UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COINT - TECNOLOGIA EM SISTEMAS PARA INTERNET
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

LUCAS VIDOTO ESTANISLAU

UMA ARQUITETURA ORIENTADA A SERVIÇOS PARA PRODUZIR SÉRIES TEMPORAIS A PARTIR DE FONTES DE DADOS HETEROGÊNEAS

TRABALHO DE CONCLUSÃO DE CURSO

TOLEDO
2021

LUCAS VIDOTO ESTANISLAU

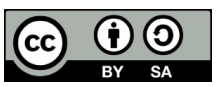
UMA ARQUITETURA ORIENTADA A SERVIÇOS PARA PRODUZIR SÉRIES TEMPORAIS A PARTIR DE FONTES DE DADOS HETEROGÊNEAS

A Service-Oriented Architecture for Producing Time Series from Heterogeneous Data Sources

Trabalho de Conclusão de Curso apresentado ao Curso de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do título de Tecnólogo.

Orientador: Sidgley Camargo de Andrade
Universidade Tecnológica Federal do Paraná - UTFPR

TOLEDO
2021



LUCAS VIDOTO ESTANISLAU

UMA ARQUITETURA ORIENTADA A SERVIÇOS PARA PRODUZIR SÉRIES TEMPORAIS A PARTIR DE FONTES DE DADOS HETEROGÊNEAS

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnológo em Sistemas para Internet da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 03 de dezembro de 2021

_____

Sidgley Camargo de Andrade, Dr.
Universidade Tecnológica Federal do Paraná
Orientador

_____

Marcelo Alexandre da Cruz Ismael, Me.
Universidade Tecnológica Federal do Paraná

_____

Eduardo Pezutti Beletato dos Santos, Me
Universidade Tecnológica Federal do Paraná

TOLEDO

2021

Dedico este trabalho à minha mãe e aos meus avós por serem sempre a razão de todo o meu esforço na vida.

AGRADECIMENTOS

Agradeço a Deus por me proporcionar saúde, paz e persistência durante o curso. Aos meus avós pelo amor, carinho e pela preocupação constante com a minha saúde e felicidade. À minha mãe que sempre foi mãe e pai que batalhou desde sempre para que eu pudesse ter uma educação boa e uma vida digna. À todos os professores que contribuíram com a minha formação acadêmica e profissional. Ao meu orientador que contribuiu com sua experiência e sabedoria e estava sempre disponível a ajudar e esclarecer dúvidas.

Sua mente é programável - se você não estiver programando sua mente, outra coisa irá programá-la para você - Jeremy Hammond.

RESUMO

ESTANISLAU, Lucas. Uma arquitetura orientada a serviços para produzir séries temporais a partir de fontes de dados heterogêneas. 2021. 40 f. Trabalho de Conclusão de Curso – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Toledo, 2021.

Fenômenos e processos do mundo real podem ser analisados a partir de séries temporais, que são coleções de observações ao longo do tempo derivadas de diferentes fontes de dados – normalmente fontes de dados heterogêneas. No entanto, produzir e analisar múltiplas séries temporais a partir de fontes de dados heterogêneas é um desafio devido aos diferentes formatos, tipos e níveis de dados, bem como pela falta de ferramentas de integração e harmonização de dados. O presente trabalho estabelece uma arquitetura de serviço para integrar fontes de dados heterogêneas e produzir séries temporais em diferentes resoluções espaço-temporais. Uma abordagem baseada em regras permite harmonizar as séries temporais produzidas a partir de fontes de dados heterogêneas. A arquitetura proposta foi implementada e aplicada em um cenário de monitoramento de chuvas a partir de duas fontes de dados heterogêneas: sensores pluviométricos, disponibilizados pelo Centro Nacional de Monitoramento e Alertas de Desastres Naturais, e sinais sociais de chuva, derivados da atividade do Twitter relacionada ao fenômeno da chuva. Como resultado, foi demonstrado que a arquitetura de serviço proposta é flexível, por permitir lidar com diferentes fontes de dados, e independente de tecnologia, por permitir a implementação das partes como serviços.

Palavras-chave: Séries temporais. Fontes de dados heterogêneas. Arquitetura de serviços. Integração de dados.

ABSTRACT

ESTANISLAU, Lucas. A service-oriented architecture to produce time series from heterogeneous data sources . 2021. 40 f. Trabalho de Conclusão de Curso – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Toledo, 2021.

Real-world phenomena and processes can be analyzed using time series, which are collections of observations over time derived from different data sources – typically heterogeneous data sources. Nonetheless, the production and analysis of multiple time series derived from heterogeneous data sources are complex tasks owing to the different data formats, types, and levels, as well as the lack of data integration and harmonization tools. This work designed a service architecture to integrate heterogeneous data sources and produce time series in different spatiotemporal resolutions. A rules-based approach that is capable of coupling different data sources to service architecture was established to harmonize the time series. A concept proof applying the architecture was performed to analyze rain events from two distinct data sources: rainfall sensors provided by the Brazilian National Center for Monitoring and Early Warning of Natural Disasters, and social rain signals derived from Twitter activity related to the rain phenomenon. The results show that the service architecture is flexible and technology-independent since it allows dealing with different data sources, coding styles, and language programming.

Keywords: Time series. Heterogeneous data sources. Services architecture. Data Integration.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

URI	Uniform Resource Identifier (Identificador Uniforme de Recurso)
HTTP	Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto)
API	Application Programming Interface (Interface de Programação de Aplicação)
CEMADEN	Centro Nacional de Monitoramento de Desastres Naturais
JSON	JavaScript Object Notation

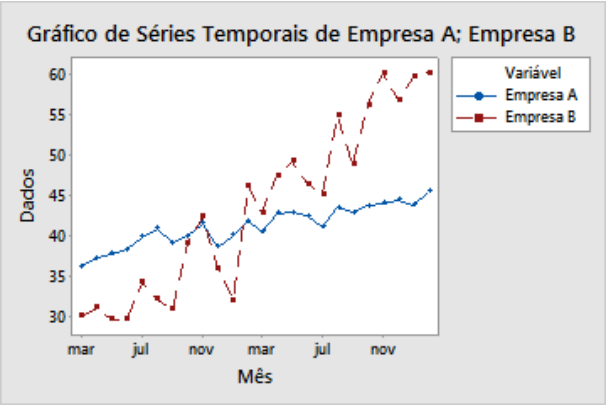
SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTO

Séries temporais são coleções de observações sequenciais em um intervalo de tempo onde há uma ordem temporal e, normalmente, dependência entre as observações (MORETTIN; TOLOI, 2004). A característica temporal dessas séries se dá pelo fato que as observações sobre um dado fenômeno físico ou social são ordenadas no tempo. Um artefato comum para visualizar séries temporais são os gráficos de linha apresentado na Figura 1, que exibem as observações como uma série de pontos conectados por uma linha. Os pontos mais à esquerda do gráfico representam os valores mais antigos, enquanto aqueles à direita os valores mais recentes. Independente do tipo de artefato de visualização, a análise de séries temporais é uma ferramenta importante para encontrar padrões ou tendências de crescimento ou decrescimento de uma ou mais variáveis de estudo. Esse tipo de análise é conduzida em diferentes domínios para apoiar a tomada de decisão – por exemplo, no mercado de ações para prever valores futuros a partir dos valores históricos (TSAY, 2005), na meteorologia para analisar o impacto de fatores meteorológicos sobre as concentrações de ozônio (SOUZA et al., 2017), e na análise de redes sociais para entender o comportamento dos usuários ao longo do tempo (STEIGER et al., 2015).

Figura 1 – Exemplo de gráfico de linha com séries temporais.



Fonte: Minitab (2019).

Embora as séries temporais sejam normalmente produzidas a partir de fontes de dados quantitativas e convencionais, como sensores físicos e registros numéricos de observações relacionadas a uma variável de estudo, elas também podem ser produzidas a partir de outras fontes de dados como mensagens de redes sociais (ANDRADE et al., 2017). Nesse caso, abordagens baseadas em regras ou diretrizes devem ser definidas para determinar como derivar séries temporais a partir de fontes de dados não tradicionais como as redes sociais.

Além disso, fenômenos ou processos do mundo real podem ser observados por diferentes instrumentos que produzem séries temporais distintas e equivalentes. Isso é resultado da percepção dos diferentes instrumentos de medição em relação ao fenômeno ou processo observado. Por exemplo, o fenômeno da chuva é percebido e aferido de maneira diferente entre os instrumentos de radar meteorológico e sensor pluviométrico. Embora as séries temporais produzidas por esses instrumentos não sejam iguais, elas são positivamente correlacionadas. De maneira similar, as pessoas também possuem uma percepção sobre o fenômeno da chuva e, portanto, podem atuar como sensores humanos (GOODCHILD, 2007; ANDRADE et al., 2017).

1.2 PROBLEMA

Sendo as fontes de dados heterogêneas um produto da observação dos fenômenos e processos do mundo real a partir de diferentes instrumentos de medição, a geração ou produção de séries temporais é uma tarefa necessária para apoiar a atividade de análise. No entanto, há alguns desafios a serem enfrentados:

- os diferentes formatos das fontes de dados (por exemplo, imagens, valores textuais, valores numéricos, vídeos);
- os diferentes tipos e níveis de mensuração (por exemplo, nominal, intervalar e razão); e
- a automatização da tarefa de produção de séries temporais levando em consideração a integração e harmonização das fontes de dados.

1.3 OBJETIVOS

Dado os desafios mecionados sobre a produção de séries temporais, este trabalho estabelece um modelo de arquitetura de serviço para integrar fontes de dados heterogêneas para a produção de séries temporais. O modelo proposto foi implementado e aplicado em um cenário de monitoramento de chuvas para a análise conjunta de séries temporais derivadas de fontes de dados heterogêneas. Como prova de conceito, duas fontes de dados foram usadas: sensores pluviométicos disponibilizados pelo Centro Nacional de Monitoramento e Alertas de Desastres Naturais (CEMADEN) e sinais sociais derivados da atividade do Twitter relacionada ao fenômeno da chuva.

