Tudo em Todo Lugar ao Mesmo Tempo: Rastreabilidade de Dados em Cidades Inteligentes por meio de Proveniência*

Maria Luiza Falci[1], Débora Pina[2], Liliane Kunstmann[3]
Vanessa Braganholo[1], Daniel de Oliveira[1]

[1]Instituto de Computação – Universidade Federal Fluminense (IC/UFF)

[2]Universidade Federal do Rio de Janeiro (COPPE/UFRJ)

[3]Mendelics Análise Genômica

marialuizafalci@id.uff.br, {vanessa,danielcmo}@ic.uff.br,
dbpina@cos.ufrj.br, liliane.kunstmann@mendelics.com.br

Resumo. *As Cidades Inteligentes (CIs) utilizam dados de diversas fontes para melhorar a qualidade de vida e apoiar políticas públicas. Garantir a rastreabilidade completa desses dados, desde a coleta até seu uso, é essencial para auditoria e confiabilidade. Dados de proveniência ajudam a representar esse caminho de derivação, mas coletá-los em CIs é complexo devido ao ecossistema variado de programas e usuários envolvidos. Para enfrentar esse desafio, foi desenvolvido o framework* `ProvInCiA`*, que captura dados de proveniência em CIs. Utilizando o conceito de meta-dataflows, o framework integra as transformações dos dados, formando um caminho de derivação contínuo. A eficácia da* `ProvInCiA` *foi avaliada em um estudo de caso sobre monitoramento de alagamentos.*

Abstract. *Smart Cities (SCs) use data from various sources to improve quality of life and support public policies. Ensuring complete traceability of this data, from collection to usage, is essential for auditing and reliability. Provenance data helps represent this derivation path, but collecting it in SCs is complex due to the diverse ecosystem of programs and users involved. To address this challenge, the* `ProvInCiA` *framework was developed to capture provenance data in SCs. Using the concept of meta-dataflows, the framework integrates data transformations, forming a continuous derivation path. The effectiveness of* `ProvInCiA` *was evaluated in a case study on flood monitoring.*

1. Introdução

Nas últimas décadas, foi possível observar um movimento contínuo de migração das áreas rurais para as cidades [Rodrigues et al. 2015]. Esse processo de urbanização acelerada gera desafios relacionados ao planejamento urbano [Bonadia et al. 2023]. À medida que a demanda por habitação nas cidades cresce e os terrenos adequados se tornam escassos ou caros, ocorre uma expansão urbana não planejada. Como resultado, populações em situação de vulnerabilidade social acabam ocupando regiões inadequadas para moradia, muitas vezes localizadas em áreas de risco e sem a infraestrutura mínima necessária. Esses locais, em geral, são áreas suscetíveis à ocorrência de desastres naturais, *e.g.* inundações, alagamentos e deslizamentos de terra. Casos recentes ilustram bem essa realidade, como as intensas chuvas que

*Os autores gostariam de agradecer pelo apoio financeiro da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) – Código de Financiamento 001, do CNPq e da FAPERJ.

afetaram o município de Petrópolis em 2022. Diante desse cenário, torna-se importante compreender a complexa dinâmica das cidades, para subsidiar a formulação de políticas públicas orientadas à mitigação de riscos e ao planejamento territorial.

Essa compreensão da dinâmica das cidades tem sido auxiliada com o advento das chamadas *Cidades Inteligentes* (CI) [Bilal et al. 2020] . O conceito de CI está diretamente relacionado à utilização intensiva de tecnologias da informação e comunicação nos processos de gestão urbana, com foco na coleta, integração, processamento e análise de grandes volumes de dados gerados [Roriz Junior et al. 2019]. Esses dados podem ser oriundos de diversas fontes, como sensores, dispositivos móveis e bases de dados públicas disponibilizadas por órgãos governamentais. Por meio do uso de soluções computacionais, esses dados são convertidos em conhecimento útil, capaz de subsidiar a tomada de decisões e orientar intervenções públicas de forma mais eficaz.

Nesse contexto, a eficácia das intervenções urbanas depende, cada vez mais, da capacidade de interpretar dados em uma escala espaço-temporal sem precedentes. A infraestrutura associada às CIs tem evoluído justamente nesse sentido, possibilitando o monitoramento contínuo de múltiplas dimensões da vida urbana, o que viabiliza decisões eficazes e embasadas em evidências concretas. O ciclo de vida dos dados, nesse cenário, torna-se cada vez mais complexo e distribuído, envolvendo múltiplas etapas, desde a coleta e integração de fontes heterogêneas até a análise e disponibilização de informações compreensíveis por agentes públicos para elaboração de políticas públicas.

Entretanto, qualquer política pública derivada de dados deve ser passível de auditoria, como forma de assegurar sua transparência, legitimidade e confiabilidade [Javed et al. 2017b]. Devemos ser capazes de compreender as motivações e justificativas que levaram à criação de determinada política pública, monitorar a sua execução por meio de mecanismos de monitoramento contínuo e, caso sejam identificadas falhas, compreender suas causas e atribuir responsabilidades. Para tanto, a rastreabilidade dos dados é um requisito-chave. É fundamental ter a capacidade de reconstruir todo o caminho de derivação de um dado, desde a sua origem até sua utilização final, permitindo uma análise sobre como ele contribuiu para a formulação de determinada política.

Os dados de proveniência [Freire et al. 2008] são uma solução natural para representar o caminho de derivação dos dados no contexto das CIs. O uso de dados de proveniência está alinhado a princípios centrais da governança de dados, tais como a transparência, a responsabilidade e a auditabilidade dos sistemas computacionais [Moreau et al. 2018]. Ao permitir a reconstituição fiel do ciclo de vida dos dados, os dados de proveniência podem tornar-se a espinha dorsal de auditorias em ecossistemas de CI. Tal rastreabilidade contribui diretamente para o fortalecimento da confiança nas soluções tecnológicas adotadas, além de conferir maior legitimidade às decisões públicas baseadas em dados.

Entretanto, a coleta de dados de proveniência em CIs não é uma tarefa trivial devido ao seu ecossistema naturalmente distribuído e heterogêneo, que envolve múltiplos usuários e ferramentas (*e.g.*, *notebooks* interativos, terminais de comando, sistemas *Web* e *pipelines* em nuvem) que produzem, coletam e processam dados em etapas diferentes do ciclo de vida do dado. Assim, as decisões críticas são tomadas com base no resultado de um *dataflow* descentralizado. A coleta de dados de proveniência já se mostra complexa mesmo em ambientes mais controlados [Pasquier et al. 2017], sendo ainda mais desafiadora em cenários urbanos, onde os *dataflows* são dinâmicos e interdependentes. Neste contexto, torna-se necessário con-

ceber mecanismos capazes de capturar um *dataflow* "global", que integre diversos *dataflows* independentes, autônomos e heterogêneos, de forma a reconstruir o caminho de derivação completo dos dados. Embora existam soluções que capturam proveniência de *dataflows* isolados [Lin et al. 2023], até o momento não se identificou uma abordagem capaz de capturar e associar *dataflows* inerentes a ecossistemas de CI.

