UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS I – CAMPINA GRANDE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
MESTRADO PROFISSIONAL EM CIÊNCIA E TECNOLOGIA EM SAÚDE

LUCAS MYLLENNO SILVA MONTEIRO LIMA

MOVVO: SISTEMA DE GESTÃO, LOCALIZAÇÃO E MONITORAMENTO DE EQUIPAMENTOS HOSPITALARES

CAMPINA GRANDE - PB
2022

LUCAS MYLLENNO SILVA MONTEIRO LIMA

MOVVO: SISTEMA DE GESTÃO, LOCALIZAÇÃO E MONITORAMENTO DE EQUIPAMENTOS HOSPITALARES

Dissertação apresentada ao Programa de Pós-graduação em Ciência e Tecnologia em Saúde da Universidade Estadual da Paraíba – UEPB, como requisito parcial à obtenção do título de Mestre em Ciência e Tecnologia em Saúde.

Área de concentração: Ciência e Tecnologia em Saúde.

Orientador: Prof. Dr. Frederico Moreira Bublitz

CAMPINA GRANDE
2022

L732m Lima, Lucas Myllenno Silva Monteiro.

MOVVO [manuscrito] : sistema de gestão, localização e monitoramento de equipamentos hospitalares / Lucas Myllenno Silva Monteiro Lima. - 2022.

136 p. : il. colorido.

Digitado.

Dissertação (Mestrado em Profissional em Ciência e Tecnologia em Saúde) - Universidade Estadual da Paraíba, Pró-Reitoria de Pós-Graduação e Pesquisa , 2022.

"Orientação : Prof. Dr. Frederico Moreira Bublitz , Coordenação do Curso de Computação - CCT."

1. Tecnologia da Saúde. 2. Modelos de negócios. 3. Serviços em nuvem. 4. Internet das Coisas. 5. Sistemas de Localização. I. Título

21. ed. CDD 003.5

Elaborada por Deivid A. M. Neves - CRB - 15/899 BC/UEPB

LUCAS MYLLENNO SILVA MONTEIRO LIMA

MOVVO: SISTEMA DE GESTÃO, LOCALIZAÇÃO E MONITORAMENTO DE EQUIPAMENTOS HOSPITALARES

Dissertação apresentada ao Programa de Pós-graduação em Ciência e Tecnologia em Saúde da Universidade Estadual da Paraíba – UEPB, como requisito parcial à obtenção do título de Mestre em Ciência e Tecnologia em Saúde.

Área de concentração: Ciência e Tecnologia em Saúde.

Aprovada em: 16/05/2022

BANCA EXAMINADORA:

Prof. Dr. Frederico Moreira Bublitz
Universidade Estadual da Paraíba (UEPB)

Prof. Dr. Wellington Candeia Araújo
Universidade Estadual da Paraíba (UEPB)

Profa. Dra. Karolina Celi Tavares Bezerra
Universidade do Minho

À Deus, pela presença diária e por toda a sabedoria que me foi dada para superar os obstáculos do caminho e seguir os meus sonhos. **À minha esposa, Ana Débora**, pelo amor, compreensão e o apoio em meio a tantos desafios em minha vida. **Aos meus pais**, **Jovelina da Silva e Marcos Antônio**, pela proteção, ensinamentos e advertências que me ajudaram a crescer e enfrentar o mundo. **E às minhas irmãs, Morganna Mayarah e Monalyza Myllenna**, por fazerem parte da minha vida com seus conselhos, incentivo e motivação.

AGRADECIMENTOS

À Deus, pela sua soberania, grandiosidade, generosidade e por me ensinar que as tribulações eram apenas ensinamentos e oportunidades para novos caminhos em minha vida. Sua companhia me guia nessa longa caminhada e devo-lhe todas as minhas realizações pessoais e profissionais.

À minha esposa, Ana Débora Coutinho Lima, pelo amor, carinho, paciência, compreensão e por compartilhar comigo todos os nossos sonhos. Seu apoio vem sendo essencial para as minhas conquistas.

Aos meus pais, Jovelina da Silva Monteiro Lima e Marcos Antônio Monteiro Lima, que me criaram, acompanharam o meu crescimento, os meus aprendizados e fizeram tudo o que foi possível para que eu descobrisse os meus objetivos e realizasse os meus sonhos desde a infância.

Às minhas irmãs, Morganna Mayarah Silva Monteiro Lima e Monalyza Myllenna Silva Monteiro Lima, por fazerem parte da minha vida e por acompanharem a minha trajetória com seus conselhos e convivência.

Ao meu orientador, Prof. Dr. Frederico Moreira Bublitz, por estar me orientando desde a metade da minha graduação. Foram 7 anos de muita confiança, paciência e ensinamentos. Suas oportunidades me levaram a desenvolver uma diversidade de projetos, desde simples aplicativos até um sistema de adequação ergonômica para ciclistas. A sua confiança me trouxe as melhores oportunidades e foi o responsável por toda a minha evolução profissional. Obrigado por nunca ter desistido de mim!

Ao Núcleo de Tecnologias Estratégicas em Saúde - NUTES, pelo acolhimento e por me proporcionar o desenvolvimento desse projeto e de todos os outros projetos que me trouxeram grandes desafios e conhecimentos desde a minha graduação.

"Nós só podemos ver um pouco do futuro, mas o suficiente para perceber que ainda há muito a fazer." (Alan Turing)

RESUMO

As organizações de saúde têm buscado soluções bem-sucedidas para ampliar a eficiência na distribuição de leitos, recursos humanos e equipe médica. Uma organização muito requisitada necessita que as suas demandas internas sejam atendidas rapidamente, pois os pequenos atrasos influenciam direta e indiretamente no aumento dos gastos. Obter um controle eficiente mantêm a equipe médica e os pacientes satisfeitos com o serviço. Visando esse problema, um sistema de monitoramento e gestão de equipamentos foi desenvolvido para essas organizações, oferecido como módulo de um modelo de negócios corporativo e fornecido como um serviço em nuvem. Esse sistema aperfeiçoará a maneira como as demandas são atendidas, de modo que, por meio de uma infraestrutura de monitoramento, a equipe médica localizará os equipamentos requisitados rapidamente e isso irá reduzir o desperdício de tempo, otimizando o fluxo no ambiente de trabalho e intensificando a atenção ao paciente.

Palavras-chave: Tecnologia da Saúde. Modelos de Negócios. Serviços em Nuvem. Internet das Coisas. Sistemas de Localização.

ABSTRACT

Health organizations have sought successful solutions to expand the efficiency in hospital beds distribution, human resources, and medical team. A highly requested organization needs its demands to be rapidly achieved, because small delays influence directly or indirectly in increasing spendings. Obtaining an effective control of activities maintains medical team and patients satisfied with the service. Aiming this problem, a monitoring and management equipment system was developed for these organizations, offered as a module of a corporate business model and provided as a cloud service. This system is going to improve how demands are controlled, so that, through a monitoring infrastructure, the medical team will locate the requested equipment quickly and it will reduce the waste of time, optimize the flow in the work environment and intensify the attention for patients.

Keywords: Health Technology. Business Models. Cloud Services. Internet of Things. Location Systems.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
AWS	*Amazon Web Services*
BLE	*Bluetooth Low Energy*
ERP	*Enterprise Resource Planning*
GPS	*Global Positioning System*
GUI	*Graphical User Interface*
HTML	*HyperText Markup Language*
IoT	*Internet of Things*
IR	*Infrared Radiation*
JSON	*JavaScript Object Notation*
MVC	*Model-View-Controller*
PWA	*Progressive Web App*
REST	*Representational State Transfer*
RFID	*Radio-Frequency IDentification*
RTLS	*Real-time Locating System*
SaaS	*Software as a Service*
SASS	*Syntactically Awesome Style Sheets*
SDK	*Software Development Kit*
UI	*User Interface*
UWB	*Ultra-wide-band*
VLC	*Visible Light Communication*
WLAN	*Wireless Local Area Network*

SUMÁRIO

1 INTRODUÇÃO

As análises de eficiência em uma organização desempenham um papel crucial para conscientização em busca de transformação dos gastos em resultados significativos (CAMPANELLA *et al.*, 2017). A indústria é um ambiente competitivo e isso exige que as organizações melhorem sua eficiência na gestão de projetos. Várias ferramentas foram criadas para aprimorar o desempenho no gerenciamento de projetos ao longo de todo o seu ciclo de vida, no entanto, a integração desses sistemas ainda é considerada um grande desafio (ABOABDO; ALDHOIENA; AL-AMRIB, 2019).

O ERP é um modelo de negócios que permite o gerenciamento de projetos usando soluções adequadas para os desafios de integração. Ele é responsável por trazer um conjunto de vantagens ao integrar os principais sistemas com diferentes responsabilidades dentro da organização. No entanto, os estudos recentes mostraram que a implantação ERP é dispendiosa, independentemente do setor para o qual está sendo usada, principalmente quando a implementação ocorre em sua totalidade dentro da organização (ABOABDO; ALDHOIENA; AL-AMRIB, 2019).

Uma maneira de diminuir os custos e a complexidade de uma implementação ERP, é usá-lo em conjunto com o SaaS. O SaaS elimina a necessidade de uma implementação ERP no ambiente organizacional, pois transmite para o provedor essa responsabilidade (ACHARGUI; ZAOUIA, 2016). Ele dispensa todos os *softwares* locais e os implementa em forma de módulos utilizando tecnologias da Computação em Nuvem (ALEEM *et al.*, 2018). O acesso às funcionalidades do ERP-SaaS ocorre de maneira remota durante o período da sua contratação, sendo oferecido como um produto "pronto para uso"por uma taxa de assinatura mensal (SAA *et al.*, 2017).

Os sistemas que utilizam tecnologias de rastreamento é outra solução inovadora que vem sendo explorada nos últimos anos, pois traz inúmeros benefícios no meio corporativo e melhora o fluxo de trabalho das equipes médicas (YOO *et al.*, 2018). Essas soluções contam com sistemas de localização e solicitação de equipamentos, tendo suas demandas respondidas rapidamente, com registro de suas localizações em tempo real, através de diferentes tecnologias de baixo custo e eficiência energética (KANAN; ELHASSAN, 2015; FRISCH, 2019).

Neste trabalho apresentaremos o desenvolvimento de um sistema de gestão e monitoramento de equipamentos utilizando o ERP-SaaS como proposta de implantação, com o propósito de aprimorar o fluxo de trabalho nas organizações de saúde. Analisamos as tecnologias de localização aptas a serem utilizadas como fornecedoras de dados e apresentamos uma arquitetura utilizando funcionalidades da Amazon AWS, tendo em vista que essa infraestrutura tem importância direta na eficiência do nosso sistema.

2 ESTADO DA ARTE

O gerenciamento de qualidade ocupa um papel estratégico nas organizações. A adoção de ferramentas com uma arquitetura alinhada promove o desenvolvimento competitivo e sustentável. Prestar um serviço bem-conceituado de maneira eficaz e barata é uma preocupação constante. A sustentabilidade é um dos maiores desafios, bem como a busca por efetividade e eficiência, com o objetivo de satisfazer o cliente, agregando valor aos produtos e serviços (ROCHA; FREIXO, 2015).

Os serviços hospitalares considerados de alta qualidade são seguros, eficazes, oportunos, equitativos, integrados, eficientes e centrado nas pessoas. Os hospitais reorganizam processos e reorientam seus esforços logísticos para tornar os cuidados acessíveis, aceitáveis e contínuos do ponto de vista do paciente. Optar por cuidados de alta qualidade também significa que as pessoas são capacitadas para tomar decisões relacionadas aos seus cuidados de saúde (WHO, 2020).

É fundamental para qualquer organização hospitalar ter um setor de tecnologia da informação, capaz de coordenar e fornecer informações de gerenciamento em tempo oportuno, abrangendo todos os aspectos de negócios e permitindo a tomada de decisões estratégicas ajustadas ao bom desempenho da organização (ROCHA; FREIXO, 2015). Vários trabalhos têm se concentrado na questão de convergência entre a medicina e os progressos tecnológicos, em busca de alcançar soluções econômicas e modelos de negócio socialmente viáveis (BERNDT *et al.*, 2012).

O crescente sucesso das tecnologias baseadas em nuvem, junto com a pressão das exigências nos ambientes de negócios, deu origem a um modelo que surgiu da combinação de sistemas de Planejamento de Recursos Empresariais - ERP com modelos de *Software* Baseado em Serviço - SaaS (SAA *et al.*, 2017). As implementações de ERPs forçam as organizações a otimizar e padronizar os processos para se ajustarem às melhores práticas dentro de cada setor (GOVINDARAJU *et al.*, 2015). Enquanto o SaaS oferece acesso remoto aos seus benefícios, reduzindo a complexidade e os custos de implementação (BERNDT *et al.*, 2012)

2.1 UTILIZANDO ERP PARA GESTÃO EMPRESARIAL

Um sistema de gestão ERP é uma solução que permite que as empresas e seus fornecedores gerenciem grandes projetos de maneira eficiente e eficaz durante todo o ciclo de vida do projeto (ABOABDO; ALDHOIENA; AL-AMRIB, 2019). É uma estrutura amplamente multifuncional, projetada para definir, organizar e padronizar os processos de negócios necessários no controle de uma organização, otimizando o desempenho e a

competitividade por eficiência, permitindo o uso de conhecimento interno para buscar vantagens externas (NIKITOVIć; MAHMUTOVIć, 2019).

O ERP tem como objetivo principal o controle sobre os processos de desenvolvimento, entrega e execução dos projetos, seguindo o fluxo apresentado na Figura 1. Os *softwares* responsáveis por gerenciar finanças, inventários, recursos humanos, planejamentos, entre outros, são substituídos por um *software* único dividido em módulos. Cada módulo funciona como um sistema independente, mas todos são vinculados e compartilham informações entre si (ABOABDO; ALDHOIENA; AL-AMRIB, 2019).

Figura 1 – Fluxo de Implementação do ERP

Fonte: Orosz, Selmeci e Orosz (2019)

A escolha por sistemas ERPs é uma das soluções mais adotadas nas organizações, mas a sua complexidade ainda costuma causar uma rejeição generalizada (GOVINDARAJU *et al.*, 2015). Apesar da sua essência ser um processo de Tecnologia da Informação, o processo de implementação é amplamente impactado pelos fatores humanos, pois o envolvimento e a conscientização da alta gerência, o treinamento e o suporte aos usuários e a composição da equipe são os fatores mais significativos para o sucesso da implantação (ABOABDO; ALDHOIENA; AL-AMRIB, 2019).

Antes de implantar uma solução ERP, é necessário seguir um rigoroso processo de examinação para garantir a sua aceitação, pois nem todos os ajustes serão totalmente alinhados às preferências das organizações (GOVINDARAJU *et al.*, 2015). Além disso, não existe uma solução única para todas as empresas, portanto pode se tornar um investimento arriscado que demanda muito tempo, esforço e orçamento (ACHARGUI; ZAOUIA, 2016).

2.2 BENEFÍCIOS DO SAAS NAS EMPRESAS

O SaaS é um modelo de entrega amplo e inovador que oferece um conjunto de benefícios ao cliente e ao provedor. Com ele as empresas e organizações não precisam instalar *software* em seus ambientes internos, pois o modelo é implantado utilizando Computação em Nuvem e oferece acesso remoto por meio de uma estrutura cliente/servidor, de acordo com a contratação dos serviços e as necessidades do cliente (ALEEM *et al.*, 2018). A plataforma é composta por um conjunto de funções básicas, dentre elas estão (BERNDT *et al.*, 2012):

- Autenticação
- Gerenciamento de usuários
- Controle de autorização
- Armazenamento e comunicação segura de dados
- Interface flexível para aplicativos de terceiros
- Ferramentas de visualização
- Conexão de aplicativos móveis e da web

Normalmente um serviço em nuvem é construído com base em princípios fundamentais, como capacidade de descoberta, alcançabilidade, economia, escalabilidade e capacidade de suporte. O desenvolvimento de uma plataforma SaaS deve incluir atividades de pesquisa e desenvolvimento para cobrir mudanças nas necessidades dos clientes, controle rígido sobre operações de segurança, desempenho, registro e monitoramento, automação de implantação e robustez para atender a todas as necessidades (VIDHYALAKSHMI; KUMAR, 2014).

O sucesso de um modelo SaaS depende muito da maneira como o *design* é construído, espera-se que ele seja ágil, dinâmico e altamente flexível. O *design* do SaaS difere de um aplicativo tradicional baseado em *web*, pois não se trata apenas do desenvolvimento de um produto, mas também da execução e manutenção dos serviços em denominação do cliente. A escolha de um *design* baseado apenas no julgamento, é um processo altamente cognitivo e tedioso que pode ser bastante propenso a erros. Algumas dessas características abaixo são interessantes para a definição de um bom *design* nas aplicações de SaaS (VIDHYALAKSHMI; KUMAR, 2014):

- *Design* de interface configurável
- Infraestrutura e plataformas distribuídas
- Interface de usuário independente do dispositivo
- Identificação e representação de serviço através do uso de APIs
- Atualizações rápidas

Um produto SaaS com alcance global deve ser personalizado para fornecer o mesmo tipo de serviço aos diferentes tipos de clientes. Uma ampla base de clientes com diferentes especificações de requisitos eleva ainda mais a complexidade. Compreender e solucionar esses fatores no processo de desenvolvimento tem um impacto positivo, resultando em um produto bem-sucedido e benéfico para os clientes (ALEEM *et al.*, 2018).

