UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

David Pierre Alves

IoT orientado a *Assets*: uma ferramenta para *assetização* de internet das coisas

Recife

2024

David Pierre Alves

IoT orientado a *Assets*: uma ferramenta para *assetização* de internet das coisas

Monografia apresentada ao Curso de Bacharelado em Ciências da Computação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Ciências da Computação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Computação

Curso de Bacharelado em Ciências da Computação

Orientador: Vanilson Burégio

Recife

2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

A474i Alves, David Pierre
IoT orientado a Assets: uma ferramenta para assetização de internet das coisas / David Pierre Alves. - 2024.
39 f. : il.

Orientador: Vanilson Buregio.
Inclui referências e apêndice(s).

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Ciência da Computação, Recife, 2024.

1. Internet das Coisas. 2. Assets. 3. Serviços. 4. Gerenciamento. 5. Framework. I. Buregio, Vanilson, orient. II. Título

CDD 004

MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
(UFRPE) BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

http://www.bcc.ufrpe.br

Ficha de Aprovação do Trabalho de Conclusão de Curso

Trabalho defendido por David Pierre Alves às 16h20min do dia 07 de março de 2024, no link https://meet.google.com/iga-xdkd-msd, como requisito para conclusão do curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, intitulado IoT orientado a Assets: uma ferramenta para assetização de internet das coisas, orientado por Vanilson Burégio e aprovado pela seguinte banca examinadora:

Vanilson Burégio
DC/UFRPE

Kellyton Brito
DC/UFRPE

Dedico este trabalho à todos os que me ajudaram ao longo desta caminhada.

Agradecimentos

Agradeço a meus pais, Teresa e Josué, meus irmãos, Victor e Vanessa e a meu sobrinho, Matheus, por todo apoio, carinho e dedicação, servindo de suporte, pilar e abrigo nos momentos mais difíceis e por todo companheirismo, alegria e orgulho nos momentos de felicidade.

À Marina que esteve ao meu lado durante todos esses anos, me encorajando e me dando o apoio necessário e incondicional. Pela compreensão a minha ausência durante a execução deste trabalho.

Aos professores pelas correções e ensinamentos que me permitiram apresentar um melhor desempenho no meu processo de formação profissional ao longo do curso. Em especial ao professor Vanilson Burégio, por ter sido meu orientador.

Aos meus colegas de curso e profissão, com quem convivi intensamente durante os últimos anos, pelo companheirismo nos momentos difíceis, pela ajuda mútua, pelos momentos de alegria e pela troca de experiências que me permitiram crescer não só como pessoa, mas também como formando.

“O mar da história é agitado. As ameaças e as guerras havemos de atravessá-las. Rompê-las ao meio, cortando-as, como uma quilha corta as ondas.”

(Vladimir Maiakóvski)

Resumo

Esse trabalho discute e apresenta um sistema para demonstrar a viabilidade técnica de servir *coisas* (como em Internet das Coisas) como *assets*. Enquanto as iniciativas existentes focam na discussão de coisas como serviços como parte do processo de *servitização*, não há iniciativas que focam na discussão de coisas como *assets* como parte do processo de *assetização*. A *assetização* permite modelar coisas de uma perspectiva gerencial, em termos de depreciação ao longo do tempo, transferibilidade entre localização, o descarte depois do uso, e a convertibilidade entre plataformas. Graças à *assetização*, coisas podem prover benefícios econômicos, informacionais, operacionais e regulatórios para seus donos (morais ou judiciais).

Palavras-chave: Assets, IoT, Internet das Coisas, serviços, gerenciamento, framework.

Abstract

This paper discusses and demonstrates the technical doability of serving *things* (as in Internet of Things) as assets. While existing initiatives focus on the conversion of things into services as part of the servitization process, there are not initiatives that focus on thing conversion into assets as part of the assetization process. Assetization permits to model things from a management perspective in terms of depreciation over time, transferability across locations, disposability after use, and convertibility across platforms. Thanks to assetization, things would provide economical, informational, operational, and regulatory benefits to their owners (whether moral or juridical).

Keywords: Assets, IoT, Internet of things, Services, Management, Framework.

Lista de ilustrações

Lista de algoritmos

Lista de abreviaturas e siglas

IoT	Internet of Things
WoT-TD	Web of Things - Thing Description
TDD	Tests Driven Development

Sumário

1 Introdução

Ao decorrer da evolução da internet, que pode ser dividida em diferentes fases(CISCO IBSG, 2006), foram exploradas e adotadas diversas tecnologias diferentes. Dentre essas tecnologias houve a ascensão da Internet das Coisas (*Internet of Things - IoT*), definido por Tarkoma e Katasonov(TARKOMA; KATASONOV, 2011) como uma "rede global e serviço de infraestrutura de densidade e conectividade variáveis com capacidades de auto-configuração baseada em padrões e protocolos e formatos interoperáveis.". De acordo com o grupo DZONE, estima-se que o impacto causado pela internet das coisas alcançará entre $3.9 trilhões e $11.1 trilhões de dólares por ano até 2025 (DZone, 2017), sendo adotado nos mais diversos cenários, desde a vida cotidiana ao setor industrial ou rural.

Outro paradigma que surgiu nos últimos anos foi a disponibilização de recursos através de uma arquitetura de serviços(SOA), definida pela organização OASIS(EVANS, 2012) como "um paradigma para organizar e utilizar capacidades distribuídas que podem estar sob o controle de diferentes domínios.". É possível ver esse paradigma sendo adotado nos mais diversos tipos de aplicações e áreas, inclusive dentro da computação, como: infraestrutura (IaaS), plataforma (PaaS), e *software* como um serviço (SaaS), dentre outros exemplos.

Eventualmente, a adoção de modelos de arquitetura para o uso de *IoT* como um serviço também surgiu, como pode ser visto na proposta feita por Maamar, Faci e Yahya (MAAMAR; FACI; YAHYA, 2021) através de meios para a disponibilização de acesso e dados a dispositivos *IoT* através de uma arquitetura voltada a serviços. E, apesar da arquitetura de serviços comportar bem as necessidades da internet das coisas, principalmente na perspectiva de interação entre requisição e resposta, essa arquitetura deixa a desejar no aspecto gerencial, quando levamos em conta alguns fatores como: custo, depreciação, transferibilidade, disponibilidade, entre outros fatores muitas vezes negligenciados.(MAAMAR et al., 2024) Uma forma de sanar esse problema e conseguir gerir essas informações é através da disponibilização de coisas (como em internet das coisas) como *assets*.

Assets são definidos pela *International Financial Reporting Standards Foundation* (IFRS) como "um recurso controlado por uma entidade como resultado de eventos passados e do qual os benefícios econômicos futuros são esperados fluir para a entidade."(IFRS, 2015). Também podem ser vistas definições de *assets* em outras áreas, como em cibersegurança, é definido por Ross et al. (2021) como "um item de valor para *stakeholders*. Um *asset* pode ser tangível ou intangível. O valor de um *asset* é

determinado como *stakeholders* considerando o desgaste em todo o ciclo de vida do sistema.”. Ao usar *assetização* é possível que uma coisa provenha ao seu dono seu valor agregado, levando em consideração custo operacional, desgaste, custos regulatórios.

Levando em consideração os pontos descritos acima, esse projeto propõe a criação e desenvolvimento de uma ferramenta para a *assetização* no contexto de IoT, com a finalidade de prover dados de propósito mais gerencial, permitindo o monitoramento do valor associado a uma Coisa ao longo de todo seu ciclo de vida, com foco na depreciação do dispositivo. Este trabalho trata do projeto e da proposta de desenvolvimento de uma ferramenta baseada no padrão REST, juntamente com uma porção web, implementando desde sua interface web até a implantação dessa ferramenta na nuvem. A aplicação desenvolvida tem o objetivo de demonstrar a viabilidade de adoção de um paradigma orientado à *assets* voltado ao contexto de internet das coisas.

2 Fundamentação teórica

Este capítulo apresenta conceitos básicos essenciais para o entendimento e desenvolvimento desse trabalho. Na seção 2.1 são abordadas as definições de *assets*, assim como suas principais características. Na seção 2.2 estão as definições de depreciação, característica principal e focal no desenvolvimento deste projeto, encontradas na literatura, técnicas utilizadas e onde esse conceito é utilizado.

2.1 *Assets*

O conceito de *asset* difere dependendo do contexto em que se encontra, sendo principalmente visto e definido em comunidades como financeira, gerenciamento de direitos digitais e cibersegurança, por exemplo.
Segundo a *International Financial Reporting Standards* (IFRS) "um *asset* é recurso controlado por uma entidade como resultado de eventos passados e do qual os benefícios econômicos futuros são esperados fluir para a entidade." (IFRS, 2015)

Já na cibersegurança, pode ser definido como "um item de valor para *stakeholders*. Um *asset* pode ser tangível ou intangível. O valor de um *asset* é determinado por *stakeholders* considerando o desgaste em todo o ciclo de vida do sistema."(ROSS et al., 2021) Ainda no contexto de cibersegurança, podemos ver *assets* sendo adotados a fim de isolar riscos de sistemas *IoT*. (KUZMINYKH; CARLSSON, 2018) Reconhecendo as características e requisitos únicos de um sistema de Internet das coisas, o autor aborda um modelo de ameaças com o intuito de identificar os *stakeholders* e *assets* relacionados à segurança, possíveis ataques e ameaças para o sistema *IoT* em questão.

