado pelo autor disponível em repositório *open-source* online.

5.2. Consumidor

Os componentes 3 e 4 da arquitetura apresentada da figura 9, atuam como *consumers*, ou seja, são *backends* responsáveis por receber e processar os eventos entregues pelo componente 2. Esses elementos possuem regras exclusivas de cada domínio, sendo que item 3 é responsável por efetuar disparos de atualização sobre as partidas que seus respectivos usuários estão acompanhando e o item 4 sendo responsável por interpretar os eventos e realizar cálculos estatísticos sobre os dados das partidas.

5.2.1. Kafka

Na figura 12, é apresentada a implementação do consumidor utilizando a ferramenta kafka. Observa-se que a função *AnalyticsConsumer*, utiliza-se de um elemento chamado *kafka.NewReader*, esse por sua vez, é responsável por estabelecer conexão com o serviço disponibilizado pelo Kafka, uma vez fornecido endereço e tópico, nomeados aqui como *broker* e *topic* respectivamente. Ademais, verifica-se também o uso do elemento *r.ReadMessage*, cuja função é receber novos eventos disponíveis.

```
func AnalyticsConsumer(ctx context.Context, logger *slog.Logger, broker, topic string) error {
    r := kafka.NewReader(kafka.ReaderConfig{
        Brokers: []string{broker},
        Topic:   topic,
    })
    defer r.Close()
    logger.Info("AnalyticsConsumer started. Waiting for events...")

    var totalScore, teamATotalScore, teamBTotalScore int
    // Function to update counters
    updateScores := func(event Event) {…
    }

    for {
        // Checks if the context has been canceled
        select {
        case <-ctx.Done():
            logger.Info("Context canceled, stopping consumer...")
            return ctx.Err() // Returns the error associated with the context cancellation
        default:
            // Try read a new message
            m, err := r.ReadMessage(ctx)
            if err != nil {
                if err == context.Canceled || err == context.DeadlineExceeded {
                    logger.Info("Context deadline exceeded or canceled, stopping consumer...")
                    return err
                }
                logger.Error("could not read message: ", slog.String("error", err.Error()))
                return err
            }
            // Converts the event from JSON to a struct in Go
            var event Event
            if err := json.Unmarshal(m.Value, &event); err != nil {
                logger.Error("unmarshal", slog.String("error", err.Error()))
                return err
            }
            // Updates the counters upon receiving a new event
            updateScores(event)
        }
    }
}
```

Figura 12. Implementação de um consumidor usando Golang e Kafka. Fonte: Projeto elaborado pelo autor disponível em repositório *open-source* online.

5.2.2. RabbitMQ

O código de implementação do consumidor usando a ferramenta RabbitMQ pode ser observado na figura 13. A função faz uso de um canal disponibilizado através do cliente da ferramenta, onde a partir desse, declara-se uma fila e suas respectivas configurações através da função *ch.QueueDeclare*. Em seguida, o método *ch.QueueBind* é invocado para vincular a fila, declarada anteriormente, e o

exchange. Em seguida, uma conexão de consumidor é estabelecida com o cliente do RabbitMQ através do método *ch.Consume*.

```
12  func AnalyticsConsumer(ctx context.Context, logger *slog.Logger, conn *amqp.Connection, queueName string) error {
13      // Opens a new channel for the consumer
14      ch, err := conn.Channel()
15      if err != nil {
16          log.Fatalf("Failed to open a channel for consumer: %v", err)
17      }
18      defer ch.Close()
19      logger.Info("AnalyticsConsumer started. Waiting for events...")
20
21      // Declare the queue
22      // name da fila --durable --auto-delete --exclusive --no-wait --arguments
23      q, err := ch.QueueDeclare(queueName,true,false,false,false,nil)
24  >   if err != nil {…
27      }
28
29      // Link the queue to the exchange
30      // nome da fila --routing key --exchange
31      err = ch.QueueBind(q.Name,"","events",false,nil)
32  >   if err != nil {…
35      }
36
37      // Start consuming messages
38      // nome da fila --consumer name --auto-ack --exclusive --no-local --no-wait --arguments
39      msgs, err := ch.Consume(q.Name,"analytics", true,false,false,false,nil)
40  >   if err != nil {…
43      }
44
45      var totalScore, teamATotalScore, teamBTotalScore int
46      // Function to update counters
47  >   updateScores := func(event Event) {…
57      }
58
59      // Message Processing Loop
60      for {
61          select {
62  >       case <-ctx.Done(): // Checks if the context has been canceled…
66          case m := <-msgs: // Processes received messages
67              // Converts the event from JSON to a struct in Go
68              var event Event
69              if err := json.Unmarshal(m.Body, &event); err != nil {
70                  logger.Error("unmarshal", slog.String("error", err.Error()))
71                  return err
72              }
73
74              // Updates the counters upon receiving a new event
75              updateScores(event)
76          }
77      }
78  }
```

Figura 13. Implementação do *consumer*, responsável por receber e processar eventos, utilizando a linguagem Golang e a ferramenta RabbitMQ. Fonte: Projeto elaborado pelo autor disponível em repositório *open-source* online.

6. Considerações Finais

Este trabalho realizou um mapeamento sistemático da literatura com o objetivo de entender um pouco mais sobre o universo da arquitetura de software, mais

especificamente no contexto de *event-driven architecture* e suas aplicabilidades. Como objetivos particulares, buscou-se compreender os casos e cenários mais apropriados para a utilização da arquitetura orientada a eventos e a identificação das principais ferramentas utilizadas nesse contexto.

Com base na análise dos estudos selecionados através do protocolo de mapeamento previamente definido, foi possível alcançar os objetivos propostos. O estudo identificou que a EDA pode ser aplicada em uma variedade de cenários, tais como processamento em tempo real, sistemas que necessitam de escalabilidade ou modularidade, integração de sistemas distribuídos e IoT, gerenciamento de fluxos de trabalho complexos e pode-se encontrar também cenários de aplicação para simulações e modelagem em ambientes dinâmicos. Além disso, através do presente estudo, foi possível observar que as principais ferramentas mencionadas ou utilizadas na implementação da EDA entre os estudos analisados, são respectivamente: Apache Kafka utilizada quatorze vezes, RabbitMQ referenciada dez vezes e Mosquitto citada em três trabalhos. Dessa forma, os resultados evidenciam que o mapeamento sistemático cumpriu seu propósito de mapear o estado da arte sobre o uso da EDA.

Portanto, as principais contribuições deste estudo estão relacionadas ao avanço no entendimento dos contextos em que a arquitetura orientada a eventos é mais apropriada. Esse conhecimento é particularmente relevante para pesquisadores e profissionais que buscam aplicar a EDA de forma mais eficaz e em cenários específicos. Nesse sentido, visualiza-se que o presente estudo abre caminho para oportunidades futuras que permeiam a exploração detalhada das ferramentas bem como suas características mais profundas, uma vez que esse trabalho realiza apenas a identificação das mais utilizadas. Observa-se também uma possível linha de pesquisa futura baseada na realização de estudos empíricos em cenários específicos levantados através desse mapeamento sistemático.

Ademais, este trabalho também apresenta como produto um estudo de caso juntamente com de um repositório *open-source* contendo tutoriais em forma de implementação das ferramentas Apache Kafka e RabbitMQ. O objetivo deste estudo de caso em conjunto com o material, é auxiliar tanto discentes da disciplina de sistemas distribuídos ministrada na Universidade Federal do Oeste do Pará quanto pesquisadores e desenvolvedores iniciantes no contexto de EDA. O repositório está disponível no github e apresenta amostras em forma de implementação, usando a linguagem Golang. Os exemplos apresentam o desenvolvimento de um consumidor e um produtor, utilizando as duas ferramentas mais citadas dentre os estudos analisados.

