UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LUÍS FERNANDO MARIOTTI PAIVA

UMA INFRAESTRUTURA CONTEINERIZADA PARA PERSISTÊNCIA DE DADOS NA SENTILO: UM ESTUDO DE CASO EM CIDADES INTELIGENTES

TOLEDO

2022

LUÍS FERNANDO MARIOTTI PAIVA

UMA INFRAESTRUTURA CONTEINERIZADA PARA PERSISTÊNCIA DE DADOS NA SENTILO: UM ESTUDO DE CASO EM CIDADES INTELIGENTES

A Containerized Infrastructure for Data Persistence at Sentilo: A Case Study in Smart Cities

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação do Curso de Bacharelado em Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Fabio Alexandre Spanhol

Coorientador: Prof. Dr. Thiago Henrique Pereira Silva

TOLEDO

2022

LUÍS FERNANDO MARIOTTI PAIVA

UMA INFRAESTRUTURA CONTEINERIZADA PARA PERSISTÊNCIA DE DADOS NA SENTILO: UM ESTUDO DE CASO EM CIDADES INTELIGENTES

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação do Curso de Bacharelado em Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 15/dezembro/2022

Fabio Alexandre Spanhol
doutorado
Universidade Tecnológica Federal do Paraná

Sidgley Camargo de Andrade
doutorado
Universidade Tecnológica Federal do Paraná

Thiago Henrique Pereira Silva
doutorado
Universidade Tecnológica Federal do Paraná

TOLEDO

2022

Dedico este trabalho aos meus pais, que empregaram anos de suas vidas para garantir minha educação formal e me ensinaram lições de vida valiosas.

AGRADECIMENTOS

Primeiramente agradeço a Deus pela dádiva de conseguir chegar com saúde até o término dessa jornada.

Especialmente agradeço aos meus pais, minha irmã e demais familiares pelo apoio durante o período de graduação, ajudando-me a superar os momentos mais difíceis.

Agradeço aos professores, Dr. Fabio Alexandre Spanhol e Dr. Thiago Henirque Pereira Silva, por guiar-me durante o desenvolvimento deste projeto, sempre compartilhando seus conhecimentos.

Também agradeço aos meus colegas de curso que dividiram comigo as angústias, frustrações, alegrias e conquistas desses anos de graduação.

E finalmente, a todos que contribuíram direta ou indiretamente para a minha formação, muito obrigado!

You can have data without information, but you cannot have information without data.
(Daniel Keys Moran, DKM)

RESUMO

A Sentilo é uma plataforma de cidades inteligentes utilizada pela UTFPR-Toledo no projeto EnvCity que visa trazer benefícios de cidades inteligentes para o município de Toledo-PR. A principal funcionalidade da Sentilo é prover uma infraestrutura para receber e persistir dados obtidos por sensores distribuídos pela cidade, comunicar-se com atuadores e facilitar a recuperação dos dados armazenados atravéd de diversos aplicativos voltados à gestão pública. Para lidar com quantidades massivas de dados normalmente gerados por cidades inteligentes, a Sentilo tem uma infraestrutura distribuída em múltiplos servidores. Essa arquitetura torna um desafio manter a plataforma garantindo escalabilidade, segurança e manutenibilidade. Diante desse cenário, este trabalho criou uma infraestrutura para Sentilo através da virtualização de contêineres usando Docker. Como resultado, facilita-se a replicação de novas instâncias devidamente configuradas com protocolos de segurança, automatizando via código a tarefa de incorporar melhorias e correções de erros. A solução conteinerizada apresentada também incorporou as tecnologias de alto desempenho ElasticSearch e OpenTSDB para pesquisar grandes volumes de dados e lidar com séries temporais.

Palavras-chave: cidades inteligentes; sentilo; elasticsearch; opentsdb; internet das coisas.

ABSTRACT

Sentilo is a smart cities platform used by UTFPR-Toledo in the EnvCity project that aims to bring the benefits of smart cities to Toledo-PR municipality. Sentilo's main functionality is to provide an infrastructure to receive and persist data obtained by sensors distributed throughout the city, communicate with actuators, and facilitate the retrieval of stored data through various applications aimed at public management. To handle massive amounts of data normally generated by smart cities, Sentilo has a distributed infrastructure across multiple servers. This architecture makes it challenging to maintain the platform ensuring scalability, security, and maintainability. Given this scenario, this work created an infrastructure for Sentilo through container virtualization using Docker. As a result, the replication of new instances properly configured with security protocols is facilitated, automating via code the task of incorporating improvements and error corrections. The containerized solution also incorporated high-performance ElasticSearch and OpenTSDB technologies to search large volumes of data and handle time series.

Keywords: smart cities; sentilo; elasticsearch; opentsdb; internet of things.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

API	*Application Programming Interface*
GPS	*Global Positioning System*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
IETF	*Internet Engineering Task Force*
IoT	Internet das Coisas, do inglês *Internet of Things*
IP	*Internet Protocol*
JSON	*JavaScript Object Notation*
REST	*Representational State Transfer*
SO	Sistemas Operacionais
SSL	*Secure Sockets Layer*
TLS	*Transport Layer Security*
TSD	*Time Series Daemon*
URL	*Uniform Resource Locator*
UTFPR-TD	Universidade Tecnológica Federal do Paraná
VVM	*Virtual Machine Monitor*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo contextualiza o tema abordado neste trabalho, apresenta a justifica para a abordagem proposta, sumariza os objetivos e finaliza com a descrição do restante do documento.

1.1 Considerações iniciais

O desenvolvimento de tecnologias de comunicação sem fio, como *bluetooth*, *wifi* e *5G*, unidos a um constante decréscimo dos preços de microcontroladores, sensores e atuadores, propiciaram uma nova tendência no mundo da tecnologia conhecida como Internet das Coisas, do inglês *Internet of Things* (IoT). O termo “coisas“ refere-se a objetos do cotidiano como, por exemplo, computadores pessoais, telefones móveis, lâmpadas, eletrodomésticos, carros e roupas. Tais objetos são geralmente equipados com (a) sensores para coletar dados sobre o ambiente em que estão inseridos, (b) atuadores que realizam tarefas remotamente e (c) alguma tecnologia que permita o compartilhamento de informações para outros dispositivos (RAMASAMY; KADRY, 2021). Essa capacidade de comunicação de objetos através da Internet incentivou diversas inovações no âmbito acadêmico, comercial e industrial, e desta forma, IoT tem impactado tanto na qualidade de vida das pessoas quanto na economia mundial (KAUR, 2018).

Paralelamente, o rápido crescimento de regiões urbanizadas impôs novos desafios para os gestores desses ambientes urbanos como: desenvolvimento sustentável, educação de qualidade, consumo energético otimizado, cuidados ao meio ambiente, segurança dos moradores, entre outros serviços públicos (ARROUB *et al.*, 2016). Perante o exposto, cidades inteligentes (*smart cities*) podem ser definidas como aquelas unidades urbanas que estão preparadas para lidar com dificuldades ocasionadas pelo crescimento populacional acelerado, utilizando principalmente meios tecnológicos para tal. Nesse contexto, a Prefeitura Municipal de Toledo-Paraná firmou uma parceria com a Universidade Tecnológica Federal do Paraná (UTFPR-TD), campus Toledo, viabilizando um projeto nomeado EnvCity. O intuito é desenvolver diversas aplicações utilizando dados gerados por dispositivos de IoT que lidam com problemas urbanos do município como rastreamento de rotas de caminhões de coleta seletiva de resíduos urbanos e monitoramento da qualidade da água e do ar.

Para administrar os sensores e atuadores do projeto EnvCity, bem como gerenciar a captação, armazenamento e uso dos dados enviados por eles, é necessário implantar uma infraestrutura preparada para lidar com o tráfego intenso de dados, comum em redes IoT. Parte fundamental de tal infraestrutura é utilizar um serviço projetado para gerenciar cidades inteligentes, visto que este procura solucioar a maioria dos problemas citados anteriormente. Apesar de existirem múltiplas soluções disponíveis no mercado, como apontado por Santana *et al.* (2017), a plataforma escolhida pelos gestores do projeto EnvCity foi a Sentilo (Barcelona City Council, 2022). Vale ressaltar que para uma operação de grande porte como esta é imprescindível tam-

bém garantir que os servidores tenham sua manutenção facilitada e que sejam escaláveis, caso precisem tratar aumentos repentinos de demanda.

Diante do cenário apresentado, este trabalho propõe uma abordagem para gerar uma infraestrutura para o projeto de cidades inteligentes EnvCity que seja escalável, disponha boa manutenibilidade, utilize protocolos de segurança na comunicação entre os componentes do sistema e persista o grande volume de dados gerado pelos dispositivos IoT. A abordagem proposta é implementada sobre a tecnologia de contêineres, utilizando como ferramenta de virtualização Docker (Docker Inc., 2022). Nela será instalada a plataforma Sentilo para gerenciar os dados gerados pelos sensores distribuídos pelo município. Atendendo ao requisito de segurança a infraestrutura entregue não somente foi configurada para persistir os dados, através dos bancos de dados OpenTSDB (The OpenTSDB, 2022) e Elasticsearch (Elasticsearch B.V., 2022), mas também foi adicionada uma camada de criptografia entre cliente e servidor.

1.2 Justificativa

Os avanços na tecnologia de IoT estão viabilizando a implementação de projetos de grande porte como os relacionados a cidades inteligentes. Por conta do grande número de dispositivos que podem ser conectados a redes dessa natureza, é essencial garantir que os servidores consigam lidar com o grande volume de dados transmitidos. Nesse sentido, a tecnologia de contêineres apresenta certas vantagens para tais ambientes devido a sua leveza e flexibilidade comparadas às máquinas virtuais convencionais. Sendo assim, a implementação utilizando o Docker para a infraestrutura de servidores do projeto EnvCity pretende assegurar a longevidade do projeto, facilitando futura expansão e manutenção. Outros fatores críticos para essa aplicação são a persistência e segurança dos dados coletados. Neste quesito, os contêineres deste trabalho foram projetados para interagir com bancos de dados externos que garantem a durabilidade das informações coletadas, bem como operem sobre protocolos de segurança para dificultar o acesso de usuários não autorizados.

1.3 Objetivos

Esta seção apresenta a forma pela qual este trabalho pretende auxiliar na solução de alguns desafios inerentes a elaboração de uma infraestrutura de servidores para cidades inteligentes no contexto do projeto EnvCity.

1.3.1 Objetivo geral

Implementar uma solução de infraestrutura no contexto de aplicações de cidades inteligentes mediante a utilização da tecnologia de contêineres.

1.3.2 Objetivos específicos

- Utilizar o Docker para substituir a máquina virtual da Sentilo;
- Criar contêineres para os bancos de dados OpenTSDB e Elasticsearch;
- Configurar a persistência de dados por meio dos agentes de histórico e de monitoramento de atividade da plataforma Sentilo;
- Configurar o protocolo de segurança SSL/TLS;
- Gerar um teste automatizado para validar o funcionamento da infraestrutura;
- Instalar os servidores em um ambiente de produção do projeto EnvCity.

1.4 Estrutura do trabalho

O restante deste trabalho está organizado da seguinte forma. O Capítulo 2 traz o referencial teórico apresentando as tecnologias adotadas bem como informações que contextualizam o leitor sobre o caso de uso deste projeto. Já no Capítulo 3 são apresentadas as tecnologias utilizadas para o desenvolvimento da solução e a metodologia empregada para tanto. No Capítulo 4 são abordados os resultados obtidos com o desenvolvimento da solução e por fim o Capítulo 5 expõe as considerações finais.