2.3 COMBINAÇÃO DE ERP COM SAAS

Um sistema ERP fornecido por SaaS, usa as vantagens da computação em nuvem para oferecer uma abordagem diferenciada. A migração do ERP local para o ERP em nuvem, ajuda as organizações a gerenciar os seus custos com eficiência, melhorando as suas operações e eliminando soluções inflexíveis. Além disso, oferece a possibilidade de escolha do provedor que melhor se adapta às suas necessidades. Apresenta-se na Tabela 1 um comparativo entre o ERP local e o ERP em nuvem (SAA *et al.*, 2017).

Tabela 1 – Comparativo do ERP Local com o ERP na Nuvem

Descrição	ERP Local	ERP na Nuvem
Dimensão da implementação	Grande	Pequeno para médio
Complexidade da solução	Alta	Baixa
Custos de capital	Alto	Baixo
Custos operacionais	Baixo para médio	Médio
Tempo de implementação	1 à 3 anos	4 à 8 meses

Fonte: Saa *et al.* (2017)

Na prática, a principal característica de uma implementação de ERP-SaaS é que o acesso ao sistema ocorre pela internet e a infraestrutura adota um modelo de pagamento por uso. Os aplicativos e os dados da organização são controlados pelo provedor do SaaS e o sistema ERP é oferecido como um produto "pronto para uso"por uma taxa de assinatura mensal. Abaixo descreve-se algumas vantagens (SAA *et al.*, 2017):

- Fornece o uso remoto de sistemas ERPs à organizações que não têm condições de configurá-lo localmente.
- Economiza gastos em infraestrutura, *softwares*, manutenção e atualização.
- Reduz a equipe necessária para suporte e manutenção.
- Permite uma implantação mais rápida e com menos esforços.
- Oferece melhor escalabilidade.
- Favorece a mobilidade.

Visando esses benefícios, uma implementação de ERP-SaaS fornece um conjunto funcionalidades para as organizações. Uma dessas funcionalidades é o monitoramento contínuo de pessoas, objetos e demandas internas. Isso é altamente desejável para garantir os cuidados e ampliar a capacidade de compreensão das necessidades de negócio (ALBAHRI *et al.*, 2018).

2.4 SERVIÇOS INTERNOS BASEADOS EM LOCALIZAÇÃO

A Internet das Coisas - IoT continua transformando o setor de saúde, aumentando a eficiência, reduzindo os custos e direcionando o foco à um melhor atendimento ao paciente. Com um sistema inteligente é possível obter um nível sem precedentes de dados críticos e em tempo real. Os dados são capturados e analisados pelo sistema para aumentar a eficiência, manter a conformidade e ajudar o setor a avançar em pesquisa, gerenciamento e atendimento. Os especialistas em tecnologia consideram a IoT como um passo importante para futuros empreendimentos (JISHA; PHILIP, 2016).

Por meio da IoT, existem instituições de saúde que incorporam tecnologias para controle em tempo real de inventários, suprimentos, equipamentos, equipes e pacientes, através de dispositivos de rastreamento. Essa incorporação ascende algumas preocupações direcionadas a custos de implementação, segurança, privacidade, escalabilidade, robustez, precisão e eficiência energética. Mas considera-se que essa tecnologia possa trazer grandes benefícios, melhorando significativamente o desempenho financeiro, aprimorando o suporte à decisão e os fluxos de trabalho, trazendo eficácia ao gerenciamento da cadeia de suprimentos, prevenindo roubos e otimizando a distribuição e localização de equipamentos (COUSTASSE; TOMBLIN; SLACK, 2013; FRISCH, 2019).

Um Sistema de Localização em Tempo Real - RTLS é uma estrutura de IoT que consiste em uma rede de dispositivos projetados internamente para localizar pessoas e objetos em ambientes fechados usando sinais e ondas de rádio (KANAN; ELHASSAN, 2015). O objetivo desse serviço é fornecer as mesmas funcionalidades de um Sistema de Posicionamento Global - GPS, tendo que esta é uma tecnologia de rastreamento e monitoramento bastante utilizada em áreas externas, não sendo aplicável em áreas internas, devido a imprecisão e atenuação do sinal por interferência dos materiais de construção e objetos que estejam dentro do ambiente (LIN *et al.*, 2015).

A avaliação dos custos da implantação de um serviço de localização leva em consideração alguns fatores: o tempo necessário para a implementação e administração do projeto, as despesas com a adaptação do ambiente e as configurações da infraestrutura, o salário do pessoal de suporte e as despesas essenciais para manter o sistema funcionando corretamente (MAUTZ, 2012).

2.4.1 Tecnologias de Localização Interna

A capacidade de localização em ambiente fechado é um desafio substancial e muitas aplicações aguardam soluções técnicas satisfatórias. O bom desempenho de tecnologias com esse intuito tem o potencial de criar oportunidades sem precedentes para as empresas (MAUTZ, 2012). Muitas tecnologias de localização foram introduzidas ao longo dos anos e hoje utilizam diversas técnicas de rastreamento, classificando-se nas categorias de longa, média e curta distância (KANAN; ELHASSAN, 2015). Uma apresentação rápida dessas tecnologias pode ser visto na Tabela 2.

Tabela 2 – Visão Geral das Tecnologias de Localização

Tecnologia	Acurácia	Custo / Potência
WLAN	1m - 5m	Alto
RFID	10cm – 1m	Médio
Bluetooth	20cm - 5m	Baixo
Ultra Banda Larga	50cm - 1m	Baixo
Infravermelho	1m - 2m	Alto
QR Code	1m	Baixo
Visão	Relativo ao Sensor	Baixo
Campo Magnético	Relativo ao Sensor	Baixo
Ultrassom	10cm - 1m	Baixo

Fonte: Hassanien e Gaber (2017)

Independente da variedade de tecnologias e suas respectivas características, o mercado possui exigências para atender a necessidades específicas. A tecnologia incorporada deve ser adequadamente de baixo custo, baixa energia, baixa necessidade de manutenção e exigir o mínimo de infraestrutura. A sua implementação deve preocupar-se com questões como robustez, segurança, privacidade e confiabilidade, seguindo os parâmetros de requisitos de precisão, cobertura, integridade, disponibilidade, taxa de atualização, latência e saída de dados (MAUTZ, 2012).

Os requisitos para cada tipo de ambiente devem ser analisados separadamente, a fim de fornecer a solução mais adequada para cada caso. Deve-se avaliar os parâmetros de desempenho de cada tecnologia e combiná-los com os requisitos que devem ser descritos precisamente, baseando-se em uma análise de mercado em que os parâmetros dos requisitos precisam ser cuidadosamente ponderados entre si (MAUTZ, 2012).

2.4.1.1 Redes Locais sem Fio - WLAN

Nos sistemas baseados em WLAN, os transmissores usam *tags* para enviar pacotes simples à diferentes pontos de acesso em uma área específica. O tempo e a força dessas leituras em cada ponto de acesso são analisados por um *software* que faz uso de algoritmos para calcular a posição das *tags*. Essas informações são coletadas e enviadas para a nuvem. A implementação dessa tecnologia para localização costuma ser de alto custo e possui baixa acurácia, mas fornecem um nível alto de precisão, no entanto, para adquirir esse nível de precisão, deve-se haver no mínimo três pontos de acesso para uma estimativa de localização aceitável (HAMEED; AHMED, 2018).

2.4.1.2 Identificação por Radiofrequência - RFID

Há também os sistemas que utilizam Identificação por Radiofrequência. Nessa tecnologia a transmissão eletromagnética é usada para transferir os dados que são recebidos e armazenados por qualquer circuito receptor compatível. Os transmissores são compostos por *tags* de RFID e enviam pacotes para os pontos de acesso disponível, esses pontos receptores utilizam protocolos e frequências de rádio predefinidas para ler os dados emitidos e estimar a localização de cada *tag* dentro do ambiente. O custo para se obter as *tags* e os receptores é considerado baixo, mas devido às limitações de alcance é necessário um número massivo deles e um posicionamento adequado para se obter uma precisão aceitável em um ambiente mais amplo. Acredita-se que essa tecnologia ainda não está madura o suficiente para ser usada de maneira única em um sistema robusto, seu benefício é maior quando utilizado em paralelo com outras tecnologias (HAMEED; AHMED, 2018).

2.4.1.3 *Bluetooth* - BL / BLE

Os sistemas que utilizam *Bluetooth* também são capazes de detectar e localizar objetos em um ambiente fechado. As aplicações usam *beacons* e *tags* como transmissores e enviam dados continuamente para diferentes dispositivos receptores. Esses dispositivos são responsáveis por enviar a identificação dos transmissores e a intensidade do sinal para um respectivo *software* que estima a posição dos transmissores. Esses transmissores são relativamente baratos e simples de implementar, mas a intensidade do sinal pode sofrer alterações se for bloqueado por algum objeto e pode sofrer interferências em uma área que contenha uma faixa alta de sinais de WiFi que estejam no mesmo intervalo de frequência (HAMEED; AHMED, 2018).

2.4.1.4 Banda Ultralarga - UWB

A Banda Ultralarga é uma tecnologia de rádio relativamente nova e promissora que pode ser usada para meios de localização interna. Durante a preparação do ambiente os receptores devem ser espalhados e suas posições registradas com horários sincronizados. Isso vai permitir que os transmissores possam enviam pulsos curtos para os receptores ao seu alcance e a aplicação possa estima a localização de cada transmissor. Essa tecnologia possui uma alta velocidade de transmissão e não causa interferência com outros sinais de rádio, destaca-se também por fornecer uma ótima precisão, acurácia, capacidade de atravessar obstáculos físicos e uma margem de erro muito baixa. No entanto, embora seja uma tecnologia de baixo custo, ela pode gerar custos altos de implantação devido o seu curto alcance, o que consequentemente amplia a necessidade de se obter um número grande de transmissores e receptores em um ambiente de atuação amplo (KOK; HOL; SCHÖN, 2015).

2.4.1.5 Radiação Infravermelha - IR

O Infravermelho é um tipo de onda eletromagnética de frequência menor que a luz visível. Os sistemas clássicos dessa tecnologia fazem uso de dispositivos de identificação eletrônica responsáveis por transmitir um sinal de infravermelho exclusivo a cada 10 segundos para um receptor posicionado no ambiente. Em seguida o receptor torna-se responsável por transmitir os dados para uma base de dados. Os sistemas de localização baseados nessa tecnologia estão entre os mais comuns, pois são baratos, fáceis de implementar e fornecem medidas precisas. No entanto, eles consomem bastante energia e são facilmente bloqueados por construções e objetos, além disso, a sua principal desvantagem está na segurança e na privacidade das informações (HAMEED; AHMED, 2018; KANAN; ELHASSAN, 2015).

2.4.1.6 Código de Resposta Rápida - *QR Code*

O *QR Code* é um modelo de código de barras bidimensional que possui recursos significativos, cuja vantagem é a capacidade de armazenar uma quantidade expressiva de informações codificadas de maneira segura, eficaz e sem custos. Os dados são anexados como *tags* em pontos estratégicos e distribuídos ao longo do ambiente, podendo guardar coordenadas de localização para fins de rastreamento, permitindo que os usuários descubram a sua posição ao escanear a *tag* com uma câmera. A principal desvantagem é que essa tecnologia funciona de maneira mais adequada aos dispositivos móveis como leitores do *QR Code*, o que impede que o rastreamento seja realizado automaticamente

e em tempo real por um sistema implantado no ambiente. Uma alternativa é utilizar câmeras espalhadas pelo ambiente e técnicas de processamento de imagem para reconhecimento dos *QR Codes*, mas essa técnica aumenta consideravelmente os custos de implantação, sendo necessário uma avaliação para saber se é compensável (ALGHAMDI; SCHYNDEL; ALAHMADI, 2013; ILKOVIČOVÁ; ERDÉLYI; KOPÁČIK, 2014; LI; HUANG, 2018).

2.4.1.7 Comunicação por Luz Visível - VLC

Em um sistema de comunicação por luz visível os dados são transmitidos com base na modulação de intensidade óptica. Os dispositivos emissores de luz (LEDs) são utilizados como referências de posicionamento seguindo um nível de iluminação predeterminado. Os LEDs emitem informações exclusivas de suas coordenadas e os dispositivos receptores decodificam essas informações, utilizando um banco de dados para determinar a sua auto-localização. Essa solução é considerada promissora, com alta precisão de posicionamento, eficiência energética, longa vida útil e infraestrutura de baixo custo. Por outro lado, os LEDs não conseguem ultrapassar paredes e objetos, portanto é necessário manter a comunicação livre de interferências visuais e o ambiente precisa ficar sempre iluminado para o funcionamento adequado dessa tecnologia. (NAKAZAWA *et al.*, 2013; XU; GONG; XU, 2018).

2.4.1.8 Campos Magnéticos

Outra estratégia proposta pela literatura são os sistemas baseados no campo magnético da terra. Inicialmente utilizam-se as anomalias e distúrbios locais para mapear o ambiente em que o sistema será implantado, em seguida um conjunto de posições são definidas pela primeira vez no ambiente e registradas em um banco de dados. Os transmissões são equipados com sensores magnéticos e uma unidade de processamento, a medição é comparada com as posições registradas no banco de dados e uma estimativa da localização de cada transmissor é obtida. A vantagem é que essa tecnologia oferece alta precisão, os sensores são baratos, compactos, robustos e podem operar mesmo em casos de obstrução da linha de visão do sinal por objetos não metálicos. A desvantagem é que os sistemas normalmente exigem um processo de mapeamento inicial complicado, consequentemente o desempenho e a precisão estão diretamente relacionados ao número de sensores e a robustez desse mapeamento (PASKU *et al.*, 2017; KANAN; ELHASSAN, 2015).

2.4.1.9 Sinais Ultrassônicos

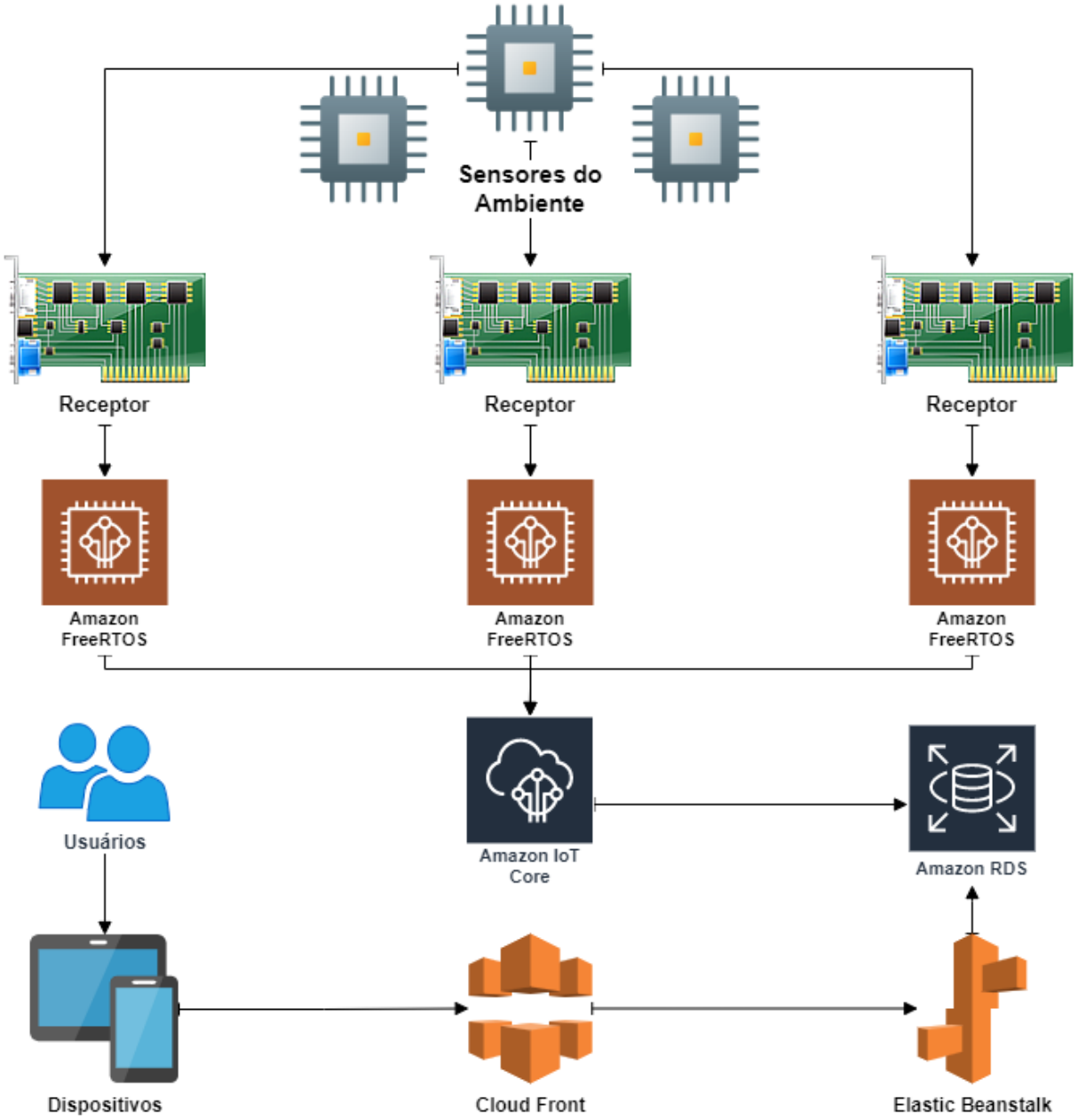
Os sistemas de sinais ultrassônicos consiste em dispositivos que utilizam emissão de ondas sonoras de alta frequência. Os sistemas implementam uma rede de sensores ultrassônicos espalhados pelo ambiente com coordenadas estratégicas, fixas e conhecidas no espaço interno. Os sensores receptam os dados dos transmissores e estimam as suas respectivas localizações de acordo com o tempo de chegada das ondas sonoras. Um sistema com essa tecnologia tem muitas vantagens, como privacidade, segurança e a velocidade de propagação lenta, o que permite que os sensores sejam simples e econômicos. No entanto, o seu sinal pode sofrer interferências devido a incapacidade de penetração em objetos e a sua precisão pode ser afetada por outros tipos de ruídos, o que amplia a necessidade de se obter muitos sensores e causa um aumento nos custos de implementação de acordo com o configuração do ambiente (QI; LIU, 2017).