Este artigo propõe a `ProvInCiA` (*Provenance in Smart Cities Approach*), um *framework* voltado à coleta de dados de proveniência no ecossistema de CI. A `ProvInCiA` foi planejada levando em consideração a diversidade de ferramentas e plataformas utilizadas no desenvolvimento de soluções para CIs, e tem como objetivo garantir a rastreabilidade dos dados nesse ecossistema. Um dos diferenciais do *framework* reside na utilização do conceito de *meta-dataflow*, o qual permite associar múltiplos *dataflows* independentes observados durante o ciclo de vida dos dados. A viabilidade da `ProvInCiA` foi avaliada por meio do estudo de caso de um sistema de monitoramento de chuvas e alagamentos, evidenciando seu potencial de aplicação em cenários urbanos complexos.

2. Preliminares

O ciclo de vida de um dado dentro do ecossistema de CI geralmente é composto por múltiplas etapas (*e.g.*, coleta de dados, pré-processamento, *etc.*), que formam um *dataflow*. Como os dados são processados em um ambiente naturalmente distribuído, eles podem ser consumidos e produzidos por múltiplos *dataflows*, cada um deles gerenciado por um usuário diferente. Formalmente (formalização inspirada em [Ikeda et al. 2013, Silva et al. 2017]), um elemento de dado $e = \{v_1, v_2, \ldots, v_n\}$ é composto por uma sequência de valores $v_n \in \mathbb{V}$. Uma coleção de dados c é uma coleção de elementos de dados. Por sua vez, um *dataset* $s = \langle A, C \rangle$ é uma tupla com uma sequência de atributos $A = \{a \mid a \in \mathbb{A}\}$ e C é um conjunto de coleções de dados em que $\forall\ c \in C \wedge e \in c,\ |e| = |A|$. Dessa forma, uma transformação de dados $t(I),\ t : I \rightarrow O$ é uma função que mapeia *datasets* de entrada I para *datasets* de saída O. Uma dependência de dados $\phi = \langle s, t, t' \rangle$ é uma tripla composta por um *dataset* s e duas transformações de dados t e t', onde $s \subseteq t(s) \wedge s \subseteq t'(t(s))$. Finalmente, um *dataflow* $d = \langle T, S, \Phi \rangle$ é uma tripla com as transformações de dados T de *datasets* S, e um conjunto de dependências Φ.

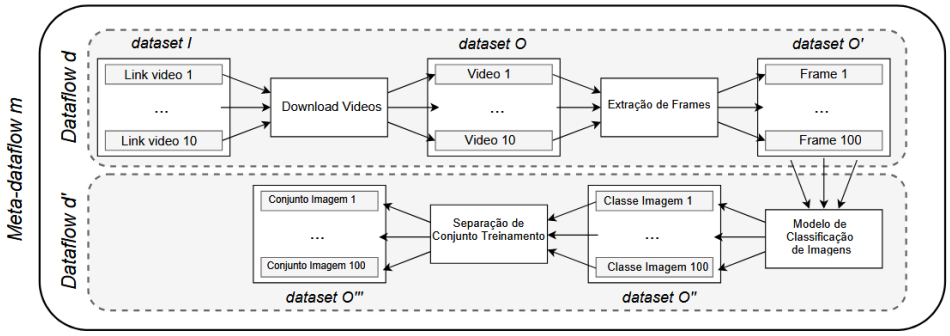

Figura 1. *Meta-dataflow* ilustrando dependência de dois *dataflows*.

Como o objetivo da `ProvInCiA` é capturar o *dataflow* global do ecossistema de CIs, devemos ser capazes de associar múltiplos *dataflows* executados de forma isolada por diferentes atores. Assim, um *meta-dataflow* $m = \langle D, S, \Xi \rangle$ é uma abstração composta por um conjunto de *dataflows* independentes D, organizados de forma sequencial, em que o

último *dataset* gerado por um *dataflow* $d \in D$ é utilizado como entrada inicial do *dataflow* subsequente $d' \in D$. Isso garante a dependência de *dataflows* $\Xi = \langle s, d, d' \rangle$, que é expressa por uma tripla composta por um *dataset* s e dois *dataflows* d e d', onde $s \subseteq d(s) \wedge s \subseteq d'(d(s))$. Essa abstração permite representar caminhos de derivação dos dados compostas por múltiplos *dataflows* conectados, promovendo a rastreabilidade em sistemas complexos, como no cenário de CIs. A Figura 1 apresenta um exemplo de um meta-dataflow m composto de dois *dataflows* d e d' onde cada um deles possui duas transformações. É importante ressaltar que o *dataset* O' é a saída do *dataflow* d e entrada do *dataflow* d', explicitando assim a dependência de *dataflows* $\xi \in \Xi$.

3. Uma Breve Introdução aos Dados de Proveniência

Dados de proveniência podem ser definidos como o conjunto de informações que descrevem as entidades, atividades e agentes (indivíduos ou sistemas) envolvidos na criação, modificação e disseminação de um determinado dado [Freire et al. 2008]. Segundo a taxonomia proposta por Freire et al. (2008), os dados de proveniência podem ser classificados em duas categorias complementares: (i) proveniência prospectiva (*p-prov*), e (ii) proveniência retrospectiva (*r-prov*). A *p-prov* refere-se à especificação da estrutura lógica do *dataflow*. Tal especificação define a sequência de execução esperada de programas ou *scripts*, bem como as dependências de dados entre cada etapa. Por outro lado, a *r-prov* está centrada no registro de uma execução concreta de um *dataflow*, capturando quais dados de entrada e parâmetros foram utilizados, quais transformações foram aplicadas em tempo de execução, e quais resultados foram produzidos ao final. A r-prov, portanto, possibilita o rastreamento de como um dado foi gerado, fornecendo um caminho de derivação que pode ser utilizado para auditoria.