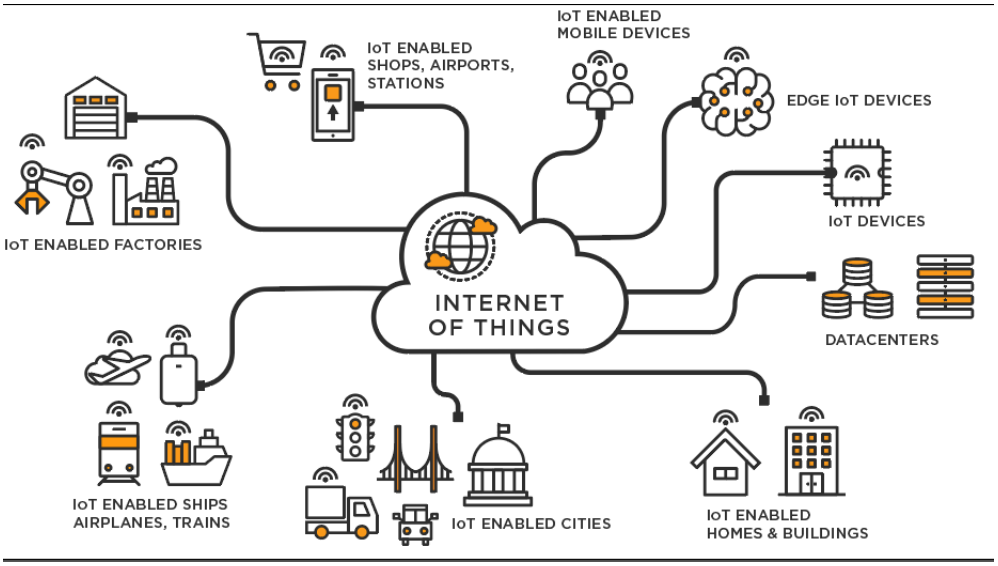
2 REFERENCIAL TEÓRICO

Este capítulo aborda as tecnologias relacionadas com este trabalho. Inicialmente são descritos os conceitos de IoT e cidade inteligentes, fundamentais para um entendimento do projeto EnvCity. Logo após, são tratados assuntos que servem de alicerce para a infraestrutura desenvolvida, tais como: diferenças entre contêineres e máquinas virtuais, Docker e os protocolos de segurança SSL/TLS. E são finalmente descritos os principais serviços do projeto virtualizados através do Docker.

2.1 Internet das coisas e cidades inteligentes

A IoT descreve uma infraestrutura de rede que interconecta diversos dispositivos heterogêneos para acumular e compartilhar dados entre si (RAJAB; CINKELR, 2018). Um dos objetivos desta tecnologia é tornar a Internet mais difusa e imersiva, visto que permite adaptá-la a diversos setores da sociedade moderna como, por exemplo, educação, saúde, esportes, mobilidade urbana, etc. Esse aspecto tem contribuído para a adoção em massa da tecnologia e um aumento expressivo de aparelhos conectados à Internet. Conforme um estudo realizado pela empresa Cisco, o número de dispositivos de IoT representarão 50% (14,7 bilhões) de todos os aparelhos globais ligados em rede até 2023 (Cisco, 2020). A Figura 1 ilustra alguns dos diferentes casos de uso de IoT.

Figura 1 – Casos de uso IoT.



Fonte: Extraído de (TIBCO, 2022).

A partir do ano de 2050, a população mundial viverá majoritariamente em centros urbanos. Este aumento da urbanização exige soluções inteligentes e sustentáveis para gerenciar as necessidades dos habitantes dessas regiões (ANAGNOSTOPOULOS *et al.*, 2017). Esse contexto está intrinsecamente relacionado com o conceito de cidades inteligentes. Apesar de

não existir uma definição formal para uma cidade inteligente, é de comum acordo que se refere àquelas que utilizam tecnologia da informação para oferecer melhores serviços e qualidade de vida para seus moradores (ISO, 2019). A seguir são apresentados possíveis cenários de implementação da IoT em cidades inteligentes:

- Gerenciamento de tráfego inteligente: congestionamentos de veículos em grandes centros urbanos, especialmente na hora do *rush*, são um grande problema para moradores, passageiros, autoridades locais e planejadores urbanos (AVATEFIPOUR; SADRY, 2018). Nesse quesito, sistemas de mapa, como o Google Maps e Waze, já utilizam o *Global Positioning System* (GPS) embutido nos aparelhos celulares para planejar rotas que evitem locais com tráfego intenso para seus usuários. A IoT pode ser útil também no monitoramento de vagas de estacionamento e sincronização de semáforos.

- Infraestrutura inteligente: uma cidade com uma infraestrutura de qualidade mínima é essencial para o bem-estar de seus habitantes. Diante disso, as prefeituras precisam construir novas pontes, estradas e prédios para sua população, além de realizar manutenções frequentemente para garantir uso ininterrupto desses serviços (SYED *et al.*, 2021). Dados levantados por sensores podem ser utilizados para balizar decisões de realização e manutenção dessas obras, garantindo o bom funcionamento da cidade.

- Rede elétrica inteligente: dezenas de milhões de vidas serão drasticamente afetadas devido às redes elétricas instáveis e não confiáveis (BARMAN *et al.*, 2018). Sendo assim, utilizar recursos de tecnologia de informação em redes elétricas terá um papel fundamental para torná-las mais eficientes, estáveis e sustentáveis. A IoT pode ser empregada para monitorar o consumo de energia, detectar falhas, reduzir o desperdício e garantir a segurança dos usuários, entre outros aspectos.

- Monitoramento inteligente do meio ambiente: se os cidadãos sentem-se pouco saudáveis vivendo na cidade, não podemos considerá-la uma cidade inteligente (RAJAB; CINKELR, 2018). A IoT permite, por exemplo, obter dados em tempo real sobre qualidade de água e do ar, permitindo que gestores públicos planejem ações para manejar melhor tais recursos fundamentais à vida. A tecnologia pode ser empregada também para reduzir desperdícios de recursos energéticos, evitar descarte irregular de dejetos urbanos e minimizar emissões de gases nocivos, como o gás carbônico.

2.2 Máquinas virtuais e contêineres

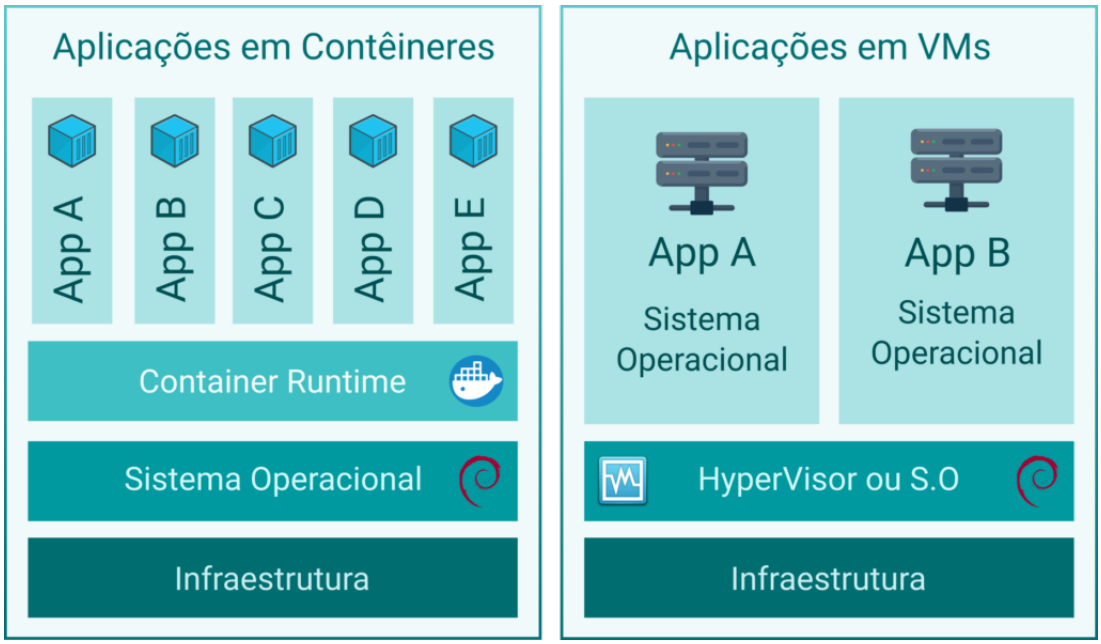
O conceito de virtualização existe desde da década de 1960, quando a IBM desenvolveu a tecnologia para possibilitar computação concorrente e iterativa nos seus computadores de grande porte (NANDA; CHIUEH, 2005). Para tanto, os engenheiros da empresa criaram **máquinas virtuais**, *softwares* capazes de emular um ou mais Sistemas Operacionais (SO) completos

num único computador físico. Essa nova técnica levou a uma diminuição dos gastos com aquisições de novos equipamentos e permitiu aumentar a produtividade da empresa com múltiplos usuários trabalhando simultaneamente no mesmo computador central.

O principal componente de uma máquina virtual é o *hypervisor* também conhecido por *Virtual Machine Monitor* (VVM). Este *software* atua entre o *hardware* dos computadores hospedeiros e as máquinas virtuais. Sua principal finalidade é alocar recursos computacionais disponíveis para serem utilizados pelos sistemas operacionais das máquinas virtuais (BAUMAN; AYOADE; LIN, 2015). Uma desvantagem de replicar um sistema operacional completo, como ocorre em uma máquina virtual, é que uma parte significante dos recursos alocados pelo VVM são utilizados apenas para o funcionamento do sistema operacional replicado em vez de serem aproveitados pelas aplicações instaladas nele (VITALINO; CASTRO, 2018).

Os avanços na tecnologia do *kernel* Linux permitiram o desenvolvimento de uma nova forma de virtualização baseada em contêineres (JOY, 2015). Um contêiner (*container*) Linux utiliza uma abordagem diferente do *hypervisor* de máquinas virtuais. Nele são gerados múltiplos ambientes isolados para executar diferentes processos que compartilham o mesmo *kernel* do computador hospedeiro. O fato dos contêineres gerarem ambientes mais leves que uma máquina virtual representa um aumento de eficiência em aplicações que utilizam desta tecnologia. A Figura 2 ilustra a diferença arquitetural entre a tecnologia de contêineres e de máquinas virtuais.

Figura 2 – Arquitetura geral de contêineres comparada com máquinas virtuais.



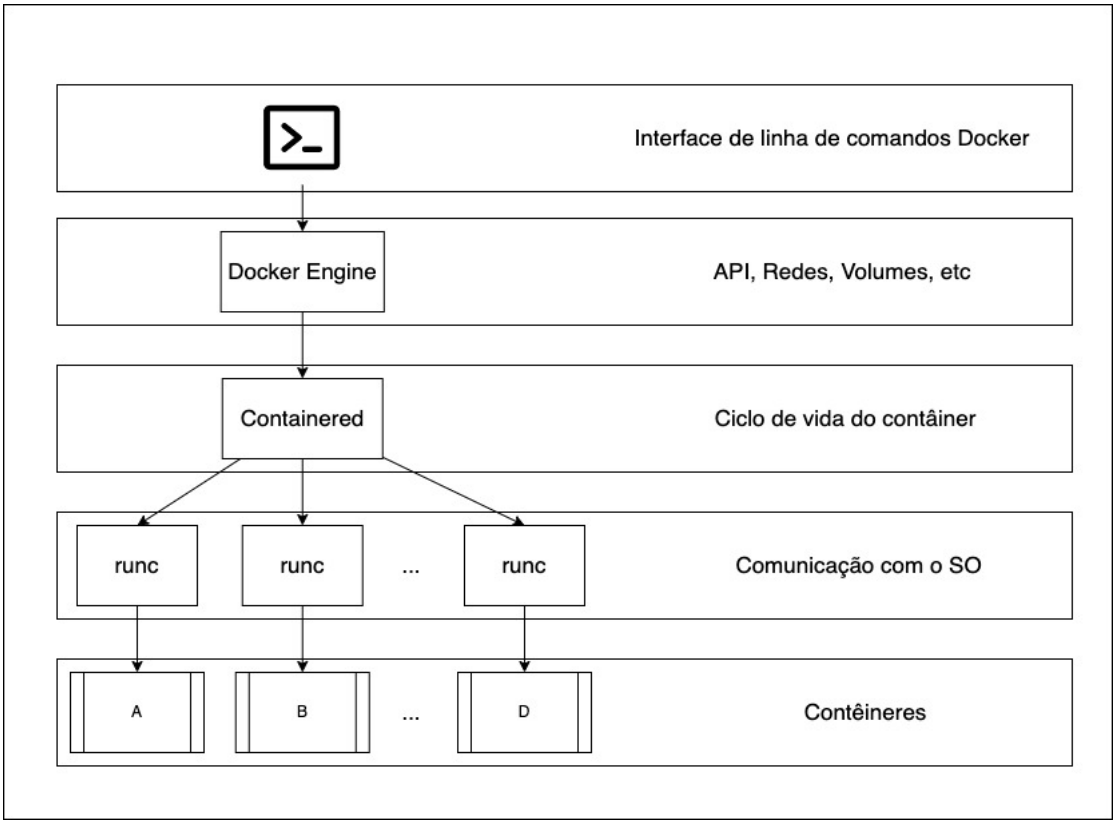
Fonte: Extraído de (4LINUX, 2022).