Na comunidade de direitos digitais, o objetivo é garantir a utilização e proteção adequadas de *assets* de mídias digitais e arquivos na era das redes sociais e compartilhamento de arquivos em massa. Uma especificação da *World Wide Web Consortium* (W3C) utilizada nesse tipo de contexto, é a *Open Digital Rights Language (ODRL)*, afirmando que "algo é permitido, proibido, ou obrigado, possivelmente limitado por algumas restrições" (W3C, 2018), além de prover um modelo de informação flexível e interoperável, vocabulário, mecanismos de codificação para representar informações acerca do uso de *assets*. Um *asset* é um recurso identificável ou uma coleção de recursos como dados e informações, conteúdo de mídia, aplicações e serviços. Já Barber (2020) define um *asset* digital como "Uma solução de sistemas e *softwares* que provêm uma abordagem sistemática para, eficientemente, armazenar, organizar, gerenciar, recuperar e distribuir os *assets* digitais de uma empresa." (BARBER, 2020)

2.1.1 Características de um *Asset*

Com a *assetização* de coisas, há uma certa interseção com alguns princípios de *assets* no contexto de gerenciamento, como nível de serviço, ciclo de vida, e o custo total de posse, a fim de lidar com alguns aspectos relacionados ao gerenciamento de itens IoT:

- Depreciação é relacionado ao valor (de venda) ou a performance de um *asset* a longo prazo. No contexto de internet das coisas vê-se a depreciação como um fator importante a ser considerado, principalmente quando uma coisa é utilizada em ambientes externos e exposto a fatores que agravam o seu desgaste e podem afetar seu funcionamento, por exemplo.

- Transferibilidade é relacionada à mudança de posse de um *asset* de uma perspectiva legal. No contexto de internet das coisas vê-se a transferibilidade como um ponto de preocupação quando coisas são movidas ou mudam de uma localização para outra, forçando-as a obedecer novas regulações de privacidade de dados, por exemplo. Sendo, neste contexto, diferente da visão mostrada por Gunnarsson e Gerhmann, que focam em requisitos de segurança durante a troca de posse. (GUNNARSSON.; GEHRMANN., 2020) Um novo dono de uma coisa não deve deduzir nada que um proprietário anterior fez antes da transferência, e *vice versa*. Gunnarsson e Gerhmann também citam que um sistema IoT não deve estar sobre posse de dois proprietários ao mesmo tempo.

- Descartabilidade é relacionada à disponibilidade de *assets*. No contexto de internet das coisas vê-se a descartabilidade como um ponto importante quando coisas não podem ser usados após uma data ou um número limite de utilizações, o que poderia resultar na substituição desse item. Para aumentar a vida útil de uma coisa, medidas como manutenções regulares devem ser planejadas.

- Convertibilidade está relacionada à monetização de *assets*. Por exemplo, quão fácil é converter um *asset* em dinheiro. No contexto de internet das coisas a convertibilidade é um ponto focal quando coisas não podem ser ajustadas ou adaptadas por causa de mudanças em seus ambientes. Em Maamar et al. (2021) é discutido três argumentos a favor da mutação de coisas: performance, para que os dispositivos possam permanecer competitivos; *awareness*, para que os dispositivos possam conhecer seu contextos e arredores; e sobrevivência, a fim de que os dispositivos continuem ativos.

2.2 Depreciação

Depreciação é uma das principais características de um *asset*, e foi a característica destacada durante a execução deste trabalho e está relacionada ao valor (de venda) ou a performance de um *asset* a longo prazo. De acordo com a Internal Revenue Service (2021) (IRS), há muitas técnicas, utilizadas por diversos países e em diferentes contextos, para calcular o desgaste de valor de um *asset* no decorrer do tempo, como a depreciação linear, acelerada, acumulada, e gerencial. Por exemplo, no governo do Canadá é utilizada a técnica de *capital-cost-allowance*, enquanto o governo do Estados Unidos utiliza técnicas de depreciação linear e saldo decrescente. Apesar de haver uma grande diversidade de técnicas, elas possuem algumas propriedades em comum (QUICSOLV, 2022): custo original (o valor de aquisição de um *asset*, incluindo impostos, fretagem, treinamento, etc.), vida útil (quantidade de anos esperados que um *asset* possa ser utilizado), e número de unidades produzidas antes que o *asset* esteja desgastado.

2.3 Trabalhos relacionados

Inicialmente, buscou-se realizar uma verificação na literatura com a adoção do modelo de *assets* em diversas áreas do conhecimento, bem como suas diferentes definições e aplicações, como por exemplo na área financeira (IFRS, 2015) e na cibersegurança (ROSS et al., 2021). Além disso, no que tange a utilização de diversos paradigmas aplicados à internet das coisas, procurou-se fazer um leitura aprofundada de trabalhos, artigos e capítulos por Zakaria Maamar e Amel Benna, que são precursores sobre o tema. A citação desses trabalhos serviu como base para fomentar a contextualização da adoção de um modelo de *assets* para dispositivos de internet das coisas.

Ademais, buscou-se realizar uma pesquisa de diferentes propostas para o desenvolvimento no contexto de IoT. Para isso, foi consultado um trabalho que possui uma abordagem semelhante, porém com ênfase na arquitetura baseada em serviços, proposta por Maamar, Faci e Yahya (2021), que propõe e discute “os passos e mecanismos necessários para a *servitização* da Internet das Coisas”, e que apresenta como um trabalho futuro o desenvolvimento de um *framework* genérico e integrado para a disponibilização de *coisas* como um serviço.

O debate sobre a adoção de *assets* como base primária tecnocientífica é discutida de forma multidisciplinar por diversos estudiosos, como pode ser visto no livro *“Assetization: Turning Things into Assets in Technoscientific Capitalism”* (BIRCH; MUNIESA, 2020), no qual é discutido, em diversas áreas do saber, do conhecimento à biomedicina, como transformar diversos ativos em *assets* e quais as implicações futu-

ras dos estudos da *assetização*.

Posteriormente, no que tange alguns aspectos mais característicos de *assets*, como definições, fórmulas e diferentes aplicações do conceito de depreciação, checamos fontes de sites organizacionais e institucionais como a IRS, do governo estadunidense. (Internal Revenue Service, 2021; QUICSOLV, 2022)

Diante do exposto, pode-se inferir que apesar da utilização de *assets* em comunidades como a financeira ou de cibersegurança e a existência do debate da *assetização* de diversos outros contextos, a adoção de um modelo de *assetização*, em especial para internet das coisas, é um nicho ainda não muito explorado. Além disso, existem estudos na área sobre outros modelos adotados em IoT, como por exemplo, a adoção de uma arquitetura orientada a serviços. Isso abre um espaço para o estudo da adoção e implicação de uma arquitetura voltada a *assets* em um meio tão crescente quanto a internet das coisas.

3 Metodologia

Nesta seção será apresentada a metodologia utilizada neste trabalho, a qual se dá a partir de uma pesquisa sobre como possibilitar o uso e desenvolvimento para internet das coisas através de paradigma orientado a *assets*, para isso a etapa de revisão de literatura foi realizada para fundamentar este estudo.

Em seguida, é relatado o processo de desenvolvimento da ferramenta proposta, mencionando as linguagens, *frameworks*, base de dados e ferramentas utilizadas, além da arquitetura da solução proposta.

3.1 Desenvolvimento da ferramenta

O presente trabalho tem como artefato tecnológico uma ferramenta para integrar-se no desenvolvimento para dispositivos de internet das coisas, desenvolvido pelo autor desta monografia, contemplando aspectos como interface, regras de negócios, *Application Programming Interface* (API), e documentação.

Para a construção da ferramenta, foi adotada uma metodologia de desenvolvimento orientado à testes (TDD), que segundo Calais e Franzini (2023), é uma técnica de desenvolvimento que combina *design* e testes de uma maneira iterativa e incremental. A técnica de TDD promove um forma estruturada de projetar soluções através de um ciclo de *feedbacks*, ao explicitar o resultado esperado, antes de, de fato, implementar um trecho de código. Para uma visualização mais abrangente do desenvolvimento do produto, a Figura 1 demonstra o processo evolutivo da ferramenta.

Na etapa de modelagem, foi utilizada, como base, as recomendações descritas no *Web of Things - Things Description* (WoT-TD) (W3C, 2023), a fim de entender o que é uma coisa e quais seus atributos. Um exemplo do modelo proposto pelo WoT-TD pode ser visto na Figura 2, onde uma Coisa, representando uma coisa concreta, é um classe central que suporta três tipos de interações com usuários finais. Todas as classes associadas a esses tipos de interação são derivadas da classe *InteractionAffordance* e corresponde, respectivamente, à *PropertyAffordance* que permite capturar e controlar parâmetros, *ActionAffordance* que se refere às operações de Coisas, e *EventAffordance* que permite comunicações de forma assíncrona, como notificações, eventos discretos e *streaming* de dados.

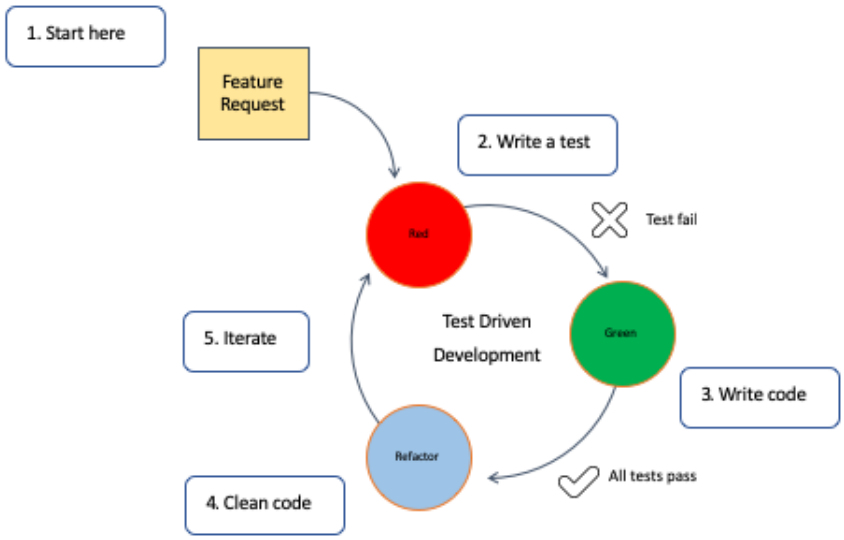


Figura 1 – Exemplo de ciclo de desenvolvimento orientado à testes (TDD) (STEINFELD, 2020)

3.1.1 Modelagem da arquitetura

A ferramenta proposta foi idealizada focando no aspecto da depreciação de um *asset*. Para isso, é preciso que haja uma forma de popular a base de dados com informações telemétricas, a fim de armazenar dados como tempo e forma de utilização de um *asset*, possibilitando o cálculo da depreciação ao longo do tempo e uma projeção do desgaste do dispositivo. Na Figura 3 temos um esquema da arquitetura geral da ferramenta e como ela se integra no fluxo de desenvolvimento para dispositivos IoT. Pode-se observar, também na Figura 3, as interações e componentes existentes na arquitetura da ferramenta proposta.