Para representar dados de proveniência de maneira estruturada e padronizada, o W3C propõe a recomendação PROV [Moreau and Missier 2013], que especifica um modelo conceitual e um conjunto de vocabulários destinados à representação formal do caminho de derivação de um dado. O PROV é baseado em três conceitos básicos: (i) entidades, (ii) atividades e (iii) agentes. As entidades referem-se a quaisquer artefatos que possam ser transformados dentro de um contexto, *e.g.*, arquivos de vídeo, *datasets*, imagens, *etc*. As atividades correspondem a processos ou ações que geram, modificam ou consomem entidades, *e.g.*, um *script* de processamento de imagens. Por fim, os agentes são os responsáveis diretos ou indiretos pela realização das atividades. Eles podem ser indivíduos, organizações ou sistemas. A partir da associação entre agentes, atividades e entidades, é possível modelar e documentar as relações de dependência, causalidade e responsabilidade envolvidas na geração e transformação dos dados. As relações entre esses elementos podem ser *wasGeneratedBy*, *used*, *wasDerivedFrom*, *wasAssociatedWith*, *wasInformedBy* e *wasAttributedTo*.

4. Trabalhos Relacionados

A captura, o armazenamento e a análise das transformações que ocorrem sobre os dados em ecossistemas de CIs são tarefas complexas e sem solução definitiva. Diversos trabalhos têm sido propostos, apresentando abordagens que buscam oferecer soluções para gerência de dados de proveniência no contexto de CIs. As abordagens propostas podem ser organizados em duas principais vertentes: (i) soluções que capturam dados de proveniência para garantir que dados sejam oriundos de fontes legítimas e seguras, e (ii) soluções que focam na rastreabilidade dados nas aplicações de CI. A seguir, discutimos essas duas vertentes.

As soluções da primeira vertente [Nepal et al. 2024b, Nepal et al. 2024a, Nepal et al. 2023, Laamech et al. 2021, Sadineni et al. 2023, Hoque and Hasan 2022] concentram-se nos dados transmitidos ou recebidos por dispositivos IoT, geralmente com ênfase na segurança da camada de aplicação e na garantia da confiabilidade dos dados. Isso se deve ao fato de que ataques furtivos realizados em camadas inferiores, como as camadas de enlace e de rede, não são detectados por soluções que operam exclusivamente na camada de aplicação. Nesse contexto, os dados de proveniência são um recurso para a investigação de incidentes de segurança que ocorrem nessas camadas inferiores das redes IoT. É importante salientar que essas abordagens, em sua maioria, registram apenas um subconjunto restrito dos dados de proveniência, mais especificamente, a fonte dos dados. Contudo, elas não têm como foco registrar o caminho de derivação dos dados, uma vez que seu principal objetivo é identificar fontes de possíveis ataques e não rastrear as transformações que os dados sofrem.

A segunda vertente de trabalhos concentra-se na rastreabilidade das transformações às quais os dados são submetidos ao longo de seu ciclo de vida no ecossistema de CI. Os trabalhos dessa vertente visam registrar, de forma estruturada, o histórico completo de derivação dos dados. [Javed et al. 2018, Javed et al. 2017b, Javed et al. 2017a, Javed et al. 2016] propõem a coleta de metadados de proveniência relacionados ao ciclo de vida de políticas públicas, incluindo informações sobre atividades executadas, os atores envolvidos, bem como as entradas e saídas de cada etapa do processo. [Bai et al. 2024] propõem o uso de ontologias para anotar o caminho de derivação dos dados, favorecendo representações semânticas. Já [Emaldi et al. 2013] se concentram na coleta de metadados sobre as ações dos usuários em relação aos dados com os quais interagem, utilizando esses dados de proveniência como base para o cálculo de um índice de confiabilidade dos dados.

[Wilms et al. 2021] propõem a coleta de dados de proveniência em veículos autônomos com o objetivo de antecipar cenários de direção perigosos. Os dados registrados concentram-se na identidade, validade e origem das informações utilizadas pelos sistemas embarcados. De forma complementar, [Bolaños-Martinez et al. 2024] realizam a coleta de dados de origem de veículos com o intuito de analisar padrões de comportamento de motoristas que estão em visita a uma determinada cidade. As informações capturadas incluem a placa do veículo, o código postal do condutor e a distância entre a cidade de origem e o local atual. No contexto de redes IoT, [Sadineni et al. 2023] registram dados de proveniência relacionados a ataques, incluindo as atividades maliciosas, horários de início e término, agentes envolvidos e o estado resultante do sistema. Apesar dessas abordagens representarem um avanço, elas se limitam a coletar o caminho de derivação de uma determinada etapa do ciclo de vida do dado em CI, sem considerar que o dado pode ser processado por diversos *dataflows* independentes. Dessa forma, não são capazes de fornecer o rastro completo dos dados.

5. `ProvInCiA`

Esta seção apresenta o *framework* `ProvInCiA`, proposto para a coleta e gerência de dados de proveniência em ecossistemas de CIs. A arquitetura da `ProvInCiA` é apresentada na Figura 2 e é composta por nove componentes principais: (i) *Eavesdrop*, (ii) API de Acesso, (iii) *Broker* de Proveniência, (iv) *Crawler*, (v) Estruturador de Proveniência, (vi) Banco de Dados de Proveniência, (vii) *Browser* de Proveniência, (viii) Gerador de Grafo de Proveniência e (ix) Processador de Consultas.

A execução da `ProvInCiA` tem início com a captura dos dados de proveniência, etapa que representa um desafio no ecossistema de CIs, considerando a diversidade de for-