Referências

1. Bogner, J.; Fritzsch, J.; Wagner, S.; Zimmermann, A. Microservices in Industry: Insights into Technologies, Characteristics, and Software Quality. Proceedings - 2019 IEEE

International Conference on Software Architecture - Companion, ICSA-C 2019, 2019, pp. 187–195.
2. Oliveira, A.; Bischoff, V.; Gonçales, L.J.; Farias, K.; Segalotto, M. BRCode: An Interpretive Model-Driven Engineering Approach for Enterprise Applications. Computers in Industry 2018, 96, 86–97.
3. Lazzari, L.; Farias, K. Uncovering the Hidden Potential of Event-Driven Architecture: A Research Agenda. arXiv Preprint arXiv:2308.05270, 2023.
4. Laigner, R.; Kalinowski, M.; Diniz, P.; Barros, L.; Cassino, C.; Lemos, M.; Arruda, D.; Lifschitz, S.; Zhou, Y. From a Monolithic Big Data System to a Microservices Event-Driven Architecture. 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA); IEEE, 2020, pp. 213–220.
5. Khriji, S.; Benbelgacem, Y.; Chéour, R.; Houssaini, D.E.; Kanoun, O. Design and Implementation of a Cloud-Based Event-Driven Architecture for Real-Time Data Processing in Wireless Sensor Networks. The Journal of Supercomputing 2022, 78, 3374–3401.
6. Cabane, H.; Farias, K. On the Impact of Event-Driven Architecture on Performance: An Exploratory Study. Future Generation Computer Systems 2024, 153, 52–69.
7. Lazzari, L.; Farias, K. An Exploratory Study on the Effects of Event-Driven Architecture on Software Modularity. arXiv Preprint arXiv:2110.14699, 2021.
8. Richards, M.W. Software Architecture Patterns: Understanding Common Architecture Patterns and When to Use Them; 2015.
9. Dermeval, D.; De, J.A.P.; Coelho, M. Capítulo 3: Mapeamento Sistemático e Revisão Sistemática da Literatura em Informática na Educação; JAQUES, Patrícia Augustin; SIQUEIRA; Sean; BITTENCOURT, Ig; PIMENTEL, Mariano.(Org.) Metodologia de Pesquisa Científica em Informática na Educação: Abordagem Quantitativa; Porto Alegre: SBC, 2020.
10. Valdez, M.G.; Guervós, J.J.M. A Container-Based Cloud-Native Architecture for the Reproducible Execution of Multi-Population Optimization Algorithms. Future Generation Computer Systems 2021, 116, 234–252.
11. Panambur, K.S.; Desai, S.; Singh, A.K.; Thoben, K.D. A Hybrid Approach for Digital Representation of Sensors in Real-Time Applications. Procedia Manufacturing 2020, 52, 14–19.
12. Bonache-Seco, J.A.; Lopez-Orozco, J.A.; Portas, E.B.; Martin, J.L.R. Adaptive Event-Driven Framework for Real-Time Multi-Agent Missions. 2018 IEEE/ACM 22nd International Symposium on Distributed Simulation and Real Time Applications (DS-RT), 2018.
13. Sburlan, D.F.; Bautu, E.; Puchianu, C.M.; Popovici, D.M. Adaptive Interactive Displaying System for In-Vehicle Use. Procedia Computer Science 2020, 176, 195–204.
14. Chan, H.Y.; Tsai, M.H. Alert Notifications for Governmental Disaster Response via Instant Messaging Applications. International Journal of Disaster Risk Reduction 2023, 96, 103984.

15. Zhang, B.; Yuan, X.; Yuan, Y.; Wang, X. An Active Perceivable Device-Oriented Modeling Framework for Hydropower Plant Simulation. Energy 2018, 165, 1009–1023.
16. Overeem, M.; Spoor, M.; Jansen, S.; Brinkkemper, S. An Empirical Characterization of Event-Sourced Systems and Their Schema Evolution—Lessons from Industry. Journal of Systems and Software 2021, 178, 110970.
17. Kaleem, M.; Kasichainula, K.; Karanjai, R.; Xu, L.; Gao, Z.; Chen, L., et al. An Event-Driven Framework for Smart Contract Execution. DEBS '21: Proceedings of the 15th ACM International Conference on Distributed and Event-Based Systems, 2021.
18. Sankaran, A.; Detterer, P.; Kannan, K.; Alachiotis, N.; Corradi, F. An Event-Driven Recurrent Spiking Neural Network Architecture for Efficient Inference on FPGA. ICONS '22: Proceedings of the International Conference on Neuromorphic Systems, 2022.
19. Pour, S.; Balouek, D.; Masoumi, A.; Brynskov, M. An Urgent Computing Oriented Architecture for Dynamic Climate Risk Management Framework. UCC '23: Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing, 2023.
20. Wohrer, M.; Zdun, U.; Rinderle-Ma, S. Architecture Design of Blockchain-Based Applications. 2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS), 2021.
21. González-Pérez, A.; Matey-Sanz, M.; Granell, C.; Díaz-Sanahuja, L.; Bretón-López, J.; Casteleyn, S. AwarNS: A Framework for Developing Context-Aware Reactive Mobile Applications for Health and Mental Health. Journal of Biomedical Informatics 2023, 141, 104359.
22. Raj, P.; Lin, J.W. Stepping into the Digitally Instrumented and Interconnected Era. In: Advances in Computers, 2020, pp. 1–34.
23. Resner, D.; De Araujo, G.M.; Fröhlich, A.A. Design and Implementation of a Cross-Layer IoT Protocol. Science of Computer Programming 2018, 165, 24–37.
24. Pérez-Muñoz, Á.G.; Gamazo-Real, J.C.; González-Bárcena, D.; Zamorano, J. Design and Implementation of a Real-Time Onboard System for a Stratospheric Balloon Mission Using Commercial Off-the-Shelf Components and a Model-Based Approach. Computers & Electrical Engineering 2023, 111, 108953.
25. Cafasso, M.; Tarral, M. Designing Flexible Sandboxing Solutions to Adapt to New Malware Trends. Computer Fraud & Security 2018, 2018(2), 5–9.
26. Figueira, J.; Coutinho, C. Developing Self-Adaptive Microservices. Procedia Computer Science 2024, 232, 264–273.
27. Dinh-Tuan, H.; Mora-Martinez, M.; Beierle, F.; Garzon, S.R. Development Frameworks for Microservice-Based Applications: Evaluation and Comparison. ESSE '20: Proceedings of the 2020 European Symposium on Software Engineering, 2020.
28. Cassell, B.; Szepesi, T.; Summers, J.; Brecht, T.; Eager, D.; Wong, B. Disk Prefetching Mechanisms for Increasing HTTP Streaming Video Server Throughput. ACM