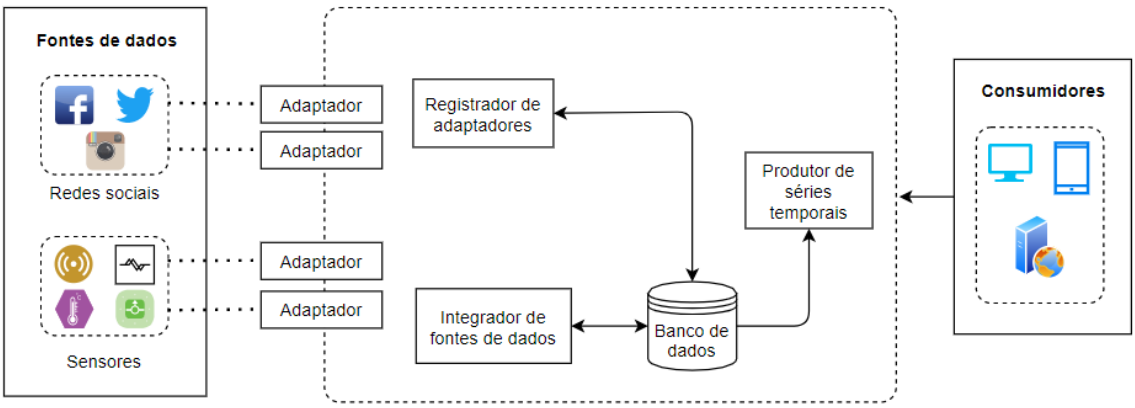
1.4 ESTRUTURA DO TRABALHO DE CONCLUSÃO DE CURSO

O Capítulo 2 descreve o modelo conceitual da arquitetura de serviço para produzir séries temporais a partir de fontes de dados heterogêneas. O Capítulo 3 apresenta uma possível implementação do modelo conceitual da arquitetura de serviço para produção de séries temporais. O Capítulo 4 demonstra o uso da arquitetura de serviço no cenário de monitoramento de chuvas por meio de uma aplicação consumidora do serviço. Por fim, o Capítulo 5 apresenta as conclusões do trabalho.

2 MODELO CONCEITUAL

Modelos conceituais são amplamente usados para descrever sistemas na forma de conjunto de ideias e conceitos sem depender de tecnologias (BRINKKEMPER, 1990). A Figura 2 apresenta os componentes do modelo conceitual da arquitetura de serviço para produzir séries temporais a partir de fontes de dados heterogêneas. O modelo foi baseado no trabalho de Castanhari et al. (2016) onde é apresentado uma arquitetura orientada a serviços chamada *AGORA Information Fusion and Management* (AGORA-IFM) para a integração de dados provenientes de sensores e informação geográfica voluntária. Como pode ser observado, existem três componentes: (i) registrador de adaptadores, (ii) integrador de fontes de dados, e (iii) produtor de séries temporais. A integração das fontes de dados ocorre por parte do componente integrador de fontes de dados, que por sua vez depende de regras cadastradas pelo registrador de adaptadores. O produtor de séries temporais faz uso das observações processadas e armazenadas pelo componente integrador de fontes de dados para agregá-las ao longo do tempo e espaço. Por exemplo, agregar médias diárias por distritos de leituras de sensores físicos distribuídos em uma cidade. As séries temporais geradas pelo produtor de séries temporais são recursos que podem ser consumidos por diferentes aplicações consumidoras. Um exemplo de aplicação consumidora é apresentado no Capítulo 4. As próximas seções detalham o funcionamento dos três componentes e como eles se conectam entre si a partir de padrões de dados interoperáveis.

Figura 2 – Modelo conceitual da arquitetura de serviço para produzir séries temporais a partir de fontes de dados heterogêneas.



Fonte: Autor.

2.1 REGISTRADOR DE ADAPTADORES

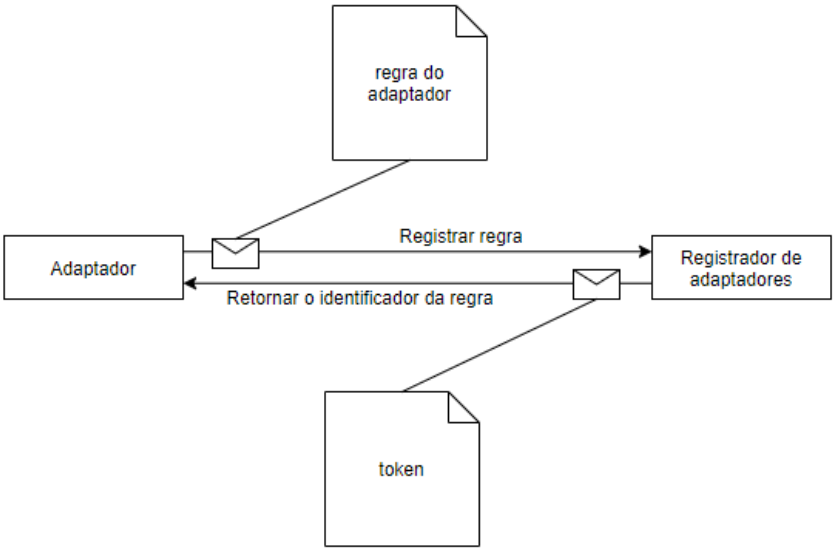
O registrador de adaptadores é responsável pelo registro das regras e informações de contexto dos adaptadores. Adaptadores são elementos físicos ou de *software* que permitem dois ou mais dispositivos ou componentes incompatíveis se conectarem com o propósito de troca de dados (GAMMA et al., 1995). Na arquitetura proposta, os adaptadores possuem duas funções: coletar os dados de diferentes fontes de dados disponíveis; e interoperabilizar os dados em um formato padrão e aceito pelos componentes da arquitetura de serviço. Os adaptadores possuem regras a serem aplicadas nas observações enviadas para o componente integrador de fontes de dados, que é o componente responsável pelo armazenamento das observações (Seção 2.2).

As regras são informações sobre os atributos de interesse obtidos das fontes de dados para a geração de séries temporais. São atributos de interesse de uma observação: *latitude*, *longitude*, *data/hora*, e *medida*. As regras devem conter o tipo de dados dos atributos – se as medições da fonte de dados é quantitativa (por exemplo, valores numéricos provenientes sensores físicos) ou qualitativa (por exemplo, densidade ou frequência de mensagens proveniente da rede social). Essas características devem estar presentes na regra para que o componente produtor de séries temporais produza séries temporais atendendo às particularidades das fontes de dados. A adoção das regras também se justifica pelo fato da agregação e análise comparativa de diferentes tipos e estruturas de dados, tais como medidas de sensores físicos e sinais sociais obtidos da percepção coletiva dos chamados "sensores humanos" (ANDRADE et al., 2021). Por exemplo, um conjunto de observações ao longo do tempo de um sensor de nível de água produz uma série temporal local. De maneira similar, a frequência coletiva de *check-ins* ao longo do tempo na rede social de um estabelecimento também é uma série temporal.

O registrador de adaptadores é o componente responsável por registrar as regras dos adaptadores. Sua função é receber uma requisição com os parâmetros necessários que cada adaptador enviará ao integrador de fontes de dados. Ao registrar uma regra, o registrador de adaptadores retorna um identificador de regra que será usado pelo adaptador para identificar cada requisição enviada ao integrador de fontes de dados. As regras servem para validar os dados que serão enviados pelos adaptadores e produzir identificadores de regras. Um identificador de regras é uma sequência de caracteres aleatórios que permite a comunicação com o componente integrador de fontes de dados. Ele permite identificar e aplicar as regras geradas para um adaptador.

A Figura 3 apresenta o processo de registro de adaptadores e regras. O adaptador envia uma requisição para o registrador de adaptadores com as informações de registro que incluem nome da regra, código da regra, fonte de dados, descrição da regra e a lista dos atributos a serem interpretados pelo integrador de fontes de dados. O registrador de adaptadores armazena as informações e envia uma mensagem de resposta ao adaptador com o identificador. O identificador é denominado de *token* e permite identificar o adaptador e a regra a ser aplicada no integrador de fontes de dados. Esse processo deve ser realizado para cada adaptador a ser conectado na arquitetura de serviço.

Figura 3 – Processo de registro dos adaptadores na arquitetura de serviço para produzir séries temporais.



Fonte: Autor.

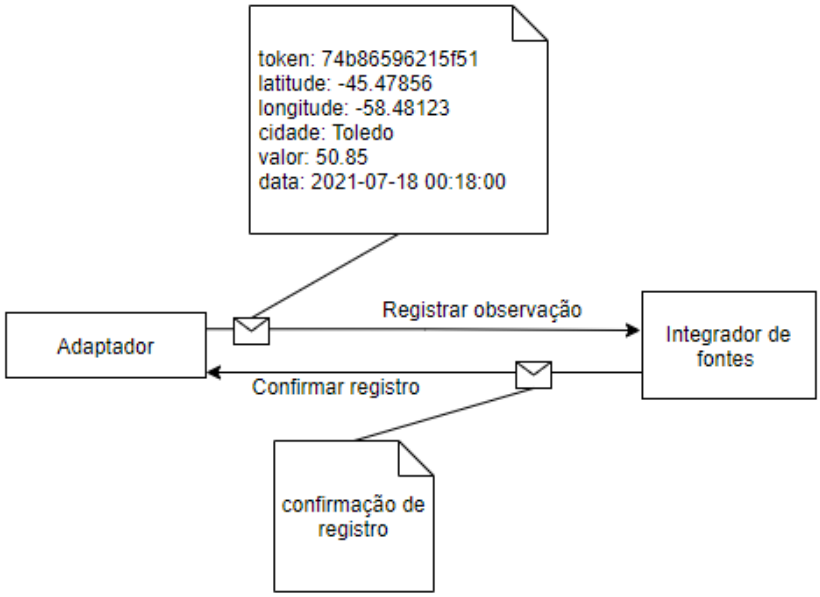
2.2 INTEGRADOR DE FONTES DE DADOS

O integrador de fontes de dados é o componente responsável pela integração e validação dos dados fornecidos pelos adaptadores previamente registrados na arquitetura de serviço. As regras cadastradas para cada adaptador são aplicadas pelo componente integrador como forma de validação dos dados, incluindo filtros e pré-processamentos. Isso inclui, mas não se limita, a seleção de atributos, definição de tipo de dados e definições de valores de referência. O *token* gerado no registro dos adaptadores é utilizado para relacionar cada requisição ao seu respectivo adaptador.

A Figura 4 apresenta o processo de envio das observações dos adaptadores para o integrador de fontes de dados. Os adaptadores devem enviar o *token* em todas as requisições, bem como demais campos indicados no registro dos adaptadores. O integrador de fontes de dados deve responder à tentativa de registro de observação, tanto no caso de sucesso quanto no caso de erros. Caso a observação seja válida e atenda às regras do adaptador, ela será persistida em um banco de dados pelo integrador de fontes de dados. Caso contrário, o integrador de fontes de dados deverá retornar uma resposta de erro de acordo com o protocolo de comunicação utilizado entre os componentes.