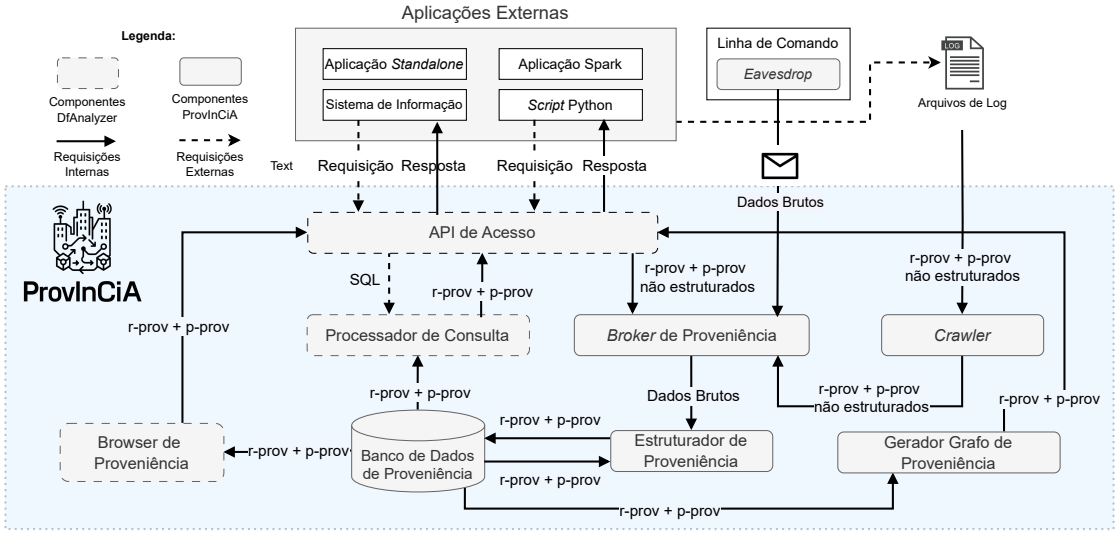

Figura 2. Arquitetura do *Framework* `ProvInCiA`

mas pelas quais os dados podem ser processados ao longo de seu ciclo de vida. Para lidar com essa heterogeneidade, a `ProvInCiA` adota três estratégias de captura de dados de proveniência: (i) instrumentação de código, (ii) escuta ativa e (iii) análise de arquivos de *logs*. A primeira estratégia assume que as aplicações e *scripts* foram previamente instrumentados para enviar mensagens contendo dados de proveniência à `ProvInCiA` por meio da sua *API de Acesso*. Essa abordagem é recomendada nos casos em que o código-fonte está disponível, pois permite que o próprio desenvolvedor explicite, por meio da injeção de chamadas à API, as transformações aplicadas aos dados. A instrumentação de código é adotada em soluções de captura de proveniência existentes [McPhillips et al. 2015]. Em sua versão atual, a `ProvInCiA` segue o mesmo tipo de instrumentação da ferramenta [Silva et al. 2018].

A segunda estratégia consiste na escuta ativa do ambiente de execução, *e.g.*, terminal de SO, com o objetivo de identificar, assim que o usuário executa comandos, transformações de dados realizadas pelo usuário por meio de chamadas ao SO. Essa escuta é realizada pelo componente *Eavesdrop*, que envia mensagens contendo os dados brutos capturados para a `ProvInCiA`. Por fim, quando dados de proveniência já estão disponíveis, porém armazenados em arquivos de *log*, a `ProvInCiA` utiliza o *Crawler*, que é responsável por acessar esses arquivos, extrair os dados relevantes e importá-los para a plataforma. Vale ressaltar que o *Crawler* deve ser customizado para cada tipo específico de *log*, de modo a garantir a correta interpretação e extração das informações. Porém, quando os *logs* estão ausentes ou incompletos e o código não é instrumentável, a reconstrução da *r-prov* torna-se parcial ou inviável. Nesse caso, é registrada uma única atividade com seus *datasets* de entrada e saída.

Uma vez que os dados de proveniência "brutos" são obtidos, eles podem ser enviados para a `ProvInCiA` por diferentes caminhos: via *API de Acesso* (no caso da instrumentação), diretamente ao *Broker de Proveniência* (no caso da escuta ativa pelo *Eavesdrop*), ou ainda pelo *Crawler* (no caso da análise de arquivos de *log*). A *API de Acesso*, ao receber dados de proveniência, também os encaminha ao *Broker*, que funciona como uma fila de processamento das requisições. O *Broker* encaminha cada requisição da fila para o componente *Estruturador de Proveniência*, responsável por identificar e estruturar os elementos do PROV

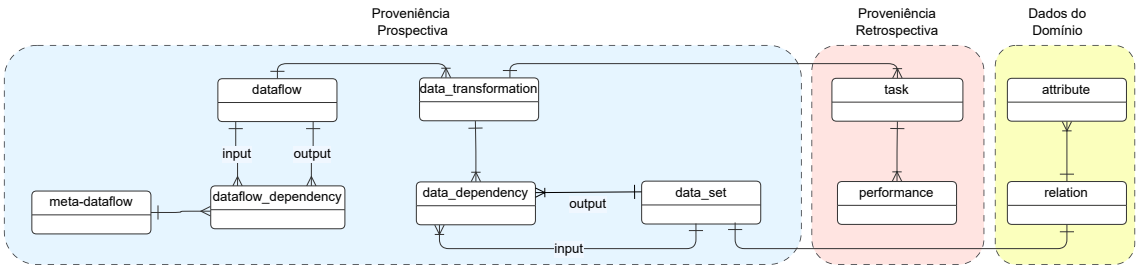

Figura 3. Modelo de Dados da `ProvInCiA`

(atividades, entidades e agentes). Esses elementos são então registrados no *Banco de Dados de Proveniência*. Além disso, o *Estruturador de Proveniência* também tem a função de identificar a existência de um *meta-dataflow*, *i.e.*, quando a saída de um *dataflow* é utilizada como entrada de outro, estabelecendo conexões de dependência entre diferentes *dataflows*. O *meta-dataflow* é capaz de representar ramificações, *e.g.*, *splits* e *merges* entre *dataflows*, mas não pode conter ciclos, uma vez que eles são Grafos Acíclicos Direcionados (DAGs).

O *Banco de Dados de Proveniência* estende o esquema já utilizado na ferramenta DfAnalyzer [Silva et al. 2018], incluindo novos conceitos como o de *meta-dataflow*, e está organizado em três conjuntos de classes, como apresentado na Figura 3: (i) classes que representam a proveniência prospectiva (em azul), (ii) classes que representam a proveniência retrospectiva (em vermelho) e (iii) classes que representam os dados de domínio (em amarelo). É importante destacar que as classes referentes aos dados de domínio, *i.e.*, as que dependem da aplicação cuja proveniência está sendo capturada, são customizadas no banco de dados à medida que novos dados de proveniência são capturados. Ou seja, sempre que a proveniência de um novo *dataflow* precisar ser capturada, serão necessárias novas customizações.

Uma vez que os dados de proveniência estão armazenados no *Banco de Dados de Proveniência*, consultas podem ser submetidas por meio do componente *Processador de Consultas*. O sistema de banco de dados utilizado na versão atual da `ProvInCiA` é o MonetDB, e, portanto, o processador atualmente oferece suporte a consultas SQL. Além disso, os dados de proveniência podem ser explorados por meio de uma visualização interativa gerada pelo *Browser de Proveniência*. Tanto o *Processador de Consultas* quanto o *Browser de Proveniência* foram estendidos a partir da ferramenta DfAnalyzer. Por fim, os dados de proveniência podem ser exportados no formato PROV-N, garantindo interoperabilidade com ferramentas compatíveis. Os dados também podem ser exportados como imagens estáticas que representam o grafo de proveniência, por meio do componente *Gerador de Grafo de Proveniência*. O código-fonte está disponível no repositório https://github.com/dew-uff/provincia.