Transactions on Modeling and Performance Evaluation of Computing Systems 2018, 3, 1–30.
29. Kolvig-Raun, E.S.; Kjærgaard, M.B.; Brorsen, R. EDDE: An Event-Driven Data Exchange to Accurately Introspect Cobot Applications. Workshop on Robotics Software Engineering (RoSE), 2023.
30. Schipor, O.A.; Vatavu, R.D.; Vanderdonckt, J. Euphoria: A Scalable, Event-Driven Architecture for Designing Interactions Across Heterogeneous Devices in Smart Environments. Information and Software Technology 2019, 109, 43–59.
31. Trabelsi, N.; Politowski, C.; Boussaidi, G.E. Event Driven Architecture: An Exploratory Study on The Gap between Academia and Industry. IEEE/ACM International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT), 2023.
32. Leveling, J.; Weickhmann, L.; Nissen, C.; Kirsch, C. Event-Driven Architecture for Sensor Data Integration for Logistics Services. IEEE International Conference on Industrial Engineering and Engineering Management, 2018.
33. Rahmatulloh, A.; Nugraha, F.; Gunawan, R.; Darmawan, I. Event-Driven Architecture to Improve Performance and Scalability in Microservices-Based Systems. Advancement in Data Science, E-learning and Information Systems (ICADEIS), International Conference on, 2022.
34. Rodríguez, D.; Thangarajah, J., Editors. Explainable Agents (XAg) by Design. AAMAS ’24: Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, 2024.
35. Chondamrongkul, N.; Sun, J.; Warren, I. Formal Security Analysis for Software Architecture Design: An Expressive Framework to Emerging Architectural Styles. Science of Computer Programming 2021, 206, 102631.
36. Zeigler, B.P.; Mittal, S.; Traore, M.K. Fundamental Requirements and DEVS Approach for Modeling and Simulation of Complex Adaptive System of Systems: Healthcare Reform. MSCIAAS ’18: Proceedings of the Symposium on Modeling and Simulation of Complexity in Intelligent, Adaptive and Autonomous Systems, 2018.
37. Chetitah, M.; Müller, J.; Deserno, L.; Waltmann, M.; Von Mammen, S. Gamification Framework for Reinforcement Learning-Based Neuropsychology Experiments. FDG ’23: Proceedings of the 18th International Conference on the Foundations of Digital Games, 2023.
38. AboElHassan, A.; Sakr, A.H.; Yacout, S. General Purpose Digital Twin Framework Using Digital Shadow and Distributed System Concepts. Computers & Industrial Engineering 2023, 183, 109534.
39. Krämer, M. GeoRocket: A Scalable and Cloud-Based Data Store for Big Geospatial Files. SoftwareX 2020, 11, 100409.
40. Kougkas, A.; Devarajan, H.; Sun, X.H. Hermes. A Heterogeneous-Aware Multi-Tiered Distributed I/O Buffering System. HPDC ’18: Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing, 2018.

41. Gómez, H.D.; Garcia-Rodriguez, J.; Azorin-Lopez, J.; Tomás, D.; Fuster-Guillo, A.; Mora-Mora, H. IA-CPS: Intelligent Architecture for Cyber-Physical Systems Management. Journal of Computational Science 2021, 53, 101409.
42. Alonso, R.; Locci, R.; Recupero, D.R. Improving Digital Twin Experience through Big Data, IoT and Social Analysis: An Architecture and a Case Study. Heliyon 2024, 10(2), e24741.
43. Seiger, R.; Malburg, L.; Weber, B.; Bergmann, R. Integrating Process Management and Event Processing in Smart Factories: A Systems Architecture and Use Cases. Journal of Manufacturing Systems 2022, 63, 575–592.
44. Łaskawiec, S.; Choraś, M.; Kozik, R.; Varadarajan, V. Intelligent Operator: Machine Learning Based Decision Support and Explainer for Human Operators and Service Providers in the Fog, Cloud and Edge Networks. Journal of Information Security and Applications 2021, 56, 102685.
45. Liu, J.; Gong, B.; Yang, L. Investigation and Implementation of Digital Software Architecture Based on Internet of Things. Measurement Sensors 2024, 101114.
46. Barriga, A.; Barriga, J.A.; Moñino, M.J.; Clemente, P.J. IoT-Based Expert System for Fault Detection in Japanese Plum Leaf-Turgor Pressure WSN. Internet of Things 2023, 23, 100829.
47. Caviglione, L.; Mazurczyk, W.; Repetto, M.; Schaffhauser, A.; Zuppelli, M. Kernel-Level Tracing for Detecting Stegomalware and Covert Channels in Linux Environments. Computer Networks 2021, 191, 108010.
48. Macdonald, J.P.; Mallick, R.; Wollaber, A.B.; Peña, J.D.; McNeese, N.; Siu, H.C. Language, Camera, Autonomy! Prompt-Engineered Robot Control for Rapidly Evolving Deployment. HRI '24: Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction, 2024.
49. Srivastava, S.; Jain, A.; Nair, P.M.; Sridharan, P. MACE Camera Controller Embedded Software: Redesign for Robustness and Maintainability. Astronomy and Computing 2020, 30, 100358.
50. Fertier, A.; Martin, G.; Barthe-Delanoë, A.M.; Lesbegueries, J.; Montarnal, A.; Truptil, S.; et al. Managing Events to Improve Situation Awareness and Resilience in a Supply Chain. Computers in Industry 2021, 132, 103488.
51. Pielmeier, J.; Theumer, P.; Schutte, C.S.L.; Snyman, S.; Bessdo, O.; Braunreuther, S.; et al. Method for Event-Based Production Control. Procedia CIRP 2019, 79, 373–378.
52. Arnedo, R.; Henao, N.; Agbossou, K.; Oviedo-Cepeda, J.C.; Dominguez, J.A.; Toquica, D. OpenATE: A Distributed Co-Simulation Engine for Transactive Energy Systems. IEEE International Conference on Smart Energy Grid Engineering (SEGE) 2023.
53. Adila, R.; Nusantara, A.B.; Yuhana, U.L. Optimization Techniques for Data Consistency and Throughput Using Kafka Stateful Stream Processing. International Seminar on Research of Information Technology and Intelligent Systems (ISRITI) 2023.

54. Mathew, A.; Andrikopoulos, V.; Blaauw, F.J.; Karastoyanova, D. Pattern-Based Serverless Data Processing Pipelines for Function-as-a-Service Orchestration Systems. Future Generation Computer Systems 2024, 154, 87–100.
55. Meißner, D.; Erb, B.; Kargl, F. Performance Engineering in Distributed Event-Sourced Systems. DEBS '18: Proceedings of the 12th ACM International Conference on Distributed and Event-Based Systems, 2018.
56. Woodside, M. Performance Models of Event-Driven Architectures. ICPE '21: Companion of the ACM/SPEC International Conference on Performance Engineering, 2021.
57. Pinnecke, M. Product-Lining the Elinvar Wealthtech Microservice Platform. SPLC '21: Proceedings of the 25th ACM International Systems and Software Product Line Conference - Volume B, 2021.
58. Alsaqaf, W.; Daneva, M.; Wieringa, R. Quality Requirements Challenges in the Context of Large-Scale Distributed Agile: An Empirical Study. Information and Software Technology 2019, 110, 39–55.
59. Weerasinghe, S.; Zaslavsky, A.; Loke, S.W.; Medvedev, A.; Abken, A.; Hassani, A.; et al. Reinforcement Learning Based Approaches to Adaptive Context Caching in Distributed Context Management Systems. ACM Transactions on Internet of Things 2024, 5(2), 1–32.
60. Illarramendi, M.; Etxeberria, L.; Elkorobarrutia, X.; Sagardui, G. Runtime Observable and Adaptable UML State Machines. SAC '19: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, 2019.
61. Alulema, D.; Criado, J.; Iribarne, L.; Fernández-García, A.J.; Ayala, R. SI4IoT: A Methodology Based on Models and Services for the Integration of IoT Systems. Future Generation Computer Systems 2023, 143, 132–151.
62. Kuhn, M.; Franke, J. Smart Manufacturing Traceability for Automotive E/E Systems Enabled by Event-Driven Microservice Architecture. International Conference on Mechanical and Intelligent Manufacturing Technologies (ICMIMT) 2020.
63. Umer, M.; Mahesh, B.; Hanson, L.; Khabbazi, M.; Onori, M. Smart Power Tools: An Industrial Event-Driven Architecture Implementation. Procedia CIRP 2018, 72, 1357–1362.
64. Scherer, M.; Menachery, K.; Magno, M. SmartAid: A Low-Power Smart Hearing Aid For Stutterers. IEEE Sensors Applications Symposium, SAS, 2019.
65. Chondamrongkul, N.; Sun, J.; Warren, I. Software Architectural Migration. ACM Transactions on Software Engineering and Methodology 2021, 30(4), 1–35.
66. Tragatschnig, S.; Stevanetic, S.; Zdun, U. Supporting the Evolution of Event-Driven Service-Oriented Architectures Using Change Patterns. Information and Software Technology 2018, 100, 133–146.
67. Liu, Z.; Sampaio, P.; Pishchulov, G.; Mehandjiev, N.; Cisneros-Cabrera, S.; Schirrmann, A.; et al. The Architectural Design and Implementation of a Digital Platform for Industry 4.0 SME Collaboration. Computers in Industry 2022, 138, 103623.