2.3 Docker

O Docker é uma plataforma de código aberto criada pela empresa Docker Inc. que facilita a utilização de contêineres Linux abstraindo do desenvolvedor múltiplos detalhes que envolvem esta tecnologia (TURNBULL, 2014). Esta ferramenta possibilita construir infraestruturas de servidores complexas que podem ser replicadas em qualquer computador que tenha um*kernel* Linux moderno e o Docker instalado. Além disso, por utilizar a virtualização baseada em contêineres, garante a seus usuários a execução de aplicações em ambientes isolados e de alto desempenho.

A tecnologia por trás do Docker pode ser separada entre duas camadas, a do *runtime* e do *daemon* (POULTON, 2018). O *runtime* é o nível mais baixo da aplicação e pode ser subdivido entre o *runc* responsável por executar comandos no SO e o *containerd* que gerencia o ciclo de vida de um contêiner. Um nível de abstração acima encontra-se o *daemon* ou *Docker engine* que é acessível aos usuários sendo encarregado de tarefas como gerenciar as redes e volumes dos contêineres. A Figura 3 ilustra a divisão entre as duas camadas.

Figura 3 – *Runtime* e *Daemon*.



Fonte: Autoria própria.

2.3.1 Imagens

Uma imagem do Docker é um pacote de arquivo que contém todos os códigos, bibliotecas, ferramentas, dependências e outros arquivos necessários para executar um aplicativo (Sumo Logic, Inc, 2022). O Docker permite que qualquer usuário crie uma imagem personalizada utilizando um arquivo denominado `Dockerfile.yml` que descreve as etapas necessárias para construir o ambiente de um contêiner. Muitas empresas e desenvolvedores disponibilizam imagens prontas de serviços famosos como bancos de dados e servidores de internet em sites de repositórios públicos que servem tanto para protótipos quanto para ambientes de produção

2.3.2 Orquestração de contêineres

Como infraestruturas que utilizam o Docker são geralmente formadas pela junção de vários contêineres, existem ferramentas especializadas em geri-los conhecidas como orquestradores. O Docker Compose é um serviço que permite gerenciar múltiplos contêineres hospedados em um único computador. Para utilizar esta ferramenta é preciso escrever um arquivo chamado `Docker-compose.yml` no qual é definido a infraestrutura. Para isto são descritas configurações importantes para a execução de cada contêiner, por exemplo, qual porta deve ser exposta para um serviço, qual imagem deve ser utilizada, em que rede os contêineres devem se comunicar, em que local da máquina hospedeira devem ser salvos dados de forma persistente, etc.

2.4 Protocolos SSL/TLS

O protocolo *Secure Sockets Layer* (SSL) foi lançado em 1995 pela empresa Netscape com o intuito de garantir privacidade, autenticação e integridade aos dados transmitidos por meio da Internet. Cinco anos depois, em 1999, o *Internet Engineering Task Force* (IETF) liberou a especificação do protocolo *Transport Layer Security* (TLS) 1.0 (DIERKS; ALLEN, 1999), sucedendo o SSL. Apesar do TLS não diferir muito do seu antecessor, trazia melhorias significativas na segurança ao possibilitar a utilização de algoritmos de criptografia mais robustos. Essa solução popularizou-se e tornou-se um padrão da Internet, sendo possivelmente a forma mais amplamente implantada de criptografia desse meio de comunicação (RHODES; GOERZEN, 2014).

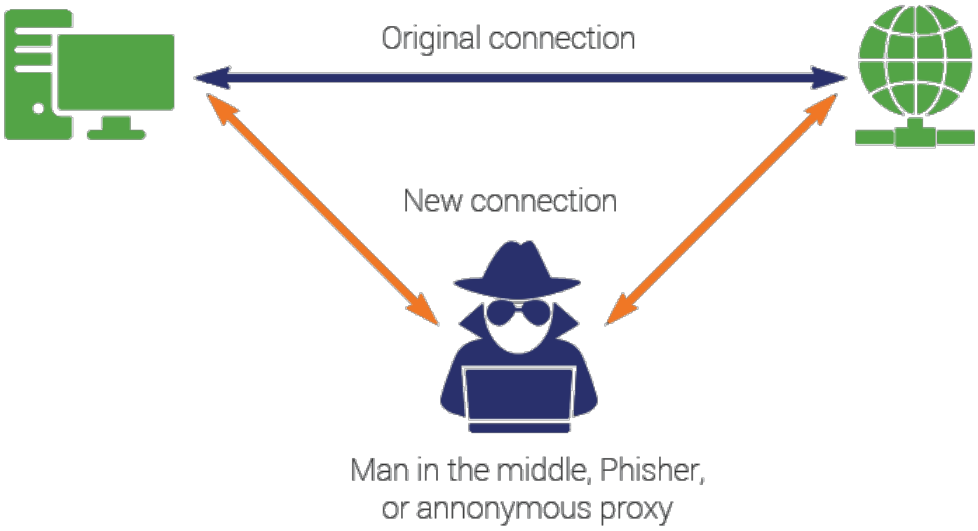
O TLS é uma abordagem prática para adicionar segurança a aplicações que utilizem protocolos de rede como o *Hypertext Transfer Protocol* (HTTP), que por padrão não apresenta nenhum mecanismo de defesa para ataques cibernéticos (CHANDRA MATT MESSIER, 2002). Por meio desta tecnologia, uma transmissão de dados que seria considerada insegura oferece

autenticação baseada em chave pública, estabelecimento de chave de sessão segura e confidencialidade de tráfego baseada em chave simétrica (DAS; SAMDARIA, 2014). Nota-se que a partir do momento que um dado é emitido com o protocolo HTTP por meio do TLS este passa a ser referido como *Hypertext Transfer Protocol Secure* (HTTPS).

Uma transação que envolva o TLS inicia pelo *handshake* entre o cliente e servidor. O *handshake* é a primeira troca de mensagens entre duas máquinas onde elas se reconhecem antes de estabelecer sua comunicação. Durante esse processo o servidor envia para o cliente um certificado que contém uma chave pública, o cliente então a utiliza para criptografar uma mensagem para o servidor. Quando o servidor recebe esta mensagem, ele é capaz de descriptografá-la com sua chave privada e enviá-la de volta ao cliente comprovando sua identidade. Junto do certificado, além da chave pública, existem mais informações que podem interessar o cliente, como o seu dono, a data de expiração e o nome do domínio ao qual este pertence.

Um possível ataque ao TLS, conhecido como *man-in-the-middle*, daria-se através da interceptação das mensagens trocadas entre cliente e servidor. A Figura 4 ilustra a execução do ataque *man-in-the-middle*. Neste tipo de ataque, um usuário mal-intencionado gera um certificado autoassinado que parece legítimo e estabelece uma conexão de *proxy* entre o cliente e o servidor, provendo ao atacante acesso aos dados enviados. Uma possível solução para este problema é permitir que o cliente não somente valide o certificado por meio do *handshake*, mas também analise sua credibilidade. Como seria impraticável que cada cliente tivesse uma lista de todos os certificados confiáveis da Internet, essa responsabilidade é repassada para terceiros intitulados autoridades certificadoras. Uma autoridade certificadora é responsável por conferir os dados presentes em um certificado, assiná-lo com sua chave privada e registrá-lo em um banco de dados.

Figura 4 – Ataque *man-in-the-middle*.



Fonte: Extraído de (Nohe, Patrick, 2021).

2.5 Sentilo

Sentilo (Barcelona City Council, 2022) é uma plataforma de código-fonte aberto, disponibilizada pela cidade espanhola de Barcelona, destinada à integração de sensores e atuadores no contexto de cidades inteligentes. Seu principal objetivo é servir como uma ponte entre os dados coletados por dispositivos de IoT instalados em cidades e aplicações que desejam consumi-los. Através dessa camada intermediária entre aplicações e *hardware*, é possível abstrair dos desenvolvedores a necessidade de gerenciar os dados gerados pela cidade, permitindo focar em como agregar valor a tal conjunto de dados. Adicionalmente, equipes de campo, responsáveis pelo *hardware*, podem usufruir da facilidade de conectar novos dispositivos à rede, valendo-se de ferramentas para catalogar, monitorar e administrar toda a infraestrutura.

2.6 Elasticsearch

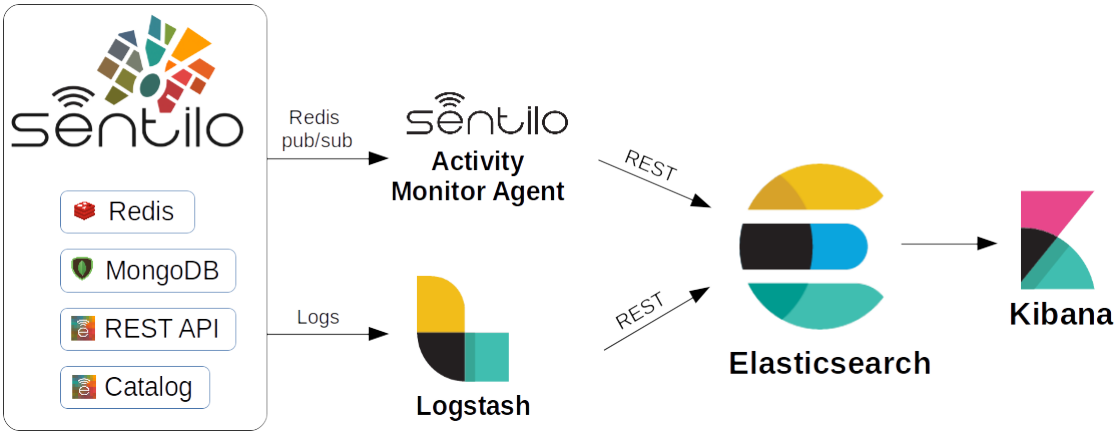
O Elasticsearch (Elasticsearch B.V., 2022) é um motor de busca desenvolvido em código aberto que permite realizar consultas complexas e agregação de dados em tempo real em ambientes com milhões de registros. Diferentemente de um sistema de banco de dados tradicional, no qual apenas um servidor atende todas as requisições, o Elasticsearch é um sistema distribuído. Essa característica lhe permite dividir as consultas em um ou mais nós (servidores) nos quais os dados são organizados utilizando a estrutura de dados de *index* invertido

disponibilizada pela biblioteca Apache Lucene resultando em consultas muito mais eficientes (ANDHAVARAPU, 2017).