Uma vez que o objetivo da arquitetura de serviço é produzir séries temporais, todo registro de observação pelo integrador de fontes de dados deve conter um *timestamp*. Esse *timestamp* pode ser ou a data/hora de coleta da observação ou a data/hora de armazenamento do registro.

Figura 4 – Processo de registro de observações pelo integrador de fontes de dados.



Fonte: Autor.

2.3 PRODUTOR DE SÉRIES TEMPORAIS

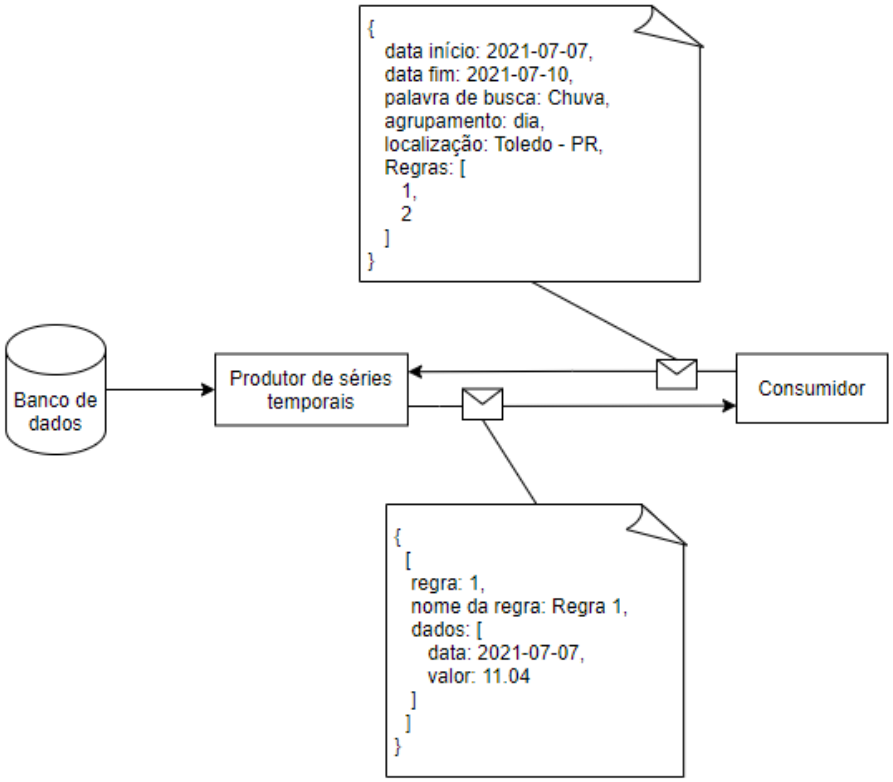
O produtor de séries temporais é o componente responsável por produzir e disponibilizar séries temporais a partir de requisições realizadas por aplicações consumidoras. As aplicações consumidoras são aplicações que estabelecem uma conexão entre a arquitetura e o usuário final. Elas podem ser sistemas Web, *mobile* ou *desktop*, e possuem como finalidade exibir as séries temporais de acordo com as requisições. As requisições são compostas por um conjunto de parâmetros que incluem as datas de início e fim e as escalas temporal e espacial da série temporal. Para as fontes de dados que exigem a extração de sinais, como o número de ocorrência de um termo nas mensagens da rede social, o parâmetro termo de busca é requerido.

A Figura 5 apresenta o processo de geração de séries temporais a partir da requisição de um consumidor. O consumidor realiza a solicitação dos dados a partir de alguns parâmetros necessários para a geração das séries temporais. No exemplo ilustrado (Figura 5), o consumidor requisita a produção de uma série temporal para o período de 7 de julho a 10 de julho de 2021. Para as observações com base textual, que são definidas no momento do registro dos adaptadores, o produtor de séries temporais retorna a frequência de ocorrência da palavra de busca que no exemplo apresentado é o termo "*chuva*". Também é apresentado os parâmetros de agregação, localização e regras de adaptadores. Enquanto o primeiro e segundo parâmetros delimitam as datas entre as observações, o terceiro parâmetro define o termo de busca. O quarto parâmetro define a escala temporal como diária, o quinto delimita as observações para a cidade de Toledo/Pr e o sexto define as regras que serão consideradas para a produção das séries temporais. Uma vez que cada regra é associada a um adaptador, esse parâmetro informa

quais observações serão consideradas de acordo com os adaptadores – pode ser informado mais de uma regra.

Após a geração da série temporal pelo componente produtor de séries temporais, uma mensagem contendo a série temporal requisitada é enviada para a aplicação consumidora.

Figura 5 – Processo de produção de séries temporais.



Fonte: Autor.

3 IMPLEMENTAÇÃO DA ARQUITETURA DE SERVIÇO

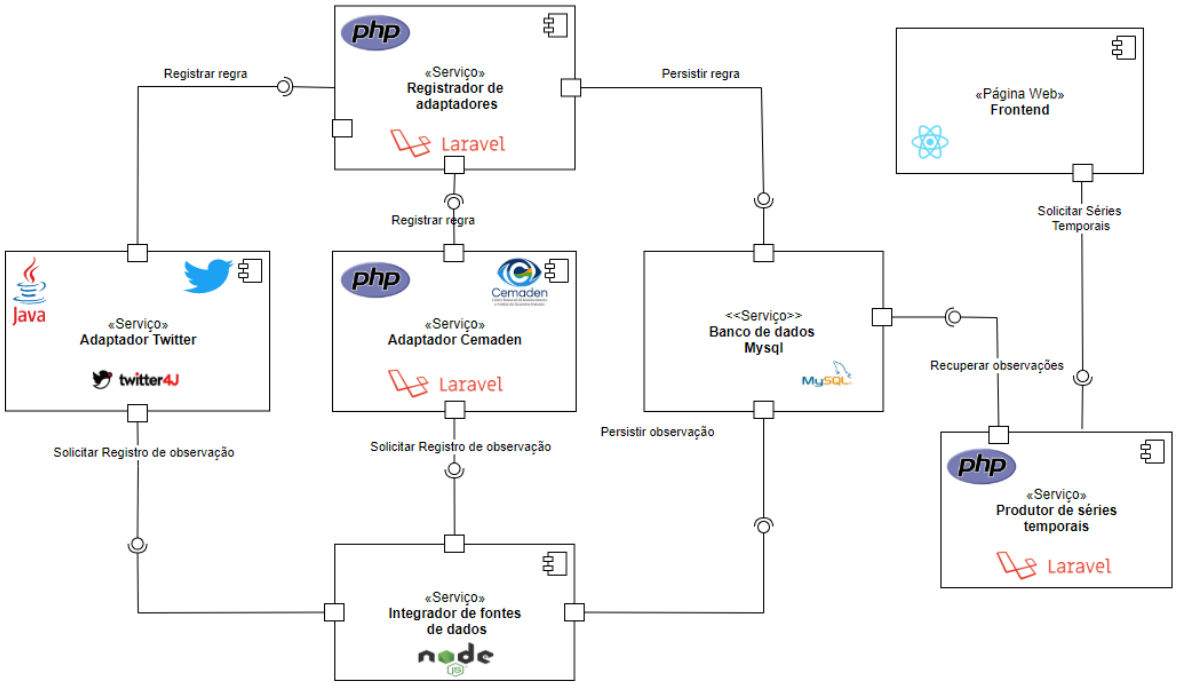
Para a implementação da arquitetura conceitual, foi construído um sistema utilizando a abordagem arquitetural baseada em microsserviços. Essa abordagem consiste na separação de funcionalidades de um sistema em pequenos serviços independentes que se comunicam entre si utilizando protocolos de comunicação (FOWLER, 2014). O protocolo de comunicação H*yperText Transfer Protocol* – HTTP e o formato de dados multiplataforma *JavaScript Object notation* – JSON foram utilizados na implementação da arquitetura de serviço para produção de séries temporais. A utilização desses padrões se justifica pelo fato de permitirem o uso de diferentes tecnologias para o desenvolvimento dos componentes (MOREIRA, 2015). Além disso, o HTTP é um protocolo assíncrono e, por isso, exige uma resposta do serviço para a aplicação cliente. Isso permite o tratamento de erros pela aplicação consumidora do serviço, tais como parâmetros obrigatórios não informados e formato de valores fora dos padrões.

Cada componente da arquitetura Figura 2 foi implementado como um microsserviço desenvolvido com tecnologias distintas e interoperáveis. Enquanto o registrador de adaptadores e o produtor de séries temporais foram desenvolvidos na linguagem de programação PHP, a linguagem Javascript foi adotada para desenvolver o integrador de fontes de dados. Essas linguagens foram utilizadas por familiaridade e por atenderem as necessidades da arquitetura. Para armazenar e recuperar as informações necessárias para o funcionamento dos microsserviços, o sistema gerenciador de banco de dados MySQL também foi escolhido por familiaridade com a tecnologia.

Como podemos observar na Figura 6, a implementação da arquitetura é composta por 6 componentes de *software*, onde cada um possui suas responsabilidades e funções definidas de acordo com o modelo conceitual. O produtor de séries possui apenas permissão para leitura no banco de dados, tendo em vista que ele apenas disponibiliza os dados em formato JSON para o componente *Frontend*. O componente registrador de adaptadores possui permissão de escrita para registrar na arquitetura de serviço as regras dos adaptadores. O integrador de fontes de dados possui permissões para leitura e escrita, pois ele é responsável por armazenar as observações e recuperar as regras cadastradas pelo registrador de adaptadores para associar a observação à regra cadastrada.

O Quadro 1 apresenta a descrição dos atributos da requisição para cadastrar regras de adaptadores. A coluna 'Atributo' representa o nome do atributo que deve ser enviado ao registrador de adaptadores na requisição de cadastro das regras, e a coluna 'Descrição', descreve os atributos da requisição enviada. No Quadro 2 podemos observar o exemplo do corpo da requisição com os valores preenchidos. Os valores são referentes ao adaptador da fonte de dados do Cemaden e para cadastrá-los o adaptador, no qual a regra pertence, deve enviar uma requisição *POST*, que é um método suportado pelo protocolo HTTP, para o serviço registrador de adaptadores. As próximas seções detalham o funcionamento de cada serviço da arquitetura.