68. Henning, S.; Hasselbring, W. TheodoLite: Scalability Benchmarking of Distributed Stream Processing Engines in Microservice Architectures. Big Data Research 2021, 25, 100209.
69. Horozal, F.; Reimer, P.; Scholze, S. Tool Support for Architectural Pattern Selection and Application in Cloud-Centric Service-Oriented IDEs. ESAAM '23: Proceedings of the 3rd Eclipse Security, AI, Architecture and Modelling Conference on Cloud to Edge Continuum, 2023.
70. Ketoma, V.K.; Vanderdonckt, J.; Meixner, G. Towards Flexible Authoring and Personalization of Virtual Reality Applications for Training. Proceedings of the ACM on Human-Computer Interaction 2023, 7(EICS), 1–37.
71. Chaves, A.J.; Martín, C.; Díaz, M. Towards Flexible Data Stream Collaboration: Federated Learning in Kafka-ML. Internet of Things 2024, 25, 101036.
72. Arjona, A.; López, P.G.; Sampé, J.; Slominski, A.; Villard, L. Triggerflow: Trigger-Based Orchestration of Serverless Workflows. Future Generation Computer Systems 2021, 124, 215–229.
73. Karsten, M.; Barghi, S. User-Level Threading. Proceedings of the ACM on Measurement and Analysis of Computing Systems 2020, 4(1), 1–30.
74. Drost, S.; Vogt, A.; Danowski-Buhren, C.; Jirka, S.; Kirstein, V.; Pakzad, K.; et al. WaCoDiS: Automated Earth Observation Data Processing within an Event-Driven Architecture for Water Monitoring. Computers & Geosciences 2022, 159, 105003.
75. Agrahari, V.; Chimalakonda, S. What's Inside Unreal Engine? - A Curious Gaze! ISEC '21: Proceedings of the 14th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference), 2021.
76. Sriraman, A.; Wenisch, T.F. μtune: Auto-Tuned Threading for OLDI Microservices. Operating Systems Design and Implementation, 2018, 177–194.
77. Lercher, A.; Glock, J.; Macho, C.; Pinzger, M. Microservice API Evolution in Practice: A Study on Strategies and Challenges. Journal of Systems and Software 2024, 215, 112110.
78. Juric, M.B. WSDL and BPEL Extensions for Event Driven Architecture. Information and Software Technology 2010, 52(10), 1023–1043.
79. Kitchenham, B. Guidelines for Performing Systematic Literature Reviews in Software Engineering. ResearchGate, 2007.
80. Roda, C.; Navarro, E.; Zdun, U.; López-Jaquero, V.; Simhandl, G. Past and Future of Software Architectures for Context-Aware Systems: A Systematic Mapping Study. Journal of Systems and Software 2018, 146, 310–355.
81. Waseem, M.; Liang, P.; Shahin, M. A Systematic Mapping Study on Microservices Architecture in DevOps. Journal of Systems and Software 2020, 170, 110798.
82. Singjai, A.; Zdun, U.; Zimmermann, O.; Pautasso, C. Patterns on deriving apis and their endpoints from domain models. In Proceedings of the 26th European Conference on Pattern Languages of Programs 2021, pp.1-15.

Apêndice A

A seguir, a tabela 10, apresenta uma relação de estudos selecionados para mapeamento sistemático contendo id, título, Autores, ano de publicação, quantidade de citações e as ferramentas utilizadas.

Tabela 10. Estudos selecionados para mapeamento sistemático.

ID	Título	Autores	Ano Publicação	Tipo Publicação	Citações	Caso de Aplicação	Ferramenta
EP1	A container-based cloud-native architecture for the reproducible execution of multi-population optimization algorithms [10]	Mario García Valdez, Juan J. Merelo Guervós	2021	Periódico	18	O estudo descreve a implementação de um sistema baseado em filas de mensagens, onde componentes como funções sem estado (stateless functions) atuam como consumidores e produtores, processando e enviando populações entre diferentes etapas do algoritmo.	RabbitMQ
EP2	A Hybrid Approach for Digital Representation of Sensors in Real-Time Applications [11]	Karthik Shenoy Panambur and Shantanoo Desai and Amit Kumar Singh and Klaus-Dieter Thoben	2020	Periódico	1	O trabalho apresenta a emulação de atributos digitais de dispositivos periféricos, como sensores, onde eventos são gerenciados por uma arquitetura de software orientada a eventos. Essa arquitetura é usada para modularizar e escalar o sistema, bem como para processar comportamentos como leituras/escritas em registradores e interrupções.	
EP3	Adaptive Event Driven Framework for Real Time Multi-Agent Missions [12]	J.A. Bonache-Seco, J.A. Lopez-Orozco, Eva Besada	2018	Conferência	4	O estudo descreve a aplicação da EDA em um framework adaptativo para o controle e monitoramento de veículos não tripulados (UVs) em missões complexas, como exemplificado no uso de um Veículo	-

		Portas, José L. Risco Martín				de Superfície Não Tripulado (USV) em um experimento de simulação e operação real.	
EP4	Adaptive Interactive Displaying System for In-Vehicle Use [13]	Dragos F. Sburlan, Elena Bautu, Crenguta M. Puchianu e Dorin Mircea Popovici	2020	Periódico	4	O estudo apresenta um caso prático de aplicação de arquitetura orientada a eventos. A arquitetura é utilizada para gerenciar a comunicação e interação entre módulos em um sistema veicular, especialmente para monitorar a orientação da cabeça do motorista e adaptar a exibição de informações na interface do veículo em tempo real.	Euphoria
EP5	Alert notifications for governmental disaster response via instant messaging applications [14]	Hao-Yung Chan e Meng-Han Tsai	2023	Periódico	1	O estudo discute a implementação de agentes conversacionais para a Agência de Recursos Hídricos (Water Resources Agency - WRA) e o Departamento de Minas (Bureau of Mines) de Taiwan, ambos afetados por desastres naturais e necessitando de soluções para gerenciar o excesso de informações durante esses eventos. A arquitetura Rich Notification Architecture (RNA) foi utilizada para a entrega de notificações e informações relevantes para as equipes dessas organizações, demonstrando a aplicação prática da arquitetura orientada a eventos na gestão de desastres.	-
EP6	An active perceivable device–oriented modeling framework for hydropower plant simulation [15]	Binqiao Zhang, Xiaohui Yuan, Yanbin Yuan e Xu Wang	2018	Periódico	2	O estudo aborda a aplicação de event-driven architecture no contexto da simulação de usinas hidrelétricas. O trabalho discute os desafios da modelagem e simulação de usinas hidrelétricas de grande escala, destacando as limitações	NServiceBus, MSMQ e RabbitMQ.