A Sentilo primordialmente utiliza o banco de dados Redis (Redis Ltda., 2022) para persistência, sendo o armazenamento por padrão temporário. Uma das integrações automatizadas do sistema é o agente de monitoramento de atividade interagindo com o Elasticsearch. Como o Elasticsearch usufrui da capacidade de persistir os dados, pode ser uma solução razoável utilizá-lo. Mas vale ressaltar que devido à natureza distribuída dessa ferramenta, existe o risco de alguns dados serem perdidos durante o processo de anexação, caso alguns dos servidores sofra uma interrupção inesperada. Isso pode ser atenuado com a criação de réplicas do banco, mas pode aumentar o custo total de operação.

A Figura 5 ilustra a integração proporcionada pela Sentilo com o Elasticsearch, o Logstash (*pipeline* de processamento de dados de *logs*) e a Kibana (ambiente gráfico para análise de dados do Elasticsearch).

Figura 5 – Integração com Elasticsearch e Kibana.



Fonte: Disponível em (Barcelona City Council, 2022).

2.7 OpenTSDB

O OpenTSDB (The OpenTSDB, 2022) é um banco de dados projetado para armazenar séries temporais. Uma série temporal é uma coleção de observações atreladas ao tempo, bastante aderante a características de dados coletados por sensores. Para exemplificar uma série temporal, pode-se imaginar um microprocessador programado para realizar aferições da temperatura ambiental a cada 1 minuto.

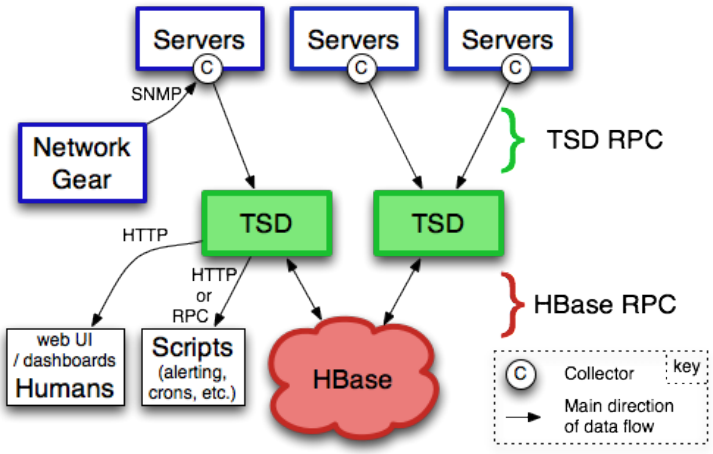
No caso do OpenTSDB, antes do envio os dados a serem armazenados devem ser estruturados da seguinte forma:

- nome de métrica;
- *UNIX timestamp* (contagem de segundos desde 1º de janeiro de 1970 no UTC);

- Os dados, que podem ser um valor (inteiro de 64 bits ou valor de ponto flutuante de precisão simples) ou um evento formatado em *JavaScript Object Notation* (JSON);
- Um conjunto de *tags* (pares de chave-valor) que descrevem a série temporal à qual o ponto de dado pertence.

O Hbase (The Apache Software Foundation, 2022b) é o banco de dados com o qual o OpenTSDB interage para fazer gravações. Implementado em Java, ele foi projetado para trabalhar com grandes conjuntos de dados, desta forma as operações de leitura e escrita são extremamente rápidas. Para permitir a interação com o banco do Hbase, o OpenTSDB disponibiliza para os usuários uma *Application Programming Interface* (API) *Representational State Transfer* (REST) e interface gráfica simples. Na Figura 6 podemos visualizar a arquitetura do sistema onde *Time Series Daemon* (TSD) é o serviço que disponibiliza a interação com o banco de dados.

Figura 6 – Arquitetura do OpenTSDB



Fonte: Disponível em (The OpenTSDB, 2022).

A fim de otimizar o sistema de busca é possível utilizar a biblioteca de Java Hadoop, que possibilita instalar o OpenTSDB em máquinas distribuídas que oferecem seus próprios recursos de computação e armazenamento. Uma das qualidades que tornam esse sistema robusto a erros é que em vez de depender de *hardware* para fornecer alta disponibilidade, a própria biblioteca foi projetada para detectar e lidar com falhas na camada de aplicação (The Apache Software Foundation, 2022a). A Sentilo oferece integração com o OpenTSDB através do agente de histórico. Para uma melhor visualização dos dados salvos no banco de dados é possível integrá-lo a Grafana (Grafana Labs, 2022) que serve como um *front-end* intuitivo para este serviço.

3 MATERIAIS E MÉTODOS

Neste capítulo são apresentados os materiais e métodos necessários elaborar uma infraestrutura para cidades inteligentes por meio da virtualização de contêineres.

3.1 Materiais

Esta seção dedica-se a apresentar as principais tecnologias empregadas para o desenvolvimento do projeto, tais como plataforma de virtualização de contêineres, sistema de versionamento de código e bibliotecas para geração de certificados.

3.1.1 Plataforma de contêineres

Para lidar com a questão relacionada a conteinerização, optou-se pela utilização do Docker (Docker Inc., 2022) para tal tarefa, embora existam outras alternativas tais como Buildah, Podman e LXD. O motivo que balizou a escolha do Docker foi sua grande comunidade de usuários, extensa documentação disponível e dominância nesse segmento do mercado. Para ilustrar, na pesquisa realizada pelo *website* "Stack Overflow" (Stack Overflow, 2022), participaram mais 59 mil usuários, sendo que Docker consta em primeiro lugar como tecnologia de virtualização de contêineres mais utilizada por desenvolvedores.

3.1.2 Sistema de versionamento de código

O GitHub é uma plataforma de hospedagem na Internet para projetos de *software*. Esse serviço permite que os usuários consigam rastrear todas as alterações feitas no código ao longo do processo de desenvolvimento. Para atingir tal feito esta plataforma utiliza o Git, um sistema de versionamento de código. O desenvolvedor que use a ferramenta terá acesso não somente a versão atual dos arquivos, mas também a um histórico de tudo que já foi escrito (BELL; BEER, 2015).

Todos os arquivos gerados no âmbito deste trabalho foram disponibilizados em um repositório público GitHub. O principal objetivo é manter um registro das alterações realizadas nos códigos escritos para a infraestrutura da Sentilo, bem como disponibilizar para o projeto da UTFPR-TD e eventualmente outros indivíduos ou organizações que desejem implementar um sistema de cidades inteligentes semelhante.

3.1.3 Shell Script

O *Shell script* aproveita-se dos comandos oferecidos pelo *kernel* Linux para roteirizar comandos que precisam ser executados, tornando-se uma ferramenta essencial para administradores de sistemas que precisam automatizar tarefas (JARGAS, 2008). Assim, *shell scripts* foram amplamente utilizados no decorrer deste do projeto, tanto para elaborar configurações nos servidores, particularmente quando estas precisavam ser executadas após a inicialização de um contêiner, quanto na criação automatizada dos certificados autoassinados para o ambiente de desenvolvimento.

3.1.4 Biblioteca OpenSSL

A biblioteca do OpenSSL pode ser dividida em duas ferramentas: uma biblioteca de criptografia e uma para implementar o protocolo SSL em sítios de Internet. Ela provém para seus usuários uma implementação de todas as versões de protocolos de segurança TLS, os algoritmos mais populares de criptografia de chave simétrica, assimétrica e *hash* e ainda simplifica a criação e validação de certificados digitais (CHANDRA; MESSIER; VIEGA, 2002). Por conta disto, foi utilizada no projeto dos servidores da Sentilo com a finalidade de gerar os certificados autoassinados para implementar o protocolo TLS no ambiente de desenvolvimento.

3.1.5 Nginx

Atualmente o Nginx é considerado um dos principais servidores de Internet do mundo e oferece aos seus usuários diversos recursos como: servidor estático HTTP, balanceamento de carga e *proxy reverso* (NGINX, 2022). Neste trabalho o Nginx foi utilizado como *proxy reverso* para assegurar que os servidores da Sentilo troquem mensagens criptografadas entre cliente e servidor a partir do protocolo SSL. Isso foi imprescindível, devido ao fato que à versão 1.9 da Sentilo implementada neste trabalho, mesmo sendo a mais recente disponibilizada, não possui esse recurso nativamente.

3.1.6 NodeJS

O NodeJS é uma biblioteca que permite que aplicações JavaScript, linguagem originalmente criada para desenvolver aplicações para serem interpretadas apenas no navegador (lado do cliente) sejam executadas no lado do servidor. Neste contexto, o NodeJS consegue gerar um servidor multiplataforma de simples configuração, compatível com sistemas operacionais como MS-Windows, GNU/Linux e MacOS (BROW, 2020).

Neste trabalho o NodeJS simula um sensor injetando requisições contínuas para a Sentilo. Um *script* é executado para chamar em intervalos predefinidos de tempo uma rota de sua API que armazena os dados dos sensores. Assim é possível verificar se os dados estão sendo retransmitidos para os servidores de persistência.

3.2 Métodos

Esta seção demonstra como as tecnologias citadas na seção 3.1 foram empregadas para realização do trabalho. A seção inicia explicando o processo de virtualização dos serviços que estão presentes na infraestrutura proposta de cidades inteligentes. Em seguida são mostradas as etapas necessárias para ativar a persistência de dados na Sentilo e por fim o processo para realizar o teste de injeção de dados sintéticos por meio da API da plataforma.

3.2.1 Infraestrutura básica

Com o intuito de alcançar as metas propostas neste trabalho, o primeiro passo foi trabalhar nos contêineres responsáveis pelo núcleo da Sentilo. Para gerar uma instância funcional desta plataforma precisamos de um servidor TomCat configurado para servir o *front-end* e a *API REST*, um banco de dados Redis para armazenar temporariamente os dados gerados pelos sensores e atuadores e um servidor com o banco de dados MongoDB destinado a guardar dados de configurações da plataforma.

Um arranjo arquitetural possível para essa infraestrutura seria agrupar todos os servidores citados previamente em um único contêiner, de forma semelhante ao que ocorre na máquina virtual disponibilizada nativamente pela Sentilo (Barcelona City Council, 2022). Todavia, neste trabalho optou-se pela construção isolada de cada contêiner, visando ampliar a escalabilidade horizontal do sistema. Ao tratá-los desta maneira, além de ser possível distribuir com maior precisão a quantidade de recursos disponíveis da máquina hospedeira, também permite-se que os serviços sobrecarregados sejam replicados para balanceamento de carga em novas máquinas interligadas via rede.

3.2.2 Contêiner da Sentilo

O primeiro arquivo feito para suprir a infraestrutura da Sentilo foi o `Dockerfile` responsável por customizar uma imagem do servidor TomCat (TomCat, 2022). Todo arquivo do tipo `Dockerfile` deve ser iniciado por uma imagem que contenha uma base sólida para o desenvolvimento do servidor, nesse sentido ao escolher uma imagem deve-se prezar por uma que contenha a maioria dos recursos necessários para o funcionamento correto do contêiner. Ante o exposto, a imagem escolhida para servir de alicerce nessa customização do servidor foi

`tomcat:9.0.58-jdk8-temurin-focal`, uma vez que ela cumpre com dois requisitos cruciais para a hospedagem da Sentilo sendo eles: uma versão funcional do servidor TomCat e o kit de desenvolvimento em Java versão 1.8.

Logo após a escolha da imagem, foram instaladas dependências extras: (a) Git, (b) Maven, ferramenta de automação de compilação em Java, e (c) Supervisor, Sistema de controle de processos. Em virtude da distribuição Linux da imagem ser baseada em Debian para instalação de tais dependências foi utilizado o gerenciador de pacotes Apt. A sequência de etapas pode ser vista na Listagem 1.

Listagem 1 – Preparação da imagem da Sentilo

```
1  FROM tomcat:9.0.58-jdk8-temurin-focal as prod
2  LABEL maintainer="luismariotti1"
3  USER root
4
5  RUN mkdir /sentilo
6  WORKDIR /sentilo
7
8  # install packages
9  RUN apt-get clean && \
10     apt-get update && \
11     apt-get install -y --no-install-recommends \
12     git \
13     maven \
14     supervisor \
15     vim
```

Fonte: Autoria própria (2023).