3 OBJETIVOS

Em conformidade com as abordagens estudadas sobre tecnologias de localização, o objetivo deste trabalho é apresentar o desenvolvimento de um sistema, chamado de Movvo, que será responsável pela gestão de equipamentos hospitalares, localização em tempo real e monitoramento das suas demandas. Ele apresenta-se como uma opção viável para implantação em organizações de saúde por meio de módulos de modelos ERP-SaaS. Nós construímos e apresentamos na Figura 2 uma arquitetura que pode ser usada como fundamento para a construção da infraestrutura de localização, utilizando como base as tecnologias e funcionalidades fornecidas pela Amazon AWS.

Os sensores do ambiente correspondem a rede de sensores projetada de acordo com as tecnologias de localização interna, descritos na Seção 2.4.1, onde cada sensor é inserido em um respectivo equipamento. Os receptores são responsáveis por detectar esses sensores e registrar as informações de localização de cada equipamento em servidores na nuvem. Os dispositivos farão uso do sistema de gestão e poderão acompanhar a área e o setor onde todos os equipamentos estão localizados dentro da organização de saúde. Vamos presumir que toda a infraestrutura já tenha sido implantada e esteja em execução com dados de monitoramento sendo registrados em tempo real. Trabalharemos apenas no módulo responsável por receber as ações do usuário, acessar a base de dados e processar suas respectivas instruções.

Figura 2 – Arquitetura do Sistema de Localização Interna



Fonte: Elaborada pelo autor, 2022

4 ETAPAS METODOLÓGICAS

Essa pesquisa possui uma abordagem qualitativa, com aspectos descritivos, fundamento positivista e de natureza aplicada (prática). Foi concluído no ano de 2022 para a dissertação do programa de Mestrado Profissional em Ciência e Tecnologia em Saúde, oferecido pelo Departamento de Computação da Universidade Estadual da Paraíba (UEPB), Campus 1, Campina Grande - Paraíba.

4.1 PROCEDIMENTOS REALIZADOS

Para a escrita desse trabalho realizou-se um revisão bibliográfica com os termos de ERP, SaaS, ERP-SaaS e *Real-Time Locating System* em várias bases de artigos e publicações, tendo o *PubMed*, *Science Direct* e *IEEE Explorer* como bases principais. Foi considerado artigos, revistas, jornais, monografias, dissertações, entre outros tipos de pesquisas, somente no idioma inglês e com data de publicação entre 2015 e 2021. Não houve definição de critérios de inclusão, exclusão e aceitação, mas só foi extraído os trabalhos destacáveis e que tinham relação direta com a nossa pesquisa.

4.2 FERRAMENTAS UTILIZADAS

Utilizamos a ferramenta *Webstorm IDE* e os recursos fornecidos pela plataforma *Angular* com as linguagens de programação *HTML*, *SASS* e *TypeScript* para desenvolver toda a parte visual do projeto. Utilizamos também a ferramenta *Inttelij IDEA* com a linguagem programação *Java* e recursos do *Spring Boot* para desenvolver a autenticação e os meios de comunicação com o banco de dados. E por fim, utilizamos diversos recursos da *Amazon Web Services - AWS* relacionados a hospedagem pública do projeto, hospedagem de dados, servidores e controle de acesso de usuários. Apresentaremos essas e outras informações técnicas de forma mais detalhada na seção 5.1.

4.3 PLANO DE ATIVIDADES

Na Figura 3 apresentamos o plano de atividades seguido para a conclusão desse projeto, especificamos por levantamento bibliográfico e desenvolvimento do sistema.

Figura 3 – Diagrama de Etapas Metodológicas

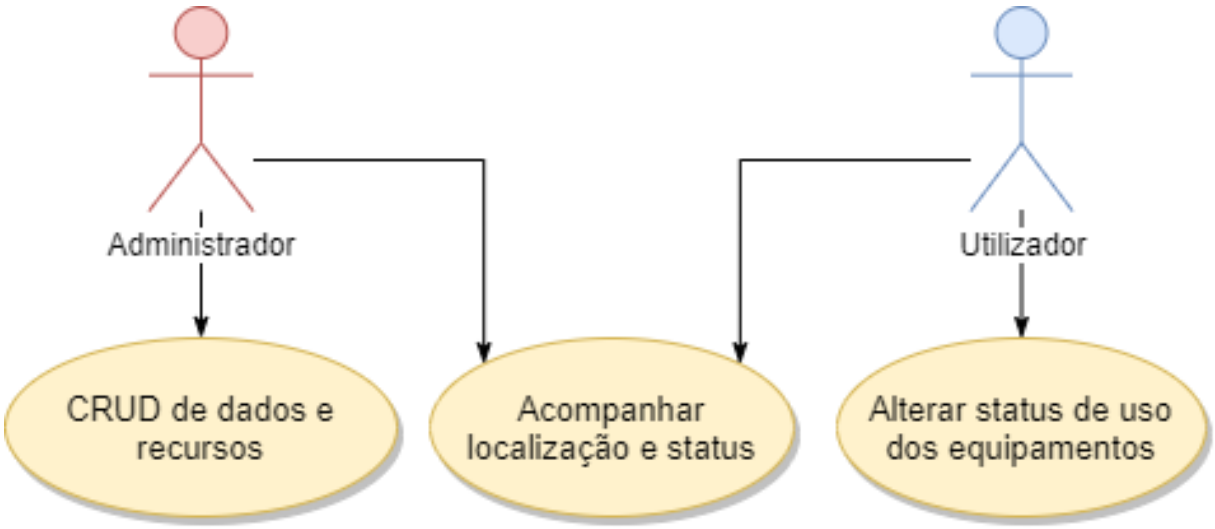
Fonte: Elaborada pelo autor, 2022

5 SOLUÇÃO PROPOSTA

Nesta seção vamos apresentar uma solução que será usada como módulo de um ERP-SaaS, tendo o propósito de gerenciar e monitorar equipamentos médicos, sejam eles do centro cirúrgico, pronto atendimento, terapia intensiva, ambulatório, higienização, dentre outros ambientes de uma organização de saúde. Devemos garantir que o Movvo tenha uma estrutura flexível aos objetivos de cada instituição para otimizar o fluxo de trabalho nas mais diversas áreas de Cirurgia, Odontologia, Enfermagem, Farmácia, Fisioterapia, Nutrição e todos os outros setores desejados pela instituição.

O sistema terá dois tipos de usuários e seguirá uma hierarquia de acordo com os casos de uso apresentados na Figura 4.

Figura 4 – Casos de Uso do Sistema



Fonte: Elaborada pelo autor, 2022

O administrador terá a responsabilidade de garantir que todos os ambientes, equipamentos, transmissores e receptores da instituição estejam cadastrados no sistema quando forem inseridos no escopo de localização. Ele terá a função de manter os dados atualizados com todas as informações necessárias para o bom funcionamento do sistema, acompanhando também os status de comunicação dos transmissores e receptores, além do período de manutenção dos equipamentos cadastrados. Por fim, o utilizador estará apto para fazer uso de todos os recursos disponíveis, com o dever de sempre atualizar o status dos equipamentos que estejam sendo utilizados.

A primeira tela do Movvo é a tela de autenticação do usuário, apresentada na Figura 5. A autenticação é importante para garantir a segurança dos dados, pois somente

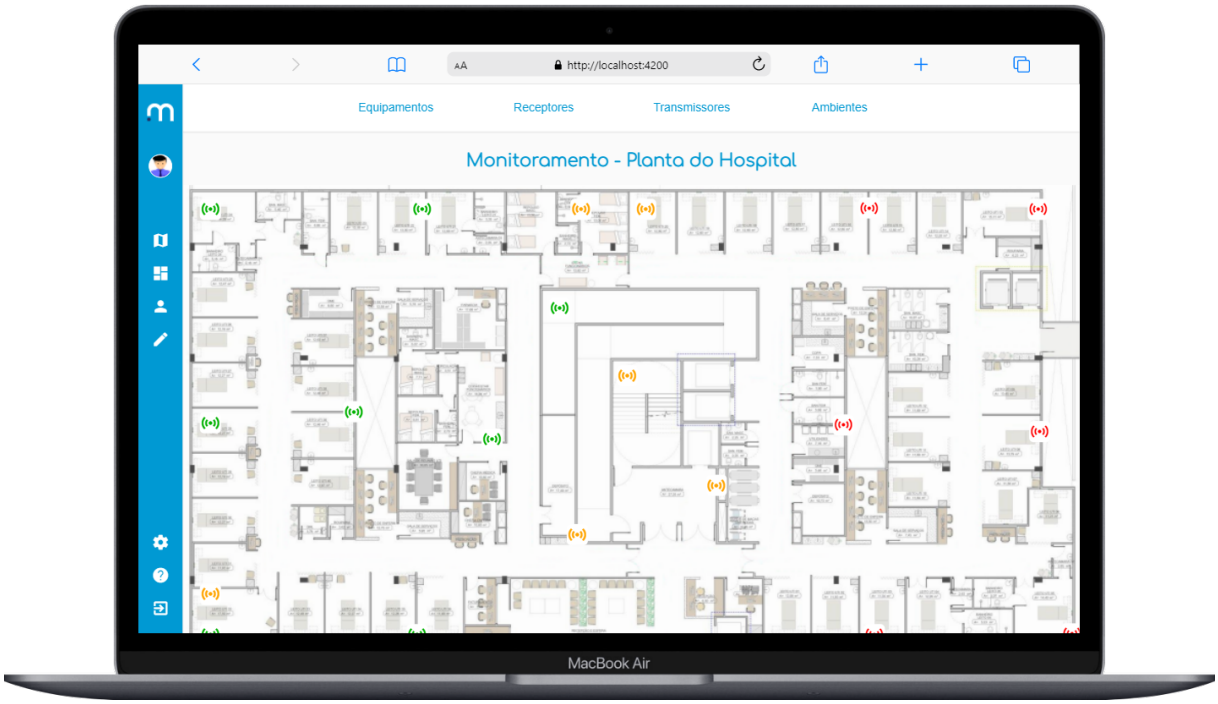
pessoas autorizadas estarão aptas à utilizar o sistema. É por meio dessa funcionalidade que o sistema irá entender quem são os administradores e utilizadores para fornecer suas devidas permissões.

Figura 5 – Tela de Autenticação

Fonte: Elaborada pelo autor, 2022

A Figura 6 contém a primeira tela após a autenticação do usuário. Ela é a tela de monitoramento que mostrará a planta baixa da instituição, a localização dos receptores e o seu *status* em tempo real. Para fazer esse mapeamento levamos em consideração as dimensões reais do local, as dimensões da imagem e fizemos a conversão da localização dos receptores baseando-se em escalas entre 0 e 1.

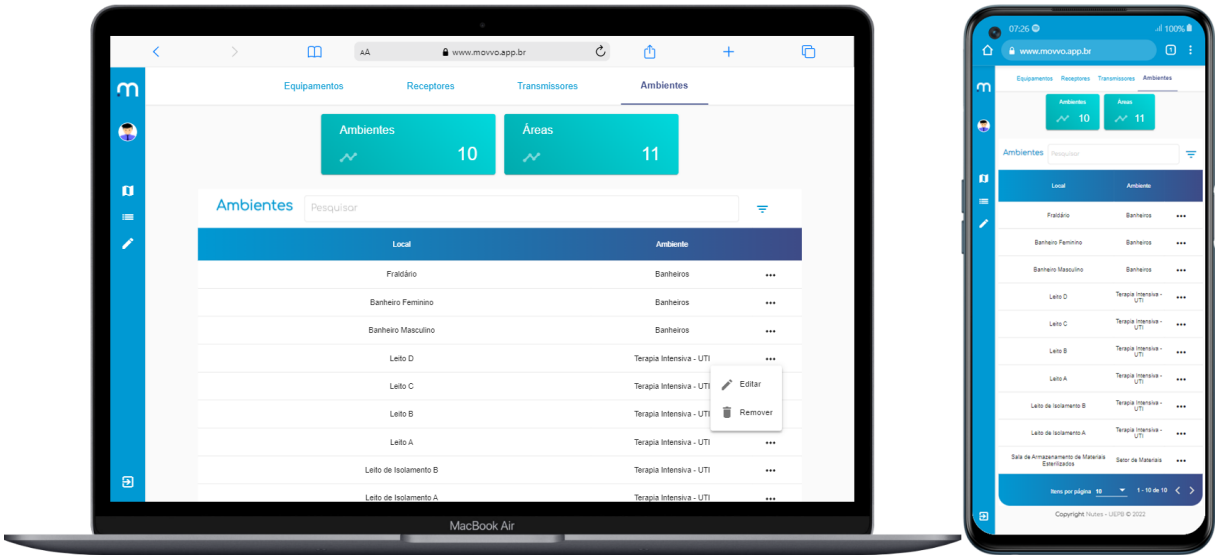
Figura 6 – Tela de Monitoramento



Fonte: Elaborada pelo autor, 2022

A Figura 7 contém a tela de apresentação dos locais e ambientes cadastrados, com um campo de busca para encontrar registros rapidamente e um botão de exibir/ocultar colunas da tabela. Através do botão de ação alinhado à direita de cada item, o administrador poderá editar e remover ambientes quando necessário.

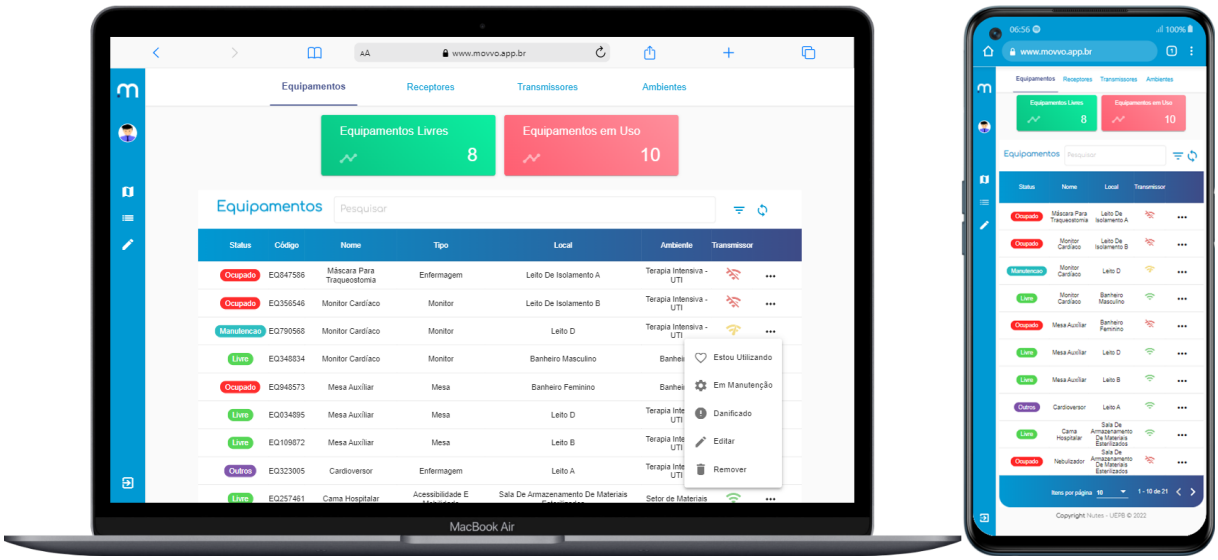
Figura 7 – Tela de Ambientes



Fonte: Elaborada pelo autor, 2022

Na Figura 8 apresenta-se a tela de equipamentos cadastrados. Ela segue o mesmo objetivo da anterior, mas aqui os usuários utilizadores do sistema irão identificar os equipamentos que estão com *status* livres/em uso e sua localização atual, podendo modificar esse *status* quando desejar. Essa tela contém também a busca rápida de equipamentos para facilitar a procura por parte do usuário. Ao selecionar um registro, um diálogo igual a Figura 9 será apresentado com todos os dados detalhados.