						das abordagens tradicionais. O trabalho explica como a EDA pode ser integrada com a SOA para superar as dificuldades associadas à orquestração centralizada e ao acoplamento forte de serviços, típicos da SOA. O foco está na proposta de um novo método de modelagem, o "Active Perceivable Device-Oriented Modeling" (APDOM), que utiliza características da EDA para melhorar a simulação de usinas hidrelétricas, mas sem citar um caso prático específico onde a EDA foi aplicada.	
EP7	An empirical characterization of event sourced systems and their schema evolution — Lessons from industry [16]	Michiel Overeem, Marten Spoor, Slinger Jansen e Sjaak Brinkkemper	2021	Periódico	15	-	-
EP8	An event driven framework for smart contract execution [17]	Kaleem, Mudabbir and Kasichainula, Keshav and Karanjai, Rabimba and Xu, Lei and Gao, Zhimin and Chen, Lin e Shi, Weidong	2021	Periódico	12	O estudo descreve um caso prático aplicado à Event-Driven Architecture (EDA) no contexto de smart contracts na plataforma Ethereum. A proposta é melhorar a implementação dos contratos inteligentes e do sistema de blockchain através de uma arquitetura orientada por eventos.	-
EP9	An Event-driven Recurrent Spiking Neural Network	Sankaran, Anand and Detterer, Paul	2022	Periódico	4	O estudo apresenta um caso prático de aplicação de uma arquitetura orientada a eventos. Eles demonstram a aplicação	-

	Architecture for Efficient Inference on FPGA [18]	and Kannan, Kalpana and Alachiotis, Nikolaos e Corradi, Federico				dessa arquitetura em dois benchmarks: reconhecimento de dígitos no conjunto de dados MNIST e a fusão sensorial de radar e imagem para a classificação de terras agrícolas. Esses casos práticos são utilizados para demonstrar como a arquitetura proposta pode ser aplicada em cenários reais, focando na eficiência e na redução do consumo de recursos, conforme mencionado no resumo do documento.	
EP10	An urgent computing oriented architecture for dynamic climate risk management framework [19]	Shahrzad Pour, Daniel Balouek, Amir Masoumi e Martin Brynskov	2024	Periódico	0	O estudo apresenta a arquitetura para um sistema de gerenciamento de riscos dinâmico e de segurança rodoviária que utiliza uma arquitetura orientada a eventos como seu modelo de coordenação e comunicação.	-
EP11	Architecture Design of Blockchain-Based Applications [20]	Maximilian Wöhrer, Uwe Zdun e Stefanie Rinderle-Ma	2021	Conferência	14	O trabalho cita a utilização da arquitetura dirigida a eventos no contexto de blockchain para comunicação assíncrona de transações e execução de smart contact.	ActiveMQ, RabbitMQ, Kafka e Pulsar
EP12	AwarNS: A framework for developing context-aware reactive mobile applications for health and mental health [21]	Alberto González-Pére z, Miguel Matey-Sanz, Carlos Granell, Laura Díaz-Sanahuja , Juana Bretón-López	2023	Periódico	7	-	-

		e Sven Casteleyn					
EP13	Stepping into the digitally instrumented and interconnected era [22]	Pethuru Raj e Jenn-Wei Lin	2020	Capítulos de Livros	1	-	-
EP14	Design and implementation of a cross-layer IoT protocol [23]	Davi Resner, Gustavo Medeiros de Araujo e Augusto Fröhlich	2018	Periódico	32	O estudo apresenta a aplicação de uma arquitetura dirigida por eventos na implementação do protocolo Trustful Space-Time Protocol (TSTP) que utiliza técnicas de metaprogramação para implementar uma arquitetura dirigida por eventos que movimenta pacotes armazenados em buffers com metadados, sem cópias desnecessárias e eliminando dependências desnecessárias.	-
EP15	Design and implementation of a real-time onboard system for a stratospheric balloon mission using commercial off-the-self components and a model-based approach [24]	Ángel-Grover Pérez-Muñoz, Jose-Carlos Gamazo-Real, David González-Bárcena e Juan Zamorano	2023	Periódico	2	A arquitetura orientada a eventos é mencionada na seção que descreve padrões arquitetônicos e de design para RTES. Essa arquitetura é caracterizada por um "processador de eventos" central, que recebe, processa e redireciona os eventos para os assinantes. A configuração é estática, visando aumentar a previsibilidade e assegurar a operação correta em sistemas de tempo real.	-
EP16	Designing flexible sandboxing solutions to adapt to new malware trends [25]	Matteo Cafasso e Mathieu Tarral	2018	Periódico	5	O trabalho apresenta o desenvolvimento de uma plataforma de sandboxing, chamada Sandboxed Execution Environment (SEE), que utiliza uma arquitetura orientada a eventos para gerenciar e controlar a execução e análise de amostras. Nessa arquitetura, eventos	-

						gerados dentro do ambiente de sandbox são tratados por diferentes plugins forenses, que reagem a esses eventos conforme eles ocorrem. Isso demonstra a aplicação real do padrão em um sistema de sandbox para análise de segurança.	
EP17	Developing self-adaptive microservices [26]	João Figueira e Carlos Coutinho	2024	Periódico	1	O trabalho apresenta o uso de Dapr e KEDA em uma arquitetura proposta para microsserviços auto-adaptáveis. Especificamente, Dapr é utilizado para comunicação entre serviços através do padrão de publicação e assinatura (publish and subscribe), e KEDA é usado para escalar horizontalmente com base em gatilhos de eventos.	Azure Service Bus, RabbitMQ, AWS SNS/SQS
EP18	Development Frameworks for Microservice-based Applications: Evaluation and Comparison [27]	Hai Dinh-Tuan, Maria Mora-Martinez, Felix Beierle e Sandro Rodriguez Garzon	2020	Periódico	26	O estudo descreve um sistema de pedágio sensível à poluição que adota uma arquitetura microservices orientada a eventos. Nesse sistema, diferentes microsserviços se comunicam por meio de mensagens JSON através de um message broker, utilizando o padrão de mensageria assíncrona publish-subscribe.	-
EP19	Disk Prefetching Mechanisms for Increasing HTTP Streaming Video Server Throughput [28]	Cassell, Benjamin and Szepesi, Tyler and Summers, Jim and Brecht, Tim and Eager, Derek e Wong, Bernard	2018	Periódico	9	-	-

EP20	EDDE: An Event-Driven Data Exchange to Accurately Introspect Cobot Applications [29]	Emil Stubbe Kolvig-Raun, Mikkel Baun Kjærgaard, Ralph Brorsen	2023	Conferência	2	O trabalho apresenta um caso prático de aplicação da arquitetura orientada a eventos (event-driven architecture, EDA). Especificamente, ele descreve a proposta, implementação e avaliação da "Event-Driven Data Exchange (EDDE)" como uma substituição ao modelo clássico baseado em frequência para geração de dados em aplicações robóticas. A implementação é apresentada no contexto de sistemas ciberfísicos (CPSs), com ênfase na introspecção e análise de eventos de execução de programas em cobots (robôs colaborativos).	-
EP21	Euphoria: A Scalable, event-driven architecture for designing interactions across heterogeneous devices in smart environments [30]	Ovidiu-Andrei Schipor, Radu-Daniel Vatavu e Jean Vanderdonckt	2019	Periódico	79	O estudo apresenta a arquitetura Euphoria, uma nova proposta de arquitetura com abordagem event-driven para desenvolvimento de software. O trabalho apresenta três cenários de aplicação do Euphoria: interação com displays públicos usando gestos e dispositivos vestíveis, acesso a conteúdo digital ancorado a locais físicos em um ambiente inteligente, e uma interface de usuário vestível para notificações sociais em um ambiente veicular. Cada cenário ilustra como a arquitetura foi aplicada para implementar interações em ambientes inteligentes.	-
EP22	Event Driven Architecture: An Exploratory Study on The Gap between Academia and Industry [31]	Nader Trabelsi, Cristiano Politowski,	2023	Conferência	1	-	Apache Kafka, AWS EventBridge, GCP Eventarc