A próxima etapa foi baixar do repositório público, disponibilizado pela Sentilo no Github (REPO, 2022), o código-fonte da plataforma. Em seguida, antes de compilar o código-fonte, foram realizadas algumas alterações em arquivos de configuração que serviram para garantir a comunicação entre os contêineres. Isso foi necessário, pois o Docker compose, utilizado para administrar múltiplos contêineres simultaneamente, gera uma rede interna na qual cada contêiner recebe um endereço de *Internet Protocol* (IP) único e a Sentilo por padrão tenta comunicar-se no endereço de *loopback* (`127.0.0.1`), o que impossibilitaria sua comunicação.

Nesse contexto, a solução do problema foi utilizar o utilitário *sed* do Linux com expressões regulares de parâmetros a fim de substituir os endereços de *loopback* pelos respectivos apelidos gerados pelo Docker para o Redis e MongoDB que se transcrevem para o endereço de IP da rede interna. Finalizada esta etapa a plataforma é compilada utilizando a ferramenta Maven. A Listagem 2 mostra os comandos necessários para baixar o código do repositório do GitHub e editar as configurações de comunicação dos contêineres e compilá-lo.

Listagem 2 – Preparação para compilar a Sentilo

```
# source code
RUN git clone https://github.com/sentilo/sentilo.git .

# change permission for execute script
RUN chmod +x -R scripts

# configure mongo for docker network
RUN sed -i "s/.*catalog.mongodb.host=.*/catalog.mongodb.host=mongo-server/"
/sentilo/sentilo-agent-alert/src/main/resources
/properties/catalog-config.properties

RUN sed -i "s/.*catalog.mongodb.host=.*/catalog.mongodb.host=mongo-server/"
/sentilo/sentilo-catalog-web/src/main/resources
/properties/catalog-config.properties

# configure redis for docker network
RUN sed -i "s/.*jedis.pool.host=.*/jedis.pool.host=redis-server/"
sentilo-platform/sentilo-platform-service/src/main/resources
/properties/jedis-config.properties

RUN sed -i "s/.*jedis.pool.host=.*/jedis.pool.host=redis-server/"
sentilo-agent-alert/src/main/resources
/properties/jedis-config.properties

RUN sed -i "s/.*jedis.pool.host=.*/jedis.pool.host=redis-server/"
sentilo-agent-relational/src/main/resources
/properties/jedis-config.properties

RUN sed -i "s/.*jedis.pool.host=.*/jedis.pool.host=redis-server/"
sentilo-agent-location-updater/src/main/resources
/properties/jedis-config.properties

# prepare compile script to run non interactively
RUN sed -i "s/^read$/#read/" ./scripts/buildSentilo.sh

# compile sentilo
RUN ./scripts/buildSentilo.sh
```

Fonte: Autoria própria (2023).

3.2.3 Contêiner dos bancos de dados MongoDB e Redis

Finalizando o processo de configuração inicial da infraestrutura básica da Sentilo, a próxima etapa consiste em gerar os contêineres para os bancos de dados Redis e MongoDB. Para o banco de dados Redis a documentação da Sentilo exige a versão 4.0.11. Como já existe uma imagem oficial que confere este requisito nenhuma alteração precisou ser feita. Já para o MongoDB, além do requisito da versão 4.0.1, algumas configurações extras tiveram que ser feitas, já que para o correto funcionamento da plataforma alguns dados precisam ser inseridos antes de inicializá-la. Os dados necessários para popular o banco de dados são primeiramente copiados do repositório da plataforma e depois inseridos por intermédio de um *shell script*. A Listagem 3 mostra o `Dockerfile` responsável por gerar o servidor do banco de dados MongoDB.

Listagem 3 – Dockerfile MongoDB

```
FROM mongo:4.0.1
USER root

RUN apt-get clean && \
    apt-get update && \
    apt-get install -y --no-install-recommends \
    git

# source code
RUN git clone https://github.com/sentilo/sentilo.git

COPY create-sentilo-user.js /

COPY docker-entrypoint.sh /docker-entrypoint-initdb.d/

RUN chmod +x create-sentilo-user.js

RUN chmod +x /docker-entrypoint-initdb.d/docker-entrypoint.sh

CMD [ "mongod" ]

EXPOSE 27017
```

Fonte: Autoria própria (2023).

Com esses servidores prontos já é possível criar o arquivo `docker-compose.yml`, que deve gerenciar múltiplos contêineres. Neste arquivos são descritos definições fundamentais para os servidores. Algumas das configurações utilizadas para a Sentilo e os bancos de dados são:

- `Build`: define qual `Dockerfile` deve ser usado para construir uma imagem;
- `Image`: serve para utilizar uma imagem que já exista em um repositório público;
- `Container_name`: gera um apelido para o contêiner que pode ser utilizado na *CLI* do Docker;
- `Ports`: mapeia uma porta da máquina hospedeira, para uma porta exposta pelo contêiner;
- `Depends_on`: define se um contêiner deve aguardar a inicialização de outro;
- `Volumes`: mapeia caminhos da máquina hospedeira onde os contêineres podem utilizar para guardar dados de forma persistente;
- `Command`: executam comandos depois que o contêiner inicializar;

A Listagem 4 apresenta o código escrito no arquivo `docker-compose.yml` o qual define a inicialização dos servidores que compõem o núcleo da aplicação da Sentilo.

Listagem 4 – Docker-compose

```
version: "3.7"
services:
  sentilo-server:
    #  image: luismariottti1/sentilo
    build: sentilo
    container_name: sentilo_servers
    depends_on:
      - "mongo-server"
      - "redis-server"
    # volumes:
      # - sentilo_plataform:/sentilo

  mongo-server:
    image: luismariotti1/sentilo-mongo
    #  build: mongo
    container_name: sentilo_mongo
    ports:
      - 27017:27017
    volumes:
      - mongo_data:/data/db
      - mongo_config:/data/configdb

  redis-server:
    image: redis:4.0.11
    container_name: sentilo_redis
    command: redis-server --requirepass sentilo
    ports:
      - 6379:6379
    volumes:
      - redis_data:/data
```

Fonte: Autoria própria (2023).

A partir deste ponto já existe uma versão funcional do servidor, contudo ela ainda não deveria ser usada em um ambiente de produção visto que o protocolo SSL não está configurado. Sem este protocolo, as mensagens transmitidas e recebidas pelo servidor não são criptografadas, tornando o sistema vulnerável a usuários mal-intencionados que podem tentar interceptar a comunicação entre cliente e servidor para usurpar informações confidenciais como a senha de outros usuários.

Devemos prezar pelo princípio da confidencialidade e ativar o protocolo HTTPS no sistema. Cabe ressaltar que até o momento de escrita deste trabalho, **a plataforma Sentilo não provê um mecanismo para lidar com requisições criptografadas**. Apesar disto, é possível adicionar este recurso através de um servidor funcionando como *proxy reverso*.

3.2.4 Contêiner para proxy reverso

Os certificados emitidos para o protocolo SSL deste trabalho foram autoassinados. É importante frisar que esse tipo de certificado só deve ser usado em ambientes de testes. Caso se opte pela utilização da solução oficialmente para o projeto EnvCity, é imprescindível substituí-los por certificados assinados por autoridades certificadoras. Posto isso, fez-se um *shell script* que gera os certificados automaticamente a partir de comandos fornecidos pela biblioteca OpenSSL. A fim de facilitar a troca por certificados oficiais, os caminhos utilizados pelos contêineres para

encontrá-los foram salvos em variáveis de ambiente. O código responsável pela criação deste certificado pode ser visto na Listagem 5.

Listagem 5 – Script para criação do certificado

```
#!/bin/sh

# remove old certs
rm ./ca/*.pem
rm ./server/*.pem
rm ./ca/*.srl

# Generate a self-signed certificate for the proxy
openssl req -nodes -x509 -newkey rsa:4096 -sha256 -days 365 -keyout
./ca/ca-key.pem -out ./ca/ca-cert.pem -subj
"/C=BR/ST=Parana/L=Toledo/O=Envcity/OU=IoT/
CN=*.envcity.com/emailAddress=envcity@gmail.com"

# Generate a certificate signing request for the proxy
openssl req -nodes -newkey rsa:4096 -sha256
-keyout ./server/server-key.pem -out ./server/server-req.pem -subj
"/C=BR/ST=Parana/L=Toledo/O=Sentilo/OU=Server/
CN=localhost.com/emailAddress=sentilo@gmail.com"

# Sign the proxy's certificate with the CA's certificate
openssl x509 -req -sha256 -in ./server/server-req.pem -CA
./ca/ca-cert.pem -CAkey ./ca/ca-key.pem
-CAcreateserial -out ./server/server-cert.pem
```

Fonte: Autoria própria (2023).

Após a emissão dos certificados foi preciso gerar um arquivo de configuração para o Nginx de forma que ele possa retransmitir os pacotes da Sentilo em um meio seguro. Este arquivo ficou dividido em duas partes, sendo a primeira responsável por ouvir requisições e reescrevê-las para uma *Uniform Resource Locator* (URL) iniciada em HTTPS. Já a segunda parte lida com o protocolo SSL. Inicia-se pela configuração da porta do servidor, neste caso 443. Logo em seguida são definidos os caminhos para o certificado e a chave privada, junto a cabeçalhos importantes para as requisições. Por fim são definidas as rotas que redirecionam os pacotes. Como o servidor da Sentilo hospeda simultaneamente o *front-end* e a API existem dois possíveis destinos para requisições, a porta 8080 e a porta 8081. No intuito de diferenciá-las foi utilizado o prefixo *sentilo-catalog-web* que está presente em todas as requisições de páginas do *front-end*. A Listagem 6 contém o arquivo de configuração do Nginx.

Listagem 6 – Configuração Nginx proxy reverso

```
server {
    listen 80;
    listen 8081;

    rewrite ^ https://$host$request_uri? permanent;
}

server {

    listen 443 ssl;

    rewrite_log on;
    access_log /var/log/nginx/sentilo-api-access.log;
    error_log  /var/log/nginx/sentilo-api-error.log info;

    underscores_in_headers on;

    ssl_certificate /etc/nginx/certs/server/server-cert.pem;
    ssl_certificate_key /etc/nginx/certs/server/server-key.pem;

    ssl_session_timeout 1m;

    ssl_protocols SSLv3 TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers "HIGH:!aNULL:!MD5 or HIGH:!aNULL:!MD5:!3DES";
    ssl_prefer_server_ciphers on;


    proxy_set_header        Host $host;
    proxy_set_header        X-Real-IP $remote_addr;
    proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header        X-Forwarded-Proto $scheme;

    location /sentilo-catalog-web/ {
        proxy_pass "http://sentilo-server:8080";
        proxy_redirect http:// https://;
    }

    location / {
        proxy_pass "http://sentilo-server:8081";
        proxy_redirect http:// https://;
    }
}
```

Fonte: Autoria própria (2023).

3.2.5 Contêiner do Elasticsearch

Ainda existe o problema de persistência de dados. A resolução desta adversidade é ativar as integrações da plataforma com o OpenTSDB e Elasticsearch que correspondem respectivamente aos agentes de histórico e de monitoramento de atividades.

Dentre os dois, o primeiro servidor adicionado a infraestrutura foi o do Elasticsearch. Para tanto, foi preciso modificar o arquivo `Docker-compose.yml` com as configurações de inicialização do novo contêiner. A maioria dos comandos utilizados se assemelham aos explicados anteriormente na Listagem 4. Como a imagem oficial do Elasticsearch necessita aumentar os limites de memória e de tamanho de arquivos para valores acima do padrão estabelecido pelo Docker, novos comandos foram utilizados. Para remover a limitação de memória e tama-

nho de arquivo foi usado o comando `unlimite`. Um problema em utilizar o comando anterior é que o contêiner pode acabar esgotando os recursos da máquina hospedeira, por esse motivo foi preciso utilizar o comando `resources` que estabelece um valor máximo para a alocação de recursos pelo contêiner. A Listagem 7 apresenta as primeiras configurações do Elasticsearch.