Figura 8 – Tela de Equipamentos



Fonte: Elaborada pelo autor, 2022

Figura 9 – Dados do Equipamento

Informações de Cadastro do Equipamento

- Tipo de Equipamento
- Nome de Identificação
- Código de Rastreamento
- Identificação do Transmissor
- Data de Fabricação
- Período do Intervalo de Manutenção
- Data da Última Manutenção
- Observações
- Última Atualização

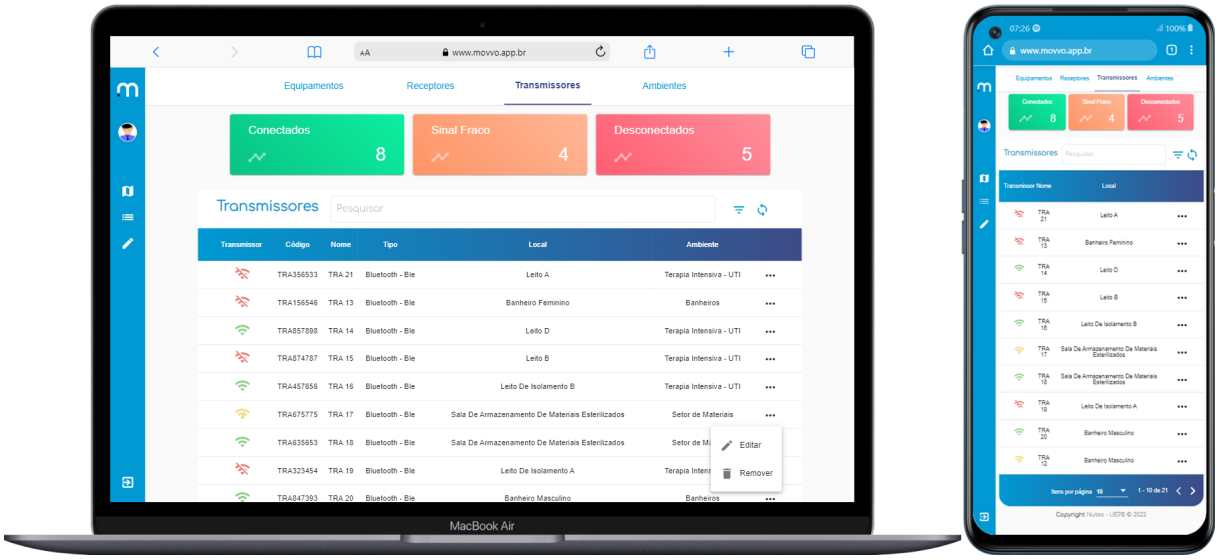
Informações Atualizadas Pelo Monitoramento

- Status de Uso
- Local Atual
- Área do Local Atual

Fonte: Elaborada pelo autor, 2022

A tela de transmissores segue o mesmo modelo da tela de equipamentos, como apresentado nas Figuras 10 e 11. Os transmissores são identificadores ligados aos equipamentos e enviam informações de localização constantemente para os receptores de acordo com a sua tecnologia de rastreamento. Na tela é possível saber quais são os transmissores que estão conectados com sinal forte, fraco e sem sinal.

Figura 10 – Tela de Transmissores



Fonte: Elaborada pelo autor, 2022

Figura 11 – Dados do Transmissor

Informações de Cadastro do Transmissor

- Tipo de Transmissor
- Nome de Identificação
- Código de Rastreamento
- Identificação do Equipamento Vinculado
- Observações

Informações Atualizadas Pelo Monitoramento

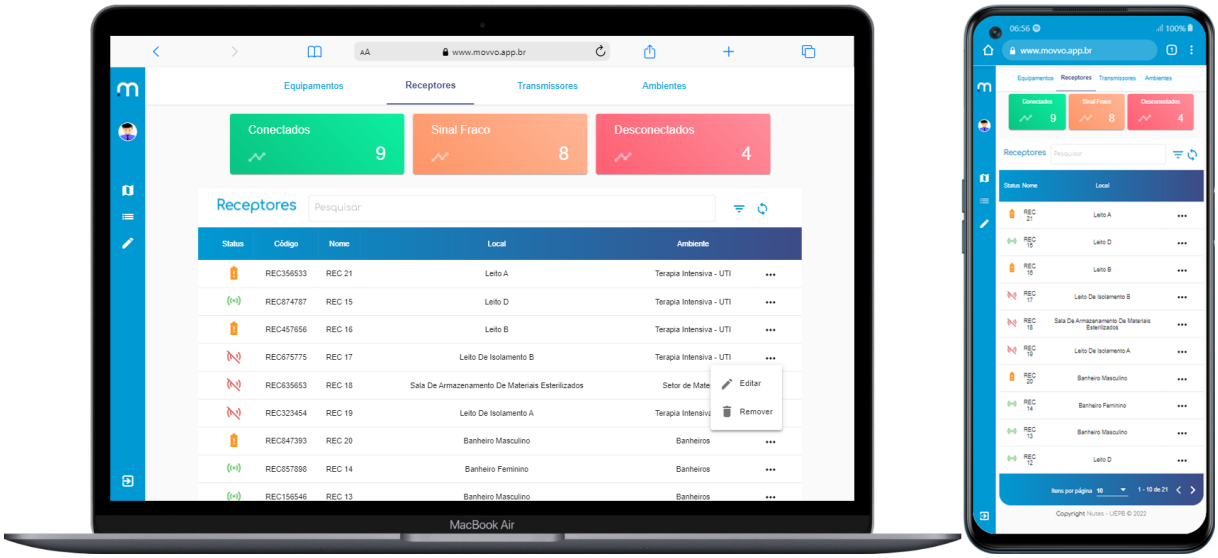
- Status de Bateria
- Status de Conexão
- Local Atual
- Área do Local Atual
- Data da última Conexão

Fonte: Elaborada pelo autor, 2022

A tela de receptores segue o mesmo modelo da tela de transmissores e equipamentos, incluindo a mesma restrição de cadastro e atualização dos dados. Apresentamos suas informações nas Figuras 12 e 13. Os receptores são dispositivos que ficam localizados

em locais espalhados pela instituição e recebem constantemente os dados de localização dos transmissores. Na tela é possível saber quais são os receptores que estão conectados com sinal forte, bateria fraca e sem sinal.

Figura 12 – Tela de Receptores



Fonte: Elaborada pelo autor, 2022

Figura 13 – Dados do Receptor

Informações de Cadastro do Receptor

- Tipo de Receptor
- Nome de Identificação
- Código de Rastreamento
- Observações

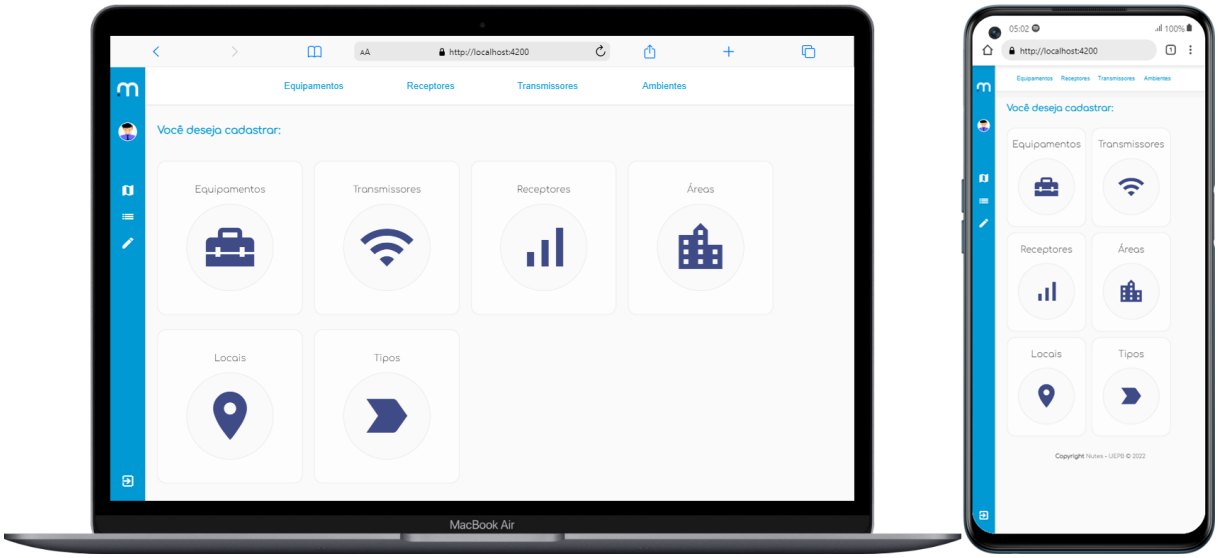
Informações Atualizadas Pelo Monitoramento

- Status de Bateria
- Status de Conexão
- Local Atual
- Área do Local Atual
- Data da última conexão

Fonte: Elaborada pelo autor, 2022

Nas figuras seguintes será apresentado o fluxo de cadastro e atualização das informações que alimentam o sistema. A tela com as opções de cadastro é apresentada na Figura 14, onde é possível escolher se deseja cadastrar equipamentos, transmissores, receptores, locais, áreas e tipos de dados.

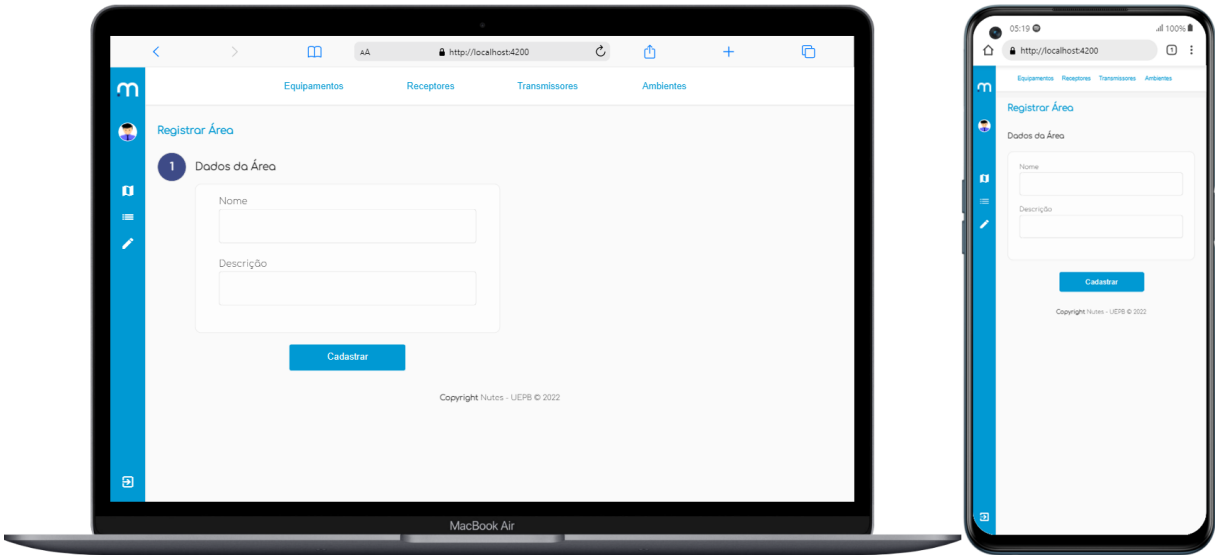
Figura 14 – Tela do Menu de Cadastro



Fonte: Elaborada pelo autor, 2022

A tela de cadastro de áreas (Figura 15) é a mais simples de todas, pois nela o usuário só precisa inserir um nome para identificação da área e uma descrição.

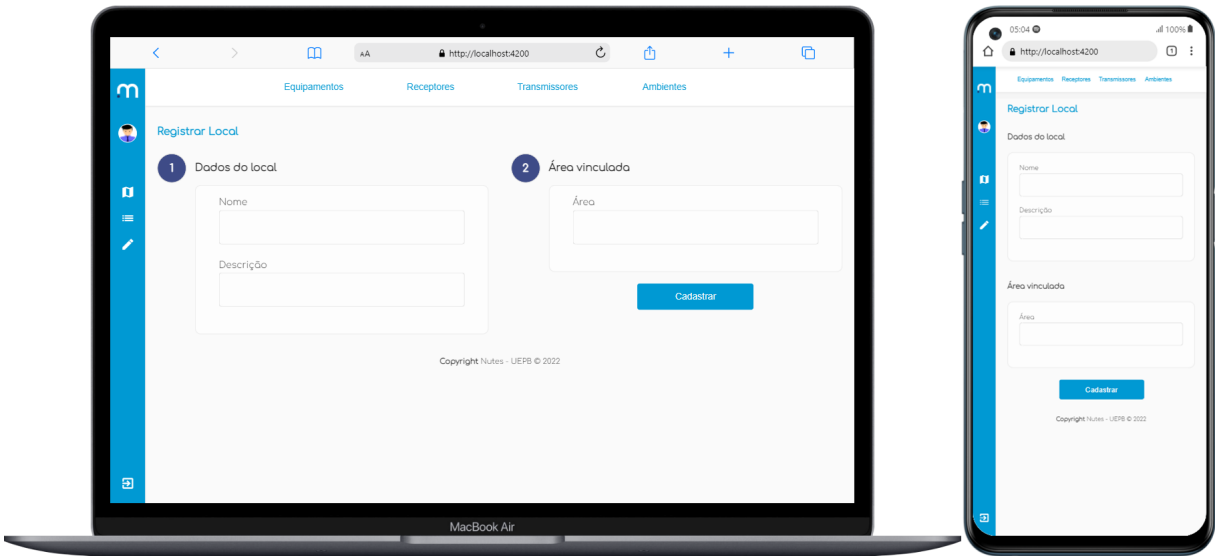
Figura 15 – Tela de Cadastro de Áreas



Fonte: Elaborada pelo autor, 2022

A tela de cadastro de locais (Figura 16) possui as mesmas características da tela anterior, mas nela é necessário que o usuário informe a qual área pertence o local que está sendo cadastrado.

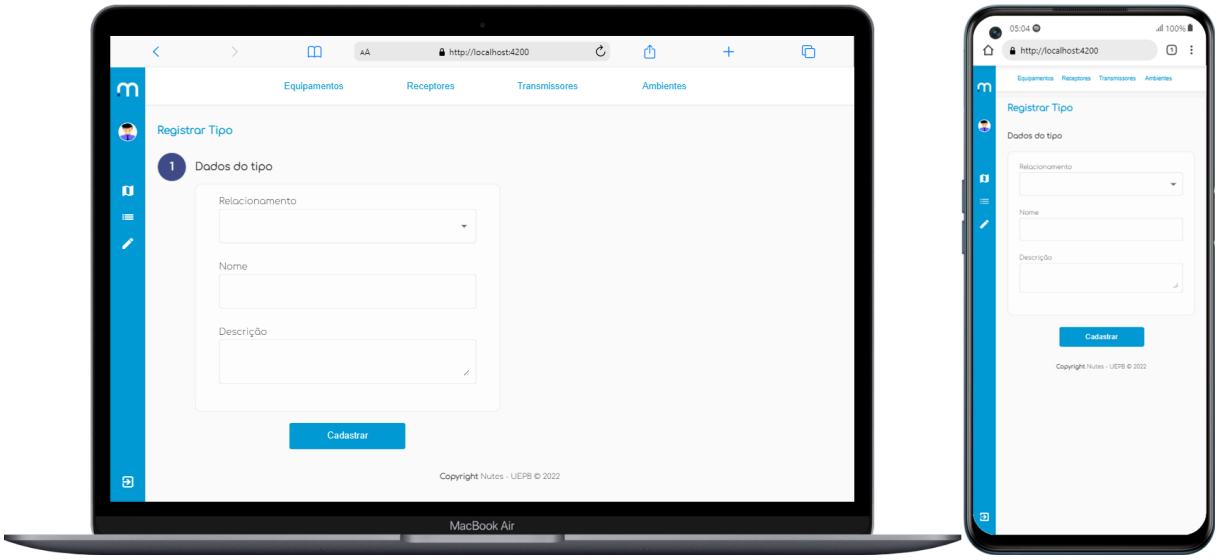
Figura 16 – Tela de Cadastro de Locais



Fonte: Elaborada pelo autor, 2022

O cadastro dos tipos de dados (Figura 17) deve ser realizado antes próximas opções de cadastro, pois é através dele que o usuário irá cadastrar os tipos de equipamentos, receptores e transmissores, com seu respectivo nome de identificação e descrição.