		Ghizlane El Boussaidi					
EP23	Event-driven Architecture for Sensor Data Integration for Logistics Services [32]	Jens Leveling, Luise Weickhmann, Christian Nissen e Christopher Kirsch	2018	Conferência	5	O trabalho apresenta uma aplicação prática da Event-Driven Architecture. Ele descreve como a arquitetura foi implementada em gateways IoT baseados em x86 e ARM para integrar dados de diferentes dispositivos sensores de um sistema de fluxo de materiais. A arquitetura foi utilizada para coletar dados com o objetivo de realizar análises posteriores, como otimização de processos e manutenção.	Mosquitto
EP24	Event-Driven Architecture to Improve Performance and Scalability in Microservices-Based Systems [33]	Alam Rahmatulloh, Fuji Nugraha, Rohmat Gunawan e Irfan Darmawan	2022	Conferência	10	O trabalho descreve um sistema de microsserviços utilizado em um processo de negócio de uma loja de tecidos, onde a arquitetura EDA é empregada para gerenciar a comunicação assíncrona entre os serviços principais por meio de um serviço de streaming de eventos, o NATS Streaming.	Apache Kafka, RabbitMQ e NATS Streaming
EP25	Explainable Agents (XAg) by Design [34]	Sebastian Rodriguez e John Thangarajah	2024	Periódico	1	O estudo apresenta o pattern TriQPAN (Trigger, Query, Process, Action and Notify), no qual utiliza-se de uma arquitetura baseada em eventos. O pattern possui como objetivo garantir que os processos de decisão são registrados para explicar os comportamentos de agentes.	Event Store DB (banco de dados para armazenar eventos)
EP26	Formal security analysis for software architecture design: An expressive framework to emerging architectural styles [35]	Nacha Chondamrongkul, Jing Sun e Ian Warren	2021	Periódico	6	O estudo apresenta um exemplo chamado "Life Net", que envolve a aplicação de arquiteturas baseadas em microservices, arquitetura orientada a eventos e tecnologia blockchain. Especificamente, o	-

						estudo descreve como o sistema Life Net utiliza a arquitetura orientada a eventos para gerenciar emergências médicas para pacientes com Alzheimer. O sistema registra eventos, como pedidos de socorro de pacientes, que são processados por serviços de microservices e armazenados em um log de eventos, permitindo que outros componentes, como os serviços Lifeguard e Lifecare, respondam a esses eventos.	
EP27	Fundamental requirements and DEVS approach for modeling and simulation of complex adaptive system of systems: healthcare reform [36]	Bernard P. Zeigler, Saurabh Mittal e Mamadou K. Traore	2018	Conferência	9	O trabalho descreve a implementação de uma arquitetura baseada em eventos para dar suporte ao modelo Pathways de coordenação de cuidados em sistemas de saúde. Essa arquitetura é usada para gerenciar a dinâmica de eventos discretos, como agendamento de tempo, transições de estado, e entrada/saída de dados dentro do sistema de coordenação de cuidados.	-
EP28	Gamification Framework for Reinforcement Learning-based Neuropsychology Experiments [37]	Mounsif Chetitah, Julian Müller, Lorenz Deserno, Maria Waltmann e Sebastian von Mammen.	2023	Conferência	4	O estudo descreve a implementação de uma arquitetura baseada em eventos para integrar quatro elementos: a descrição do experimento, o modelo de aprendizado por reforço (RL), o design do jogo e o módulo de coleta de dados. Essa arquitetura permite a criação de experimentos neuropsicológicos gamificados, facilitando a integração e a execução dos experimentos.	-
EP29	General purpose digital twin framework using digital	Ayman AboElHassan,	2023	Periódico	7	O trabalho descreve a arquitetura de um Digital Twin (DT) que utiliza conceitos de	Kafka

	shadow and distributed system concepts [38]	Ahmed H. Sakr e Soumaya Yacout				sistemas distribuídos, incluindo comunicação via canais Pub/Sub, que são fundamentais em arquiteturas orientadas a eventos. Esses elementos indicam uma abordagem que pode ser considerada compatível com EDA	
EP30	GeoRocket: A scalable and cloud-based data store for big geospatial files [39]	Michel Krämer	2020	Relatórios Técnicos	11	O caso prático apresentado no estudo é o uso do GeoRocket para o gerenciamento de grandes conjuntos de dados geoespaciais na nuvem. GeoRocket utiliza uma arquitetura reativa e baseada em eventos, onde componentes independentes (verticles) se comunicam através de um barramento de eventos para realizar importação, indexação e consulta de dados.	Vert.x
EP31	Hermes: a heterogeneous-aware multi-tiered distributed I/O buffering system [40]	Kougkas, Anthony and Devarajan, Hariharan e Sun, Xian-He	2018	Periódico	94	O trabalho apresenta um novo sistema I/O distribuído, dinâmico, multicamadas e com consciência da heterogeneidade. O sistema utiliza uma camada de buffer baseada em uma arquitetura orientada a eventos afim de evitar perdas de dados.	-
EP32	IA-CPS: Intelligent architecture for cyber-physical systems management [41]	Henry Duque Gómez, Jose Garcia-Rodriguez, Jorge Azorin-Lopez, David Tomás, Andres Fuster-Guillo e Higinio Mora-Mora	2021	Periódico	3	O estudo apresenta dois casos de uso que utilizam a arquitetura orientada a eventos: Um sistema de vigilância por vídeo que realiza análise inteligente em tempo real de imagens e áudio, gerando alarmes automáticos para melhorar a resposta e a tomada de decisão. Arquitetura baseada em microsserviços e protocolos de mensageria para gerenciar sensores em ambientes inteligentes e conectados à	RabbitMQ e Mosquitto

						nuvem, permitindo que empresas de serviços e clientes monitorem o consumo de energia e água através de dispositivos sensores, com acesso por meio de uma interface web.	
EP33	Improving digital twin experience through big data, IoT and social analysis: An architecture and a case study [42]	Rubén Alonso, Riccardo Locci e Diego Reforgiato Recupero	2024	Periódico	1	O caso prático apresentado é a implementação de um "Digital Twin" para um escritório da empresa italiana R2M Solution. Esse sistema integra diversos sensores, dados de redes sociais, informações de APIs de terceiros e dados processados da empresa para criar um ambiente virtual imersivo e interativo que representa o espaço físico real.	Apache kafka e Apache Spark
EP34	Integrating process management and event processing in smart factories: A systems architecture and use cases [43]	Ronny Seiger, Lukas Malburg, Barbara Weber e Ralph Bergmann	2022	Periódico	43	O estudo descreve a utilização de uma arquitetura baseada em eventos para integrar dados em tempo real de sensores IIoT com sistemas de gestão de processos de negócios (WfMS). A arquitetura permite o monitoramento e a ativação de eventos complexos no nível de processos de negócios.	Siddhi
EP35	Intelligent operator: Machine learning based decision support and explainer for human operators and service providers in the fog, cloud and edge networks [44]	Sebastian Łaskawiec, Michał Choraś, Rafał Kozik e Vijayakumar Varadarajan	2021	Periódico	12	O estudo descreve a aplicação da arquitetura orientada a eventos em ambientes de nuvem baseados em contêineres, como Kubernetes e OpenShift. Especificamente, ele discute como o Kubernetes utiliza controladores para reagir às mudanças de estado dos objetos no API Server através de um mecanismo de notificação chamado "Watch".	-