3.1.2 RESTful API

REST *(Representational State Transfer)* é um padrão de arquitetura de comunicação baseado na web introduzido por Fielding (2000), utilizando HTTP como protocolo para comunicação dos dados trafegados. Além de utilizar uma URL, também chamada de *endpoint*, o padrão REST espera chamadas de métodos específicas para cada URL, sendo esses métodos: GET, POST, PUT e DELETE. Algumas das vantagens propostas pelo padrão RESTful são consequência de sua característica *stateless*, o que significa que cada requisição pode ser feita independentemente de outras e cada requisição possui os dados necessários para serem completadas, possibilitando escalabili-

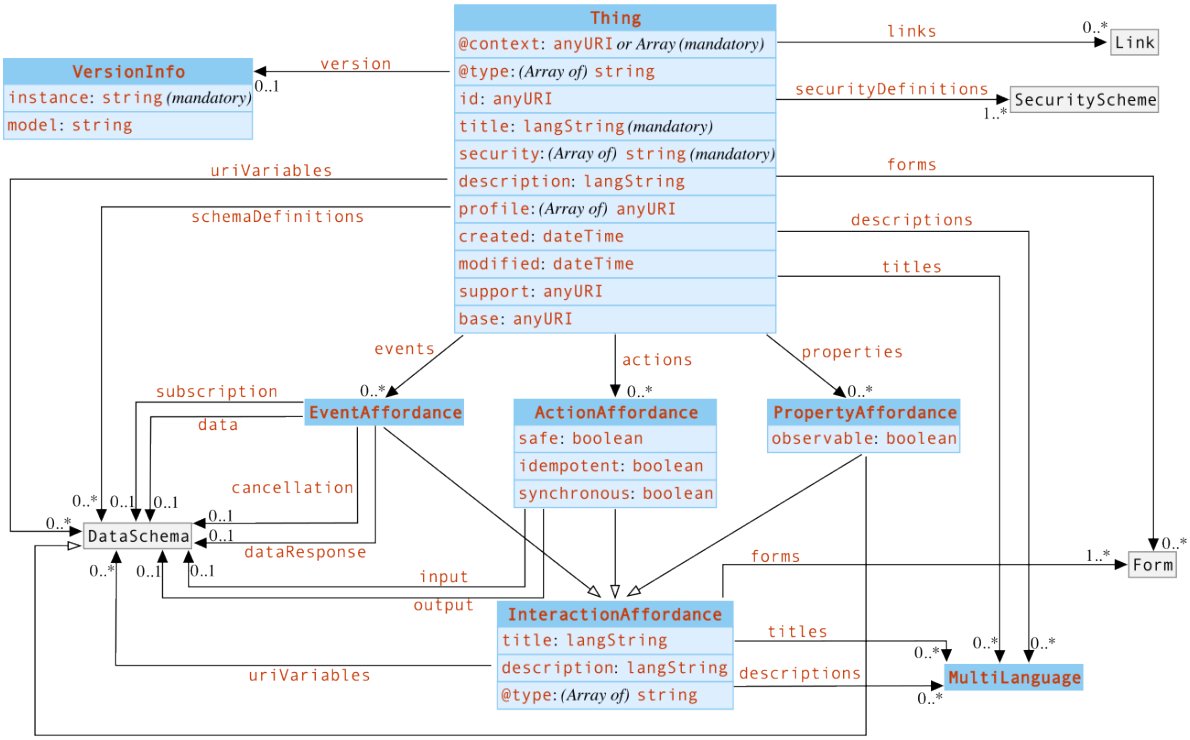


Figura 2 – Exemplo da modelagem de coisas descrito no WoT-TD (W3C, 2023)

dade, interoperabilidade e simplicidade(MORON-RODRIGUEZ; LONGA-CHEVARRIA; SHIGUIHARA-JUAREZ, 2017).

A API foi criada com o intuito de ser integrada utilizando Node-RED[1], uma ferramenta de programação, originalmente criada pela IBM, para integrar dispositivos *hardware*, APIs, e serviços *online*. Contudo, outras ferramentas podem ser utilizadas para substituir o Node-RED. A API foi desenvolvida utilizando o *framework* FastAPI[2], desenvolvido na linguagem de programação Python[3]. FastAPI é um *framework web* focado no desenvolvimento de APIs e conhecido por ser moderno, rápido e eficiente, além de trazer suporte a funcionalidades como anotações de tipos e concorrência, facilitando o desenvolvimento de APIs e sendo flexível o suficiente para ser utilizado em diversos tipos de projetos.

3.1.3 *Dashboard*

A ferramenta também conta com a integração com um *dashboard*, para visualização facilitada dos dados referentes aos *assets* registrados. O *dashboard* foi desenvolvido utilizando Svelte[4], um compilador utilizado para o desenvolvimento de *interfa-*

1 https://nodered.org/, acessado em: 15 fev. 2024
2 https://fastapi.tiangolo.com/, acessado em: 15 fev. 2024
3 https://python.org/, acessado em: 15 fev. 2024
4 https://svelte.dev/, acessado em: 15 fev. 2024

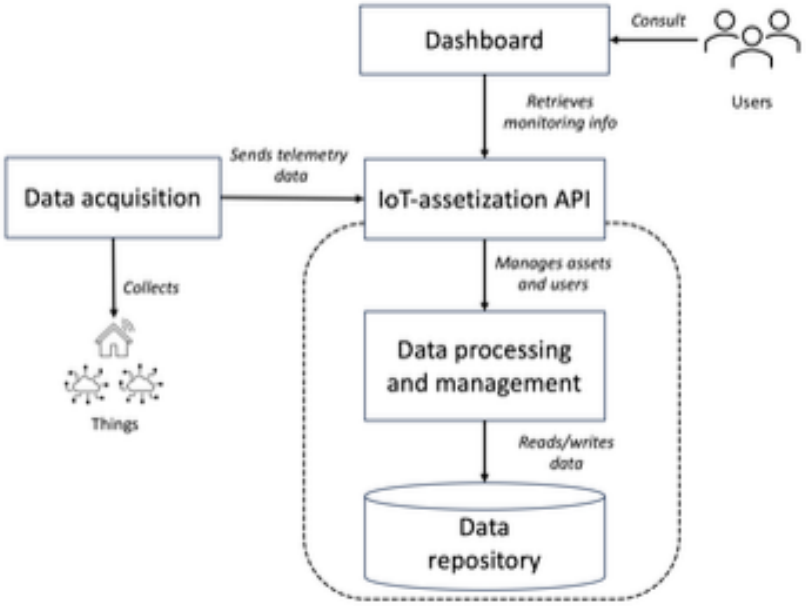


Figura 3 – Arquitetura, componentes e interações na arquitetura da ferramenta (MAAMAR et al., 2024)

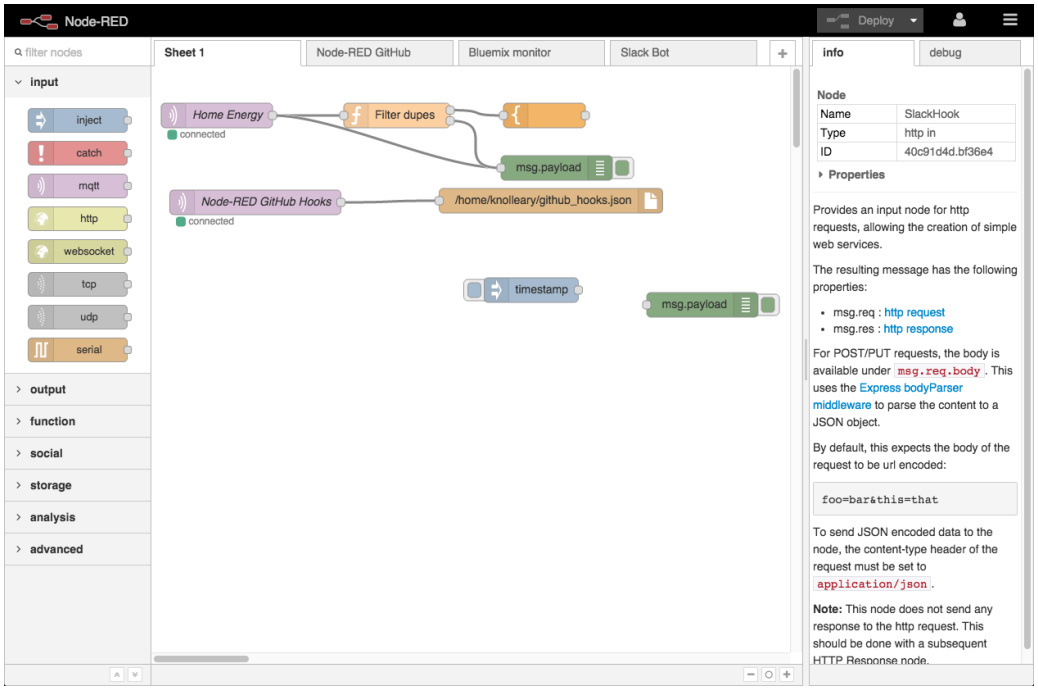


Figura 4 – Exemplo de projeto Node-RED

ces web, traduzindo componentes declarativos e os convertendo em código Javascript, fazendo assim, alterações no *Document Object Mode* (DOM) do *browser*.

3.1.4 Processamento

Além disso, para todo o processamento feito na ferramenta foi utilizada Python, uma linguagem de programação de alto nível, de propósito geral e *multiparadigma*. A linguagem foi escolhida juntamente com o *framework web* FastAPI, pela facilidade de desenvolvimento, maturidade da linguagem, no que se diz respeito à bibliotecas, ferramentas, e facilidade de suporte. Além de ser uma linguagem amplamente adotada em diversos contextos e projetos pela facilidade de leitura e entendimento do código. Em conjunto a isso, para o armazenamento dos dados, foi escolhido o banco de dados PostgreSQL[5], um banco de dados objeto-relacional de código aberto que é ativamente desenvolvido há mais de 35 anos, utilizado em larga escala por ser performático, escalável e robusto.