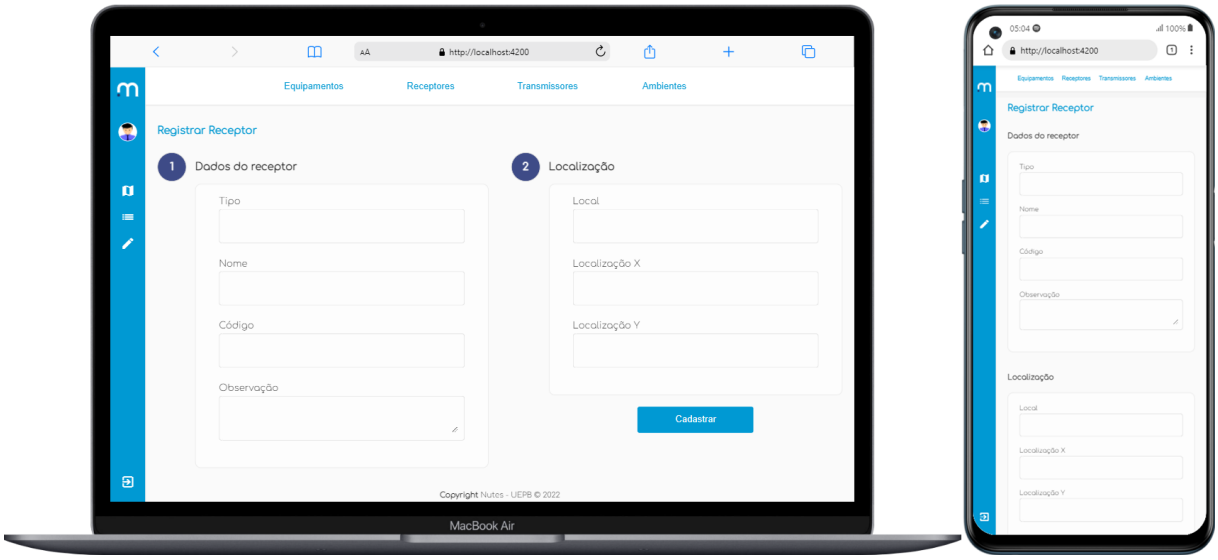
Figura 17 – Tela de Cadastro de Tipos



Fonte: Elaborada pelo autor, 2022

A tela de cadastro de receptores (Figura 18) é uma das mais importantes do sistema. Nela será informado o tipo de conexão do receptor, nome, código de identificação e observação. Será exigido também que o usuário informe em qual local está o receptor e suas coordenadas (X, Y) em escalas entre 0 e 1, caso o usuário deseje que acompanhar esse receptor na tela de monitoramento que contém a planta baixa da organização.

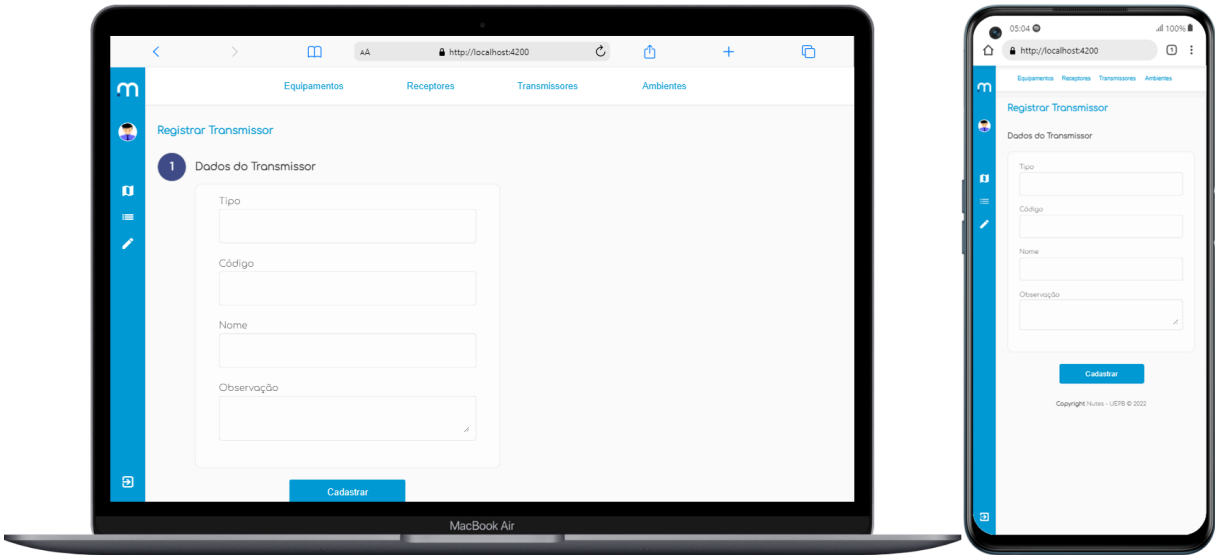
Figura 18 – Tela de Cadastro de Receptores



Fonte: Elaborada pelo autor, 2022

Na tela de cadastro de transmissores (Figura 19) o usuário deverá informar o tipo de conexão, nome, código de identificação e suas observações. Aqui não é necessário informar o equipamento que esse transmissor será relacionado, pois eles poderão ser colocados em estoque para ser relacionados futuramente.

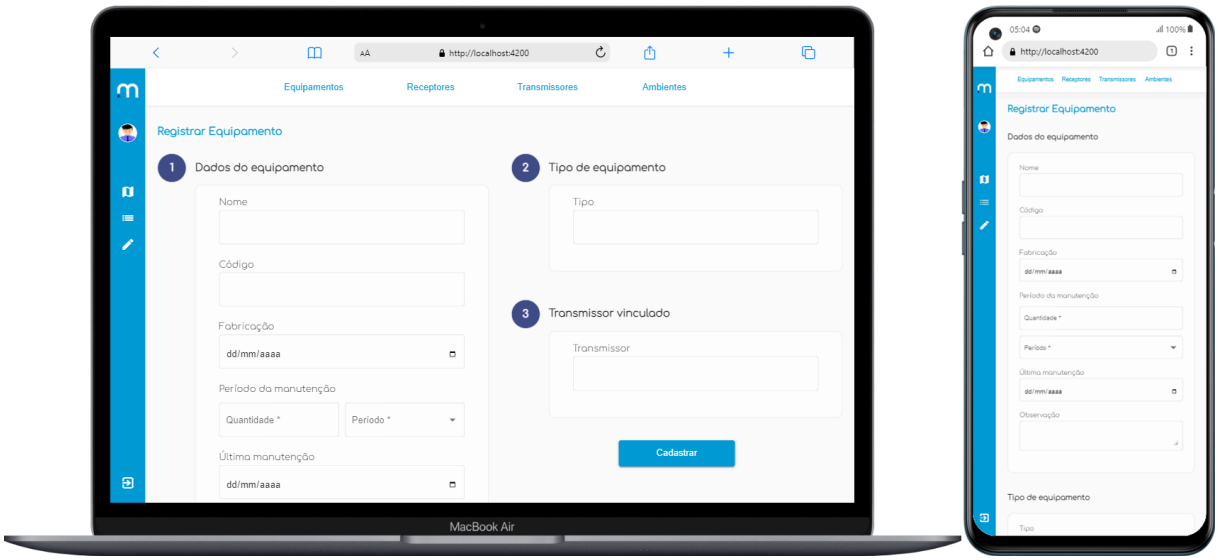
Figura 19 – Tela de Cadastro de Transmissores



Fonte: Elaborada pelo autor, 2022

A tela de cadastro de equipamentos (Figura 20) é a que compõe o maior número de informações de cadastro. Nela será informado o nome, código de identificação e tipo de equipamento, além da data de fabricação, período de manutenção, última data de manutenção e outras observações. Será exigido que o usuário informe qual é o transmissor que será anexado ao equipamento, portanto esse transmissor deverá ser cadastrado primeiro.

Figura 20 – Tela de Cadastro de Equipamentos



Fonte: Elaborada pelo autor, 2022

5.1 INFORMAÇÕES TÉCNICAS E TECNOLOGIAS UTILIZADAS

O Movvo é composto por dois projetos com tecnologias distintas que se complementam e formam uma única aplicação: o projeto de interface do usuário e o projeto de interface da aplicação. Essa sessão contém a descrição de todos os recursos utilizados nos dois projetos e todas as informações técnicas importantes no desenvolvimento e na implantação deles.

5.1.1 Projeto de Interface do Usuário

O projeto de Interface de Usuário (UI) é o ambiente responsável pelos componentes visuais da aplicação e pela interação direta com o usuário. Toda a implementação dos elementos gráficos, configurações relacionadas a responsividade de tela e o suporte a plataformas diversificadas é realizada nessa interface. Através dela o usuário consegue gerar eventos e receber respostas para esses eventos. Nesse ambiente utilizamos o *Webstorm v2022.1* para construir a aplicação por meio do kit de desenvolvimento do *Angular v12.2.8* em conjunto com a linguagem de programação *TypeScript v4.6.3*.

O *Angular* é uma tecnologia multiplataforma de código aberto que fornece recursos para a construção de aplicações eficientes e sofisticadas. É uma ferramenta que vem aumentando a sua popularidade nos últimos anos e vem recebendo atualizações constantes que buscam melhores práticas e experiências de desenvolvimento. Junto com o *Angular*, utilizamos vários pacotes para expandir funcionalidades e simplificar o desenvolvimento, inserindo-os no escopo do arquivo de configuração do projeto, chamado *package.json*. Abaixo inserimos uma breve descrição de alguns dos principais pacotes inclusos.

- ***NPM v7.21.1***: Esse é o gerenciador de pacotes que permite baixar, compartilhar, gerenciar e inserir diferentes pacotes de código desenvolvidos na linguagem de programação *JavaScript*. Ele utilizado para incorporar todos os pacotes necessários para o desenvolvimento.
- ***RXJS v7.3.0***: Essa é uma biblioteca de extensões reativas para o *JavaScript*. É através dela que implementamos os métodos assíncronos da aplicação.
- ***@angular/material v12.2.8***: Esse pacote traz uma coleção de componentes visuais e ferramentas que fornecem uma aparência bonita e o suporte às melhores práticas de *design*. Esse conjunto de componentes é conhecido como *Material Design* e foi desenvolvido pela Google para padronizar as suas interfaces gráficas. É um dos principais pacotes do projeto, pois é utilizado em todas as páginas.

- ***@angular/flex-layout v12.2.8***: O *FlexLayout* facilita o desenvolvimento de páginas responsivas, pois possui um conjunto de métodos para implementação de *layouts* em *grid* com variações de ordenação, alinhamento, comportamento e organização dos componentes.

- ***@angular/service-worker v12.2.8***: Esse pacote possibilita a inclusão do recurso *PWA* no projeto. Com ele o usuário consegue baixar a aplicação através do navegador, passando a ter o comportamento de um aplicativo móvel em seu dispositivo.

No Quadro 1 apresentamos o escopo do arquivo *package.json* com todos os pacotes inseridos no projeto, além de todos os descritos anteriormente.

Quadro 1 – Arquivo de Configuração da Interface do Usuário

```
// package.json
{
  "name": "movvo",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "watch": "ng build --watch --configuration development",
    "test": "ng test",
    "lint": "ng lint",
    "mock": "json-server --watch src/mocks/index.js --port 5210"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^12.2.8",
    "@angular/cdk": "^12.2.8",
    "@angular/common": "^12.2.8",
    "@angular/compiler": "^12.2.8",
    "@angular/core": "^12.2.8",
    "@angular/flex-layout": "^12.0.0-beta.35",
    "@angular/forms": "^12.2.8",
    "@angular/material": "^12.2.8",
    "@angular/material-moment-adapter": "^12.2.8",
    "@angular/platform-browser": "^12.2.8",
    "@angular/platform-browser-dynamic": "^12.2.8",
    "@angular/router": "^12.2.8",
    "@angular/service-worker": "^12.2.8",
    "@fortawesome/angular-fontawesome": "^0.9.0",
```

```
    "@fortawesome/fontawesome-free": "^5.15.4",
    "@fortawesome/fontawesome-svg-core": "^1.2.35",
    "@fortawesome/free-solid-svg-icons": "^5.15.3",
    "json-server": "^0.16.3",
    "ngx-device-detector": "^3.0.0",
    "npm": "^7.21.1",
    "rxjs": "^7.3.0",
    "tooltip-js": "^3.0.0",
    "tslib": "^2.3.1",
    "uuid": "^8.3.2",
    "zone.js": "^0.11.4"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "^12.2.8",
    "@angular/cli": "^12.2.8",
    "@angular/compiler-cli": "^12.2.8",
    "@angular/language-service": "^12.2.8",
    "@fortawesome/fontawesome-free": "^5.15.4",
    "@types/jasmine": "~3.6.0",
    "@types/node": "^12.11.1",
    "@types/uuid": "^8.3.3",
    "jasmine-core": "~3.7.0",
    "karma": "~6.3.0",
    "karma-chrome-launcher": "~3.1.0",
    "karma-coverage": "~2.0.3",
    "karma-jasmine": "~4.0.0",
    "karma-jasmine-html-reporter": "^1.5.0",
    "protractor": "^7.0.0",
    "ts-node": "^10.2.1",
    "tslint": "^6.1.3",
    "typescript": "^4.4.2"
  },
  "browser": {
    "crypto": true
  }
}
```

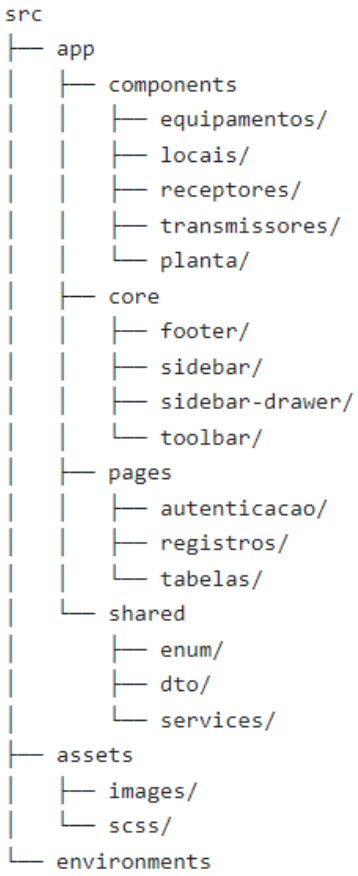
Fonte: Elaborada pelo autor, 2022

5.1.2 Arquitetura do Projeto de Interface do Usuário

A documentação do *Angular* já prevê uma arquitetura bem definida. O conceito de componentização é fundamental em todo o projeto, o que torna o código-fonte mais legível, organizado e reutilizável. Cada página se torna um conjunto componentes, onde cada componente é bem estruturado e possui seus arquivos do tipo *HTML*, *SCSS* e

TypeScript. Além da organização dos módulos, a documentação também traz uma série de conceitos e recomendações de boas práticas de arquitetura (ANGULARDOC, 2022). Na Figura 21 apresentamos a hierarquia de pastas e componentes do projeto.

Figura 21 – Hierarquia de Pastas e Componentes da Interface do Usuário



Fonte: Elaborada pelo autor, 2022

5.1.3 Projeto de Interface de Aplicação

O projeto de Interface de Aplicação (API) é o ambiente responsável por receber os eventos realizados pelo usuário na interface gráfica, processar as requisições de autenticação e interações com o banco de dados, além de toda a parte de codificação da infraestrutura do projeto, e por fim, retornar uma resposta com os dados requisitados ou mensagens específicas com códigos de erro.

O *Spring Boot* é uma tecnologia de código livre que tem o objetivo de facilitar o desenvolvimento de APIs em *Java*, baseado em padrões de projetos. Ele fornece infraestrutura pronta em nível de aplicativo, especialmente injeção de dependências e programação orientada a aspectos, tornando-se adequado à projetos que vão desde pequenos serviços até aplicações mais robustas (BOOT, 2018).

Neste projeto utilizamos a ferramenta *Intellij IDEA v2022.1* para desenvolver o código fonte em *Java v17*, os recursos fornecidos pelo *Spring Boot v2.6.6* e diversos outros módulos inseridos na aplicação. Estes módulos são injetados em um arquivo chamado *pom.xml* dentro do projeto e compilados através do *Apache Maven*, uma ferramenta de automação de compilação para projetos desenvolvidos em *Java*. Abaixo apresentaremos o escopo desse arquivo e uma breve descrição dos principais módulos utilizados neste projeto, de acordo com as suas documentações (SPRINGDOCS, 2021; MAVENREPOSITORY, 2021).

- ***Spring Boot Starter Web v2.6.6***: Fornece o conjunto de classes e métodos para criação dos serviços *REST* utilizando o padrão de projetos *MVC*. A classe controladora receber as anotações *@RestController* e *@RequestMapping* com especificação da rota que os métodos farão parte. Dentro da classe os métodos com anotações *@GetMapping*, *@PostMapping*, *@PutMapping* e *@DeleteMapping* são criados para expandir a comunicação e retornar respostas à interface do usuário.

- ***Spring Boot Starter Data JPA v2.6.6***: Combina o *Java Persistence API* com o *Hibernate*. Ambos são conjuntos de classes, métodos e mapeamentos utilizados para criação de entidades e manipulação da base de dados. Primeiro cria-se as classes com anotações *@Entity* e *@Table*, em seguida todas as colunas com suas respectivas características e relacionamentos com outras tabelas. Por fim, cria-se a interface de *@Repository* com os métodos de manipulação de dados que serão requisitados para esta entidade. O sistema encarrega-se de criar a tabela automaticamente na base de dados através do *Hybernate*.

- ***Spring Boot Starter Security v2.6.6***: Foi utilizada na criação de todos os recursos de segurança para proteção de acesso à aplicação e autenticação dos usuários. Com a inclusão das anotações *@EnableWebSecurity* e a extensão da classe *WebSecurityConfigurerAdapter* foi possível realizar todas as configurações de segurança das requisições HTTP e liberar as permissões de acesso ao compartilhamento de dados para todas as rotas de comunicação REST da API.