6. Avaliação da `ProvInCiA`

Essa seção apresenta o estudo de viabilidade conduzido com o *framework* `ProvInCiA`, com o propósito de avaliar suas funcionalidades que permitem rastrear, reproduzir e auditar o ciclo de vida dos dados no contexto do ecossistema de CIs.

Estudo de Caso. A Figura 4 apresenta uma aplicação de CI voltada à identificação de alagamentos, utilizando imagens capturadas por câmeras de monitoramento de tráfego urbano. Essa aplicação foi escolhida como estudo de caso e é composta por quatro *dataflows* independentes e executados por diferentes usuários, mas que combinados formam um *meta-dataflow*

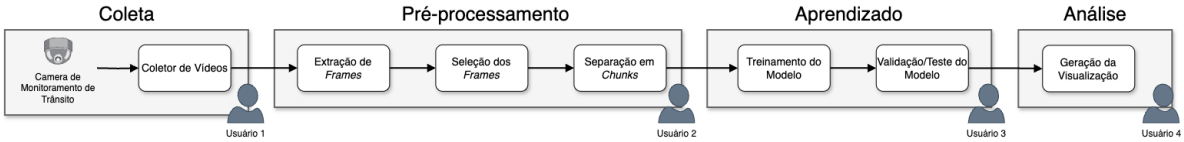

Figura 4. *Meta-dataflow* da avaliação experimental do `ProvInCiA`.

Tabela 1. Consultas de proveniência utilizadas para avaliação da `ProvInCiA`.

#	Consulta	Classe
Q1	Qual video que deu origem a um *frame* específico?	C1
Q2	Qual usuário executou a extração dos *frames* em uma determinada data e hora?	C1
Q3	Quais tarefas da atividade treinamento do modelo foram executadas pelo usuário `mfalci`?	C1
Q4	Quais vídeos foram utilizados para treinar um modelo específico de aprendizado de máquina?	C2

único. O *meta-dataflow* inicia-se com um *dataflow* de captura de dados, onde vídeos de câmeras instaladas em vias públicas são coletados. Em seguida, cada vídeo é submetido a um *dataflow* de pré-processamento, que compreende etapas como a extração de *frames*, ajustes visuais e seleção dos *frames* mais relevantes.

Os *frames* selecionados são então agrupados em *chunks*, que servem como base para o treinamento de redes neurais profundas (DNNs) [Cichy and Kaiser 2019], executado em um terceiro *dataflow*. Os modelos treinados são integrados a um sistema de monitoramento automatizado, que compõe o quarto *dataflow*, sendo responsável por identificar, a partir de novas imagens, a ocorrência de alagamentos. Esse sistema tem aplicação tanto em ações imediatas, *e.g.*, alertas, quanto no apoio à formulação de políticas públicas, *e.g.*, na destinação de investimentos em infraestrutura de drenagem ou na construção de reservatórios de contenção.

Na Figura 4, cada retângulo cinza representa um *dataflow* distinto, permitindo visualizar as diferentes etapas do *meta-dataflow*. Todos os *scripts* associados às transformações envolvidas na aplicação foram instrumentados por meio da inserção direta de chamadas à API da `ProvInCiA` nos próprios *scripts*, sem modificar a lógica original de cada processo, assegurando assim a fidelidade ao comportamento do sistema em ambiente de produção. A execução do *meta-dataflow* ocorreu em um ambiente conteinerizado, no qual cada *dataflow* foi alocado em um contêiner distinto, garantindo isolamento entre as etapas do processo e maior controle sobre sua execução. Em termos *overhead* de captura de dados de proveniência, a diferença entre o tempo de execução sem captura de proveniência (0,15s) e com captura (0,37s) do *script* de separação de *frames* foi de 0,22s, enquanto a diferença de tempo de execução sem captura (24,5s) e com captura (39,8s) do *script* de classificação foi de 15s.

Consultas. A `ProvInCiA` tem como objetivo oferecer suporte a uma análise do caminho de derivação dos dados de *meta-dataflows* no ecossistema de CIs. Assim, de forma a avaliar esse suporte, a Tabela 1 apresenta um conjunto de consultas que devem ser respondidas pelo banco de dados de proveniência. Essas consultas foram projetadas junto com especialistas da Secretaria de Defesa Civil da cidade de Niterói [Victorino et al. 2023] e foram inspiradas no *First Provenance Challenge*[1]. As consultas estão organizadas em duas classes: (C1) consultas que consideram *dataflows* isolados; (C2) consultas que consideram *meta-dataflows*.

[1]https://openprovenance.org/provenance-challenge/FirstProvenanceChallenge.html

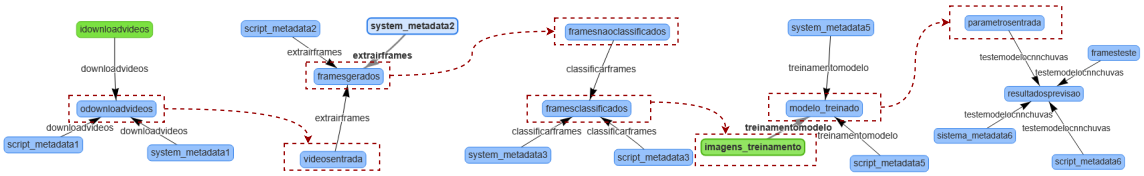

Figura 5. p-prov dos *dataflows* envolvidos na aplicação estudo de caso.

Discussão dos Resultados. Com o objetivo de avaliar a viabilidade do *framework* `ProvInCiA`, foram executadas todas as consultas apresentadas na Tabela 1, e seus resultados foram analisados. No entanto, dado que essas consultas operam sobre a combinação de *p-prov* e *r-prov*, optamos inicialmente por realizar uma análise isolada da *p-prov*, a fim de compreender como os dados fluem ao longo do *meta-dataflow*. A Figura 5 apresenta a visualização interativa gerada pelo *Browser de Proveniência* com todos os *datasets* produzidos e consumidos durante a execução dos diferentes *dataflows* de forma independente.