Listagem 7 – Configurações iniciais Elasticsearch

```
es01:
    container_name: es01
    image: elasticsearch:7.17.5
    ulimits:
      memlock:
        soft: -1
        hard: -1
      nofile:
        soft: 65536
        hard: 65536
    deploy:
      resources:
        limits:
          cpus: "0.50"  # Use at most 50% of one CPU core
          memory: 2000M  # Use at most 2 GB of RAM
    volumes:
      - "es01_data:/usr/share/elasticsearch/data"
      - "./certs:$CERTS_DIR"
    ports:
      - '9200:9200'
```

Fonte: Autoria própria (2023).

Diferentemente da Sentilo, o Elasticsearch possui suporte nativo ao protocolo SSL. Para ativá-lo no contêiner basta adicionar algumas variáveis de ambiente para o pacote de segurança Xpack. Em um arquivo `Docker-compose.yml` o comando `enviroment` é responsável por gerenciar automaticamente a inserção de variáveis de ambiente. A última configuração definida foi o *healthcheck* que verifica a integridade da instalação. A Listagem 8 é responsável por adicionar esses dois recursos citados a instalação do Elasticsearch.

Listagem 8 – Variáveis de ambiente e *healthcheck* do Elasticsearch

```
    environment:
      - node.name=es01
      - discovery.seed_hosts=es01
      - cluster.initial_master_nodes=es01
      - ELASTIC_PASSWORD=$ELASTIC_PASSWORD
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
      - xpack.security.enabled=true

      - xpack.security.http.ssl.enabled=true
      - xpack.security.http.ssl.certificate_authorities=
      $CERTS_DIR/ca/ca-cert.pem
      - xpack.security.http.ssl.certificate=$CERTS_DIR/server/server-cert.pem
      - xpack.security.http.ssl.key=$CERTS_DIR/server/server-key.pem

      - xpack.security.transport.ssl.enabled=true
      - xpack.security.transport.ssl.verification_mode=certificate
      - xpack.security.transport.ssl.certificate_authorities=
      $CERTS_DIR/ca/ca-cert.pem
      - xpack.security.transport.ssl.certificate=
      $CERTS_DIR/server/server-cert.pem
      - xpack.security.transport.ssl.key=$CERTS_DIR/server/server-key.pem
    healthcheck:
      test: curl --cacert $CERTS_DIR/ca/ca.crt -s
      https://localhost:9200 >/dev/null; if [[ $$? == 52 ]];
      then echo 0; else echo 1; fi
      interval: 30s
      timeout: 10s
      retries: 5
```

Fonte: Autoria própria (2023).

3.2.6 Contêiner do OpenTSDB

O próximo servidor instalado foi o OpenTSDB no qual foi necessário construir uma imagem personalizada para seguir as demandas de versões descritas na documentação da Sentilo. Perante ao exposto, o primeiro passo da criação deste arquivo foi escolher uma imagem oficial para ser editada. A imagem escolhida foi o `openjdk:8-jdk`, isto porque este ambiente já vem preparado para trabalhar com o Java e utiliza a versão 8 do JDK que é um requisito para a instalação do banco HBASE e do OpenTSDB. Em seguida são instaladas outras dependências necessárias utilizando o Apt sendo elas: o Supervisor, o wget (ferramenta que propicia o download de dados da Web) e o gnuplot (programa para plotar gráficos de funções matemáticas).

Finalizado esse processo de instalação utiliza-se o Wget para baixar as versões corretas do HBASE e OpenTSDB que são respectivamente a 1.2.1 e 2.3.0. Depois disso são instalados ambos os programas por meio do comando dpkg do Linux. Também foi alterada uma configuração requisitada pela Sentilo para gerar métricas automaticamente no OpenTSDB. Por fim se inicializa o servidor com o programa supervisor. Este processo pode ser vista de forma detalhada em código na Listagem 9.

Listagem 9 – Imagem customizada do banco OpenTSDB

```
FROM openjdk:8-jdk
USER root

RUN apt-get clean && \
    apt-get update && \
    apt-get install -y --no-install-recommends \
    supervisor \
    wget \
    gnuplot

ENV HBASE_VERSION 1.2.1
ENV OPENTSDB_VERSION 2.3.0

RUN wget http://archive.apache.org/dist/hbase/${HBASE_VERSION}/
hbase-${HBASE_VERSION}-bin.tar.gz

RUN tar xzvf hbase-${HBASE_VERSION}-bin.tar.gz

RUN rm -r hbase-${HBASE_VERSION}-bin.tar.gz

RUN wget https://github.com/OpenTSDB/opentsdb/releases/download/
v${OPENTSDB_VERSION}/opentsdb-${OPENTSDB_VERSION}_all.deb

RUN dpkg -i opentsdb-${OPENTSDB_VERSION}_all.deb

ENV COMPRESSION=NONE

ENV HBASE_HOME=/hbase-${HBASE_VERSION}

# configure for auto generate metrics1212
RUN sed -i "s/.*#tsd.core.auto_create_metrics =
false.*/tsd.core.auto_create_metrics = true/"
/etc/opentsdb/opentsdb.conf

# start.sh
COPY start.sh /

RUN chmod +x start.sh

# supervisor conf
COPY supervisord.conf /etc/supervisor/conf.d/supervisord.conf


CMD [ "/usr/bin/supervisord" ]

EXPOSE 4242
```

Fonte: Autoria própria (2023).

Por conseguinte, deve-se adicioná-lo também ao `Docker-compose.yml` para que o Docker possa gerenciá-lo em sua rede interna. Antes de prosseguir para o *script* que prepara a plataforma da Sentilo para transmitir os dados ao Elasticsearch e OpenTSDB é preciso enfatizar um problema de segurança deste último. Até o momento de escrita do artigo, o OpenTSDB não implementa um sistema de autenticação para sua API e, segundo sua documentação, não está apto para uso em produção (The OpenTSDB, 2022). Uma forma de solucionar este problema seria criar uma aplicação que autentique o usuário antes de dar acesso a informações registradas no banco de dados, mas está além do escopo deste trabalho.

3.2.7 Ativação dos agentes

Antes de ativar os agentes da Sentilo foi preciso fazer algumas alterações no arquivo da Listagem 2 que geram o servidor principal da plataforma. Nestes arquivos foram adicionados mais alguns comandos objetivando alterar as URLs de conexão do sistema com os contêineres de persistência, onde estas foram trocadas por seus apelidos gerados para os IPs da rede interna do Docker. As alterações realizadas podem ser vistas na Listagem 10.

Listagem 10 – Alterações na imagem Sentilo

```
# configure opentsdb for docker network
RUN sed -i "s/.*catalog.mongodb.host=.*/catalog.mongodb.host=mongo-server/"
sentilo-agent-historian/src/main/resources/properties/
catalog-config.properties
RUN sed -i "s/.*jedis.pool.host=.*/jedis.pool.host=redis-server/"
sentilo-agent-historian/src/main/resources/properties/
jedis-config.properties
RUN sed -i "s/opentsdb.url=http:\/\/127.0.0.1:4242/opentsdb.url=
http:\/\/opentsdb-server:4242/g"
sentilo-agent-historian/src/main/resources/properties/
historian-config.properties

#configure elasticsearch for docker network
RUN sed -i "s/.*catalog.mongodb.host=.*/catalog.mongodb.host=mongo-server/"
sentilo-agent-activity-monitor/src/main/resources/properties
/catalog-config.properties
RUN sed -i "s/.*jedis.pool.host=.*/jedis.pool.host=redis-server/"
sentilo-agent-activity-monitor/src/main/resources/properties/
jedis-config.properties
RUN sed -i "s/elasticsearch.url=http:\/\/localhost:9200/elasticsearch.url=
http:\/\/agent-proxy:9000/g" sentilo-agent-activity-monitor/
src/main/resources/properties/monitor-config.properties
```

Fonte: Autoria própria (2023).

Os agentes da Sentilo não são compilados com a plataforma pois são considerados módulos extras que pode ser adicionados conforme a necessidade dos usuários. Nesse caso, implementou-se um *shell script* adicional para ser executado após a inicialização do contêiner, compilando automaticamente ambos os agentes de persistência utilizados neste projeto. O *script* que automatiza o processo de instalação dos agentes pode ser visto na Listagem 11.

Listagem 11 – Script para compilar agentes

```
#!/bin/sh

# save agent path in variable
agent_path=/sentilo/sentilo-agent-activity-monitor
agent_server_path=$agent_path/target/appassembler/bin

echo installing elasticsearch ...

# check if appassembler folder exists
if [ ! -d $agent_path"/target/appassembler/" ]; then
    cd $agent_path
    mvn clean install; mvn package appassembler:assemble -P dev
fi

cd $agent_server_path

# check if the sentilo-agent-activity-monitor-server fille has +x permission
if [ ! -x sentilo-agent-activity-monitor-server ]; then
    chmod +x sentilo-agent-activity-monitor-server
fi

agent_path=/sentilo/sentilo-agent-historian
agent_server_path=$agent_path/target/appassembler/bin

echo installing opentsdb ...

# check if appassembler folder exists
if [ ! -d $agent_path"/target/appassembler/" ]; then
    cd $agent_path
    mvn clean install; mvn package appassembler:assemble -P dev
fi

cd $agent_server_path

# check if the sentilo-agent-activity-monitor-server fille has +x permission
if [ ! -x sentilo-agent-historian ]; then
    chmod +x sentilo-agent-historian-server
fi
```

Fonte: Autoria própria (2023).

3.2.8 Teste automatizado de inserção de dados

A fim de testar o comportamento global da plataforma foi elaborado um teste NodeJS, a ser executado em segundo plano no servidor, fazendo requisições periódicas para inserir dados sintéticos. Tais dados simulam o comportamento de um dispositivo IoT equipado com um sensor de temperatura. O código utiliza duas bibliotecas, o Axios (para fazer requisições HTTP) e o Https (Módulo NodeJS adicional que permite trabalhar com o protocolo SSL).

O primeiro passo do código é configurar o SSL para aceitar requisições de certificados autoassinados por meio de uma classe de agente. Na sequência, o Axios é configurado para utilizar o agente do Https e adicionar no cabeçalho da requisição o *token* de identidade. Em seguida é construída a função que repete as requisições a cada 2 segundos para o servidor Sentilo. Nesta função é gerado um valor aleatório para a temperatura e adicionada a URL da API. O código final para o teste é apresentado na Listagem 12.