- ***MySQL Connector Java***: Fornece os *drivers* e recursos para comunicação com o banco de dados *MySQL*. Todas as informações são inseridas no arquivo de recursos (*application.properties*) do projeto.

- ***Project Lombok v3.12.0***: Responsável pela criação e gerenciamento automático dos métodos de *get*, *set*, *equals* e *toString*, utilizando apenas a anotação *@Data*.

- ***Model Mapper v3.1.0***: Mapeador automático de objetos de forma simples e segura. Foi utilizado na conversão dos objetos de *DTOs* para *@Entity* e vice-versa.

- ***Amazon SDK v2.17***: Essa dependência possibilita a implementação do conjunto de recursos de comunicação com a *Amazon Web Services - AWS*. Dentre os recursos utilizados está o *Cognito*, uma solução otimizada para cadastro, autenticação, geração de *tokens* e controle de acessos de usuários.

No Quadro 2 apresentamos o escopo do arquivo *pom.xml* com todas as dependências utilizadas no projeto, incluindo as descritas anteriormente.

Quadro 2 – Arquivo de Configuração da Interface de Aplicação

```
// pom.xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
      https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.6.6</version>
    <relativePath/>
  </parent>

  <groupId>br.edu.nutes</groupId>
  <artifactId>movvo</artifactId>
  <version>1.0.0</version>
  <name>movvo</name>
  <description>Sistema de gesto , localizao e monitoramento de equipamentos
      hospitalares</description>

  <properties>
    <java.version>17</java.version>
    <aws.java.sdk.version>2.17.121</aws.java.sdk.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
  </properties>

  <dependencyManagement>
    <dependencies>
      <!-- AWS SDK -->
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
```

```
            <version>${aws.java.sdk.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<dependencies>
    <!-- WEB -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- STARTER -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter</artifactId>
    </dependency>

    <!-- JPA -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
        <exclusions>
            <exclusion>
                <artifactId>hibernate-entitymanager</artifactId>
                <groupId>org.hibernate</groupId>
            </exclusion>
        </exclusions>
    </dependency>

    <!-- SECURITY -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>

    <!-- MYSQL -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
    </dependency>

    <!-- LOMBOK -->
    <dependency>
```

```
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.22</version>
    <scope>provided</scope>
</dependency>

<!-- COMMONS LANG 3 -->
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.12.0</version>
</dependency>

<!-- JAVAX VALIDATION -->
<dependency>
    <groupId>javax.validation</groupId>
    <artifactId>validation-api</artifactId>
    <version>2.0.1.Final</version>
</dependency>

<!-- ACTUATOR -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>

<!-- MODEL MAPPER -->
<dependency>
    <groupId>org.modelmapper</groupId>
    <artifactId>modelmapper</artifactId>
    <version>3.1.0</version>
</dependency>

<!-- AWS COGNITO IDENTITY -->
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>cognitoidentity</artifactId>
    <version>${aws.java.sdk.version}</version>
</dependency>

<!-- AWS COGNITO IDENTITY PROVIDER -->
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>cognitoidentityprovider</artifactId>
    <version>${aws.java.sdk.version}</version>
</dependency>
```

```
        <!-- TEST -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <!-- SPRINGDOC -->
        <dependency>
            <groupId>org.springdoc</groupId>
            <artifactId>springdoc-openapi-ui</artifactId>
            <version>1.6.7</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                <configuration>
                    <source>${java.version}</source>
                    <target>${java.version}</target>
                    <verbose>true</verbose>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-resources-plugin</artifactId>
                <version>3.1.0</version>
            </plugin>
        </plugins>
        <resources>
            <resource>
                <directory>src/main/resources</directory>
                <filtering>true</filtering>
            </resource>
        </resources>
    </build>
</project>
```

Fonte: Elaborada pelo autor, 2022

5.1.4 Arquitetura do Projeto de Interface da Aplicação

Definir a arquitetura do projeto é o passo inicial e o mais importante para manter as boas práticas de desenvolvimento, pois sua execução facilita a manutenção do código-fonte e traz mais performance e segurança para a aplicação. Desse modo, construímos uma arquitetura bem estruturada com algumas camadas que definimos por ser a abordagem mais adequada para a nossa solução. Abaixo apresentamos uma descrição de cada camada utilizando apenas o contexto de equipamentos, pois todos os outros recursos seguem a mesma lógica de negócio. A documentação completa pode ser consultada no Apêndice A.

5.1.4.1 Camada de Controle

Essa camada contém as classes responsáveis por receber as requisições vindas da interface de usuário, ela realiza a comunicação com a camada de serviço e retorna uma resposta com seus respectivos *status* (Quadro 3).

Quadro 3 – Classe de Controle de Equipamentos

```
// EquipamentoController.java
package com.nutes.movvo.controller.equipamento;

import com.nutes.movvo.dto.PaginacaoDTO;
import com.nutes.movvo.dto.equipamento.EquipamentoDTO;
import com.nutes.movvo.service.equipamento.EquipamentoService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;

@CrossOrigin
@RestController
@RequestMapping("equipamento")
public class EquipamentoController {

    @Autowired
    private EquipamentoService equipamentoService;

    @GetMapping
    public ResponseEntity<Page<EquipamentoDTO>> listarEquipamentos(
```

```
            @Valid PaginacaoDTO paginacao
    ) {
        return equipamentoService.list(paginacao);
    }

    @GetMapping("buscar")
    public ResponseEntity<Page<EquipamentoDTO>> buscarEquipamentos(
            @RequestParam(value = "term") String term,
            @Valid PaginacaoDTO paginacao
    ) {
        return equipamentoService.search(term, paginacao);
    }

    @GetMapping("{id}")
    public ResponseEntity<EquipamentoDTO> detalharEquipamento(
            @PathVariable(value = "id") Long id
    ) {
        return equipamentoService.get(id);
    }

    @PostMapping
    public ResponseEntity<EquipamentoDTO> cadastrarEquipamento(
            @RequestBody EquipamentoDTO dto
    ) {
        return equipamentoService.create(dto);
    }

    @PutMapping("{id}")
    public ResponseEntity<EquipamentoDTO> atualizarEquipamento(
            @PathVariable(value = "id") Long id,
            @RequestBody EquipamentoDTO dto
    ) {
        return equipamentoService.update(id, dto);
    }

    @DeleteMapping("{id}")
    public ResponseEntity<Boolean> removerEquipamento(
            @PathVariable(value = "id") Long id
    ) {
        return equipamentoService.delete(id);
    }

    @PostMapping("utilizar/{id}")
    public ResponseEntity<EquipamentoDTO> utilizarEquipamento(
            @PathVariable(value = "id") Long id
    ) {
        return equipamentoService.utilizar(id);
```

```
    }

    @PostMapping("desocupar/{id}")
    public ResponseEntity<EquipamentoDTO> desocuparEquipamento(
            @PathVariable(value = "id") Long id
    ) {
        return equipamentoService.desocupar(id);
    }
}
```

Fonte: Elaborada pelo autor, 2022

5.1.4.2 Camada de Serviço

Essa camada se comunica com todas as outras camadas da arquitetura, pois nela são implementadas as classes de serviço com todos os métodos contendo as lógicas de negócio da aplicação (Quadro 4).

Quadro 4 – Classe de Serviço de Equipamentos

```
// EquipamentoService.java
package com.nutes.movvo.service.equipamento;

import com.nutes.movvo.dto.PaginacaoDTO;
import com.nutes.movvo.dto.equipamento.EquipamentoDTO;
import com.nutes.movvo.entity.equipamento.EquipamentoEntity;
import com.nutes.movvo.repository.equipment.EquipamentoRepository;
import org.modelmapper.ModelMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestParam;

import javax.validation.Valid;
import java.util.Optional;

@Service("EquipamentoService")
public class EquipamentoService {
```

```
@Autowired
private EquipamentoRepository repository;

@Autowired
private ModelMapper modelMapper;

public ResponseEntity<Page<EquipamentoDTO>> list(
        @Valid PaginacaoDTO paginacao
) {
    PageRequest pageRequest = PageRequest.of(paginacao.getPage(), paginacao.getPageSize(),
        paginacao.getSort());
    Page<EquipamentoDTO> page =
        repository.findAll(pageRequest).map(EquipamentoDTO::new);
    return new ResponseEntity<>(page, HttpStatus.OK);
}

public ResponseEntity<Page<EquipamentoDTO>> search(
        @RequestParam(value = "term") String term,
        @Valid PaginacaoDTO paginacao
) {
    PageRequest pageRequest = PageRequest.of(paginacao.getPage(), paginacao.getPageSize(),
        paginacao.getSort());
    String terms = '%' + term.replace(" ", "%") + '%';
    Integer termsCount = term.split(" ").length;
    Page<EquipamentoDTO> page = repository.findEquipamentoEntitiesByTerms(
            terms, termsCount, pageRequest
    ).map(EquipamentoDTO::new);
    return new ResponseEntity<>(page, HttpStatus.OK);
}

public ResponseEntity<EquipamentoDTO> get(
        @PathVariable(value = "id") Long id
) {
    Optional<EquipamentoEntity> salvo = repository.findById(id);
    return salvo.map(entidade -> new ResponseEntity<>(
            new EquipamentoDTO(entidade), HttpStatus.OK)
    ).orElseGet(() -> new ResponseEntity<>(null, HttpStatus.NOT_FOUND));
}

public ResponseEntity<EquipamentoDTO> create(
        @RequestBody EquipamentoDTO dto
) {
    EquipamentoEntity salvo = repository.saveAndFlush(
            modelMapper.map(dto, EquipamentoEntity.class));
    return new ResponseEntity<>(new EquipamentoDTO(salvo), HttpStatus.OK);
}
```

```
    public ResponseEntity<EquipamentoDTO> update(
            @PathVariable(value = "id") Long id,
            @RequestBody EquipamentoDTO dto
    ) {
        Optional<EquipamentoEntity> salvo = repository.findById(dto.getId());
        if (salvo.isPresent()) {
            dto.setId(id);
            repository.save(modelMapper.map(dto, EquipamentoEntity.class));
            return new ResponseEntity<>(dto, HttpStatus.OK);
        }
        return new ResponseEntity<>(null, HttpStatus.NOT_FOUND);
    }

    public ResponseEntity<Boolean> delete(
            @PathVariable(value = "id") Long id
    ) {
        Optional<EquipamentoEntity> salvo = repository.findById(id);
        if (salvo.isPresent()) {
            repository.deleteById(id);
            return new ResponseEntity<>(true, HttpStatus.OK);
        }
        return new ResponseEntity<>(false, HttpStatus.NOT_FOUND);
    }

    public ResponseEntity<EquipamentoDTO> utilizar(
            @PathVariable(value = "id") Long id
    ) {
        return new ResponseEntity<>(null, HttpStatus.OK);
    }

    public ResponseEntity<EquipamentoDTO> desocupar(
            @PathVariable(value = "id") Long id
    ) {
        return new ResponseEntity<>(null, HttpStatus.OK);
    }
}
```

Fonte: Elaborada pelo autor, 2022

5.1.4.3 Camada de Entidades

É nessa camada onde são inseridas as entidades com seus respectivos parâmetros, características individuais, relacionamentos com outras entidades e partir dela são geradas todas as tabelas na base de dados por meio do *Hybernate* (Quadro 5).

Quadro 5 – Classe de Entidade de Equipamentos

```
// EquipamentoEntity.java
package com.nutes.movvo.entity.equipamento;

import com.nutes.movvo.entity.transmissor.TransmissorEntity;
import lombok.Data;
import org.hibernate.annotations.*;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.Table;
import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.Date;
import java.util.HashSet;
import java.util.Set;

@Data
@Entity
@Table(name = "equipamentos")
public class EquipamentoEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @NotFound(action = NotFoundAction.IGNORE)
    @JoinColumn(name = "tipo", referencedColumnName = "id")
    private EquipamentoTipoEntity tipo;

    @Column(name = "nome", nullable = false, length = 20)
    private String nome;

    @Column(name = "codigo", nullable = false, length = 10, unique = true)
    private String codigo;

    @Column(name = "fabricacao", nullable = false)
    private Date fabricacao;

    @Column(name = "periodo_manutencao")
    private String periodoManutencao;

    @Column(name = "ultima_manutencao")
```

```
    private Date ultimaManutencao;

    @OneToOne(fetch = FetchType.LAZY)
    @NotFound(action = NotFoundAction.IGNORE)
    @JoinColumn(name = "transmissor", referencedColumnName = "id")
    private TransmissorEntity transmissor;

    @OneToMany(mappedBy = "equipamento", orphanRemoval = true, cascade =
        {CascadeType.ALL}, fetch = FetchType.LAZY)
    private Set<EquipamentoStatusEntity> statusHistorico = new HashSet<>();

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinFormula("(SELECT status.id FROM equipamento_status status WHERE status.equipamento
        = id " +
            "ORDER BY status.updated_at DESC LIMIT 1)")
    private EquipamentoStatusEntity statusAtual;

    @Column(name = "observacao", length = 200)
    private String observacao;

    @CreationTimestamp
    @Column(name = "created_at", updatable = false, nullable = false)
    private LocalDateTime createdAt;

    @UpdateTimestamp
    @Column(name = "updated_at", nullable = false)
    private LocalDateTime updatedAt;
}
```

Fonte: Elaborada pelo autor, 2022

5.1.4.4 Camada de Modelos de Transferência de Dados

Os objetos de transferência de dados são inseridos nessa camada, sejam eles de recebimento através do corpo das requisições ou de resposta para a interface de usuário (Quadro 6).

Quadro 6 – Classe de Dados de Equipamentos

```
// EquipamentoDTO.java
package com.nutes.movvo.dto.equipamento;

import com.nutes.movvo.dto.transmissor.TransmissorDTO;
import com.nutes.movvo.entity.equipamento.EquipamentoEntity;
```

```
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDateTime;
import java. util .Date;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class EquipamentoDTO {

    private Long id;
    private EquipamentoTipoDTO tipo;
    private String nome;
    private String codigo;
    private Date fabricacao;
    private String periodoManutencao;
    private Date ultimaManutencao;
    private TransmissorDTO transmissor;
    private EquipamentoStatusDTO statusAtual;
    private String observacao;
    private LocalDateTime createdAt;
    private LocalDateTime updatedAt;

    public EquipamentoDTO(EquipamentoEntity equipamento) {
        this.id = equipamento.getId();
        if (equipamento.getTipo() != null)
            this.tipo = new EquipamentoTipoDTO(equipamento.getTipo());
        this.nome = equipamento.getNome();
        this.codigo = equipamento.getCodigo();
        this.fabricacao = equipamento.getFabricacao();
        this.periodoManutencao = equipamento.getPeriodoManutencao();
        this.ultimaManutencao = equipamento.getUltimaManutencao();
        if (equipamento.getTransmissor() != null)
            this.transmissor = new TransmissorDTO(equipamento.getTransmissor());
        if (equipamento.getStatusAtual() != null)
            this.statusAtual = new EquipamentoStatusDTO(equipamento.getStatusAtual());
        this.observacao = equipamento.getObservacao();
        this.createdAt = equipamento.getCreatedAt();
        this.updatedAt = equipamento.getUpdatedAt();
    }
}
```

Fonte: Elaborada pelo autor, 2022

5.1.4.5 Camada de Repositório

Essa camada inclui as interfaces de repositório que realizam a persistência das entidades e se comunicam diretamente com o meio de acesso aos dados. As *querys* personalizadas são inseridas dentro dessas interfaces (Quadro 7).

Quadro 7 – Classe de Repositório de Equipamentos

```
// EquipamentoRepository.java
package com.nutes.movvo.repository.equipment;

import com.nutes.movvo.entity.equipamento.EquipamentoEntity;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

@Repository
public interface EquipamentoRepository extends JpaRepository<EquipamentoEntity, Long> {

    @Query("SELECT equipamento FROM EquipamentoEntity equipamento WHERE
        function('repeat', function('concat_ws', ';', equipamento.codigo, equipamento.nome,
        COALESCE(equipamento.tipo.nome, ''),
        COALESCE(equipamento.transmissor.statusAtual.receptor.local.nome, ''),
        COALESCE(equipamento.transmissor.statusAtual.receptor.local.area.nome, ''),
        function('date_format', equipamento.fabricacao, '%d/%m/%Y')), :termsCount) LIKE :terms")
    Page<EquipamentoEntity> findEquipamentoEntitiesByTerms(
            @Param("terms") String terms,
            @Param("termsCount") Integer termsCount,
            @Param("pageable") Pageable pageable);
}
```

Fonte: Elaborada pelo autor, 2022

5.1.4.6 Camada de Configuração

Nessa camada inserimos as classes de configurações necessárias para o funcionamento adequado do sistema. O *WebSecurityConfig* é a classe mais importante dessa camada, pois ela contém o mapeamento de todas as rotas da *API*, onde são realizadas as checagens de permissão e a liberação/bloqueio da autenticação do usuário (Quadro 8).