EP36	Investigation and implementation of digital software architecture based on internet of things [45]	Jie Liu, Boxiang Gong e Lan Yang	2024	Periódico	1	-	-
EP37	IoT-based expert system for fault detection in Japanese Plum leaf-turgor pressure WSN [46]	Arturo Barriga, José A. Barriga, María José Moñino e Pedro J. Clemente a	2023	Periódico	3	O estudo apresenta a implementação de um sistema IoT para detecção de falhas em sensores de pressão de turgor das folhas. Neste sistema, o MQTT é utilizado para a comunicação entre camadas e componentes, permitindo uma arquitetura baseada em eventos para gerenciar dados e detecção de falhas.	Mosquitto
EP38	Kernel-level tracing for detecting stegomalware and covert channels in Linux environments [47]	Luca Caviglione, Wojciech Mazurczyk, Matteo Repetto, Andreas Schaffhausere Marco Zuppelli	2021	Periódico	40	-	-
EP39	Language, Camera, Autonomy! Prompt-engineered Robot Control for Rapidly Evolving Deployment [48]	Jacob P. Macdonald, Rohit Mallick, Allan B. Wollaber, Jaime D. Peña, Nathan McNeese e Ho Chit Siu	2024	Periódico	2	O estudo apresenta um caso prático de aplicação de arquitetura orientada a eventos. A seção "CLEAR_worker_server" descreve como os serviços do CLEAR utilizam uma arquitetura orientada a eventos, aguardando requisições HTTP POST para entregar dados de entrada e emitir sinais. Eventos específicos acionam processos de trabalho que produzem dados utilizáveis pelo coordenador.	-

EP40	MACE camera controller embedded software: Redesign for robustness and maintainability [49]	S. Srivastava, A. Jain, P.M. Nair e P. Sridharan	2020	Periódico	4	-	
EP41	Managing events to improve situation awareness and resilience in a supply chain [50]	Audrey Fertier, Guillaume Martin, Anne-Marie Barthe-Delanoë, Julien Lesbegueries, Aurélie Montarnal, Sébastien Truptil, Frédérick Bénaben e Nicolas Salatgé	2021	Periódico	23	O estudo testa os benefícios do sistema de informação AIC (que utiliza uma arquitetura orientada a eventos e modelos) em uma cadeia de suprimentos farmacêutica na França. Ele monitora e interpreta eventos em tempo real para melhorar a resiliência da cadeia de suprimentos, detectando novos riscos e incidentes.	-
EP42	Method for event-based production control [51]	Julia Pielmeier, Philipp Theumer, Corné S.L. Schutte, Stephan Snyman, Olaf Bessdo, Stefan Braunreuther e Gunther Reinhart	2019	Periódico	6	-	-

EP43	Microservice API Evolution in Practice: A Study on Strategies and Challenges [77]	Alexander Lercher, Johann Glock, Christian Macho e Martin Pinzger	2024	Periódico	4	-	RabbitMQ e Apache Kafka
EP44	On the impact of event-driven architecture on performance: An exploratory study [6]	Hebert Cabane e Kleinner Farias	2024	Periódico	10	O estudo realiza um estudo exploratório em volta da arquitetura orientada a eventos e a arquitetura monolítica. Para isso, o trabalho implementou uma plataforma de e-commerce utilizando as duas arquiteturas com o objetivo de avaliar métricas que indicam desempenho.	RabbitMQ e Azure Service Bus
EP45	OpenATE: A Distributed Co-simulation Engine for Transactive Energy Systems [52]	Rafael Arnedo, Nilson Henao, Kodjo Agbossou, Juan Carlos Oviedo-Cepeda, Juan Antonio Dominguez e David Toquica	2023	Conferência	0	O estudo descreve a implementação da OpenATE (Open-source Automated Transactive Energy Engine), que utiliza uma arquitetura orientada a eventos para controlar simulações e gerenciar mensagens de transação. O sistema permite que cada módulo opere de forma autônoma, abordando problemas de coordenação do tempo simulado, o que é um exemplo claro de como a arquitetura orientada a eventos pode ser aplicada em um cenário de simulação para sistemas de energia transacional (TES).	-
EP46	Optimization Techniques for Data Consistency and Throughput Using Kafka Stateful Stream Processing [53]	Rida Adila, Adetiya Bagus Nusantara, Umi Laili Yuhana	2023	Conferência	0	-	Apache Kafka

EP47	Pattern-based serverless data processing pipelines for Function-as-a-Service orchestration systems [54]	Anil Mathew, Vasilios Andrikopoulos, Frank J. Blaauw e Dimka Karastoyanova	2024	Periódico	1	O estudo menciona a aplicação prática de uma arquitetura orientada a eventos ao descrever o uso do Zeebe, um motor de orquestração de workflows baseado em BPMN para microsserviços e aplicações serverless. O Zeebe usa uma arquitetura orientada a eventos por meio de eventos de mensagem para coordenar essas aplicações.	SQS e SNS
EP48	Patterns on Deriving APIs and their Endpoints from Domain Models [82]	Apitchaka Singjai, Uwe Zdun, Olaf Zimmermann e Cesare Pautasso	2022	Periódico	8	-	-
EP49	Performance Engineering in Distributed Event-sourced Systems [55]	Dominik Meißner, Benjamin Erb e Frank Kargl	2018	Conferência	2	-	-
EP50	Performance Models of Event-Driven Architectures [56]	Murray Woodside	2021	Periódico	3	O trabalho apresenta um sistema simplificado de e-commerce. Nele, um componente "Buy" envia uma solicitação para o componente "InventoryMgr". A solicitação é gerenciada por um EventBus, e o componente "Buy" não aguarda a resposta imediatamente. Quando a resposta é gerada, ela também é gerenciada pelo EventBus e pode ser recebida por uma instância diferente do componente "Buy". O estado da operação é salvo em um Persistence Store para garantir que a	-

						resposta seja corretamente associada à instância que fez a solicitação original.	
EP51	Product-lining the elinvar wealthtech microservice platform [57]	Marcus Pinnecke	2021	Periódico	3	O trabalho descreve a arquitetura da plataforma Elinvar, que utiliza é um event-driven architecture. O trabalho destaca o uso de eventos como uma parte essencial de sua comunicação entre serviços e a implementação de um RPC utilizando Kafka.	Apache Kafka
EP52	Quality requirements challenges in the context of large-scale distributed agile: An empirical study [58]	Wasim Alsaqaf, Maya Daneva e Roel Wieringa	2019	Periódico	119	-	-
EP53	Reinforcement Learning Based Approaches to Adaptive Context Caching in Distributed Context Management Systems [59]	Weerasinghe, Shakthi and Zaslavsky, Arkady and Loke, Seng W. and Medvedev, Alexey and Abken, Amin and Hassani, Alireza and Huang e Guang-Li	2024	Periódico	1	O trabalho descreve a aplicação de uma arquitetura baseada em eventos para agentes de caching adaptativos em um sistema distribuído. A arquitetura utiliza padrões de design como *publisher-subscriber* e *observer* para gerenciar fluxos de informações contextuais, realizar decisões de caching e calcular recompensas.	-
EP54	Runtime observable and adaptable UML state machines: models@run.time approach [60]	Miren Illarramendi, Leire Etxeberria, Xabier Elkorobarrutia	2019	Periódico	3	-	-