Figura 6 – Diagrama de componentes da implementação da arquitetura de serviço para produzir séries temporais a partir de fontes de dados heterogêneas. As setas indicam a comunicação e o fluxo de dados entre os componentes. O estereótipo ≪ `seviço` ≫ indica que o componente é independente. A estereótipo ≪ `Página Web` ≫ indica que o componente é uma página Web para visualização das séries temporais.



Fonte: Autor.

3.1 SERVIÇO REGISTRADOR DE ADAPTADORES

O serviço registrador de adaptadores é responsável por conectar as fontes de dados na arquitetura e permitir que as observações sejam enviadas para o serviço integrador de fontes de dados. O serviço foi desenvolvido utilizando a linguagem PHP juntamente com o *framework* Laravel versão 8. Internamente, esse serviço possui um *endpoint* onde é possível registrar os adaptadores e receber como resposta um *token* de acesso. Um *endpoint* é um canal de uma API com o propósito de fornecer acesso aos recursos de um determinado serviço disponível da API. O *token* é uma cadeia de caracteres formada pela função *hash* MD5 aplicada ao código da regra – informado no momento da solicitação de registro. O *hash* MD5 retorna uma cadeia de caracteres de 128 bits contendo 32 caracteres utilizado para identificar as observações que os adaptadores enviam ao serviço integrador de fontes de dados. Todas as requisições enviadas para o serviço devem ser identificadas com o *token*.

A Figura 7 apresenta o diagrama de classes do serviço registrador de adaptadores. A classe `Regra` representa a regra a ser enviada para o arquitetura de serviço. A classe `GerenciadorRegra` recebe a requisição de solicitação de cadastro de regra, realiza a construção do objeto `Regra`, produz o token utilizando a função `gerarToken()`, onde é aplicada a função

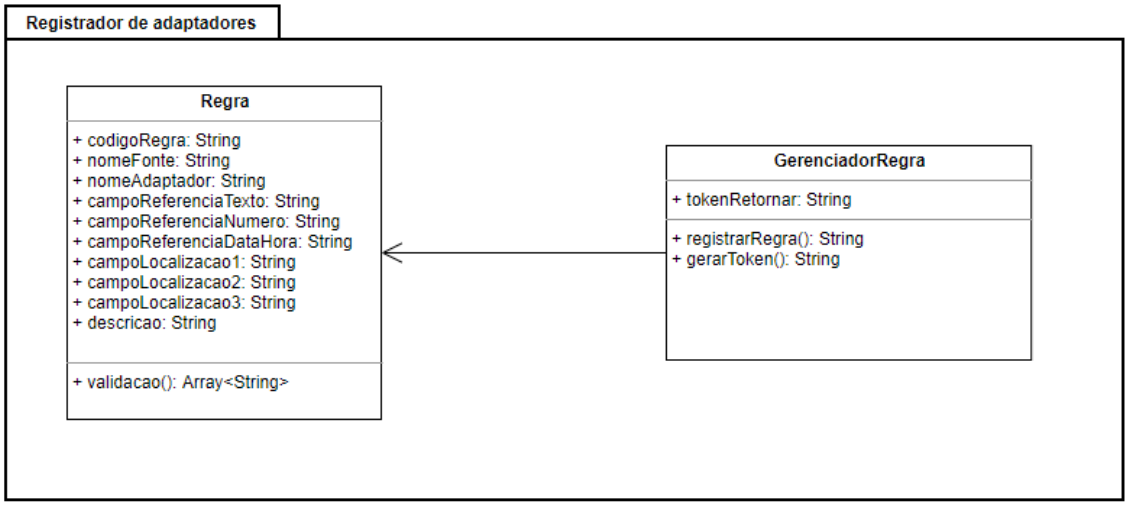
Quadro 1 – Descrição dos atributos da requisição de registro de regras.

Atributo	**Descrição**
`codigoRegra`	Código para identificar a regra.
`nomeFonte`	Nome da fonte de dados de onde são coletadas as observações.
`nomeAdaptador`	Nome do adaptador.
`atributos`	Vetor de objetos com nome e tipo do atributo.
`campoReferenciaTexto`	Campo texto referência para geração das séries temporais.
`campoReferenciaNumero`	Campo numérico referência para a geração das séries temporais.
`campoReferenciaDataHora`	Campo que representa a data da observação.
`campoLocalizacao1`	Campo referência para localização.
`campoLocalizacao2`	Campo referência para a localização.
`campoLocalizacao3`	Campo referência para a localização.
`campoCodigo`	Campo que representa o código das observações
`descricao`	Campo que descreve a regra

Fonte: Autor.

hash MD5 ao atributo `codigoRegra` apresentado no Quadro 2, realiza a persistência da regra no banco de dados e retorna o *token* para o solicitante.

Figura 7 – Diagrama de classe do serviço registrador de adaptadores.



Fonte: Autor.

Quadro 2 – Exemplo de uma requisição de registro de regras.

```
{
        "codigoRegra" : "cemaden-pluviometrica",
        "nomeFonte" : "cemaden",
        "nomeAdaptador" : "cemaden-p",
        "atributos" : [
                {"nome": "codestacao", "tipo" : "string"},
                {"nome": "tipo", "tipo" : "dateTime"},
                {"nome": "latitude", "tipo" : "number"},
                {"nome": "longitude", "tipo" : "number"},
                {"nome": "chuva", "tipo" : "number"},
                {"nome": "dataHora", "tipo" : "dateTime"},
                {"nome": "nome", "tipo" : "string"},
                {"nome": "uf", "tipo" : "string"},
                {"nome": "cidade", "tipo" : "string"}
        ],
        "campoReferenciaTexto" : null,
        "campoReferenciaNumero" : "chuva",
        "campoReferenciaDataHora" : "dataHora",
        "campoLocalizacao1" : "latitude",
        "campoLocalizacao2" : "longitude",
        "campoLocalizacao3" : "cidade",
        "campoCodigo" : "codestacao",
        "descricao" : "Regra referente ao
        adaptador que coleta dados do cemaden"
}
```

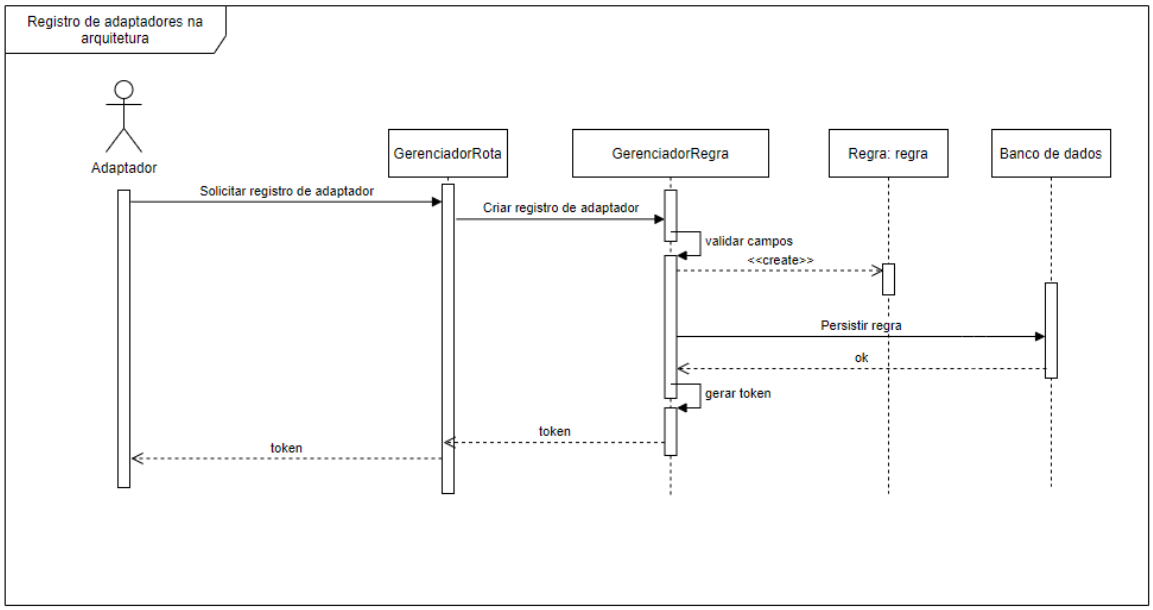
Fonte: Autor.

A Figura 8 apresenta o diagrama de sequência que demonstra o processo de registro de adaptadores na arquitetura. O usuário externo `Adaptador` envia uma requisição solicitando o registro de um adaptador na arquitetura e o objeto 'GerenciadorRota' recebe a requisição e encaminha ao 'GerenciadorRegra' que realiza a validação de acordo com os campos apresentados no Quadro 1. O objeto 'GerenciadorRegra' também realiza a criação de um objeto da classe `Regra` e realiza a persistência dele no banco de dados. Em seguida, o *token* é gerado e enviado ao adaptador.

3.2 SERVIÇO INTEGRADOR DE FONTES DE DADOS

O serviço integrador de fontes de dados foi desenvolvido utilizando a linguagem de programação Javascript juntamente com o *framework* Node Js versão 14 adotando o paradigma de programação orientada a eventos. A programação orientada a eventos é um paradigma de programação onde a execução do programa se baseia em eventos externos e definidos por ciclos desses eventos e *callbacks* (BRYANT; LAGAR-CAVILLA, 2012). O *framework* Node Js foi escolhido por permitir programação assíncrona com bom desempenho, requisito exigido pelos adaptadores para realizar requisições simultaneamente. O integrador de fontes de dados é o

Figura 8 – Diagrama de sequência do registro de adaptadores na arquitetura.



Fonte: Autor.

serviço mais importante da arquitetura, pois ele recebe as requisições dos adaptadores, com as observações, e realiza uma validação dos dados de acordo com as regras cadastradas para esse adaptador e logo após isso, persiste as observações no banco de dados. O *token* que foi gerado pelo componente registrador de adaptadores, quando as regras para esse adaptador foram registradas, é usado como um identificador de cada requisição. O componente realiza uma consulta no banco de dados, buscando a regra na qual possui o *token* informado, recupera a regra do adaptador, valida as observações que foram repassadas pelos adaptadores e persiste as informações no banco de dados. Esse processo se aplica a cada requisição feita ao serviço.