3.1.5 Ferramentas auxiliares e dependências

A fim de facilitar a configuração do ambiente necessário para execução da ferramenta, foi criado um arquivo Docker[6]. Docker é um sistema baseado em contêineres, provendo um nível de abstração para aplicações, criando um sistema virtualizado, onde o *kernel* do máquina hospedeira é compartilhado com a máquina virtualizada. Portanto, uma pessoa desenvolvedora pode reproduzir seus projetos de forma mais fácil e agnóstica de sistemas operacionais, já contendo as dependências e instruções necessárias para execução do projeto.

Para versionamento de código foi utilizado o sistema Git[7], e a plataforma de repositório Github[8]. Neste consta a estrutura do projeto, arquitetura e código fonte das etapas de desenvolvimento, o versionamento permite uma praticidade no controle de versão de uma base de código, possibilitando, entre muitas coisas, o acesso ao histórico do projeto e acesso de forma de forma remota.

A documentação da API do projeto foi gerada utilizando duas ferramentas: ReDoc[9] e Swagger UI[10]. Ambas podem ser integradas ao FastAPI e, utilizando as especificações descritas pelo OpenAPI[11], gerar automaticamente uma documentação detalhada dos *endpoints* implementados, provendo a URL descrita, os campos esperados em cada requisição e as possíveis respostas do servidor. Uma documentação descritiva de uma API é importante para facilitar a adoção da ferramenta por outros desenvolvedores, sanando possíveis dúvidas e demonstrando o uso detalhado de cada

[5] https://www.postgresql.org/, acessado em: 25 fev. 2024
[6] https://www.docker.com/, acessado em: 15 fev. 2024
[7] https://git-scm.com/, acessado em: 24 fev. 2024
[8] https://github.com/, acessado em: 24 fev. 2024
[9] https://redocly.github.io/redoc/, acessado em: 20 fev. 2024
[10] https://swagger.io/tools/swagger-ui/, acessado em: 20 fev. 2024
[11] https://www.openapis.org/, acessado em: 20 fev. 2024

endpoint disponível.

3.1.6 Autorização e autenticação

Para oferecer um acesso seguro à API, aos dados armazenados e às operações disponíveis, foi implementada uma autenticação baseada em *tokens*. Permitindo que usuários possam ter seus *assets* diretamente relacionados a suas contas. Garantindo separação entre diferentes usuários sobre quais *assets* possuem e o gerenciamento de cada dispositivo cadastrado. Além de prover uma forma de autenticação e de garantia de acesso autorizado, também foi implementado um mecanismo que permite a atualização do *token* de acesso, possibilitando o acesso ininterrupto de usuários autenticados, ou de dispositivos que se integrem à API de forma automatizada.

3.1.7 Ambientes executados

A API foi testada e executada em ambiente local e hospedada na plataforma Fly.io[12]. A documentação da API pode ser acessada através do endereço <https://assetizationapp.fly.dev/docs>. Para a hospedagem, foi utilizada a máquina gratuita disponibilizada, com 256 megabytes de memória, até 3 gigabytes de volume persistente e um limite máximo de até 160 gigabytes de dados trafegados. Fly.io é uma plataforma global de distribuição de aplicações na nuvem, oferecendo serviços para diversas linguagens de programação e *frameworks*. Permitindo a hospedagem de aplicações em contêineres *docker* e disponibilizando, também, bancos de dados para serem usados em suas aplicações, como PostgreSQL e Redis.

[12] https://fly.io/, acessado em: 28 fev. 2024

4 Implementação da proposta

Neste capítulo serão apresentados os resultados do desenvolvimento da ferramenta e da viabilidade de uso da ferramenta para a internet das coisas. Para isso, serão explicitadas as funcionalidades presentes na ferramenta, detalhando seus *endpoints* e sua respectiva documentação e telas. Em seguida será demonstrado o resultado obtido na implementação dessa ferramenta.

4.1 IoT-assetization API

Após a pesquisa levantada sobre a adoção de *assets* em diversos contextos e a aplicabilidade de seus conceitos no cenário da internet das coisas e o desenvolvimento de uma ferramenta prova de conceito para mostrar a viabilidade dessa abordagem, originou-se a ferramenta: IoT-assetization API.

A proposta deste projeto consiste em demonstrar a utilização e consumo dos dados provenientes de dispositivos IoT servidos como *assets*. Utilizando, para esse trabalho, o conceito de depreciação, a fim de prover dados relativos ao custo depreciativo de um *asset*, além de trazer outros dados gerenciais característicos de um *asset*.

Para demonstrar a transição de coisas para *assets*, foi desenvolvido um sistema de *assetização* IoT que disponibiliza uma API para ser integrada ao fluxo de desenvolvimento de uma coisa, garantindo uma comunicação transparente entre o dispositivo e o processamento dos dados *assetizados*. Ademais, o sistema disponibiliza um *dashboard* para cada usuário a fim facilitar a visualização dos dados referentes a seus dispositivos em forma de lista ou em uma visualização mais detalhada. Nas seções a seguir, são apresentadas todas as funcionalidades do IoT-assetization API.

4.2 Arquitetura

4.2.1 Casos de Uso

Na figura 5 abaixo estão dispostos os casos de uso projetados para a aplicação. São eles : fazer cadastro, editar usuário, se autenticar, criar *asset*, listar *assets*, deletar *asset*, editar *asset*, e registrar ação de um *asset*.

Dentre as principais funcionalidades propostas estão a criação de *assets*, listagem de *assets* e o registro de ações de um *asset*. Esse registro de ação é feito somente por usuários autenticados e proprietários do dispositivo em questão.

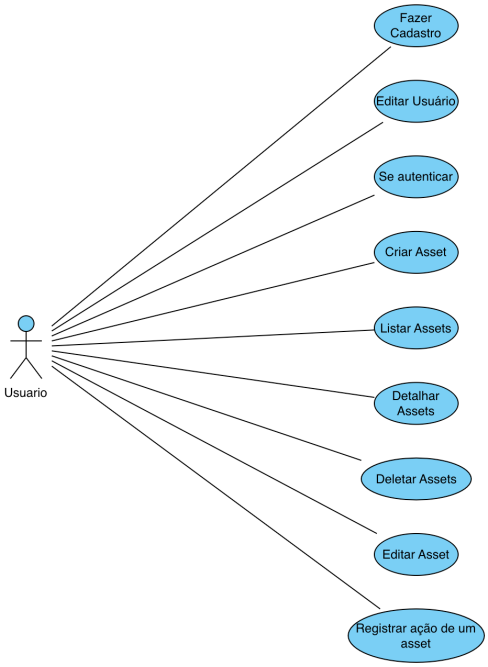


Figura 5 – Diagrama de casos de uso da aplicação proposta

4.2.2 Camada de modelo

Uma das camadas seguindo a arquitetura proposta pelo FastAPI é a camada de modelo, ou *model*. Nesta camada estão definidas as entidades básicas do sistema, representando os atributos de um objeto que foi abstraído. Sendo assim, os elementos que modelam ou representam os elementos de um negócio.

Na Figura 6 está o diagrama de classes da camada de modelo da ferramenta proposta. A camada é composta pelas classes: Assets e Users. A seguir é feita uma breve descrição sobre suas funções e seus atributos.

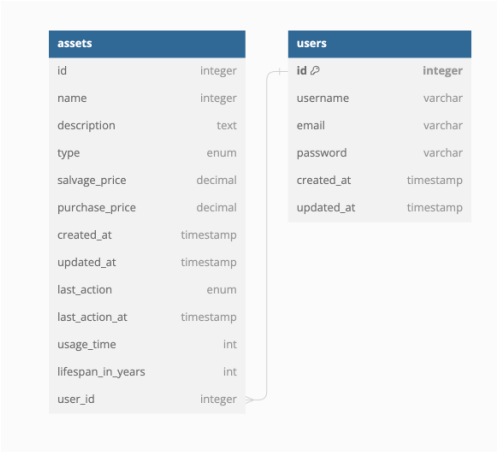


Figura 6 – Camada de modelo da ferramenta proposta

Asset é a entidade responsável por caracterizar um *asset* cadastrado no sistema. Ela possuirá os seguintes atributos: id, name, description, type, salvagePrice, purchasePrice, createdAt, updatedAt, lastAction, lastActionAt, usageTime, lifespanInYears, userId. Os atributos name, description, type, salvagePrice e purchasePrice representam, respectivamente o nome, descrição, tipo, custo de salvamento e valor de compra de um *asset*. Já os atributos usageTime, lastAction, lastActionAt, lifespanInYears e userID representam respectivamente o tempo de uso ativo de um *asset*, qual a última ação realizada, a data da última ação, o tempo de vida útil do *asset* e a qual usuário aquele *asset* está relacionado. Os campos updatedAt e createdAt são automaticamente preenchidos pelas ações de atualização e criação de um *asset*.

User é uma classe que caracterizará o usuário do sistema, possuindo atributos como *username*, *email*, *password*, *createdAt*, *updatedAt*. A propriedade email armazenará um texto para ser utilizado pelo usuário para autenticação, juntamente com o atributo *password*, que representa a senha definida pelo usuário.

4.2.3 Persistência de dados

O sistema gerenciador de banco de dados utilizado foi o PostgreSQL. Foram criadas as tabelas: *user* e *asset*. Estas tabelas estão diretamente relacionadas às classes descritas na aplicação, pois foram definidas pela ferramenta de mapeamento objeto-relacional SQLAlchemy[1], integrada ao *framework* FastAPI. Dessa forma, para cada classe definida no diretório de *models* do projeto, será criada uma tabela correspondente, juntamente com seus atributos no banco de dados.