Quadro 8 – Classe de Configuração da Segurança das Requisições

```
// WebSecurityConfig.java
package com.nutes.movvo.config;

import com.nutes.movvo.aws.AwsCognito;
import com.nutes.movvo.dto.CredenciaisDTO;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
    org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.AuthenticationEntryPoint;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;
import org.springframework.security.web.util.matcher.OrRequestMatcher;
import org.springframework.security.web.util.matcher.RequestMatcher;
import org.springframework.stereotype.Component;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.CorsConfigurationSource;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;
import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
```

```
@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private AuthenticationFilter jwtTokenFilter;

    @Autowired
    private JwtAuthenticationEntryPoint jwtAuthenticationEntryPoint;

    @Autowired
    private EndpointsConfig endpointsConfig;

    @Bean
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Override
    public void configure(WebSecurity web) throws Exception {
        web.ignoring().antMatchers(HttpMethod.OPTIONS, "/**");
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        List<RequestMatcher> requestMatchers = new ArrayList<>();
        endpointsConfig.list().forEach(endpoint -> requestMatchers.add(new
            AntPathRequestMatcher(endpoint)));

        http.csrf().disable()
            .addFilterBefore(jwtTokenFilter, UsernamePasswordAuthenticationFilter.class)
            .authorizeRequests()
            .antMatchers("/actuator/**")
            .permitAll()
            .antMatchers("/auth/**")
            .permitAll()
            .requestMatchers(new OrRequestMatcher(requestMatchers))
            .authenticated()
            .and()
            .exceptionHandling()
```

```
            .authenticationEntryPoint(jwtAuthenticationEntryPoint)
            .and()
            .sessionManagement()
            .sessionCreationPolicy(SessionCreationPolicy.STATELESS);
    }

    @Bean
    CorsConfigurationSource corsConfigurationSource() {

        CorsConfiguration config = new CorsConfiguration();
        config.setAllowedOrigins(Collections.singletonList("*"));
        config.setAllowedMethods(Arrays.asList("POST", "GET", "DELETE", "PUT", "UPDATE",
            "OPTIONS"));
        config.setAllowedHeaders(Collections.singletonList("*"));
        config.setAllowCredentials(true);
        config.setMaxAge(3600L);

        UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
        endpointsConfig.list().forEach(
            endpoint -> source.registerCorsConfiguration(endpoint, config));

        return source;
    }

    @Configuration
    public static class AuthenticationFilter extends OncePerRequestFilter {

        @Autowired
        private EndpointsConfig endpointsConfig;

        @Autowired
        private AwsCognito awsCognito;

        @Override
        protected void doFilterInternal(
                HttpServletRequest request, HttpServletResponse response, FilterChain chain
        ) throws ServletException, IOException {
            String authorization = request.getHeader("Authorization");
            if (authorization != null && authorization.startsWith("Bearer ")) {
                String path = request.getServletPath();
                endpointsConfig.list().forEach(endpoint -> {
                    String routeUrl = endpoint.replace("/**", "");
                    if (path.startsWith(routeUrl) && awsCognito.isTokenValid(authorization)) {
                        acceptCredentials(request, authorization);
                    }
                });
            }
```

```
            chain.doFilter(request, response);
        }

        private void acceptCredentials(HttpServletRequest request, String authorization) {
            CredenciaisDTO accountCredentials = new CredenciaisDTO(
                    awsCognito.getUserByToken(authorization).username());
            UsernamePasswordAuthenticationToken auth = new
                UsernamePasswordAuthenticationToken(
                    accountCredentials, null, accountCredentials.getAuthorities());
            auth.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));
            SecurityContextHolder.getContext().setAuthentication(auth);
        }
    }

    @Component
    public static class JwtAuthenticationEntryPoint implements AuthenticationEntryPoint,
        Serializable {
        @Override
        public void commence(HttpServletRequest request, HttpServletResponse response,
                             AuthenticationException authException) throws IOException {
            response.sendError(HttpServletResponse.SC_UNAUTHORIZED, "Unauthorized");
        }
    }
}
```

Fonte: Elaborada pelo autor, 2022

5.1.5 Recursos de Autenticação da Interface de Aplicação

O *AWS Cognito* foi o recurso que utilizamos para o controle de cadastro e autenticação dos usuários. Ele fornece todas as funcionalidades necessárias de forma segura e de fácil implementação. Apresentamos no Quadro 9 a classe de controle com as chamadas de autenticação que serão acessadas pela interface do usuário.

Quadro 9 – Classe de Controle de Autenticação

```
// AutenticacaoController.java
package com.nutes.movvo.controller.autenticacao;

import com.nutes.movvo.dto.autenticacao.*;
import com.nutes.movvo.service.AutenticacaoService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
```

```
import org.springframework.web.bind.annotation.*;

@CrossOrigin
@RestController
@RequestMapping("autenticacao")
public class AutenticacaoController {

    @Autowired
    private AutenticacaoService autenticacaoService;

    @PostMapping("login")
    public ResponseEntity<TokenDTO> login(
            @RequestBody LoginDTO dto
    ) {
        return autenticacaoService.login(dto);
    }

    @PostMapping("cadastrar")
    public ResponseEntity<CadastrarDTO> cadastrarUsuario(
            @RequestBody CadastrarDTO dto
    ) {
        return autenticacaoService.cadastrar(dto);
    }

    @PostMapping("confirmar-cadastro")
    public ResponseEntity<Boolean> confirmarCadastro(
            @RequestBody ConfirmarCadastroDTO dto
    ) {
        return autenticacaoService.confirmarCadastro(dto);
    }

    @PostMapping("recuperar-senha")
    public ResponseEntity<Boolean> recuperarSenha(
            @RequestParam("usuario") String usuario
    ) {
        return autenticacaoService.recuperarSenha(usuario);
    }

    @PostMapping("confirmar-nova-senha")
    public ResponseEntity<Boolean> confirmarNovaSenha(
            @RequestBody ConfirmarNovaSenhaDTO dto
    ) {
        return autenticacaoService.confirmarNovaSenha(dto);
    }

    @PostMapping("alterar-senha")
    public ResponseEntity<Boolean> alterarSenha(
```

```
            @RequestHeader("Authorization") String authorization,
            @RequestBody AlterarSenhaDTO dto
    ) {
        return autenticacaoService.alterarSenha(authorization, dto);
    }
}
```

Fonte: Elaborada pelo autor, 2022

Essa classe de controle fará a comunicação com a classe de serviço de autenticação (Quadro 10) que contém os métodos com a lógica negócio.

Quadro 10 – Classe de Serviço de Autenticação

```
// AutenticacaoService.java
package com.nutes.movvo.service;

import com.nutes.movvo.dto.autenticacao.*;
import com.nutes.movvo.service.aws.AwsCognitoService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestParam;
import software.amazon.awssdk.services.cognitoidentityprovider.model.*;

@Service("AutenticacaoService")
public class AutenticacaoService {

    @Autowired
    private AwsCognitoService awsCognito;

    public ResponseEntity<TokenDTO> login(
            @RequestBody LoginDTO dto
    ) {
        try {
            AuthenticationResultType authenticationResultType = awsCognito.login(dto);
            TokenDTO token = new TokenDTO(authenticationResultType.accessToken(),
                    authenticationResultType.refreshToken());
            return new ResponseEntity<>(token, HttpStatus.OK);
        } catch (UserNotConfirmedException | NotAuthorizedException e) {
            if (e.getClass().equals(UserNotConfirmedException.class)) {
                awsCognito.reenviarCodigoCadastro(dto.getUsuario());
```

```
            }
            return new ResponseEntity<>(null, HttpStatus.valueOf(e.statusCode()));
        }
    }

    public ResponseEntity<CadastrarDTO> cadastrar(
            @RequestBody CadastrarDTO dto
    ) {
        try {
            awsCognito.cadastrar(dto);
            return new ResponseEntity<>(dto, HttpStatus.OK);
        } catch (InvalidParameterException | UsernameExistsException | InvalidPasswordException
                 e) {
            return new ResponseEntity<>(null, HttpStatus.valueOf(e.statusCode()));
        }
    }

    public ResponseEntity<Boolean> confirmarCadastro(
            @RequestBody ConfirmarCadastroDTO dto
    ) {
        try {
            awsCognito.confirmarCadastro(dto);
            return new ResponseEntity<>(true, HttpStatus.OK);
        } catch (ExpiredCodeException | CodeMismatchException e) {
            if (e.getClass().equals(ExpiredCodeException.class)) {
                awsCognito.reenviarCodigoCadastro(dto.getUsuario());
            }
            return new ResponseEntity<>(false, HttpStatus.valueOf(e.statusCode()));
        }
    }

    public ResponseEntity<Boolean> recuperarSenha(
            @RequestParam("usuario") String usuario
    ) {
        try {
            awsCognito.recuperarSenha(usuario);
            return new ResponseEntity<>(true, HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<>(false, HttpStatus.BAD_REQUEST);
        }
    }

    public ResponseEntity<Boolean> confirmarNovaSenha(
            @RequestBody ConfirmarNovaSenhaDTO dto
    ) {
        try {
            awsCognito.confirmarNovaSenha(dto);
```

```
            return new ResponseEntity<>(true, HttpStatus.OK);
        } catch (InvalidParameterException | ExpiredCodeException | CodeMismatchException e) {
            return new ResponseEntity<>(false, HttpStatus.valueOf(e.statusCode()));
        }
    }

    public ResponseEntity<Boolean> alterarSenha(
            @RequestHeader("Authorization") String authorization,
            @RequestBody AlterarSenhaDTO dto
    ) {
        if (awsCognito.isTokenValid(authorization)) {
            try {
                awsCognito.alterarSenha(authorization, dto);
                return new ResponseEntity<>(true, HttpStatus.OK);
            } catch (UsernameExistsException | InvalidPasswordException
                    | NotAuthorizedException | LimitExceededException e) {
                return new ResponseEntity<>(false, HttpStatus.valueOf(e.statusCode()));
            }
        }
        return new ResponseEntity<>(false, HttpStatus.UNAUTHORIZED);
    }
}
```

Fonte: Elaborada pelo autor, 2022

Por sua vez, a classe de serviço de autenticação irá se comunicar com uma segunda classe de serviço apresentada no Quadro 11. Essa segunda classe é responsável pela comunicação direta com os recursos do *AWS Cognito*.

Quadro 11 – Classe de Serviço do Cognito

```
// AwsCognitoService.java
package com.nutes.movvo.service.aws;

import com.nutes.movvo.dto.autenticacao.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import software.amazon.awssdk.services.cognitoidentityprovider.model.*;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
```

```
@Service("AwsCognitoService")
public class AwsCognitoService {

    @Autowired
    private AwsCredenciaisService credenciais;

    @Value("${cognito.clientId}")
    private String clientId;

    @Value("${cognito.userPoolId}")
    private String userPoolId;

    public CognitoIdentityProviderClient cognitoIdentityProviderClient() {
        return CognitoIdentityProviderClient.builder()
                .credentialsProvider(credenciais)
                .region(Region.US_EAST_1)
                .build();
    }

    private String getTokenString(String authorization) {
        return authorization.startsWith("Bearer ") ? authorization.substring(7) : authorization;
    }

    public boolean isTokenValid(String authorization) {
        try {
            return getUsuarioInfo(getTokenString(authorization)).hasUserAttributes();
        } catch (Exception e) {
            return false;
        }
    }

    public boolean usuarioExiste(String username) {
        try {
            ListUsersResponse usuarios = cognitoIdentityProviderClient()
                    .listUsers(ListUsersRequest
                            .builder()
                            .userPoolId(userPoolId)
                            .filter(String.format("username=\"%s\"", username))
                            .build()
                    );
            return usuarios.users().size() > 0;
        } catch (Exception e) {
            return false;
        }
    }
```

```
public GetUserResponse getUsuarioInfo(String authorization) {
    return cognitoIdentityProviderClient()
        .getUser(GetUserRequest
            .builder()
            .accessToken(getTokenString(authorization))
            .build()
        );
}

public AuthenticationResultType login(LoginDTO dto) {
    HashMap<String, String> params = new HashMap<>() {{
        put("USERNAME", dto.getUsuario());
        put("PASSWORD", dto.getSenha());
    }};
    return cognitoIdentityProviderClient()
        .initiateAuth(InitiateAuthRequest
            .builder()
            .authFlow(AuthFlowType.USER_PASSWORD_AUTH)
            .clientId(clientId)
            .authParameters(params)
            .build()
        ).authenticationResult();
}

public void cadastrar(CadastrarDTO dto) {
    List<AttributeType> attrs = new ArrayList<>() {{
        add(AttributeType.builder().name("email").value(dto.getEmail()).build());
        add(AttributeType.builder().name("name").value(dto.getNome()).build());
        add(AttributeType.builder().name("nickname").value(dto.getUsuario()).build());
    }};
    SignUpRequest signUpRequest = SignUpRequest.builder()
        .userAttributes(attrs)
        .clientId(clientId)
        .username(dto.getUsuario())
        .password(dto.getSenha())
        .build();
    cognitoIdentityProviderClient().signUp(signUpRequest);
}

public void confirmarCadastro(ConfirmarCadastroDTO dto) {
    cognitoIdentityProviderClient()
        .confirmSignUp(ConfirmSignUpRequest
            .builder()
            .username(dto.getUsuario())
            .confirmationCode(dto.getCodigoConfirmacao())
            .clientId(clientId)
            .build()
```

```
        );
    }

    public void reenviarCodigoCadastro(String username) {
        cognitoIdentityProviderClient()
            .resendConfirmationCode(ResendConfirmationCodeRequest
                .builder()
                .username(username)
                .clientId(clientId)
                .build()
            );
    }

    public void recuperarSenha(String username) {
        cognitoIdentityProviderClient()
            .forgotPassword(ForgotPasswordRequest
                .builder()
                .username(username)
                .clientId(clientId)
                .build()
            );
    }

    public void confirmarNovaSenha(ConfirmarNovaSenhaDTO credentials) {
        cognitoIdentityProviderClient()
            .confirmForgotPassword(ConfirmForgotPasswordRequest
                .builder()
                .username(credentials.getUsuario())
                .confirmationCode(credentials.getCodigoConfirmacao())
                .password(credentials.getNovaSenha())
                .clientId(clientId)
                .build()
            );
    }

    public void alterarSenha(String authorization, AlterarSenhaDTO dto) {
        cognitoIdentityProviderClient()
            .changePassword(ChangePasswordRequest
                .builder()
                .accessToken(getTokenString(authorization))
                .previousPassword(dto.getSenha())
                .proposedPassword(dto.getNovaSenha())
                .build()
            );
    }

    public AuthenticationResultType atualizarToken(String refreshToken) {
```

```
        HashMap<String, String> params = new HashMap<>() {{
            put("REFRESH_TOKEN", refreshToken);
        }};
        return cognitoIdentityProviderClient()
            .initiateAuth(InitiateAuthRequest
                .builder()
                .authFlow(AuthFlowType.REFRESH_TOKEN)
                .clientId(clientId)
                .authParameters(params)
                .build()
            ).authenticationResult();
    }

    public void revogarToken(String refreshToken) {
        cognitoIdentityProviderClient()
            .revokeToken(RevokeTokenRequest
                .builder()
                .token(refreshToken)
                .clientId(clientId)
                .build()
            );
    }
}
```

Fonte: Elaborada pelo autor, 2022

Assim que o usuário realizar a autenticação no sistema, será retornado um *token* para que ele possa usá-lo nas requisições e ter acesso a todos os outros recursos da *API*. No Quadro 12 está contido a classe de controle relacionada ao *token*.

Quadro 12 – Classe de Controle de *Tokens*

```
// TokenController.java
package com.nutes.movvo.controller.autenticacao;

import com.nutes.movvo.dto.autenticacao.TokenDTO;
import com.nutes.movvo.service.TokenService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@CrossOrigin
@RestController
@RequestMapping("token")
public class TokenController {
```

```
    @Autowired
    private TokenService tokenService;

    @GetMapping("validar")
    public ResponseEntity<Boolean> validarToken(
            @RequestHeader("Authorization") String authorization
    ) {
        return tokenService.validar(authorization);
    }

    @GetMapping("atualizar")
    public ResponseEntity<TokenDTO> atualizarToken(
            @RequestParam("refreshToken") String refreshToken
    ) {
        return tokenService.atualizar(refreshToken);
    }

    @GetMapping("revogar")
    public ResponseEntity<Boolean> revogarToken(
            @RequestParam("refreshToken") String refreshToken
    ) {
        return tokenService.revogar(refreshToken);
    }
}
```

Fonte: Elaborada pelo autor, 2022

Para completar o fluxo de comunicação, a classe de controle do *token* também se comunica com sua respectiva classe de serviço, nela contém os métodos de validação, atualização e revogação do *token* (Quadro 13).