O Quadro 3 contém uma representação de como os dados são gravados no banco de dados pelo integrador de fontes de dados. Os dados de todos os adaptadores são convertidos para os campos que se encontram na primeira coluna do quadro. Esses valores apresentados representam uma observação coletada pelo Adaptador Cemaden Pluviométrico. Os campos cidade e estados são preenchidos com base na latitude e longitude fornecida, para isso foi utilizado a API de Geocoding reverso Nominatim API, que retorna os dados de endereço de uma determinada coordenada. O Geocoding reverso consiste em uma extração de informações textuais, como nome, endereço, cidade, através de coordenadas geográficas como latitude e longitude (KOUNADI et al., 2013). A utilização desse serviço para preencher a cidade e o estado da observação, se deve ao fato de que as observações coletadas pelo adaptador Twitter, podem não ter a localização exata do registro, pois os usuários podem alterar esses dados em seus perfis na rede social, sendo assim, no momento da integração, utiliza-se a latitude e a

longitude no qual se originou essa observação e com esses valores e a utilização da Api de Geocoding reverso, conseguimos o endereço exato do registro.

Quadro 3 – Exemplo de uma observação registrada no banco de dados.

Coluna no banco de dados	**Valor**
`id`	6167
`regra_id`	1
`latitude`	-23.58935
`longitude`	-48.04417
`cidade`	Itapetininga
`estado`	São Paulo
`dataHora`	2021-05-31 23:10:00
`valorNumero`	0.2
`valorTexto`	NULL
`codigo`	352703A

Fonte: Autor.

A Figura 9 apresenta o diagrama de sequência que demonstra o registro das observações no integrador de fontes de dados. O ator externo 'Adaptador' requisita o registro de uma observação ao serviço integrador. O objeto 'GerenciadorRota' recebe a requisição e encaminha para o objeto 'IntegradorFonteController' que recupera a regra cadastrada do adaptador requisitante através do token informado na requisição, realiza a validação com base nas regras e persiste a observação no banco de dados, logo em seguida retorna uma mensagem de sucesso.

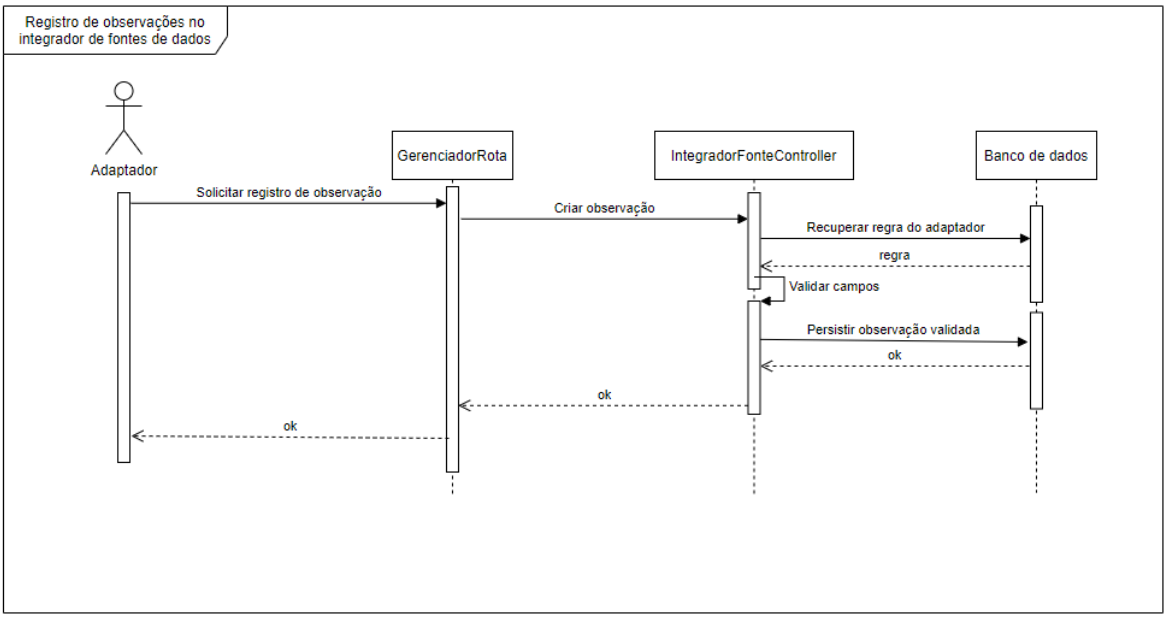
3.3 PRODUTOR DE SÉRIES TEMPORAIS

O produtor de séries é o componente responsável por gerar séries temporais das observações registradas no banco de dados pelo integrador de fontes de dados. O componente foi desenvolvido também utilizando a linguagem PHP com o framework Laravel versão 8. O componente possui um *endpoint* onde recebe uma requisição GET dos consumidores que desejam ter acesso às observações registradas no banco de dados. Essa requisição enviada ao Produtor de séries contém alguns parâmetros na URL para que seja possível a busca de dados na base. O Quadro 4 apresenta todos os parâmetros que devem ser enviados na URL ao produtor de séries temporais.

A Figura 10 apresenta o diagrama de classes do serviço produtor de séries temporais. A classe ControladorRequisicao é responsável por receber as requisições das aplicações consumidoras, validar os parâmetros informados na requisição e enviar os dados para a classe ProdutorSeries. A classe ProdutorSeries é responsável por receber os dados validados do controlador da requisição e produzir as séries temporais.

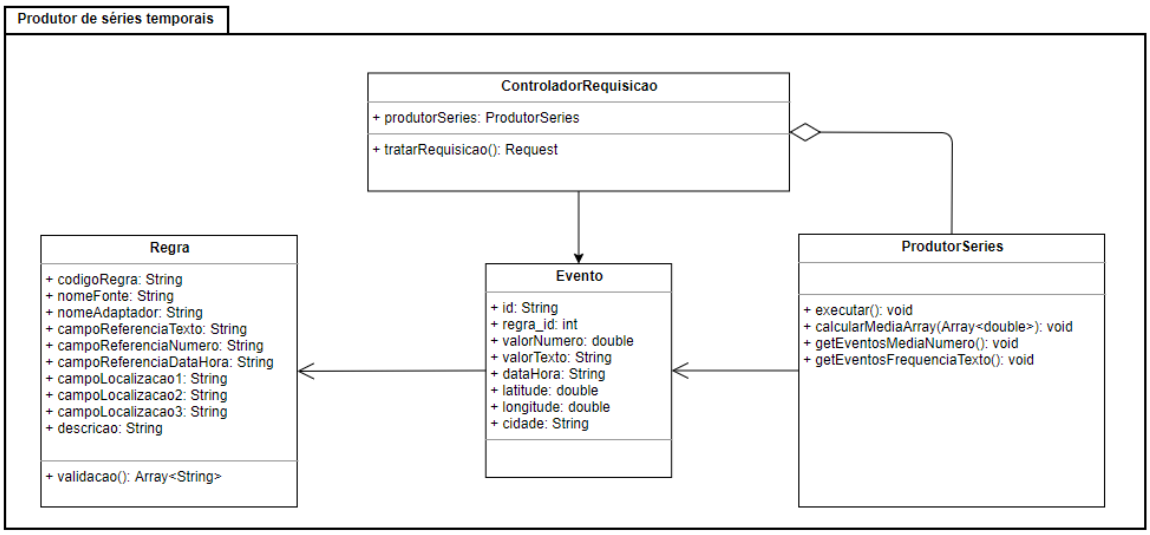
A Figura 11 apresenta o diagrama de sequência do processo de produção de séries temporais. O 'Consumidor' solicita a produção enviando uma requisição para com os parâmetros apresentados no Quadro 4. O objeto 'GerenciadorRota' recebe a requisição e envia para o

Figura 9 – Diagrama de sequência do registro de observações no serviço integrador de fontes de dados.



Fonte: Autor.

Figura 10 – Diagrama de classe do serviço produtor de séries temporais.



Fonte: Autor.

'ControladorRequisicao' que faz um tratamento para verificar se os parâmetros informados estão válidos. Depois é criado uma instância da classe 'ProdutorSeries' e acionado os métodos que recuperam as observações do banco de dados e então é realizado um filtro de acordo

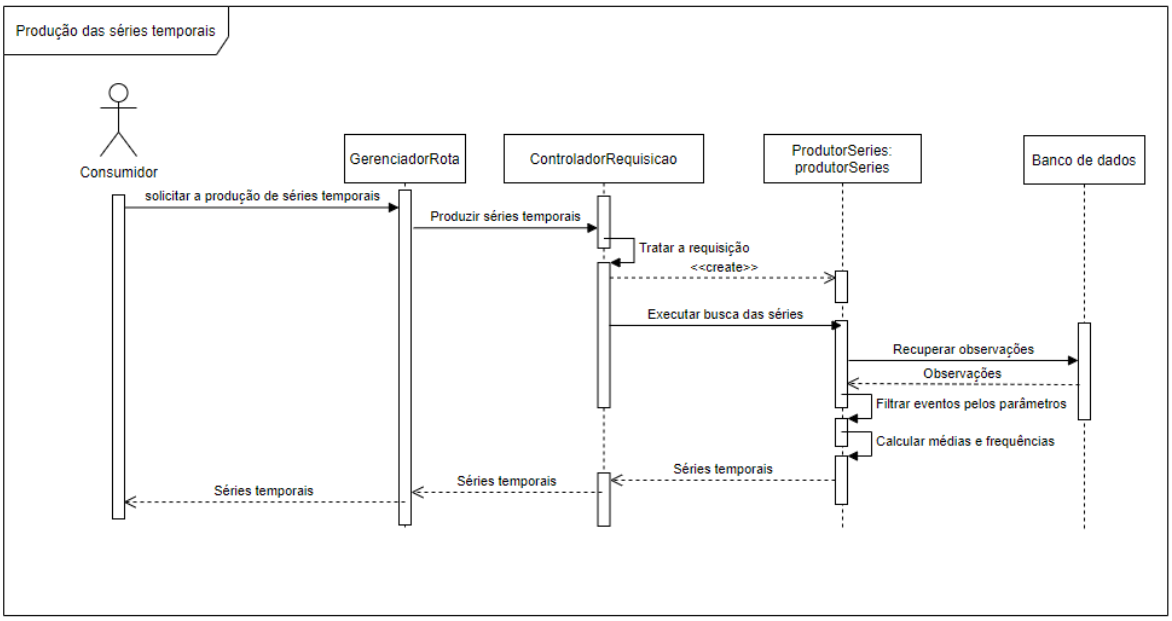
Quadro 4 – Parâmetros da mensagem de requisição para solicitação de séries temporais.