É importante destacar que, como a instrumentação dos *scripts* que compõem a aplicação pode ser realizada por diferentes usuários dentro do ecossistema de CIs, é comum que os nomes dos *datasets* variem. Apesar disso, a `ProvInCiA` é capaz de identificar *datasets* equivalentes por meio da análise dos atributos A que compõem as coleções de dados C, os quais, por sua vez, definem cada *dataset* $s = \langle A, C \rangle$. Na Figura 5, foram adicionadas manualmente setas tracejadas vermelhas que indicam essas equivalências entre *datasets*, evidenciando como a `ProvInCiA` consegue associar os diversos *dataflows* registrados, compondo assim uma representação do *meta-dataflow* completo.

A consulta `Q1` tem como objetivo analisar exclusivamente a proveniência associada ao segundo *dataflow* do *meta-dataflow*. Especificamente, busca-se identificar a origem de um determinado *frame*, *i.e.*, a qual vídeo original ele está associado. Esse tipo de consulta é particularmente relevante em situações em que algum problema é identificado em um *frame* específico, *e.g.*, a presença de imagens de pessoas capturadas sem autorização, algo plausível considerando a utilização de câmeras de trânsito. Nesses casos, a identificação do vídeo de origem torna-se essencial para que todos os *frames* derivados desse vídeo possam ser localizados e, se necessário, removidos dos conjuntos utilizados para o treinamento das DNNs *a posteriori*, de modo a preservar a conformidade legal e ética do sistema. A consulta `Q1` pode ser implementada utilizando a linguagem SQL, conforme exemplificado a seguir.

```
SELECT f.path, v.video_path
FROM ds_framesgerados f, ds_videosentrada v
WHERE f.extrairframes_task_id = v.extrairframes_task_id
AND f.path = '/frames/089/frame2.jpg';
```

Uma vez submetida a consulta e fixando como parâmetro o *frame* de interesse localizado em `/frames/089/frame2.jpg`, obteve-se como resultado o vídeo correspondente `/videos/local1/20250507_000653.mp4`, conforme ilustrado a seguir. Com base nesse resultado, torna-se possível desconsiderar esse vídeo para uso na aplicação, caso se identifique qualquer irregularidade ou inadequação relacionada ao *frame* em questão.

```
+------------------------+-------------------------------------+
| path                   | video_path                          |
+========================+=====================================+
| /frames/089/frame2.jpg | /videos/local1/20250507_000653.mp4  |
+------------------------+-------------------------------------+
1 tuple
```

A consulta `Q2` tem como objetivo identificar qual usuário foi responsável pela execução da transformação de extração de *frames* no segundo *dataflow* do *meta-dataflow*. Especificamente, busca-se permitir a atribuição de responsabilidades aos usuários envolvidos na execução. Uma das funções fundamentais dos dados de proveniência é justamente possibilitar essa atribuição, ou seja, identificar o agente responsável por uma determinada execução de transformação de dados. A consulta `Q2` pode ser implementada utilizando a linguagem SQL, conforme exemplificado a seguir.

```
SELECT sm.executed_by AS executado_por, s.start_time AS data_execucao,
    t.id AS task_id, dt.tag AS transformacao
FROM task t JOIN data_transformation dt ON dt.id = t.dt_id
JOIN ds_system_metadata2 sm ON sm.extrairframes_task_id = t.id
JOIN ds_script_metadata2 s ON s.extrairframes_task_id = t.id
WHERE dt.tag = 'extrairframes' AND s.start_time LIKE '2025-05-06T21:11:53%';
```

Uma vez submetida a consulta `Q2`, e tendo como parâmetro a data e o horário de interesse definidos como `2025-05-06T21:11:53`, obteve-se como resultado o nome do usuário responsável pela execução da transformação, identificado aqui como `marialuizafalci`, juntamente com o horário exato em que a transformação foi executada, o identificador único da execução (campo `task_id`) e o nome da transformação executada. A seguir, apresentamos o resultado retornado pela consulta.

```
+-----------------+----------------------------+---------+---------------+
| executado_por   | data_execucao              | task_id | transformacao |
+=================+============================+=========+===============+
| marialuizafalci | 2025-05-06T21:11:53.606965 |      36 | extrairframes |
+-----------------+----------------------------+---------+---------------+
1 tuple
```

A consulta `Q3` tem como objetivo identificar todas as tarefas de treinamento do modelo, que são pertencentes ao *dataflow* de Aprendizado que foram executadas por um usuário previamente definido. Assim como a consulta `Q2`, esta também pode ser utilizada em cenários nos quais seja necessário atribuir responsabilidades, com a diferença de que, neste caso, o foco recai sobre o processo de treinamento dos modelos de aprendizado de máquina, uma etapa crítica e sensível da aplicação, pois influencia diretamente na qualidade, precisão e ética das decisões que podem ser tomadas. A consulta `Q3` pode ser implementada em SQL, conforme apresentado a seguir.

```
SELECT t.id AS task_id, dt.tag AS transformacao, sm.executed_by
AS executado_por, s.script_last_modified
FROM task t JOIN data_transformation dt ON dt.id = t.dt_id
JOIN ds_system_metadata5 sm ON sm.treinamentomodelo_task_id = t.id
JOIN ds_script_metadata5 s ON s.treinamentomodelo_task_id = t.id
WHERE dt.tag = 'treinamentomodelo';
```

Uma vez que a consulta `Q3` foi submetida, utilizando como parâmetro a transformação associada ao treinamento do modelo de aprendizado de máquina, obteve-se como retorno o nome do usuário responsável pela execução da referida transformação, identificado no resultado como `marialuizafalci`. Além disso, o resultado incluiu a data e o horário em que o *script* de treinamento foi modificado pela última vez, o identificador único da execução (`task_id`) e o nome da transformação executada, confome ilustrado a seguir. Esses dados são fundamentais para fins de auditoria, uma vez que permitem verificar quando e por quem determinada versão do modelo foi gerada.

```
+---------+------------------+-----------------+----------------------------+
| task_id | transformacao    | executado_por   | script_last_modified       |
+=========+==================+=================+============================+
|    2477 | treinamentomodelo | marialuizafalci | 2025-04-30T16:20:54.326034 |
```

```
+---------+------------------+-----------------+---------------------------+
1 tuple
```

Finalmente, a consulta `Q4` tem como finalidade rastrear todos os vídeos que foram utilizados no treinamento do modelo responsável por detectar situações de alagamento. Diferentemente das consultas anteriores, que se restringem a um único *dataflow*, essa consulta precisa atravessar múltiplas etapas do *meta-dataflow*, especificamente as que envolvem a extração de *frames* a partir de vídeos e o posterior uso desses *frames* na construção do conjunto de dados de treinamento do modelo. Essa consulta é particularmente relevante em contextos de validação e auditoria do sistema, pois permite que usuários identifiquem possíveis causas para o mau desempenho de um modelo. Por exemplo, caso um modelo apresente resultados incorretos, como no caso da câmera à direita na Figura 6, que sinalizou indevidamente a ocorrência de alagamento, torna-se importante verificar quais vídeos foram usados no seu treinamento. A partir dessa informação, é possível refinar o conjunto de treinamento ou até mesmo desconsiderar modelos treinados com bases não representativas (devido a restrições de espaço o SQL da consulta `Q4` não é apresentado, mas pode ser obtido em `https://github.com/dew-uff/provincia/tree/main/Evaluation/sql`).