4.2.4 Cálculo da depreciação

O cálculo de depreciação de um *asset* é descrito como duas propriedades da classe Asset, presente na camada de modelos do projeto, chamadas *depreciable_cost* e *depreciation_per_year*, representando, respectivamente, o valor mínimo aceitável de depreciação e a taxa de depreciação por ano. Isso significa que apesar de serem descritas como métodos da classe Asset, seus valores são acessados de forma similar a um atributo de classe, mas são calculados em tempo de execução. Isso possibilita que esses dados sejam facilmente acessados e reutilizados em outras seções do projeto, além de garantir a atualização e transparência do cálculo de depreciação. A técnica utilizada para calcular a depreciação neste trabalho, para fins de simplicidade, foi a depreciação linear que pode ser calculada através da fórmula descrita na Figura 7.

$$depreciation = (purchaseprice - salvageprice)/lifespan$$

Figura 7 – Fórmula de cálculo de depreciação linear

[1] https://www.sqlalchemy.org/, acesso em: 20 fev. 2024

Além disso, é armazenada a informação de uso ativo de um *Asset*, além de ser um dado importante para entendimento da saúde de um dispositivo. Esse valor pode ser utilizado para o cálculo da depreciação em técnicas como a depreciação acelerada, que leva em consideração o tempo de uso dos *assets* para calcular a perda de valor ao longo do tempo. Para que essa informação seja corretamente armazenada, é feita uma checagem para salvar a última ação registrada, e sua respectiva data em formato *timestamp*. Além disso, há uma verificação do tipo de ação executada, a fim de calcular o tempo gasto entre quando um dispositivo aciona uma ação e quando a ação de desligar é registrada.

Algoritmo 1 – Implementação da função de registro de uso de um *asset*

```
def register_usage(self, action: AssetsActions):
    last_action_at = (
        self.last_action_at
        if self.last_action_at
        else datetime.datetime.now()
    )
    if (
        action == AssetsActions.turn_off
        and self.last_action != AssetsActions.turn_off
    ):
        result = datetime.datetime.utcnow() - last_action_at
        self.usage_time = result.days

    self.last_action = action
    self.last_action_at = datetime.datetime.utcnow()
```

4.2.5 Token de acesso

Para ter acesso aos *endpoints*, o usuário deverá se autenticar. A autenticação pode ser feito através da URI token, utilizando o método HTTP POST e passando, no cabeçalho da requisição, um objeto JSON, contendo os dados: e-mail e senha. Na figura 8 é possível observar as possíveis respostas do servidor para uma solicitação de login. O *token* de acesso deve, então, ser inserido no cabeçalho das requisições posteriores para que o usuário tenha autorização de acesso às rotas. A chave de acesso é um JWT (JSON Web Token), gerada pelo servidor a partir de uma chave secreta. Essa chave é definida pelo desenvolvedor no arquivo de configuração do projeto.

Além disso, a fim de manter a validade do seu *token*, o usuário poderá utilizar o *endpoint* de atualização de *token* através de uma requisição HTTP POST para a URI refresh_token, enviando no cabeçalho o *token* de acesso atualmente válido. Dessa

forma, o usuário consegue manter sua autenticação ativa desde que atualize seu *token* de acesso antes da expiração do *token* anterior.

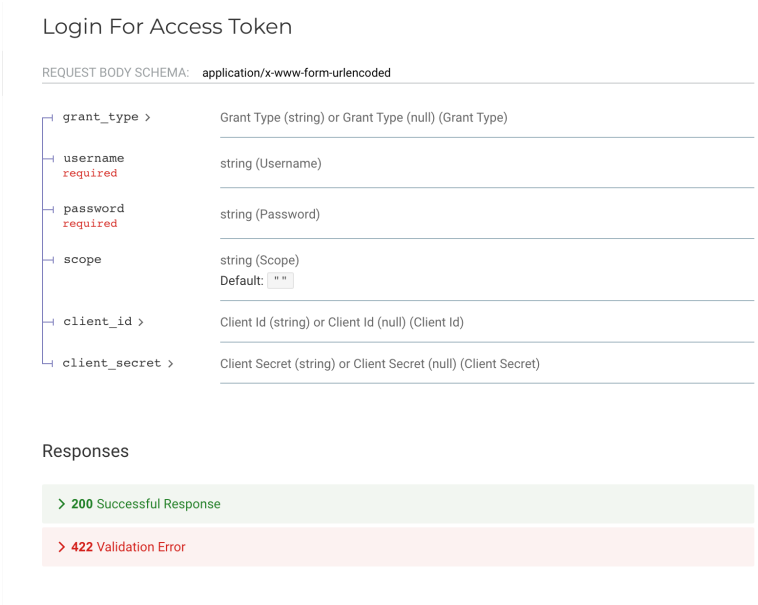


Figura 8 – Documentação de autenticação da aplicação

4.2.6 Camada de controle e rotas

4.2.6.1 API de usuários

Também é possível ter um controle dos usuários através da API Rest disponibilizada. Através da API é possível realizar criação de usuário e autenticação de um usuário, de forma similar que é feita através da interface gráfica. Além dos métodos similares ao disponibilizado pela interface, também é possível realizar edição de um usuário através da API, a fim de promover maior controle ao usuário.

4.2.6.2 API de *assets*

Assim como nas funcionalidades de usuários, o sistema também provê uma API Rest para operações relacionadas a um *asset*, além da visualização disponibilizada pela interface gráfica. A primeira funcionalidade disponível é de criação de um *asset*, permitindo ao usuário registrar um novo item contendo as seguintes informações: nome, descrição, tipo, valor de compra, custo de salvamento e vida útil em anos. Um exemplo de um registro utilizando JSON pode ser visto na Figura 9, além da documentação descritiva, como consta na Figura 10. Para que a criação de um *asset* seja sucedida, o usuário deve estar autenticado e com um *token* de acesso válido.

Além da criação, também é possível realizar operações de edição e atualização de um *asset*, a fim de corrigir eventuais erros ou manter os dados atualizados de forma manual. Um exemplo dos dados necessários para realizar a atualização de um *asset*

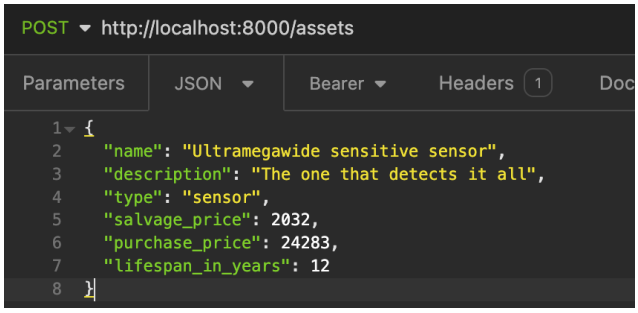


Figura 9 – Exemplo de requisição para criação de um *asset*

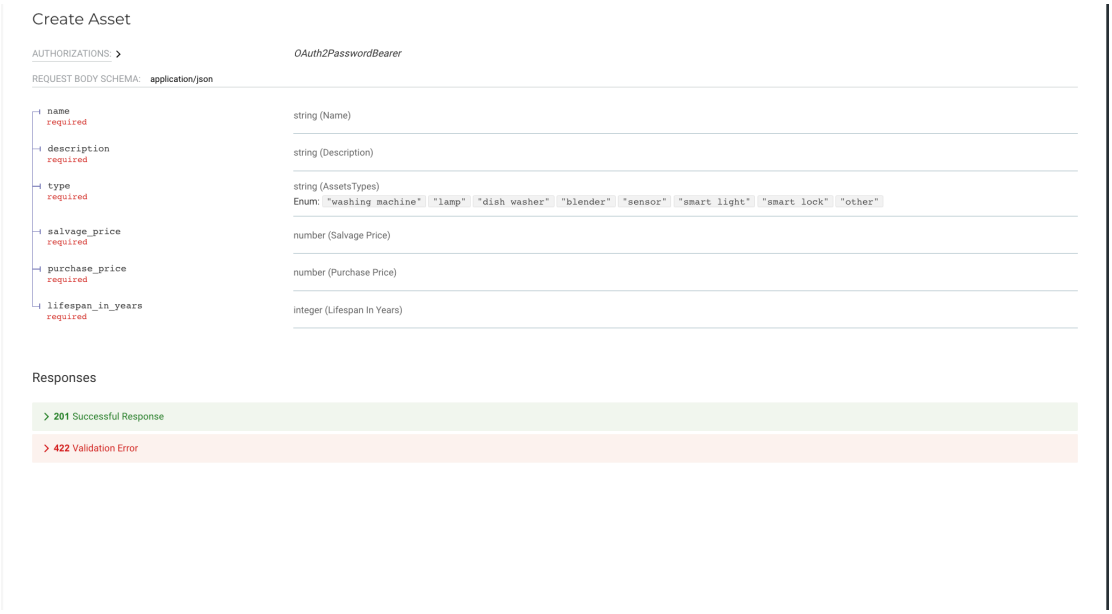


Figura 10 – Documentação de criação de *asset*

pode ser visto na Figura 11. Para que a edição de um *asset* seja sucedida, o usuário deve estar autenticado e com um *token* de acesso válido.

Além da visualização pela interface gráfica, o usuário também pode utilizar a API Rest a fim de listar seus *assets* registrados. Para isso, o usuário pode utilizar o método GET, descrito pelo padrão REST. Para que a listagem de *assets* seja sucedida, o usuário deve estar autenticado e com um *token* de acesso válido. A documentação dessa funcionalidade pode ser vista na Figura 12.

É possível também, realizar, através da API, a deleção de um *asset* registrado. Para realizar essa ação, o usuário precisa estar autenticado e com um *token* de acesso válido. O usuário deve enviar o ID de um *asset* como parâmetro na URL e, caso tenha a devida permissão sobre aquele *asset*, a deleção é feita com sucesso. Caso o usuário envie um ID inválido, ou que não possua permissão para deletar, será retornado um erro de permissão, informando ao usuário o motivo da deleção não ter sido sucedida.