Parâmetro	**Descrição**
`regras_ids`	Esse campo representa os identificadores das regras registradas no banco de dados, caso a busca seja por mais de uma regra, o valor pode ser separado por vírgula, por exemplo: 13,11,45.
`data_inicio`	Esse campo representa a data de início dos eventos requisitados. Exemplo de uso: 2021-04-01
`agrupador`	Esse campo representa a escala temporal dos dados, estão disponíveis: hora e dia. Exemplo de uso: dia
`palavra _chave`	Esse campo representa o texto de busca para os dados do tipo string que são coletados pelo integrador de fontes de dados. No exemplo implementado, apenas o adaptador Twitter utiliza o valor texto para o campo de referência. Exemplo de uso: chuva
`data_fim`	Esse campo representa a data final dos eventos requisitados. Exemplo de uso: 2021-04-15
`cidade`	Esse campo representa a cidade no qual os eventos são gerados. Exemplo de uso: São paulo
`estado`	Esse campo representa o estado no qual os eventos são gerados. Exemplo de uso: São Paulo

Fonte: Autor.

com os parâmetros. Logo após isso, é realizado os cálculos de médias para as observações de base numérica e os cálculos de frequência para as observações com base textual e finalmente retornando as séries temporais.

Figura 11 – Diagrama de sequência da produção de séries temporais.



Fonte: Autor.

4 APLICAÇÃO DA ARQUITETURA DE SERVIÇO

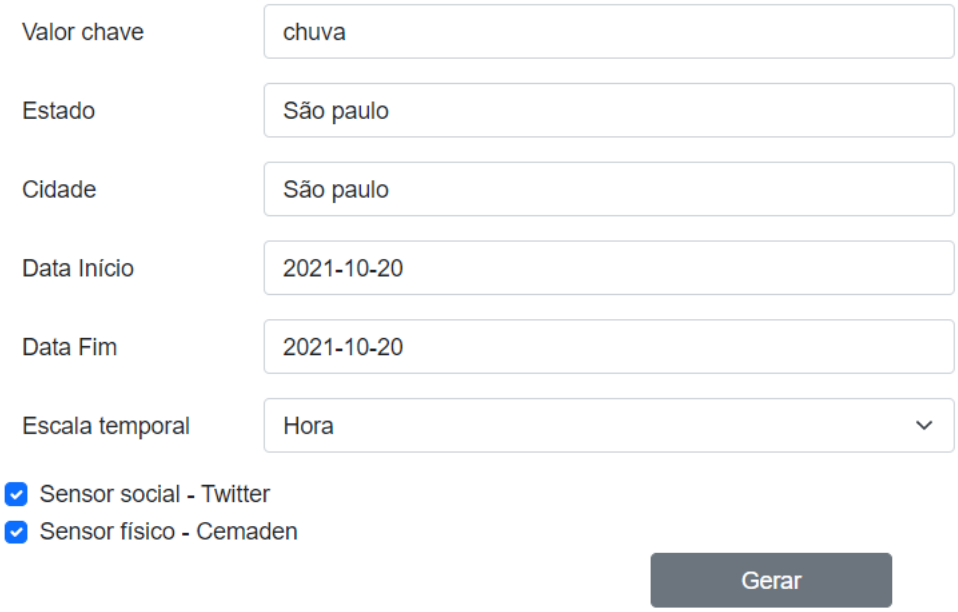
Para demonstrar a aplicação da arquitetura de serviço para produção de séries temporais, uma aplicação cliente foi desenvolvida para consumir e contrastar séries temporais relacionadas à percepção do fenômeno de chuva. Para isso, duas fontes de dados foram registradas na arquitetura de serviço através dos adaptadores implementados nos apêndices A e B. A primeira fonte de dados corresponde aos sensores pluviométricos do CEMADEN e a segunda o Twitter. Sendo a atividade do Twitter uma fonte de dados rica e potencialmente usada para derivar sinais sociais relacionados à percepção da chuva (ANDRADE et al., 2017), ela foi usada como sensor humano (GOODCHILD, 2007).

A aplicação consumidora foi desenvolvida com a linguagem de programação *Javascript* juntamente com a biblioteca *ReactJs* para construção de interface com o usuário. Também foi utilizado a biblioteca *Axios*, que permite realizar de maneira simples requisições para serviços externos. A escolha dessas tecnologias se justifica por serem novas e amplamente utilizadas em serviços de troca de mensagem. Para a exibição dos gráficos de séries temporais, foi utilizado a biblioteca *Chart.js*. Essa biblioteca possui uma configuração simples e objetiva e permite plotar gráficos com um ótimo desempenho e uma boa aparência.

O serviço produtor de séries temporais disponibiliza um `endpoint` onde o consumidor envia uma requisição com alguns parâmetros e obtém como resposta as séries temporais produzidas. Na aplicação consumidora desenvolvida, esses parâmetros são enviados através de um formulário. Um formulário é um ponto de interação entre o usuário e um sistema web, com ele é possível que usuários enviem dados para o sistema web através do preenchimento de campos. A Figura 12 apresenta o formulário da aplicação consumidora onde é possível preencher os valores que são os parâmetros para o componente produtor de séries temporais produzir as séries temporais. Os campos no formulário equivalem aos parâmetros enviados ao componente produtor de séries temporais apresentados no Quadro 4 da seção 3.3. Ao final do formulário há um componente Checkbox onde é listado as fontes de dados disponíveis onde é possível selecionar quais fontes serão utilizadas para a produção das séries temporais.

Ao submeter o formulário clicando no botão Gerar, a aplicação envia uma requisição para o produtor de séries temporais com os parâmetros informados no formulário. As séries temporais obtidas como resposta da requisição são plotadas em um gráfico de linha gerado com a biblioteca *Chart.js* (Figura 13). Cada linha pode ser ocultada de acordo com a preferência de quem utilizar a tela. Os adaptadores que trabalham com fontes de dados baseadas em texto, cada ponto no gráfico irá representar a frequência de aparição da palavra-chave informada no campo 'Valor chave'. A palavra-chave é um parâmetro enviado do consumidor para o produtor de séries temporais e com ela podemos filtrar as observações que serão utilizadas nas séries temporais de fontes de dados textuais. Como foi analisado o fenômeno da chuva, o campo 'Valor chave' informado foi a palavra 'chuva', sendo assim, cada ponto no gráfico gerado

Figura 12 – Aplicação consumidora do serviço de produção de séries temporais. Os campos correspondem aos parâmetros que serão enviados ao produtor de séries temporais para a produção das séries temporais.

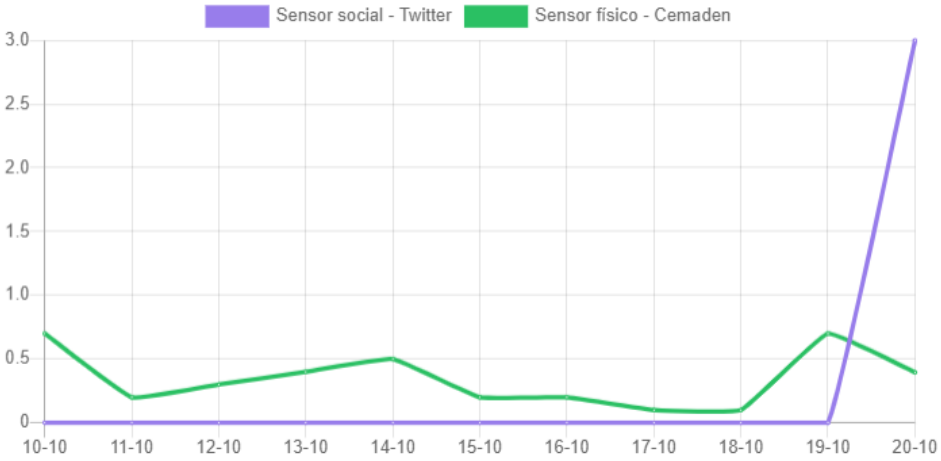


Fonte: Autor.

representa a frequência absoluta de mensagens coletadas da fonte de dados do Twitter que contenham a palavra 'chuva'. Para os adaptadores que trabalham com as fontes de dados onde o valor referência é numérico, cada ponto no gráfico irá representar a média dos valores registrados para aquela determinada data.

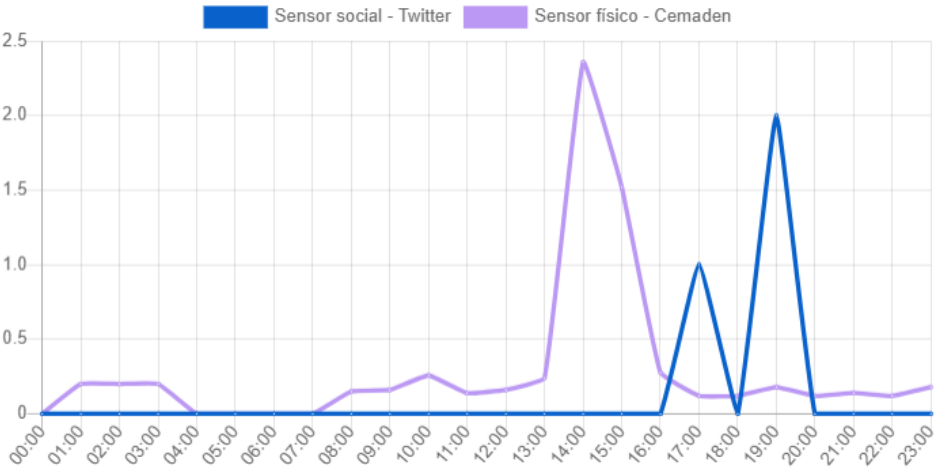
A Figura 14 apresenta as séries temporais com escala temporal em horas para a data de 20 de outubro de 2021, ou seja, no campo 'Escala temporal' do formulário apresentado na Figura 12 foi informado o valor 'hora' sendo assim cada ponto no gráfico representa os valores para aquele determinado horário.

Figura 13 – Séries temporais com resolução diária derivadas das fontes de dados Twitter (linha roxa) e Cemaden pluviométrico (linha verde).



Fonte: Autor.

Figura 14 – Séries temporais com resolução por hora derivadas das fontes de dados do Twitter (linha azul) e Cemaden pluviométrico(linha rosa).



Fonte: Autor.

5 CONCLUSÃO

Séries temporais são artefatos usados em diversos domínios para apoiar análises e tomadas de decisão. O processo de produção de séries temporais pode ser uma tarefa complexa dependendo da fonte de dados. Por exemplo, redes sociais podem fornecer dados em formato de texto, áudio, vídeo ou imagem. Essa complexidade pode ser ainda maior quando envolve múltiplas fontes de dados heterogêneas que requerem abordagens baseadas em regras para automatizar a produção e integração de séries temporais.