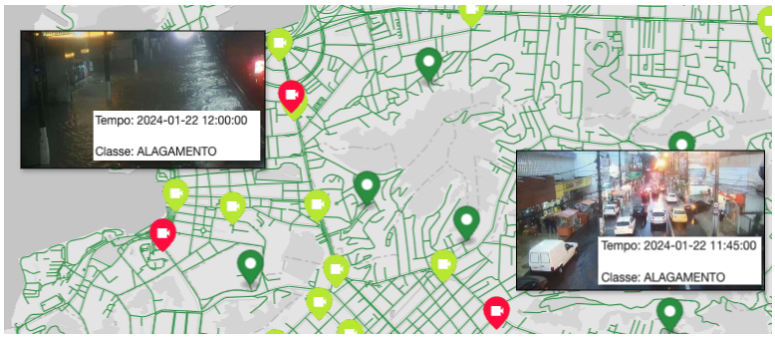

Figura 6. Sistema de monitoramento apresentando a classificação de alagamento.

O resultado da execução da consulta `Q4`, usando como parâmetro de entrada a visualização do sistema que incorretamente sinalizou a ocorrência de alagamento, identifica de forma precisa os vídeos que serviram como base para o treinamento do modelo utilizado nessa visualização. O resultado contém uma lista com os nomes e os respectivos caminhos dos arquivos de vídeo utilizados durante o processo de treinamento do modelo, conforme ilustrado no fragmento a seguir. O resultado dssa consulta evidencia a capacidade da `ProvInCiA` em oferecer rastreabilidade e suporte à auditoria em aplicações de CI.

```
+------------------------------+
| video_path                   |
+==============================+
| /videos/20250506_200355.mp4  |
| /videos/20250506_200331.mp4  |
| /videos/20250504_085458.mp4  |
................
| /videos/20250507_000629.mp4  |
+------------------------------+
500 tuples
```

Além da análise dos resultados obtidos a partir das consultas sobre os dados de proveniência armazenados na base da `ProvInCiA`, realizamos também uma avaliação do componente responsável pela geração dos grafos de proveniência. A Figura 7 apresenta um fragmento representativo do grafo correspondente ao *meta-dataflow* executado. Nesse

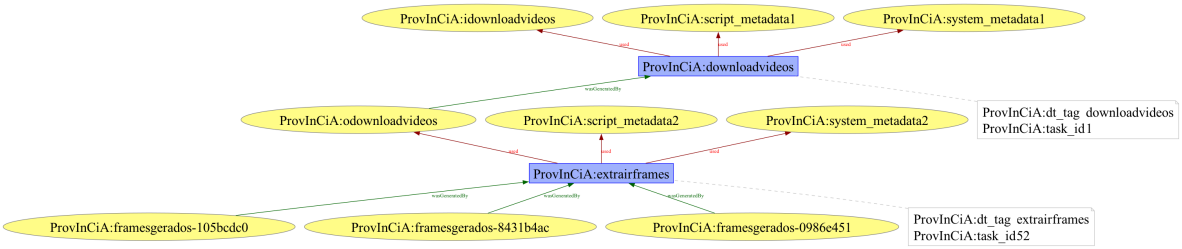

Figura 7. Fragmento do grafo de proveniência do *meta-dataflow*.

grafo, é possível observar como diversas atividades, como por exemplo a transformação `extrairframes`, levaram à geração de diferentes *frames*. Todos esses arquivos estão logicamente conectados a um mesmo *script* responsável pela transformação, aos metadados do sistema no momento da execução, e aos arquivos de vídeo originais. Essa forma de visualização fornece uma ferramenta de apoio à análise, pois permite identificar, de maneira clara, as dependências entre diferentes execuções, os caminhos de derivação dos dados, além de possibilitar a verificação da completude da proveniência, *i.e.*, se todos os dados produzidos estão devidamente associados às atividades que os geraram. Grafos como esse são particularmente relevantes em cenários complexos, pois oferecem suporte à auditoria, permitindo que se rastreie exatamente como cada dado foi produzido.

7. Conclusão

Este artigo apresentou o *framework* `ProvInCiA`, que tem como objetivo coletar e gerenciar os dados de proveniência gerados em ecossistemas de CIs. A `ProvInCiA` é capaz de capturar e gerenciar dados oriundos de diversos *dataflows* interdependentes, formando o que chamamos de *meta-dataflow*. A `ProvInCiA` foi avaliada por meio de um estudo de caso centrado em uma aplicação real de monitoramento de alagamentos, a qual envolve diversas etapas: desde o *download* de vídeos provenientes de câmeras de tráfego, passando pela extração de *frames*, classificação das imagens, organização dos dados em conjuntos de treinamento, validação e teste, até o treinamento e avaliação de modelos de aprendizado de máquina. Por meio da `ProvInCiA`, foi possível coletar, de forma semi-automática (uma vez que os *scripts* foram instrumentados), informações detalhadas sobre os dados de entrada e saída, metadados do ambiente de execução e relacionamentos entre as diversas transformações aplicadas ao longo do *meta-dataflow*.

A modelagem adotada, que considera tanto *p-prov* quanto *r-prov*, mostrou-se eficaz para responder a consultas analíticas que envolvem rastreabilidade dos dados. Consultas como a identificação do vídeo de origem de um determinado *frame* e a atribuição de autoria sobre uma execução específica puderam ser respondidas com sucesso. Em especial, `ProvInCiA` permite responder perguntas sobre dados que permeiam vários *dataflows* diferentes, garantindo a correta rastreabilidade dos dados. Tais funcionalidades são essenciais para garantir a auditoria nas aplicações de CI, especialmente naquelas com impacto direto sobre a formulação de políticas públicas e serviços urbanos.