		e Goiuria Sagardui					
EP55	SI4IoT: A methodology based on models and services for the integration of IoT systems [61]	Darwin Alulema, Javier Criado, Luis Iribarne, Antonio Jesús Fernández-García e Rosa Ayala	2023	Periódico	13	O estudo apresenta utiliza a arquitetura no contexto de smart home para detecção de presença na casa e realizar ações em tempo real.	AMQ e Kafka
EP56	Smart Manufacturing Traceability for Automotive E/E Systems enabled by Event-Driven Microservice Architecture [62]	Marlene Kuhn e Jörg Franke	2020	Conferência	10	O trabalho descreve a aplicação prática de Event-Driven Architecture ao mencionar o uso de mensagens para notificar diferentes serviços sobre a conclusão de tarefas, como na integração entre um sistema de rastreamento de veículos e outros serviços da empresa.	-
EP57	Smart Power Tools: An Industrial Event-Driven Architecture Implementation [63]	Muhammad Umer, Bhargav Mahesh, Lars Hanson, M.R. Khabbazi e Mauro Onori	2018	Periódico	11	O estudo descreve a implementação de uma arquitetura orientada a eventos na linha de montagem de carros de pedal na Scania, Sodertalje. A arquitetura foi utilizada para transformar dados gerados pelas ferramentas de potência em tempo real e publicá-los em um barramento de mensagens (Kafka), permitindo que diferentes sistemas se inscrevessem e consumissem esses dados. A implementação do EDA facilitou a coleta e utilização dos dados do chão de fábrica em tempo real, substituindo ferramentas antigas por modernas e permitindo a comunicação eficiente entre os sistemas.	Kafka

EP58	SmartAid: A Low-Power Smart Hearing Aid For Stutterers [64]	Moritz Scherer, Kiran Menachery e Michele Magno	2019	Conferência	5	-	-
EP59	Software Architectural Migration: An Automated Planning Approach [65]	Nacha Chondamrongkul, Jing Sun e Ian Warren	2021	Periódico	6	O estudo apresenta um exemplo prático de sistema de compartilhamento de caronas (Ride-Sharing System - RSS), que é usado para demonstrar como a arquitetura orientada a eventos pode ser aplicada para melhorar aspectos como recuperação de falhas e escalabilidade. O estudo propõe a aplicação de padrões orientados a eventos, como o Event-Sourcing e CQRS, para transformar o design arquitetônico do sistema RSS.	Apache Kafka
EP60	Supporting the evolution of event-driven service-oriented architectures using change patterns [66]	Simon Tragatschnig, Srdjan Stevanetic e Uwe Zdun	2018	Periódico	19	-	-
EP61	The architectural design and implementation of a digital platform for Industry 4.0 SME collaboration [67]	Zixu Liu, Pedro Sampaio, Grigory Pishchulov, Nikolay Mehandjiev, Sonia Cisneros-Cabrera, Arnd Schirrmann,	2022	Periódico	72	O trabalho descreve a arquitetura DIGICOR, que é baseada em uma abordagem de Event-Driven Service-Oriented Architecture (EDSOA). O foco está na arquitetura e nas capacidades da plataforma para suportar a colaboração na cadeia de suprimentos e a integração de serviços através de eventos.	-

		Filip Jiru e Nisrine Bnouhanna					
EP63	Theodolite: Scalability Benchmarking of Distributed Stream Processing Engines in Microservice Architectures [68]	Sören Henning e Wilhelm Hasselbring	2021	Periódico	62	O trabalho cita quatro casos de uso para a arquitetura e executam benchmarks de escalabilidade em ambiente de nuvem. 1. Armazenamento de mensagens ou eventos processadas para consulta posterior. 2. Agregação de eventos em janelas de tempo com objetivo de reduzir a quantidade de eventos consecutivos 3. Agregação de eventos com mesmo atributo de tempo. ex: dia, hora, mês ou ano. 4. Agregação de eventos por grupos de sensores no contexto de IoT	Apache Kafka Streams e Apache Flink
EP64	Tool Support for Architectural Pattern Selection and Application in Cloud-Centric Service-Oriented IDEs [69]	Fulya Horozal, Philip Reimer e Sebastian Scholze	2023	Periódico	0	-	-
EP65	Towards Flexible Authoring and Personalization of Virtual Reality Applications for Training [70]	Vix Kemanji Ketoma, Jean Vanderdonckt e Gerrit Meixner	2023	Periódico	1	O trabalho apresenta o VRTrain framework, que segue uma arquitetura baseada em eventos, para o desenvolvimento de aplicações de treinamento em realidade virtual (VR). A arquitetura dirige os comportamentos dos objetos virtuais, as tarefas de treinamento e	-

						a coleta de dados, tudo baseado em eventos.	
EP66	Towards flexible data stream collaboration: Federated Learning in Kafka-ML [71]	Antonio Jesús Chaves, Cristian Martín e Manuel Díaz	2024	Periódico	2	O trabalho apresenta um novo módulo para o Kafka-ML com objetivo de treinar modelos de forma federada assíncrona, sendo capaz de combinar recursos de aprendizado federado e fluxo de dados em aplicativos de ML/IA.	Kafka-ML
EP67	Triggerflow: Trigger-based orchestration of serverless workflows [72]	Aitor Arjona, Pedro García López, Josep Sampé, Aleksander Slominski e Lionel Villard	2021	Periódico	79	O trabalho descreve o Triggerflow, uma arquitetura de orquestração baseada em eventos para workflows serverless. O Triggerflow utiliza uma arquitetura de Event-Condition-Action (ECA) e é projetado para lidar com eventos de diversas fontes, executar ações e criar filtros personalizados.	Kafka e RabbitMQ
EP68	User-level Threading: Have Your Cake and Eat It Too [73]	Karsten, Martin and Barghi e Saman	2020	Periódico	17	O estudo discute a programação orientada a eventos como uma alternativa ao modelo thread-per-session. A programação orientada a eventos é usada para gerenciar conjuntos de estados de sessão em resposta a eventos de I/O, e é diretamente suportada por interfaces típicas dos sistemas operacionais	kqueue e epoll
EP68	WaCoDiS: Automated Earth Observation data processing within an event-driven architecture for water monitoring [74]	Sebastian Drost, Arne Vogt, Christian Danowski-Buhren, Simon Jirka, Verena Kirstein, Kian	2022	Periódico	3	O Trabalho descreve como o WaCoDiS System implementa uma arquitetura orientada a eventos para gerenciar o processamento de dados de observação da terra (Earth Observation). Especificamente, o WaCoDiS utiliza padrões baseados em mensagens assíncronas e o protocolo AMQP para implementar fluxos de dados	Apache Kafka e RabbitMQ

		Pakzad e Matthes Rieke				orientados a eventos, que são acionados por eventos de disponibilidade de dados e outros eventos relevantes, como eventos climáticos extremos.	
EP69	What's Inside Unreal Engine? - A Curious Gaze! [75]	Vartika Agrahari e Sridhar Chimalakonda	2021	Periódico	10	O trabalho discute a utilização da arquitetura orientada a eventos no Unreal Engine, onde os componentes comunicam-se usando um modelo de publicação-inscrição (publish-subscribe) para manter a consistência de dados e reduzir o tempo de acesso.	-
EP70	µtune: auto-tuned threading for OLDI microservices [76]	Akshitha Sriraman e Thomas F. Wenisch	2018	Conferência	141	-	-