O presente trabalho propôs uma arquitetura de serviço para a produção de séries temporais a partir de fontes de dados heterogêneas. O modelo conceitual da arquitetura de serviço foi apresentado no Capítulo 2, onde os componentes foram descritos em termos de objetivo, função e interconexão. Em seguida, uma implementação da arquitetura de serviço foi apresentada no Capítulo 3. Além das tecnologias utilizadas para implementar os componentes conceituais, diagramas de componentes, classe e sequência detalharam o projeto da arquitetura de serviço.

Um dos benefícios da arquitetura proposta é permitir adicionar de forma flexível e extensível fontes de dados heterogêneas através de adaptadores. A arquitetura realiza a integração dessas fontes de dados e produz séries temporais que contribuem para analisar e contrastar as informações de um determinado fenômeno ou processo do mundo real percebido por diferentes fontes de dados. Uma aplicação foi desenvolvida para demonstrar como consumir o serviço da arquitetura (Capítulo 4).

Possíveis extensões deste trabalho incluem a adoção de controles de acesso com *tokens*, a utilização de *docker* para gerenciar os microsserviços, a melhoria de desempenho no componente produtor de séries temporais e extensão de adaptadores para outras redes sociais. Segundo os preceitos da Engenharia de Software, o desenvolvimento de *software* é um trabalho contínuo e envolve correções, melhorias, aprimoramentos e evoluções (PRESSMAN, 1994). Dessa maneira, são sugestões de melhoria da arquitetura de serviço para produção de séries temporais:

- Criação de documentação para os serviços.
- Adicionar recursos para captação *online* das informações da arquitetura, utilizando ferramentas como *Stream API* que possibilita a visualização das informações em tempo real.
- Melhorar o desempenho da geração das séries temporais, pois quanto maior é o número de consumidores da arquitetura, mais lento se torna a geração das séries temporais.
- Realizar testes em escalas e com diferentes fontes.
- Implementar um sistema de autenticação na comunicação entre os componentes.
- Utilizar o ATAM (*Architecture tradeoff analysis method*) para validar a arquitetura (KAZMAN et al., 1998).

- Experimentação de usabilidade e aplicabilidade com usuários.

Referências

ANDRADE, S. C. de et al. Mining rainfall spatio-temporal patterns in twitter: A temporal approach. In: BREGT, A. et al. (Ed.). **Societal Geo-innovation**. Cham: Springer International Publishing, 2017. p. 19–37. ISBN 978-3-319-56759-4. Citado 3 vezes nas páginas 12, 13 e 28.

ANDRADE, S. C. de et al. A multicriteria optimization framework for the definition of the spatial granularity of urban social media analytics. **International Journal of Geographical Information Science**, Taylor and Francis, v. 35, n. 1, p. 43–62, 2021. Disponível em: <https://doi.org/10.1080/13658816.2020.1755039>. Citado na página 15.

ANDRADE, S. de et al. An architectural model for retrieving data from social media apis: an instantiation for consuming the twitter stream api. 12 2018. Citado na página 36.

BRINKKEMPER, S. Formalisation of information systems modelling. 01 1990. Citado na página 14.

BRYANT, R.; LAGAR-CAVILLA, H. A. Snowflock. In: BROWN, A.; WILSON, G. (Ed.). **The Architecture of Open Source Applications**. [S.l.: s.n.], 2012. Citado na página 22.

CASTANHARI, R. et al. A software architecture to integrate sensor data and volunteered geographic information for flood risk management. In: . [S.l.: s.n.], 2016. Citado na página 14.

FOWLER, J. L. M. **Microservices**. 2014. Disponível em: <http://martinfowler.com/articles/microservices.html>. Acesso em: 10 de outubro de 2021. Citado na página 19.

GAMMA, E. et al. **Design Patterns: Elements of Reusable Object-Oriented Software**. USA: Addison-Wesley Longman Publishing Co., Inc., 1995. ISBN 0201633612. Citado na página 15.

GOODCHILD, M. F. Citizens as sensors: the world of volunteered geography. **GeoJournal**, v. 69, n. 4, p. 211–221, Aug 2007. ISSN 1572-9893. Disponível em: <https://doi.org/10.1007/s10708-007-9111-y>. Citado 2 vezes nas páginas 13 e 28.

KAZMAN, R. et al. The architecture tradeoff analysis method. In: **Proceedings. Fourth IEEE International Conference on Engineering of Complex Computer Systems (Cat. No.98EX193)**. [S.l.: s.n.], 1998. p. 68–78. Citado na página 31.

KOUNADI, O. et al. Accuracy and privacy aspects in free online reverse geocoding services. **Cartography and Geographic Information Science**, v. 40, p. 140–153, 2013. Acesso em: 10 de outubro de 2021. Citado na página 23.

MINITAB. **Suporte ao Minitab 18**. 2019. Disponível em: <https://support.minitab.com/pt-br/minitab/18/help-and-how-to/graphs/how-to/time-series-plot/before-you-start/overview>. Acesso em: 06 de dezembro de 2021. Citado na página 12.

MOREIRA, D. M. B. P. F. M. Desenvolvimento de aplicações e micro serviços: Um estudo de caso. **Revista TIS**, v. 4, n. 3, p. 391–410, 2015. Acesso em: 10 de outubro de 2021. Citado na página 19.

MORETTIN, P. A.; TOLOI, C. M. d. C. **Análise de séries temporais**. [S.l.]: Edgard Blucher, 2004. Citado na página 12.

PRESSMAN, R. S. **Software Engineering: A Practitioner's Approach**. European. [S.l.]: McGraw-Hill, 1994. Adapted by Darrel Ince. Citado na página 31.

SOMMERVILLE, I. **Software Engineering**. 9. ed. Harlow, England: Addison-Wesley, 2010. ISBN 978-0-13-703515-1. Citado na página 36.

SOUZA, A. de et al. Impacto ee fatores meteorológicos sobre as concentrações de ozônio modelados por análise de séries temporais e métodos estatísticos multivariados holos. **Instituto Federal De Educação, Ciência E Tecnologia Do Rio Grande Do Norte Natal, Brasil**, v. 5, p. 2–16, 2017. Citado na página 12.

STEIGER, E. et al. Twitter as an indicator for whereabouts of people? Correlating Twitter with UK census data. **Computers, Environment and Urban Systems**, v. 54, p. 255–265, 2015. ISSN 0198-9715. Citado na página 12.

TSAY, R. **Analysis of financial time series**. 2. ed.. ed. Hoboken, NJ: Wiley-Interscience, 2005. (Wiley series in probability and statistics). ISBN 978-0-471-69074-0. Disponível em: <http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+483463442&sourceid=fbw_bibsonomy>. Citado na página 12.

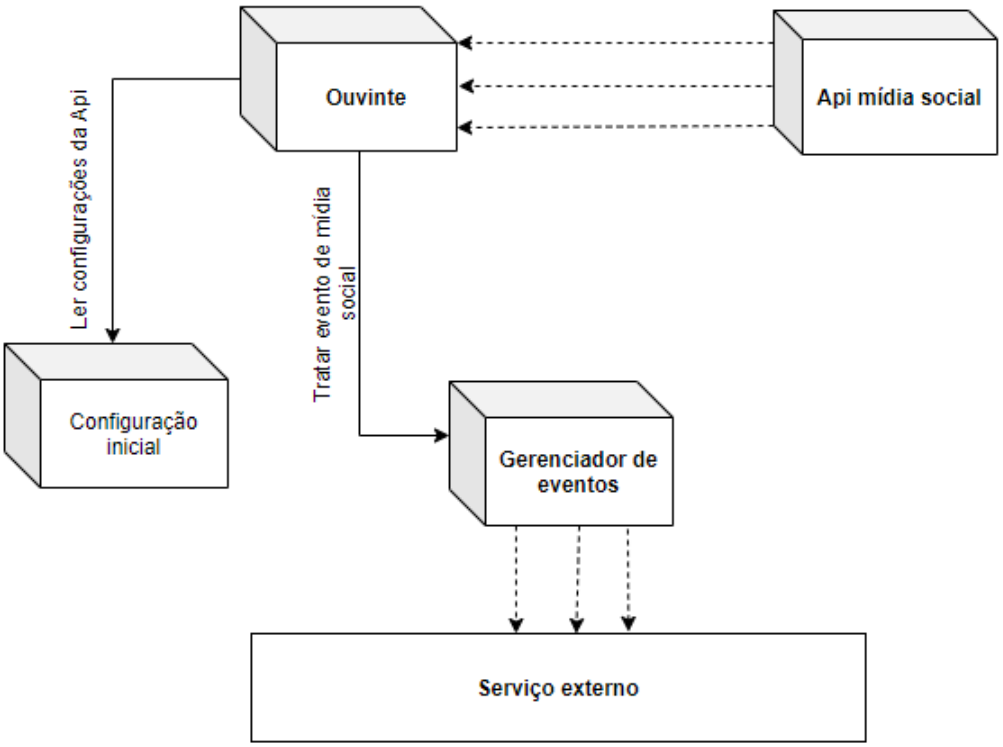
Apêndices

APÊNDICE A – Adaptador da fonte de dados do Twitter

O Twitter é uma plataforma de rede social que permite usuários publicarem mensagens de até 140 caracteres denominadas *tweets*. Os *tweets* podem ser recuperados por meio de uma *Application Programming Interface* (API) (Interface de Programação de Aplicação) quando os usuários configuram seus perfis como públicos dentro da plataforma do Twitter. Uma API é um conjunto de definições e protocolos que permite integrar aplicações ou componentes de software (SOMMERVILLE, 2010). No caso do Twitter, a API permite integrar aplicações na plataforma ou recuperar os tweets públicos via streaming.

O adaptador Twitter foi desenvolvido com base no *crawler* de Andrade et al. (2018). A principal diferença está no artefato gerado – para este trabalho foi desenvolvido um adaptador para ser registrado na arquitetura de serviço, e não um *crawler* com armazenamento de dados. A Figura 15 apresenta o modelo conceitual do adaptador desenvolvido para consumir *tweets* via Twitter API Stream.

Figura 15 – Modelo conceitual do adaptador da fonte de dados Twitter.



Fonte: Autor.