Atualmente, a `ProvInCiA` encadeia os *dataflows* com base nos conjuntos de dados consumidos e produzidos, mas não gerencia dependências assíncronas acionadas por eventos. Estender o *framework* para contemplar esse tipo de dependência permanece como trabalho futuro. Além disso, planejamos incorporar uma camada de *Text-to-SQL*, capaz de converter consultas em linguagem natural para SQL.

Referências

Bai, J., Lee, K. F., Hofmeister, M., Mosbach, S., Akroyd, J., and Kraft, M. (2024). A derived information framework for a dynamic knowledge graph and its application to smart cities. *Future Generation Computer Systems*, 152:112–126.

Bilal, M., Usmani, R. S. A., Tayyab, M., Mahmoud, A. A., Abdalla, R. M., Marjani, M., Pillai, T. R., and Targio Hashem, I. A. (2020). *Smart Cities Data: Framework, Applications, and Challenges*, pages 1–29. Springer International Publishing, Cham.

Bolaños-Martinez, D., Bermudez-Edo, M., and Garrido, J. L. (2024). Clustering pipeline for vehicle behavior in smart villages. *Information Fusion*, 104:102164.

Bonadia, S., Gama, R., Oliveira, D., Miranda, F., and Lage, M. (2023). Visual analytics using heterogeneous urban data. In *Conference on Graphics, Patterns and Images*, pages 25–30, Porto Alegre, RS, Brasil. SBC.

Cichy, R. M. and Kaiser, D. (2019). Deep neural networks as scientific models. *Trends in cognitive sciences*, 23(4):305–317.

Emaldi, M., Pena, O., Lazaro, J., Lopez-de Ipina, D., Vanhecke, S., and Mannens, E. (2013). To trust, or not to trust: Highlighting the need for data provenance in mobile apps for smart cities. In *International Workshop on Semantic Sensor Networks*, pages 1–4.

Freire, J., Koop, D., Santos, E., and Silva, C. T. (2008). Provenance for computational tasks: A survey. *Computing in science & engineering*, 10(3):11–21.

Hoque, M. A. and Hasan, R. (2022). A trust management framework for connected autonomous vehicles using interaction provenance. In *IEEE International Conference on Communications*, pages 2236–2241. IEEE.

Ikeda, R., Sarma, A. D., and Widom, J. (2013). Logical provenance in data-oriented workflows? In *IEEE International Conference on Data Engineering*, pages 877–888. IEEE.

Javed, B., Khan, Z., and McClatchey, R. (2017a). A network-based approach to capture provenance of a policy-making process. In *International Database Engineering & Applications Symposium*, pages 283–286.

Javed, B., Khan, Z., and McClatchey, R. (2017b). Using a model-driven approach in building a provenance framework for tracking policy-making processes in smart cities. In *International Database Engineering & Applications Symposium*, pages 66–73.

Javed, B., Khan, Z., and McClatchey, R. (2018). An adaptable system to support provenance management for the public policy-making process in smart cities. *Informatics*, 5(1):3:1–26.

Javed, B., McClatchey, R., Khan, Z., and Shamdasani, J. (2016). A provenance framework for policy analytics in smart cities. In *International Conference on Internet of Things and Big Data*, pages 429–434.

Laamech, N., Munier, M., and Pham, C. (2021). Towards a data provenance model for private data sharing management in iot. In *IEEE International Enterprise Distributed Object Computing Workshop*, pages 210–215. IEEE.

Lin, S., Xiao, H., Jiang, W., Li, D., Liang, J., and Li, Z. (2023). A survey of provenance in scientific workflow. *J. High Speed Networks*, 29(2):129–145.

McPhillips, T. M. et al. (2015). Yesworkflow: A user-oriented, language-independent tool for recovering workflow information from scripts. *CoRR*, abs/1502.02403.

Moreau, L., Batlajery, B. V., Huynh, T. D., Michaelides, D., and Packer, H. (2018). A templating system to generate provenance. *IEEE Transactions on Software Engineering*, 44(2):103–121.

Moreau, L. and Missier, P. (2013). PROV-DM: the PROV data model. W3C Recommend.

Nepal, A., Amanullah, M. A., Doss, R., and Jiang, F. (2024a). Secure data provenance in internet of vehicles with data plausibility for security and trust. In *IEEE World AI IoT Congress*, pages 612–618. IEEE.

Nepal, A., Doss, R., and Jiang, F. (2023). Secure data provenance for internet of vehicles with verifiable credentials. In *IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference*, pages 0210–0218. IEEE.

Nepal, A., Doss, R., and Jiang, F. (2024b). Secure data provenance in internet of vehicles with verifiable credentials for security and privacy. In *Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume*, pages 59–61. IEEE.

Pasquier, T., Han, X., Goldstein, M., Moyer, T., Eyers, D., Seltzer, M., and Bacon, J. (2017). Practical whole-system provenance capture. In *Symposium on Cloud Computing*, page 405–418, New York, NY, USA. Association for Computing Machinery.

Rodrigues, A. J., Vieira, J., Fontana, R. L., de Cássia Barroso, R., Silva, J. A., et al. (2015). a urbanização no mundo e no brasil sob um enfoque geográfico. *Caderno de Graduação-Ciências Humanas e Sociais-UNIT-SERGIPE*, pages 95–106.

Roriz Junior, M., de Oliveira, R. P., Carvalho, F., Lifschitz, S., and Endler, M. (2019). M ensageria: A smart city framework for real-time analysis of traffic data streams. In *Big Social Data and Urban Computing Workshop*, pages 59–73. Springer.

Sadineni, L., Pilli, E. S., and Battula, R. B. (2023). Provlink-iot: A novel provenance model for link-layer forensics in iot networks. *Forensic Science International: Digital Investigation*, 46:301600.

Silva, V., de Oliveira, D., Valduriez, P., and Mattoso, M. (2018). Dfanalyzer: Runtime dataflow analysis of scientific applications using provenance. *Proceedings of the VLDB Endowment*.

Silva, V., Leite, J., Camata, J. J., De Oliveira, D., Coutinho, A. L. G. A., Valduriez, P., and Mattoso, M. (2017). Raw data queries during data-intensive parallel workflow execution. *Future Generation Computer Systems*, 75:402–422.

Victorino, F., Amorim, A., et al. (2023). Pluv-web: um gateway científico orientado a dados para análise e monitoramento de chuvas na cidade de niterói. In *Anais Estendidos do Simpósio Brasileiro de Bancos de Dados*, pages 108–113, Belo Horizonte, Brasil. SBC.

Wilms, D., Stoecker, C., and Caballero, J. (2021). Data provenance in vehicle data chains. In *IEEE Vehicular Technology Conference*, pages 1–5. IEEE.