Listagem 12 – Código JavaScript para teste da API Sentilo

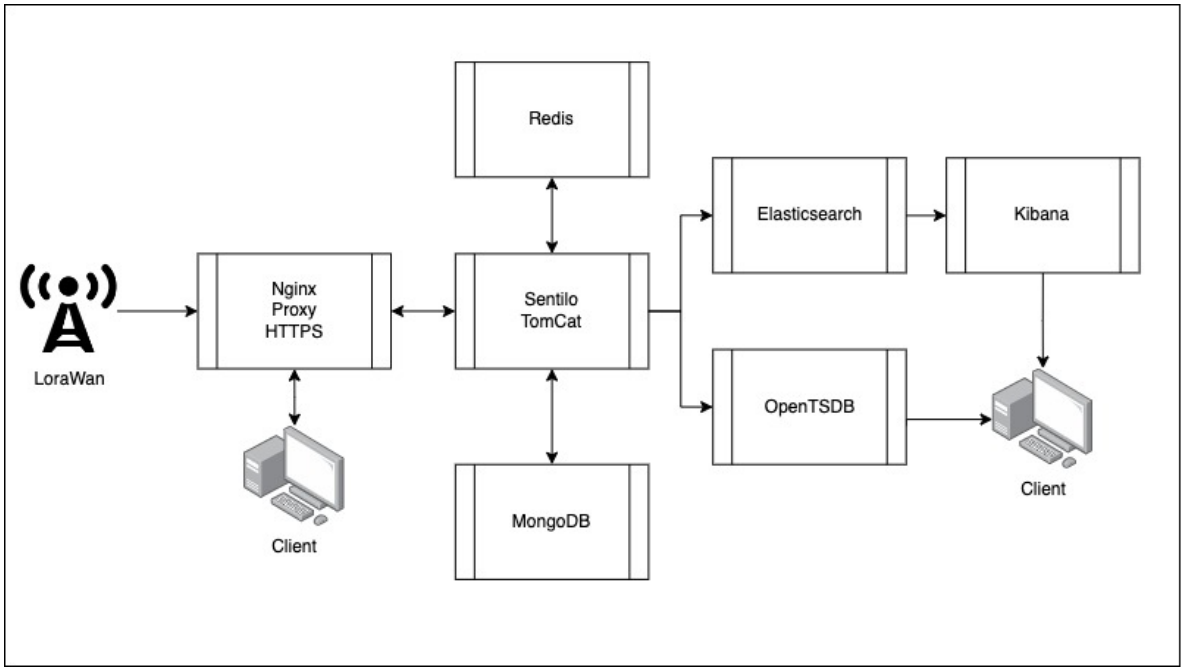
```
const axios = require('axios');
const https = require('https');


// configure https agent to ignore self-signed certificate
const httpsAgent = new https.Agent({
  rejectUnauthorized: false
});

// configure axios
const axiosInstance = axios.create({
  httpsAgent,
  headers: {'IDENTITY_KEY':
  '563093ec5252147edc8860c2d667be5db0c010325b6953ed5b323724bcc00e05'}
});

// every 2 seconds make a request to the api
setInterval(() => {
  temp = randomInt(10, 40);
  axiosInstance.put('https://localhost/data/testApp_provider/temp/'+temp)
    .then(() => {
      console.log(temp);
    })
    .catch((err) => {
      console.log(err);
    });
}, 2000);

randomInt = (min, max) => {
  return Math.floor(Math.random() * (max - min + 1)) + min;
}
```

Fonte: Autoria própria (2023).

4 RESULTADOS

Este capitulo apresenta os resultados obtidos a partir dos experimentos realizados com a solução de infraestrutura apresentada no capítulo anterior. A Figura 7 mostra um diagrama da situação final dos servidores. Para esse fim, a infraestrutura foi instalada em um servidor do projeto EnvCity totalmente novo. Essa instalação ficou operacional durante o período de 7 semanas consecutivas, nas quais um servidor NodeJS fez requisições contínuas com dados gerados sinteticamente no intuito de avaliar o comportamento da plataforma Sentilo e a persistência dos dados nos bancos de dados OpenTSDB e Elasticsearch.

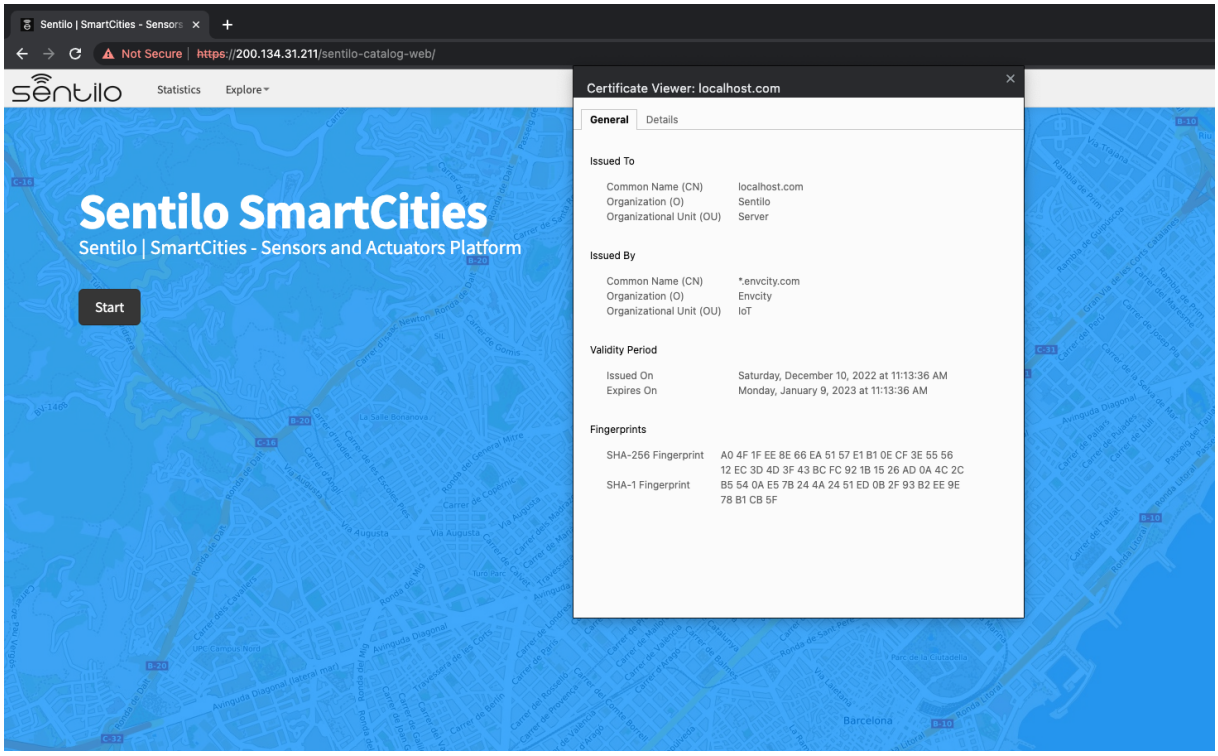
Figura 7 – Diagrama da infraestrutura de servidores



Fonte: Autoria própria.

O primeiro resultado positivo foi que a instalação ocorreu com sucesso, sendo possível acessá-la fora da rede interna da UTFPR-TD, a partir do endereço público de IPv4 da máquina. Por ser um servidor novo, a universidade ainda não tinha expedido um certificado para ele junto a uma autoridade certificadora. Por conta disto, a instalação utilizou um certificado autoassinado garantindo que os dados transmitidos estão sendo criptografados. Porém, em produção tal certificado deverá ser substituído. Além disso, o certificado autoassinado fazia com que os navegadores considerassem o *site* Web inseguro, exibindo um aviso de risco de ataque ao acessar o *site* pela primeira vez. Mesmo com esse bloqueio parcial da plataforma realizado pelo navegador era possível prosseguir com o acesso se o usuário concordasse com potenciais riscos. A Figura 8 mostra a plataforma Sentilo sendo acessada pelo endereço IP público do servidor. Na mesma figura são exibidas as informações referentes ao certificado autoassinado que o servidor disponibiliza ao navegador.

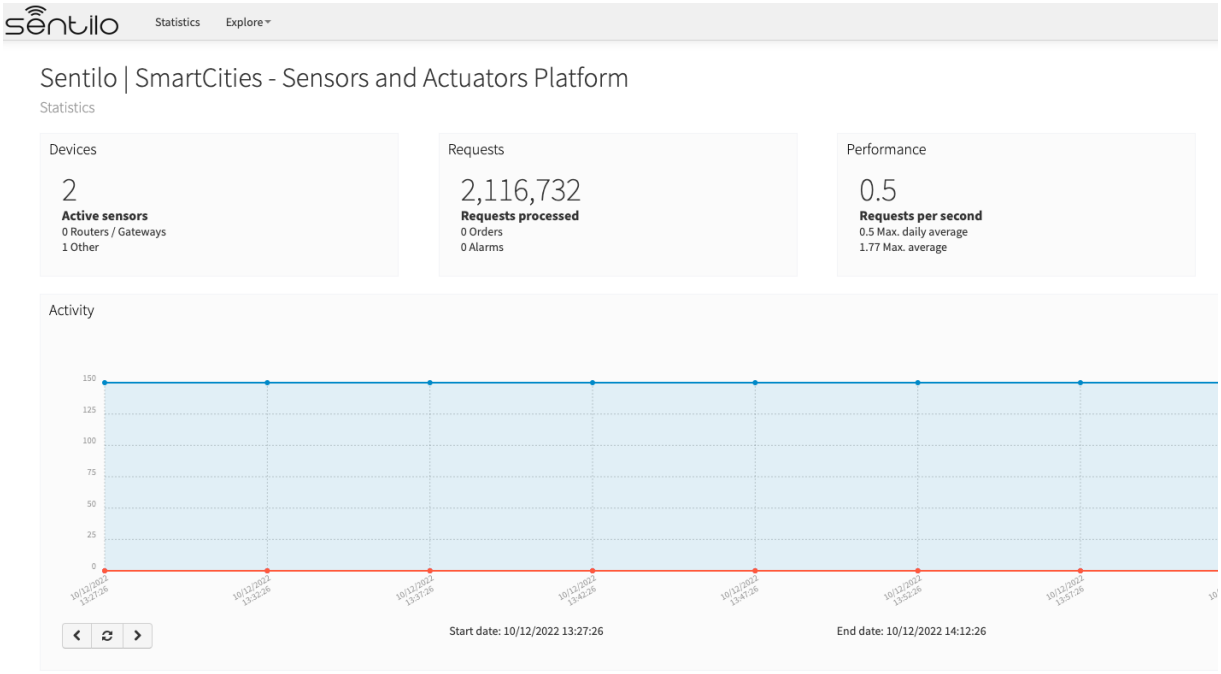
Figura 8 – Plataforma Sentilo hospedada no servidor do EnvCity com HTTPS



Fonte: autoria própria.

Durante o período de testes, o servidor dedicado a fazer requisições realizou mais de 2 milhões de inserções que simulavam o comportamento de um sensor de temperatura de dados na Sentilo. Nesse código os valores eram gerados randomicamente e enviados a plataforma utilizando sua API. O servidor conseguiu lidar com todos os dados enviados, indicando que a infraestrutura com contêineres estava funcionando corretamente. O número total de requisições feitas pode ser observado na Figura 9.

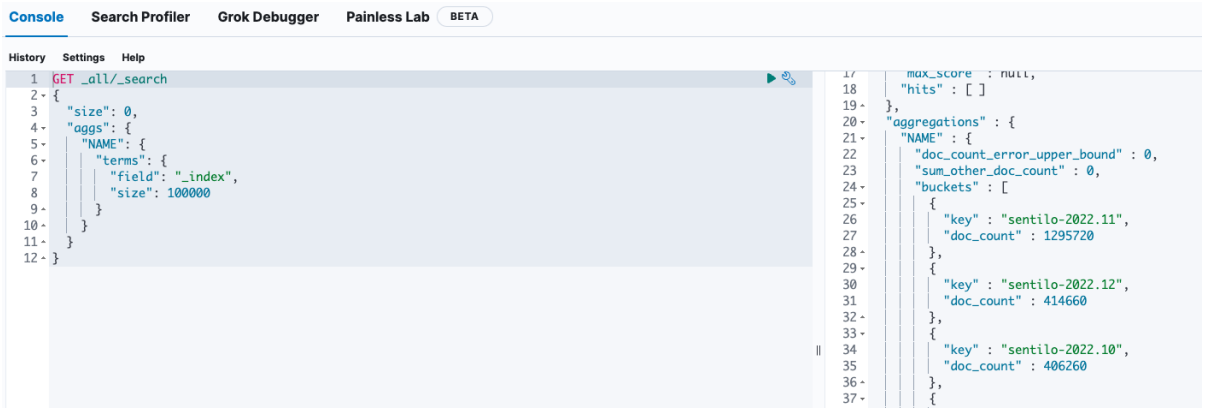
Figura 9 – Dados enviados a Sentilo



Fonte: autoria própria.