Por fim, há também a funcionalidade de coleta de dados de telemetria dos *assets*. Esse *endpoint* é responsável por registrar o uso de cada dispositivo cadastrado

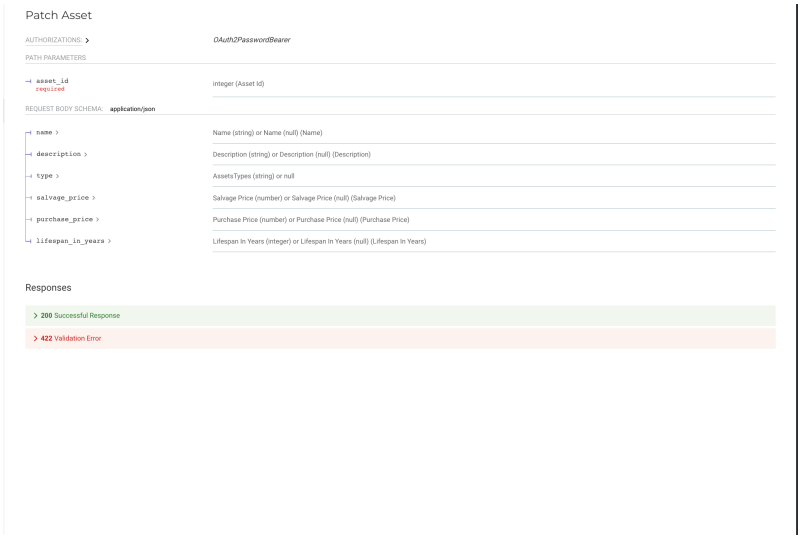


Figura 11 – Documentação de atualização de *asset*

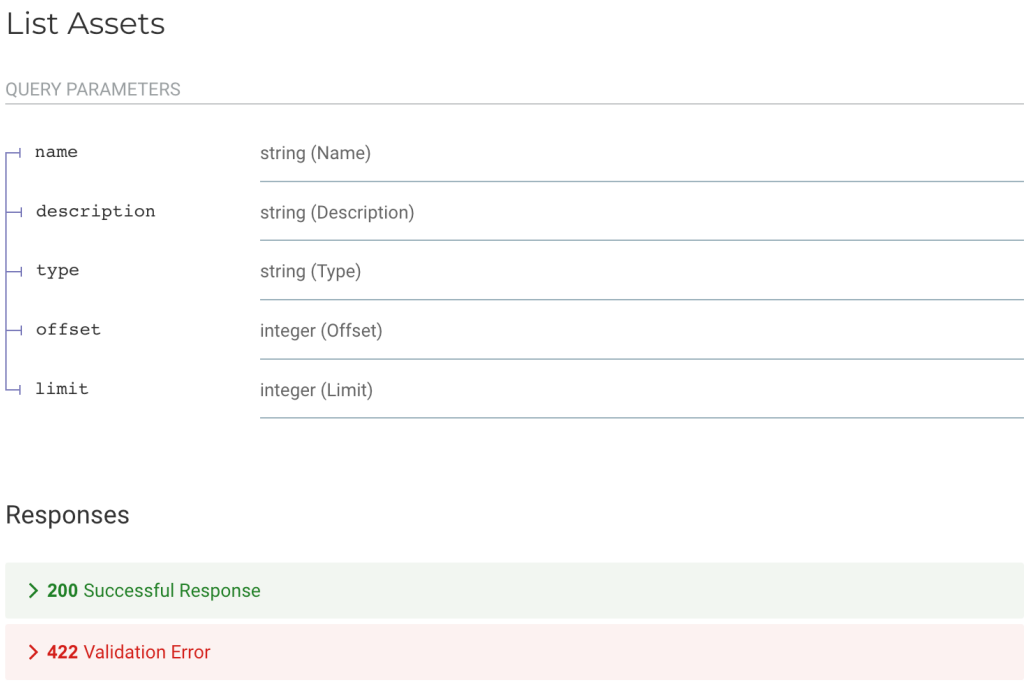


Figura 12 – Documentação de listagem de *assets*

e fazer os cálculos do tempo de vida ativo do *asset*, dado esse utilizado, posteriormente, para calcular o custo de depreciação do dispositivo. Um exemplo de como a funcionalidade funciona pode ser visto na Figura 13.

4.2.7 Telas

A primeira tela da aplicação voltada para ser acessada pelo navegador, é a tela de login do usuário, contendo os campos necessários para a autenticação do usuário: e-mail e senha. Além do botão de "*Login*". Em caso de e-mail ou senha inválidos, o sis-

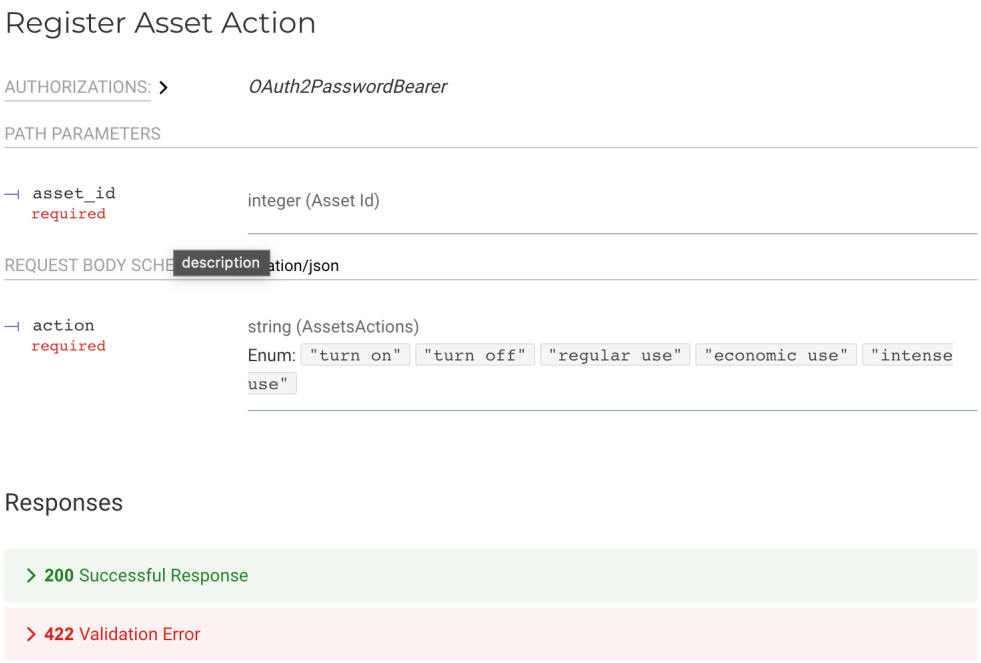


Figura 13 – Documentação de registro de ação de um *asset*

tema exibe uma mensagem de erro contendo a informação “Email ou senha inválidos”. Caso dados válidos de login sejam inseridos, o usuário é redirecionado para a página de listagem de seus *assets*. Caso o usuário não possua uma conta, ele poderá clicar no texto “Sign up” e será redirecionado para a tela de cadastro.

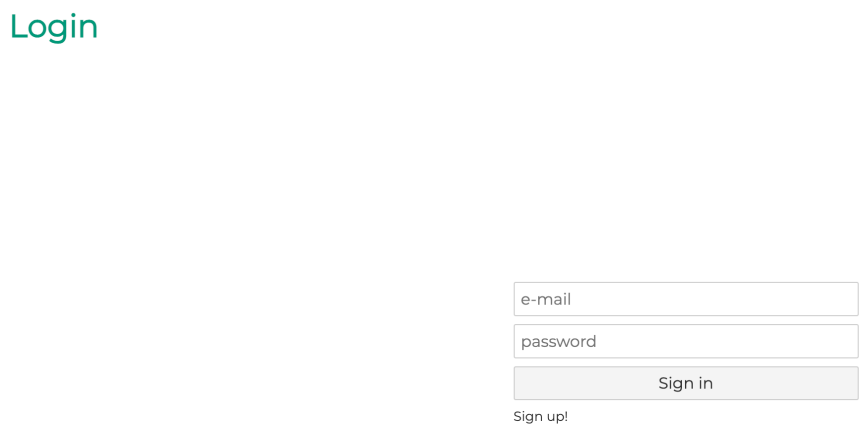


Figura 14 – Tela de login da aplicação

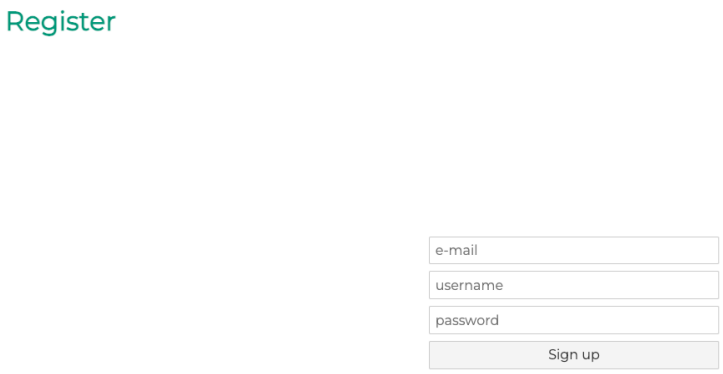
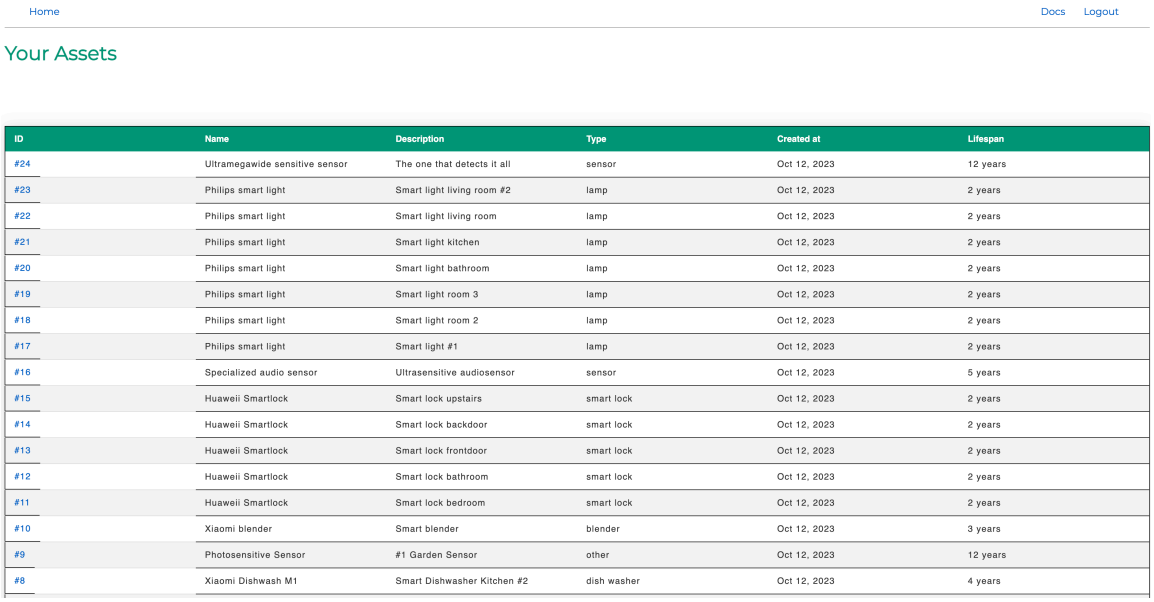