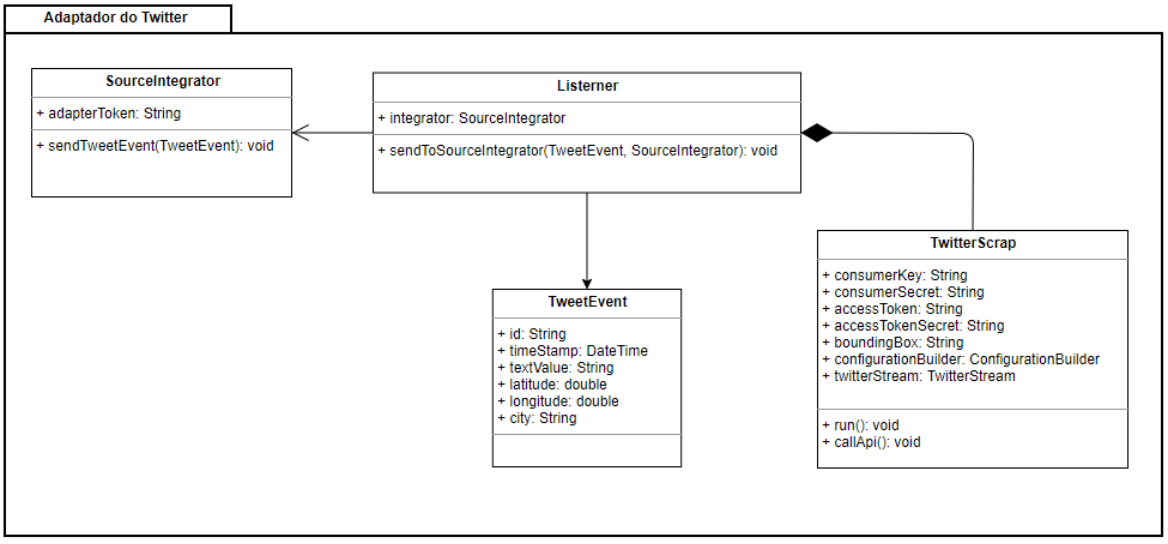
A linguagem de programação Java e a biblioteca Twitter4j (<https://twitter4j.org/>) foram usadas para construir o adaptador Twitter. A biblioteca Twitter4j permite abstrair a complexidade da comunicação com a Twitter API Stream, aumentando a produtividade no

desenvolvimento de crawlers e API REST consumidoras.

O componente *Configuração inicial* (Figura 15) é responsável por definir as configurações iniciais para a coleta dos *tweets*, como: definição das chaves de acesso à API, instanciação de objetos, e chamada do componente `Ouvinte`. O componente `Ouvinte`, por sua vez, mantém uma conexão ativa com a API do Twitter e permite observar e recuperar os tweets a serem enviados para o componente `Gerenciador de eventos`. O componente `Gerenciador de eventos` possui uma interface que permite a integração com a arquitetura de serviço proposta. As informações presentes no `Gerenciador de eventos` são essenciais para a comunicação com os demais componentes da arquitetura, tais como: a URL do integrador de fontes os métodos HTTP. Também é responsabilidade do `Gerenciador de eventos` formatar os dados de acordo com as regras do adaptador.

A Figura 16 apresenta o diagrama de classes da implementação do adaptador do Twitter. A classe `SourceIntegrator` é equivalente ao componente *Gerenciador de eventos* apresentado anteriormente no modelo conceitual. A classe `Listener` se equivale ao componente *Ouvinte* e a classe *TwitterScrap* se equivale ao componente *Configuração inicial*. A classe TweetEvent é usada pela classe Listener para a criação do objeto que é enviado para o SourceIntegrator.

Figura 16 – Diagrama de classes da implementação do adaptador da fonte de dados Twitter.



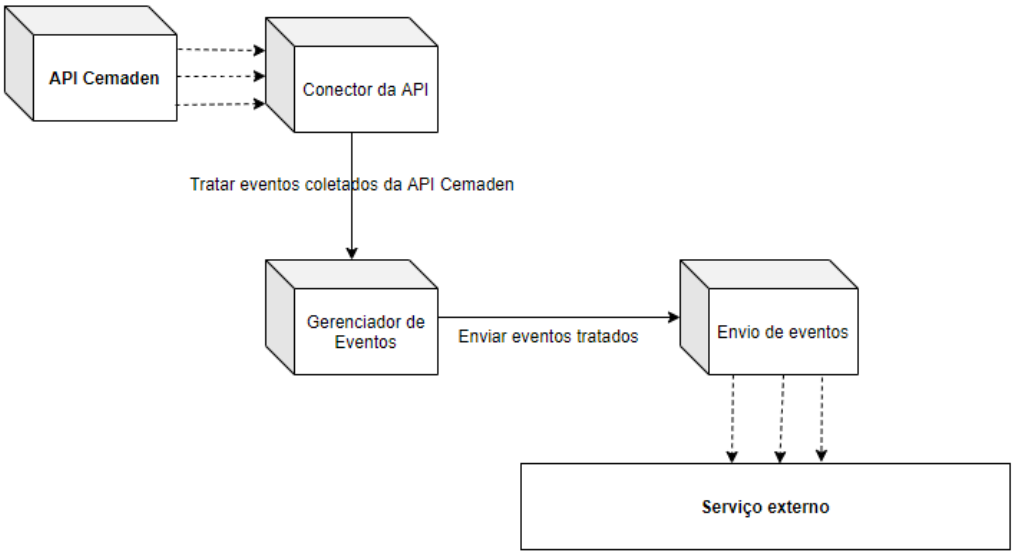
Fonte: Autor.

APÊNDICE B – Adaptador da fonte de dados do CEMADEN

O adaptador CEMADEN coleta os dados de sensores pluviométricos – medidas de intensidade de chuva em uma determinada área. Para desenvolver este adaptador, foi utilizado a linguagem de programação PHP juntamente com o *framework* Laravel versão 8. O Laravel é um *framework* PHP de código aberto que segue o modelo de arquitetura (*Model-View-Controller* (MVC). A escolha dessas tecnologias se deve a curva de aprendizado relativamente baixa do Laravel em relação a outros *frameworks*, sua relevância em relação aos demais *frameworks* e para demonstrar que os adaptadores da arquitetura proposta são inteiramente independentes de tecnologias.

A Figura 17 apresenta o modelo conceitual para o adaptador CEMADEN, o componente *Conector da API* é responsável por requisitar os dados da API do CEMADEN e enviá-los para o *Gerenciador de Eventos*. O gerenciador de eventos trata os dados em um formato padrão de envio para o serviço externo. Ao finalizar o tratamento dos dados, o gerenciador de eventos envia para o componente *Envio de Eventos*, responsável pela requisição ao *Serviço externo*.

Figura 17 – Modelo conceitual do adaptador da fonte de dados Cemaden

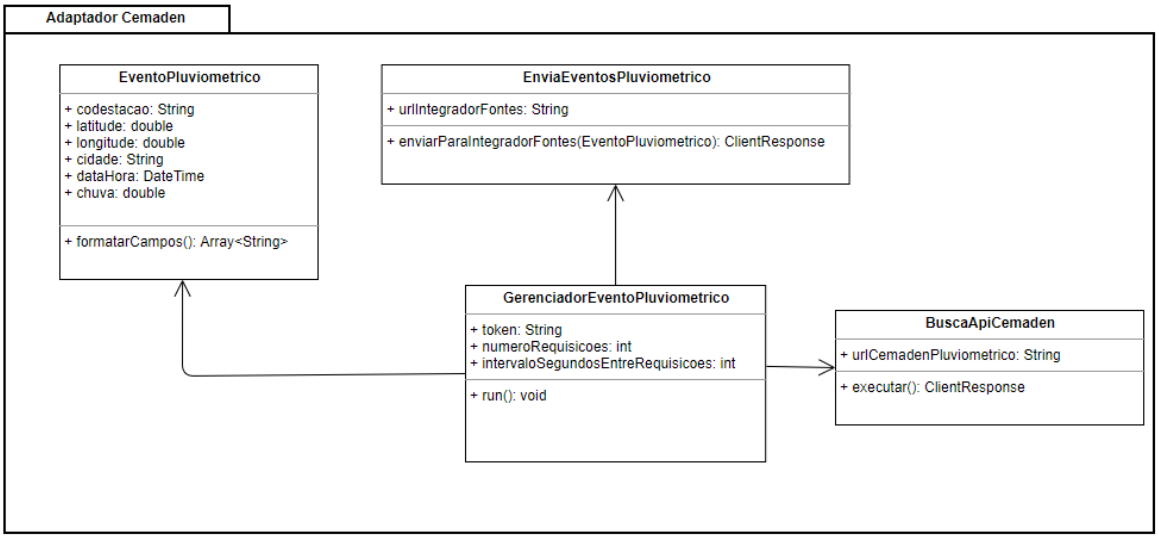


Fonte: Autor.

A Figura 18 apresenta o diagrama de classes para o adaptador da fonte de dados CEMADEN. A classe `EventoPluviometrico` possui os atributos que são enviados para o *Serviço externo* e um método que formata os dados. A classe `BuscaApiCemaden` é responsável por requisitar os dados da api do CEMADEN. Ela possui um atributo que representa a URL da api e um método que executa a ação de coleta. A classe `GerenciadorEventoPluviometrico`

possui os atributos *token*, que representa o identificador da regra cadastrada no Registrador de Adaptadores, `numeroRequisicoes`, que define a quantidade de requisições que serão feitas para o serviço externo, e `intervaloSegundosEntreRequisicoes`, esse último atributo foi adicionado pois há um intervalo de 10 minutos para que os valores sejam atualizados, com isso pode-se controlar esse tempo através do atributo. O método `run()` é o primeiro a ser executado no serviço e é responsável pela criação das instâncias necessárias para o funcionamento completo do serviço. A classe `EnviaEventosPluviometrico` faz o envio da observação tratada para o serviço externo que é o serviço integrador de fontes de dados.

Figura 18 – Diagrama de classes da implementação do adaptador da fonte de dados Cemaden.



Fonte: Autor.

APÊNDICE C – Disponibilidade do código fonte

O código fonte está disponível no github no endereço: <https://github.com/lucasestanislau/tcc-projects>. Cada componente está separado em sua respectiva pasta. O Github é uma plataforma de hospedagem de código colaborativo, construída com o sistema de controle de versão Git. O objetivo pelo qual o Github foi escolhido para disponibilização dos códigos, se dá ao fato de que ele é uma das plataformas mais populares atualmente e permite usufruir de todas as vantagens de estar trabalhando com uma ferramenta de controle de versão.