O agente de monitoramento de atividade também estava operando durante a realização destes testes. Esse agente é um módulo configurável da Sentilo para retransmitir os dados para o Elasticsearch. Como o agente teve êxito na comunicação entre os dois contêineres, os dados que antes eram voláteis, visto que eram salvos no banco de dados Redis, foram persistidos. No final das 7 semanas utilizou-se a Kibana para contar o número de dados inseridos nos índices do Elasticsearch por meio da sua API. Todos os valores tinham sidos salvos com sucesso e o resultado desta pesquisa pode ser verificado na Figura 10, mostrando como resposta os índices criados automaticamente pela Sentilo junto ao número de registros. A soma resulta no total de dados transmitidos no período de testes.

Figura 10 – Dados persistidos no Elasticsearch



Fonte: autoria própria.

Outro agente configurado foi o agente de histórico. Esse agente é mais uma forma da Sentilo possibilitar que os usuários salvem os dados enviados à plataforma. A principal limitação dessa abordagem é que o banco de dados OpenTSDB utilizado para persistir as informações não possui um sistema de autenticação. Por conta desta limitação, que pode comprometer a segurança dos dados, este banco não foi testado diretamente no servidor do EnvCity. Contudo, ainda foram realizados testes em ambientes controlados utilizando a mesma metodologia de testes para o agente de monitoramento no intuito de avaliar se a infraestrutura conseguiria transmitir os dados utilizando este módulo. Findados os testes os resultados mostraram-se promissores, uma vez que o agente transmitiu de forma adequada os dados sintéticos. O resultado desse experimento pode ser visto na Figura 11. É mostrado um gráfico produzido pela interface gráfica do OpenTSDB, exibindo os dados enviados a API da Sentilo ao longo do tempo.

Figura 11 – Dados persistidos no OpenTSDB

Fonte: autoria própria.

5 CONSIDERAÇÕES FINAIS

O conceito de cidades inteligentes é de suma importância para a sociedade moderna, visto que o rápido desenvolvimento da tecnologia de informação permite trazer comodidades e qualidade de vida para moradores de regiões urbanas. Neste sentido, a tecnologia da IoT destaca-se por permitir o desenvolvimento de diversas aplicações nos mais variados segmentos de uma cidade.Tais aplicações devem não somente ajudar os moradores, tornando suas vidas mais práticas, mas também aos gestores públicos que terão dados para subsidiar o processo decisório.

O município de Toledo-PR está ciente da revolução de cidades inteligentes e buscou apoiar o o projeto EnvCity da UTFPR-TD. Nesse grande projeto estão sendo desenvolvidas iniciativas ambientais e logísticas utilizando a IoT. Como redes de IoT comumente contam com muitos dispositivos conectados que geram quantidades massivas de dados, a infraestrutura deste projeto deve estar preparada para lidar com situações de estresse.

Neste cenário, o presente trabalho contribuiu na implementação de uma infraestrutura focada em escalabilidade, desempenho e manutenibilidade para o projeto EnvCity. Para esse fim, a abordagem convencional baseada em máquinas virtuais foi substituída pela tecnologia de virtualização de contêineres.

A plataforma de virtualização de contêineres utilizada foi o Docker. Nele foram definidos os principais servidores do projeto:

- infraestrutura de cidades inteligentes – uma imagem customizada Sentilo;
- persistência de dados – bancos de dados Elasticsearch e OpenTSDB, facilmente integráveis à Sentilo.

Os servidores virtualizados foram preparados também para trabalhar com o protocolo SSL, incrementando a segurança dos dados potencialmente sensíveis.

Para testar se a solução apresentada estava funcionando devidamente, *scripts* JavaScript foram desenvolvidos para realizar requisições periódicas para a API da Sentilo durante 7 semanas. Os *scripts* de teste juntamente aos servidores containerizados em Docker foram instalados em uma máquina utilizada no projeto EnvCity. Ao término dos testes constatou-se que a solução comportava-se conforme o esperado e os dados estavam sendo persistidos adequadamente.

No que diz respeito a trabalhos futuros, é esperado a realização de testes de desempenho nas soluções de persistências aqui apresentadas. Adicionalmente é necessário definir uma solução para autenticar os usuários que terão acesso ao banco de dados OpenTSDB para viabilizar sua utilização respeitando os critérios de segurança. Além disso, o código pode receber otimizações e novos recursos para melhorar a infraestrutura de cidades inteligentes do projeto EnvCity.

REFERÊNCIAS

4LINUX. *Containers e VMs: Diferenças e usos*. 2022. Disponível em: <https://4linux.com.br/diferenca-containers-maquinas-virtuais/>. Acesso em: 1 jun. 2022.

ANAGNOSTOPOULOS, T. *et al.* Challenges and opportunities of waste management in iot-enabled smart cities: A survey. **IEEE Transactions on Sustainable Computing**, v. 2, n. 3, p. 275–289, 2017.

ANDHAVARAPU, A. **Learning Elasticsearch : distributed real-time search and analytics with Elasticsearch 5.x**. [*S.l.*: *s.n.*], 2017. 404 p. ISBN 978-1787128453.

ARROUB, A. *et al.* A literature review on smart cities: Paradigms, opportunities and open problems. In: **2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)**. [*S.l.*: *s.n.*], 2016. p. 180–186.

AVATEFIPOUR, O.; SADRY, F. Traffic management system using iot technology - a comparative review. In: **2018 IEEE International Conference on Electro/Information Technology (EIT)**. [*S.l.*: *s.n.*], 2018. p. 1041–1047.

Barcelona City Council. **Sentilo**. 2022. Disponível em: https://www.sentilo.io/wordpress/. Acesso em: 18 mai. 2022.

BARMAN, B. K. *et al.* Iot based smart energy meter for efficient energy utilization in smart grid. In: **2018 2nd International Conference on Power, Energy and Environment: Towards Smart Technology (ICEPE)**. [*S.l.*: *s.n.*], 2018. p. 1–5.

BAUMAN, E.; AYOADE, G.; LIN, Z. A survey on hypervisor-based monitoring: Approaches, applications, and evolutions. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 48, n. 1, aug 2015. ISSN 0360-0300. Disponível em: https://doi-org.ez48.periodicos.capes.gov.br/10.1145/2775111.

BELL, P.; BEER, B. **Introdução ao GitHub**. São Paulo: Novatec, 2015.

BROW, E. **Programação web com Node e Express**. São Paulo: Novatec, 2020.

CHANDRA MATT MESSIER, J. V. P. **Network Security with OpenSSL**. [*S.l.*]: O'Reilly, 2002. ISBN 0-596-00270-X.

CHANDRA, P.; MESSIER, M.; VIEGA, J. **Segurança da internet com OpenSSL**. [*S.l.*]: O'Reilly, 2002.

Cisco. **Cisco Annual Internet Report (2018–2023)**. 2020. Disponível em: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. Acesso em: 20 nov. 2022.

DAS, M. L.; SAMDARIA, N. On the security of ssl/tls-enabled applications. **Applied Computing and Informatics**, v. 10, n. 1, p. 68–81, 2014. ISSN 2210-8327. Disponível em: https://www.sciencedirect.com/science/article/pii/S2210832714000039.

DIERKS, T.; ALLEN, C. **Rfc2246: The TLS protocol version 1.0**. [*S.l.*]: RFC Editor, 1999.

Docker Inc. **Use containers to Build, Share and Run your applications**. 2022. Disponível em: https://www.docker.com/resources/what-container/. Acesso em: 10 nov. 2022.

Elasticsearch B.V. **Elastic Enterprise Search**. 2022. Disponível em: https://www.elastic.co/. Acesso em: 10 nov. 2022.

Grafana Labs. **Grafana: The open observability platform**. 2022. Disponível em: https://grafana.com/. Acesso em: 1 jun. 2022.

ISO. **ISO 37122:2019, Sustainable cities and communities — Indicators for smart cities**. 2019. Disponível em: https://www.iso.org/obp/ui/#iso:std:iso:37122:ed-1:v1:en.

JARGAS, A. M. **Shell Script Profissional**. São Paulo: Novatec, 2008.

JOY, A. M. Performance comparison between linux containers and virtual machines. In: **2015 International Conference on Advances in Computer Engineering and Applications**. [*S.l.*: *s.n.*], 2015. p. 342–346.

KAUR, K. A survey on internet of things - architecture, applications, and future trends. **ICSCCC 2018 - 1st International Conference on Secure Cyber Computing and Communications**, Institute of Electrical and Electronics Engineers Inc., p. 581–583, 7 2018.

NANDA, S.; CHIUEH, t.-c. A survey on virtualization technologies. 01 2005.

NGINX. **What is NGINX?** 2022. Disponível em: https://www.nginx.com/resources/glossary/nginx/. Acesso em: 18 mai. 2022.

Nohe, Patrick. **Executing a Man-in-the-Middle Attack in just 15 Minutes**. 2021. Disponível em: https://www.wallarm.com/what/what-is-mitm-man-in-the-middle-attack. Acesso em: 10 nov. 2022.

POULTON, N. **Docker Deep Dive: Zero to Docker in a single book**. [*S.l.*]: Nigel Poulton, 2018.

RAJAB, H.; CINKELR, T. Iot based smart cities. In: **2018 International Symposium on Networks, Computers and Communications (ISNCC)**. [*S.l.*: *s.n.*], 2018. p. 1–4.

RAMASAMY, L. K.; KADRY, S. Internet of things (iot). In: **Blockchain in the Industrial Internet of Things**. IOP Publishing, 2021, (2053-2563). p. 1–1 to 1–16. ISBN 978-0-7503-3663-5. Disponível em: https://dx.doi.org/10.1088/978-0-7503-3663-5ch1.

Redis Ltda. **Introduction to Redis**. 2022. Disponível em: https://redis.io/docs/about/. Acesso em: 18 mai. 2022.

REPO. 2022. Disponível em: https://github.com/sentilo/sentilo. Acesso em: 10 nov. 2022.

RHODES, B.; GOERZEN, J. Tls/ssl. In: ____. **Foundations of Python Network Programming: Third Edition**. Berkeley, CA: Apress, 2014. p. 93–114. ISBN 978-1-4302-5855-1. Disponível em: https://doi.org/10.1007/978-1-4302-5855-1_6.

SANTANA, E. F. Z. *et al.* Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 50, n. 6, nov 2017. ISSN 0360-0300. Disponível em: https://doi.org/10.1145/3124391.

Stack Overflow. **2022 Developer Survey**. 2022. Disponível em: https://survey.stackoverflow.co/2022/#most-loved-dreaded-and-wanted-tools-tech-love-dread. Acesso em: 10 nov. 2022.

Sumo Logic, Inc. **Docker Swarm - definition overview**. 2022. Disponível em: https://www.sumologic.com/glossary/docker-swarm. Acesso em: 16 nov. 2022.

SYED, A. S. *et al.* Iot in smart cities: A survey of technologies, practices and challenges. **Smart Cities**, v. 4, n. 2, p. 429–475, 2021. ISSN 2624-6511. Disponível em: https://www.mdpi.com/2624-6511/4/2/24.

The Apache Software Foundation. **Apache Hadoop**. 2022. Disponível em: https://hadoop.apache.org/. Acesso em: 1 jun. 2022.

The Apache Software Foundation. **Welcome to Apache HBase™**. 2022. Disponível em: https://hbase.apache.org/. Acesso em: 10 nov. 2022.

The OpenTSDB. **OpenTSDB overview**. 2022. Disponível em: http://opentsdb.net/overview.html. Acesso em: 1 jun. 2022.

TIBCO. **What is the Internet of Things (IoT)?** 2022. Disponível em: https://www.tibco.com/reference-center/what-is-the-internet-of-things-iot. Acesso em: 4 jun. 2022.

TomCat. **Apache TomCat**. 2022. Disponível em: https://tomcat.apache.org/. Acesso em: 18 mai. 2022.

TURNBULL, J. **The Docker Book: Containerization is the new virtualization**. [*S.l.*]: James Turnbull, 2014.

VITALINO, J. F. N.; CASTRO, M. A. N. PDF. **Descomplicando o Docker**. [*S.l.*: *s.n.*], 2018.