Figura 15 – Tela de cadastro da aplicação

Ao realizar a autenticação através da interface gráfica, o usuário será redirecionado para a listagem de seus *assets*, como pode ser visto na Figura 16. Nessa tela, o usuário poderá ter uma visão geral de seus dispositivos cadastrados além de alguns atributos importantes, como: ID, nome, descrição, tipo, data de criação e tempo de vida útil do dispositivo.



ID	Name	Description	Type	Created at	Lifespan
#24	Ultramegawide sensitive sensor	The one that detects it all	sensor	Oct 12, 2023	12 years
#23	Philips smart light	Smart light living room #2	lamp	Oct 12, 2023	2 years
#22	Philips smart light	Smart light living room	lamp	Oct 12, 2023	2 years
#21	Philips smart light	Smart light kitchen	lamp	Oct 12, 2023	2 years
#20	Philips smart light	Smart light bathroom	lamp	Oct 12, 2023	2 years
#19	Philips smart light	Smart light room 3	lamp	Oct 12, 2023	2 years
#18	Philips smart light	Smart light room 2	lamp	Oct 12, 2023	2 years
#17	Philips smart light	Smart light #1	lamp	Oct 12, 2023	2 years
#16	Specialized audio sensor	Ultrasensitive audiosensor	sensor	Oct 12, 2023	5 years
#15	Huaweii Smartlock	Smart lock upstairs	smart lock	Oct 12, 2023	2 years
#14	Huaweii Smartlock	Smart lock backdoor	smart lock	Oct 12, 2023	2 years
#13	Huaweii Smartlock	Smart lock frontdoor	smart lock	Oct 12, 2023	2 years
#12	Huaweii Smartlock	Smart lock bathroom	smart lock	Oct 12, 2023	2 years
#11	Huaweii Smartlock	Smart lock bedroom	smart lock	Oct 12, 2023	2 years
#10	Xiaomi blender	Smart blender	blender	Oct 12, 2023	3 years
#9	Photosensitive Sensor	#1 Garden Sensor	other	Oct 12, 2023	12 years
#8	Xiaomi Dishwash M1	Smart Dishwasher Kitchen #2	dish washer	Oct 12, 2023	4 years

Figura 16 – Tela de listagem de *assets*

Ao clicar no ID de um dos itens listados, o usuário será redirecionado para a página de detalhes de um *asset*. A tela conta com um gráfico de previsão de custo de depreciação, demarcando o valor esperado do ativo a cada ano. Uma linha vermelha demonstra o custo de salvamento do *asset*, informação cadastrada pelo próprio usuário

na criação de um item.

Logo abaixo do gráfico de depreciação, há uma seção com informações de uso do *asset*, contendo dados como: anos de uso, tempo de uso ativo, e condições gerais de uso.

Por último, temos dois gráficos que mostram informações gerais sobre os *assets* daquele usuário, demonstrando, no primeiro gráfico, a quantidade de *assets* cadastrados agrupados por seus tipos, enquanto o segundo gráfico demonstra dados gerais sobre a saúde dos *assets*, em um gráfico de setores, dividindo os dispositivos acima do custo de salvamento, e os dispositivos abaixo do custo de salvamento.

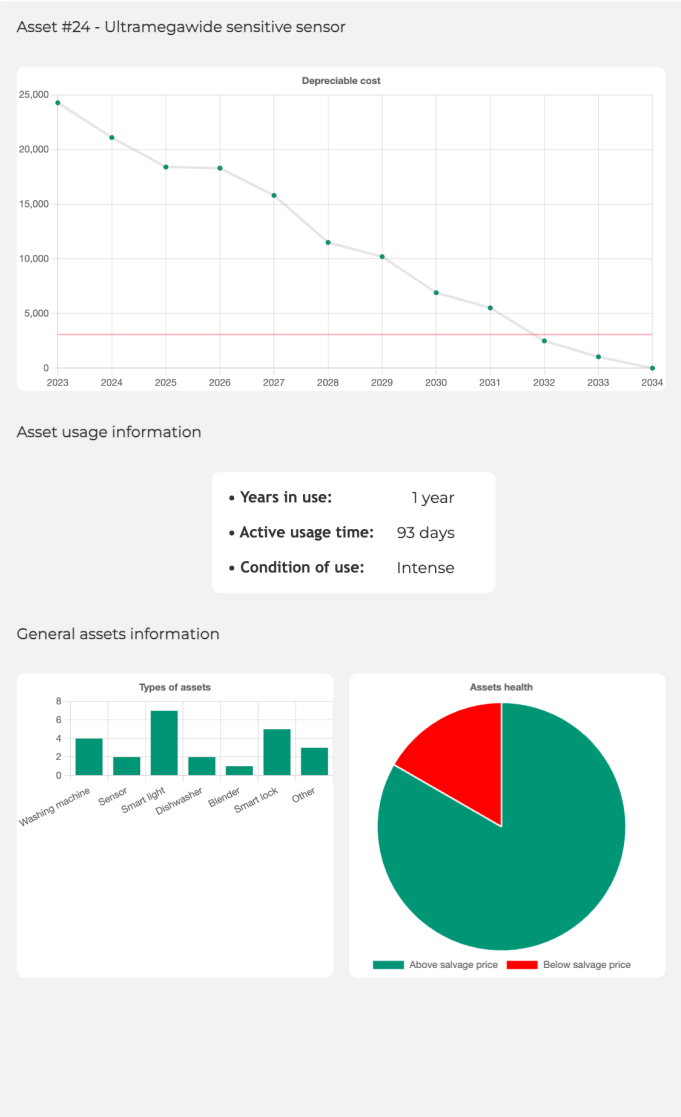


Figura 17 – Tela de detalhe de *asset*

5 Conclusão

Este trabalho tratou do projeto e do desenvolvimento de uma ferramenta baseada no padrão REST, juntamente com uma porção web, implementando desde sua interface web até a implantação dessa ferramenta na nuvem. A aplicação desenvolvida tem o objetivo de demonstrar a viabilidade de adoção de um paradigma orientado à *assets* voltado ao contexto de internet das coisas. Para propor uma melhor visualização e gerenciamento de dados relacionados a atributos como: depreciação, transferibilidade, descartabilidade, e convertibilidade. Em geral, o retorno foi positivo, foi possível mostrar a aplicabilidade de uma implementação de *assets* voltadas para a internet das coisas e intenciona-se que a ferramenta desenvolvida fique disponível para a comunidade.

Somado a isso, este trabalhou visou empregar e utilizar tecnologias e serviços atuais, como FastAPI, Svelte, ReDoc, PostgreSQL e Docker. Foram utilizados planos gratuitos para a implantação, desenvolvimento e manutenção da ferramenta na nuvem. Dessa forma, foi possível planejar e atuar em todo o processo de idealização e desenvolvimento do projeto, sem necessidade de aportes financeiros.

Foram utilizados conceitos aprendidos e apresentados principalmente nas disciplinas de Desenvolvimento de Aplicações para Web, Banco de Dados, Projeto e Desenvolvimento de Software, Metodologias Ágeis de Desenvolvimento de Software, e Testes de Software. Essas disciplinas, em conjunto com as disciplinas básicas do curso, foram fundamentais na construção do conhecimento necessário para o desenvolvimento deste projeto.

Dentre as lições aprendidas, uma das principais foi a importância da utilização de testes unitários e uma documentação detalhada das funcionalidades e funcionamento de uma API. A partir dessa documentação, o trabalho pode ser desenvolvido ao longo do período de forma organizada e mantendo um grau de confiança a cada modificação e refatoração feita no código. Além disso, foi possível a percepção de muitas funcionalidades que poderiam ser adicionadas ao sistema.

Diante da possibilidade de manter o sistema disponível gratuitamente, como prova de conceito, pretende-se o incrementar e implementar novas funcionalidades para melhor utilização do *dashboard* e da experiência de uso da API.

Como possíveis trabalhos futuros, sugere-se que a ferramenta tenha uma evolução para abranger outras características típicas de um *asset*, focando em aspectos e implicações econômicas da *assetização* de IoT. Além disso, sugere-se testar a adoção de um modelo de persistência poliglota, a fim de permitir uma melhor escalabilidade

da ferramenta, avaliando a viabilidade de adoção de bancos de dados não-relacionais que sejam otimizados para um grande número de operações de escrita e leitura, melhorando o desempenho da ferramenta.

Referências

BARBER, N. *The Digital Asset Management Cookbook*. [S.l.], 2020. Acessado em: 29 de Janeiro de 2024. Disponível em: <https://www.forrester.com/report/the-digital-asset-management-cookbook/RES159480>. Citado na página 14.

BIRCH, K.; MUNIESA, F. *Assetization: Turning Things into Assets in Technoscientific Capitalism*. The MIT Press, 2020. ISBN 9780262359030. Disponível em: <https://doi.org/10.7551/mitpress/12075.001.0001>. Citado na página 16.

CALAIS, P.; FRANZINI, L. Test-driven development benefits beyond design quality: Flow state and developer experience. In: *2023 IEEE/ACM 45th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. [S.l.: s.n.], 2023. p. 106–111. Citado na página 18.

CISCO IBSG. *The Internet of Everything: How More Relevant and Valuable Connections Will Change the World*. 2006. Acessado em: 20 de Agosto de 2023. Disponível em: <https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoE_Economy_FAQ.pdf>. Citado na página 12.

DZone. *IoT: Applications, Protocols, and Best Practices*. 2017. Acessado em: 20 de Agosto de 2023. Disponível em: <https://dzone.com/guides/iot-applications-protocols-and-best-practices>. Citado na página 12.

EVANS, D. *Reference Model for Service Oriented Architecture 1.0*. [S.l.], 2012. Citado na página 12.

FIELDING, R. T. *Architectural styles and the design of network-based software architectures*. Tese (Publication) — University of California, Irvine, 2000. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>. Citado na página 19.

GUNNARSSON., M.; GEHRMANN., C. Secure ownership transfer for the internet of things. In: INSTICC. *Proceedings of the 6th International Conference on Information Systems Security and Privacy - ICISSP*. [S.l.]: SciTePress, 2020. p. 33–44. ISBN 978-989-758-399-5. ISSN 2184-4356. Citado na página 15.

IFRS. *Conceptual Framework: Elements of financial statements— definitions and recognition*. 2015. Acessado em: 21 de Agosto de 2023. Disponível em: <https://www.ifrs.org/content/dam/ifrs/project/conceptual-framework/webcast-2015/cf-webcast-5-pdf.pdf>. Citado 3 vezes nas páginas 12, 14 e 16.

Internal Revenue Service. *How to Depreciate Property*. 2021. Acessado em: 24 de Fevereiro de 2024. Disponível em: <https://www.irs.gov/publications/p946>. Citado 2 vezes nas páginas 16 e 17.

KUZMINYKH, I.; CARLSSON, A. Analysis of assets for threat risk model in avatar-oriented iot architecture. *Internet of Things, Smart Spaces, and Next Generation Networks and Systems. NEW2AN 2018, ruSMART 2018. Lecture Notes in Computer Science, vol 11118. Springer, Cham. (pp. 52-63).*, 2018. Citado na página 14.

MAAMAR, Z. et al. From IoT Servitization to IoT Assetization. A ser publicado. 2024. Citado 2 vezes nas páginas 12 e 21.

MAAMAR, Z. et al. Towards a cell-inspired approach for a sustainable internet-of-things. v. 14, abr. 2021. Publisher Copyright: © 2021 Elsevier B.V. Citado na página 15.

MAAMAR, Z.; FACI, N.; YAHYA, F. A Guiding Framework for IoT Servitization. In: *Next-Gen Digital Services. A Retrospective and Roadmap for Service Computing of the Future. Lecture Notes in Computer Science*. [S.l.]: Springer, 2021. cap. Services and the Internet of Things, p. 179–188. Citado 2 vezes nas páginas 12 e 16.

MORON-RODRıGUEZ, L.; LONGA-CHEVARRIA, B.; SHIGUIHARA-JUAREZ, P. Analysis of image transfer mechanisms in a restful api client-server architecture and its application to lane detection. In: *2017 IEEE International Conference on Aerospace and Signals (INCAS)*. [S.l.: s.n.], 2017. p. 1–4. Citado na página 20.

QUICSOLV. *Types of Depreciation and their Calculations*. 2022. Acessado em 29 de Janeiro de 2024. Disponível em: <https://www.quicsolv.com/blog/internet-of-things/asset-tracking/types-depreciation-calculations>. Citado 2 vezes nas páginas 16 e 17.

ROSS, R. et al. *Developing Cyber-Resilient Systems: A Systems Security Engineering Approach*. [S.l.], 2021. Disponível em: <https://csrc.nist.gov/pubs/sp/800/160/v2/r1/final>. Citado 3 vezes nas páginas 12, 14 e 16.

STEINFELD, G. *5 steps of test-driven development*. 2020. Acessado em: 1 de Março de 2024. Disponível em: <https://developer.ibm.com/articles/5-steps-of-test-driven-development/>. Citado na página 19.

TARKOMA, S.; KATASONOV, A. Internet of things strategic research agenda (IoT-SRA). *Finnish Strategic Centre for Science, Technology, and Innovation: For Information and Communications (ICT) Services, businesses, and technologies*, 2011. Citado na página 12.

W3C. *ODRL Information Model 2.2*. 2018. Acessado em: 27 de Fevereiro de 2024. Disponível em: <https://www.w3.org/TR/2018/REC-odrl-model-20180215/>. Citado na página 14.

W3C. *Web of Things (WoT) Thing Description*. 2023. Acessado em: 24 de Fevereiro de 2024. Disponível em: <https://www.w3.org/TR/wot-thing-description/>. Citado 2 vezes nas páginas 18 e 20.

APÊNDICE A – Documentação restante de *endpoints*

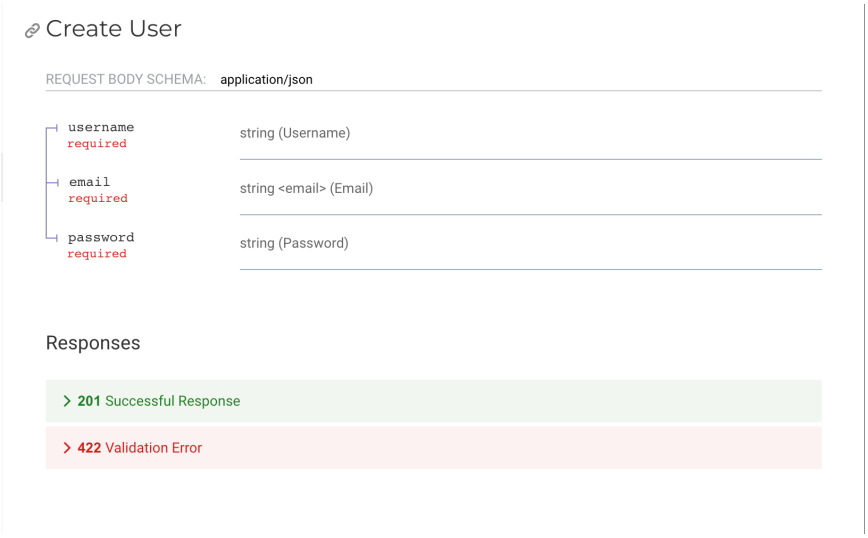


Figura 18 – Documentação de criação de usuário

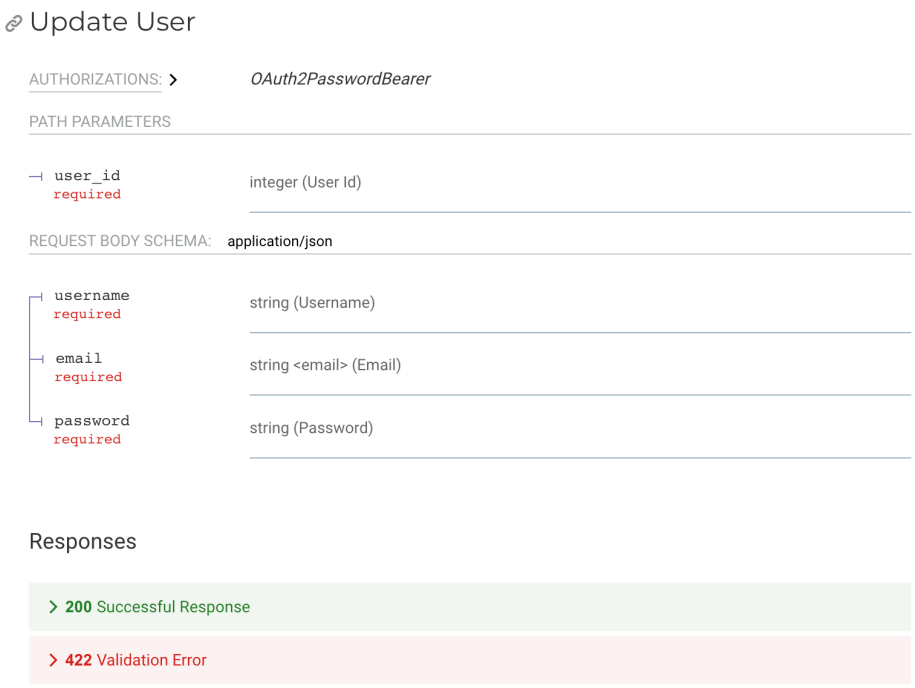


Figura 19 – Documentação de edição de usuário

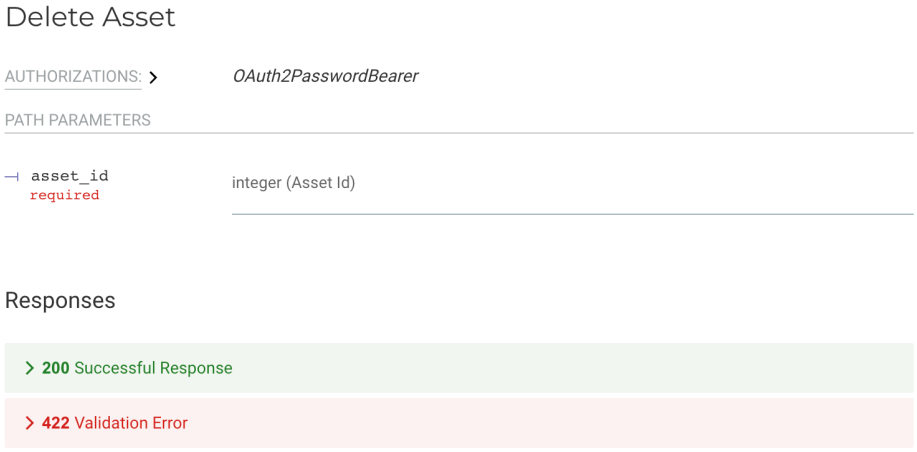


Figura 20 – Documentação de deleção de *asset*