Quadro 13 – Classe de Serviço de *Tokens*

```
// TokenService.java
package com.nutes.movvo.service;

import com.nutes.movvo.dto.autenticacao.TokenDTO;
import com.nutes.movvo.service.aws.AwsCognitoService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestParam;
```

```
import software.amazon.awssdk.services.cognitoidentityprovider.model.AuthenticationResultType;
import software.amazon.awssdk.services.cognitoidentityprovider.model.NotAuthorizedException;

@Service("TokenService")
public class TokenService {

    @Autowired
    private AwsCognitoService awsCognito;

    public ResponseEntity<Boolean> validar(
            @RequestHeader("Authorization") String authorization
    ) {
        if (awsCognito.isTokenValid(authorization)) {
            return new ResponseEntity<>(true, HttpStatus.OK);
        }
        return new ResponseEntity<>(false, HttpStatus.UNAUTHORIZED);
    }

    public ResponseEntity<TokenDTO> atualizar(
            @RequestParam("refreshToken") String refreshToken
    ) {
        try {
            AuthenticationResultType authenticationResultType =
                awsCognito.atualizarToken(refreshToken);
            TokenDTO token = new TokenDTO(authenticationResultType.accessToken(),
                refreshToken);
            return new ResponseEntity<>(token, HttpStatus.OK);
        } catch (NotAuthorizedException e) {
            return new ResponseEntity<>(null, HttpStatus.valueOf(e.statusCode()));
        } catch (Exception e) {
            return new ResponseEntity<>(null, HttpStatus.BAD_REQUEST);
        }
    }

    public ResponseEntity<Boolean> revogar(
            @RequestParam("refreshToken") String refreshToken
    ) {
        try {
            awsCognito.revogarToken(refreshToken);
            return new ResponseEntity<>(true, HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<>(false, HttpStatus.BAD_REQUEST);
        }
    }
}
```

Fonte: Elaborada pelo autor, 2022

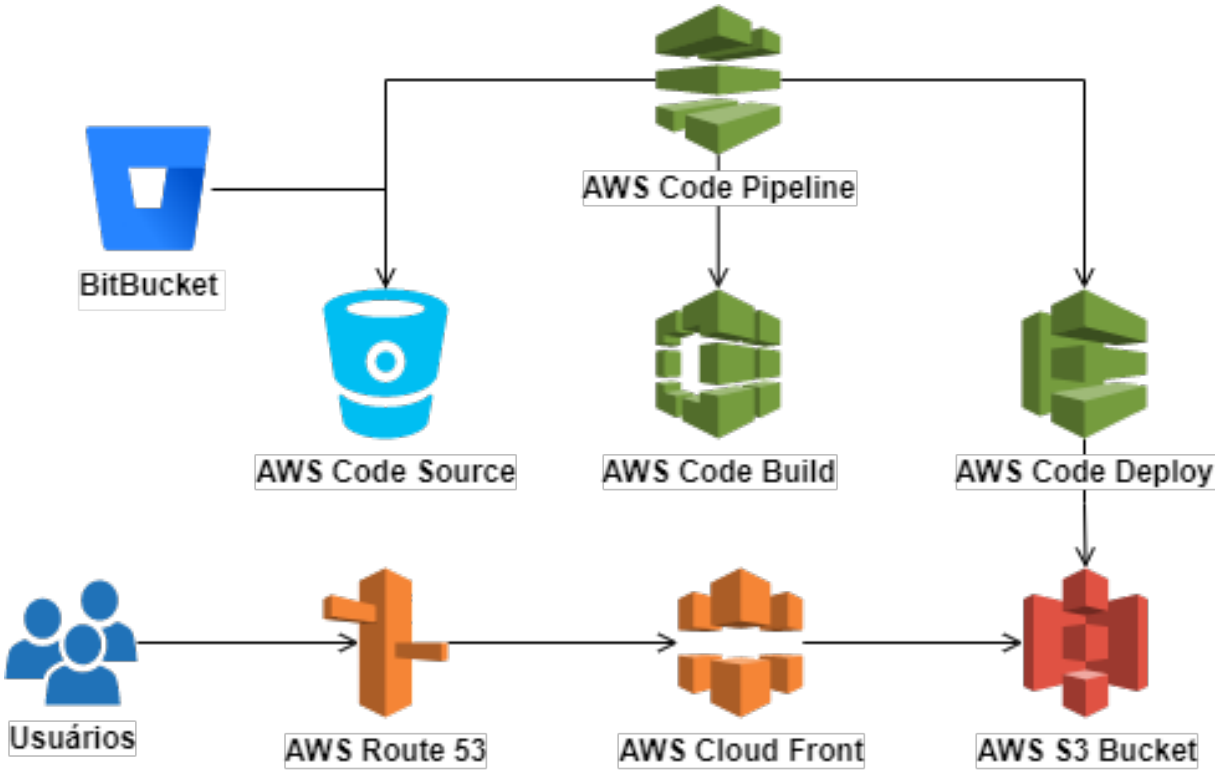
5.2 IMPLANTAÇÃO

Assim como a insfraestrutura de rastreamento, é uma opção do fornecedor do ERP-SaaS os meios para implantação da interface do usuário e da aplicação do Movvo. Não há um pré-requisito ou exigência para essa realização. No entanto, para fins demonstrativos executamos a implantação de ambos os projetos na *AWS* com todos os recursos necessários.

Na prática, os recursos utilizados na implantação da interface do usuário são diferente dos recursos utilizados na implantação da interface da aplicação, mas há semelhanças entre eles. O *AWS Code Pipeline* é utilizado nos dois casos. Ele é responsável por automatizar a configuração dos três passos necessários para a implantação do projeto: o código (*AWS Code Source*), a compilação (*AWS Code Build*) e a implantação (*AWS Code Deploy*). Esses passos são executados em sequência e mantém o projeto sempre atualizado.

Apresentamos na Figura 22 a arquitetura de implantação da interface do usuário. Nela podemos ver que o primeiro contato que o usuário tem é com o *AWS Route 53*, pois é com esse recurso que configuramos a rota que pode ser acessada através do navegador. Essa rota faz a associação com o *AWS Cloud Front* que por sua vez se comunica com o *AWS S3 Bucket* onde armazena-se o código-fonte do projeto.

Figura 22 – Arquitetura de Implantação da Interface do Usuário



Fonte: Elaborada pelo autor, 2022

Nas configurações do *AWS Code Build* precisamos informar qual a sequência de passos necessários para a compilação do projeto. Para isso temos que editar o arquivo *buildspec.yml* e inserir no escopo as instruções contidas no Quadro 14.

Quadro 14 – Arquivo do Compilador da Interface do Usuário

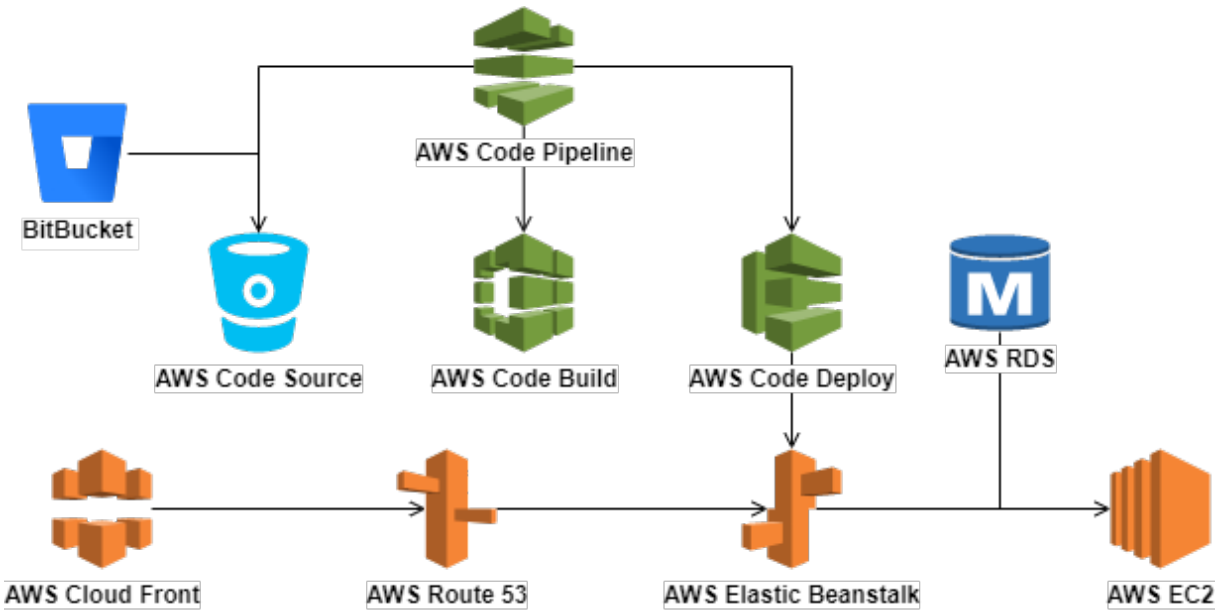
```
// Buildspec.yml
version: 0.2

phases:
  install :
    runtime-versions:
      nodejs: 12

  pre_build:
    commands:
      - echo Installing source NPM dependencies
      - npm install
      - npm install -g @angular/cli
```

```
build:
  commands:
    – echo Build started
    – ng build ––configuration production

post_build:
  commands:
    – echo Build completed on ‘date‘

artifacts :
 files :
  – ’**/*’
base–directory: ’dist’
```

Fonte: Elaborada pelo autor, 2022

Na Figura 23 demonstramos a arquitetura de implantação da interface de aplicação. O *AWS Cloud Front* é o recurso responsável por fazer a comunicação com *AWS Elastic Beanstalk* através de uma rota configurada no *AWS Route 53*. Esse recurso traz vários benefícios para a aplicação, principalmente benefícios relacionados a performance e segurança. Por fim, a aplicação comunica-se com o *AWS RDS* onde ficam armazenados os dados e é implantada no *AWS EC2*, onde cria-se uma instância escalável com o sistema operacional escolhido.

Figura 23 – Arquitetura de Implantação da Interface da Aplicação



Fonte: Elaborada pelo autor, 2022

Aqui também é necessário informar os passos para a compilação através do

AWS Code Build. É importante ressaltar que o projeto foi desenvolvido utilizando o *Java v17* e ele necessita do compilador *Correto v17* para que possa executar a aplicação. No momento dessa implantação percebemos que a versão máxima predeterminada era o *Correto v11*, mas conseguimos instalar o *Correto v17* manualmente utilizando as instruções no arquivo *buildspec.yml* apresentado no Quadro 15.

Quadro 15 – Arquivo do Compilador da Interface da Aplicação

```
// Buildspec.yml
version: 0.2

phases:
  install :
    runtime-versions:
      java: corretto11

  pre_build:
    commands:
      - echo Installing Corretto17
      - yum install -y java-17-amazon-corretto-devel
      - echo Configuring JAVA_HOME
      - export JAVA_HOME=/usr/lib/jvm/java-17-amazon-corretto.x86_64
      - export PATH=$PATH:$JAVA_HOME/bin
      - echo Updating Java configurations
      - update-alternatives --auto javac
      - update-alternatives --auto java
      - mvn -version
      - java -version

  build:
    commands:
      - echo Build started on 'date'
      - mvn clean -B package

  post_build:
    commands:
      - echo Build completed on 'date'

artifacts :
  files :
    - '**/*'
  base-directory: 'target'
```

Fonte: Elaborada pelo autor, 2022

6 CONCLUSÕES

Os sistemas ERPs prometem uma integração perfeita dos processos de negócio de toda a organização. No entanto, a implantação desses sistemas é significativamente desafiadora e gera gastos enormes de recursos em termos de tempo, dinheiro e energia nos processos de implementação interna. Dado esse cenário, é importante definir os fatores críticos de sucesso do ERP, pois o bom entendimento desses fatores contribui diretamente para a escolha das melhores estratégias de implementação (ABOABDO; ALDHOIENA; AL-AMRIB, 2019).

Nesse trabalho optamos por focar nos sistemas com arquitetura ERP integrados aos SaaS, o que chamamos de ERP-SaaS. Esse modelo transfere a necessidade de implementação local para um provedor em nuvem. A tecnologia em nuvem traz um conjunto de vantagens, como o fácil uso e acessibilidade, recursos virtualizados, escalabilidade e disponibilidade garantidos pelo provedor do serviço. A tecnologia SaaS permite que os sistemas ERP otimizem algumas de suas fraquezas típicas, como inflexibilidade, escalabilidade e consumação de enormes recursos locais (SAA *et al.*, 2017).

Nosso sistema se beneficiará das funcionalidades do ERP-SaaS para promover a criação de um módulo de gestão e monitoramento de equipamentos com auxílio de um serviço interno de rastreamento. Houve na última década um interesse crescente em técnicas de rastreamento interno, principalmente porque permite a localização de objetos e pessoas de maneira rápida, eficiente, de baixo custo e tendo uma infraestrutura facilmente implantável nas organizações (HOSSAIN; SOH, 2015). Dentre as tecnologias disponíveis para esse objetivo, descrevemos as mais conhecidas na literatura: Bluetooth, RFID, infravermelho, WLAN, entre outros. Tecnologias essas que atendem a categorias de baixa, curta e longa distância (KANAN; ELHASSAN, 2015).

Esperamos que o desenvolvimento do nosso projeto potencialize o controle dentro de uma organização de saúde, e que com auxílio de tecnologias de baixo custo e eficiência energética, permita um melhor controle das demandas internas. Focamos em uma interface intuitiva, ágil e de fácil compreensão, pois o mercado é exigente e uma boa usabilidade resulta também na qualidade da implantação dentro da organização e na satisfação dos seus funcionários. Acreditamos que essa solução reduzirá desperdícios de tempo, recursos e otimizará a produtividade da equipe médica, diminuindo a preocupação em encontrar os equipamentos requisitados e favorecendo uma maior atenção ao paciente que precisa de cuidados.

REFERÊNCIAS BIBLIOGRÁFICAS

AboAbdo, Aldhoiena e Al-Amrib 2019ABOABDO, S.; ALDHOIENA, A.; AL-AMRIB, H. Implementing enterprise resource planning erp system in a large construction company in ksa. **Procedia Computer Science**, Elsevier, v. 164, p. 463–470, 2019.

Achargui e Zaouia 2016ACHARGUI, A.; ZAOUIA, A. Hosted, cloud and saas, off-premises erp systems adoption by moroccan smes: a focus group study. In: IEEE. **2016 7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)**. [S.l.], 2016. p. 344–348.

Albahri *et al.* 2018ALBAHRI, O.; ZAIDAN, A.; ZAIDAN, B.; HASHIM, M.; ALBAHRI, A.; ALSALEM, M. Real-time remote health-monitoring systems in a medical centre: A review of the provision of healthcare services-based body sensor information, open challenges and methodological aspects. **Journal of medical systems**, Springer, v. 42, n. 9, p. 164, 2018.

Aleem *et al.* 2018ALEEM, S.; BATOOL, R.; AHMED, F.; KHATTAK, A. M. Design guidelines for saas development process. In: IEEE. **2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)**. [S.l.], 2018. p. 825–831.

Alghamdi, Schyndel e Alahmadi 2013ALGHAMDI, S.; SCHYNDEL, R. V.; ALAHMADI, A. Indoor navigational aid using active rfid and qr-code for sighted and blind people. In: IEEE. **2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing**. [S.l.], 2013. p. 18–22.

AngularDoc 2022ANGULARDOC. **AngularDoc**. 2022. Disponível em: <https://angular.io/guide/architecture/>. Acesso em: 30 Apr. 2022.

Berndt *et al.* 2012BERNDT, R.; TAKENGA, M.; KUEHN, S.; PREIK, P.; SOMMER, G.; BERNDT, S. Saas-platform for mobile health applications. In: IEEE. **International Multi-Conference on Systems, Signals & Devices**. [S.l.], 2012. p. 1–4.

Boot 2018BOOT, S. **Spring Boot**. 2018.

Campanella *et al.* 2017CAMPANELLA, P.; AZZOLINI, E.; IZZI, A.; PELONE, F.; MEO, C. D.; MILIA, D. L.; SPECCHIA, M. L.; RICCIARDI, W. Hospital efficiency: how to spend less maintaining quality? **Annali dell'Istituto superiore di sanita**, v. 53, n. 1, p. 46–53, 2017.

Coustasse, Tomblin e Slack 2013COUSTASSE, A.; TOMBLIN, S.; SLACK, C. Impact of radio-frequency identification (rfid) technologies on the hospital supply chain: a literature review. **Perspectives in Health Information Management**, American Health Information Management Association, v. 10, n. Fall, 2013.

Frisch 2019FRISCH, P. H. Rfid in today's intelligent hospital enhancing patient care & optimizing hospital operations. p. 458–463, 2019.

Govindaraju *et al.* 2015GOVINDARAJU, R.; SALAJAR, R.; CHANDRA, D. R.; SUDIRMAN, I. Acceptance and usage of erp systems: The role of institutional factors in erp post-implementation. p. 1292–1296, 2015.

Hameed e Ahmed 2018HAMEED, A.; AHMED, H. A. Survey on indoor positioning applications based on different technologies. p. 1–5, 2018.

Hassanien e Gaber 2017HASSANIEN, A. E.; GABER, T. **Handbook of research on machine learning innovations and trends**. [S.l.]: IGI Global, 2017.

Hossain e Soh 2015HOSSAIN, A. M.; SOH, W.-S. A survey of calibration-free indoor positioning systems. **Computer Communications**, Elsevier, v. 66, p. 1–13, 2015.

Ilkovičová, Erdélyi e Kopáčik 2014ILKOVIČOVÁ, L.; ERDÉLYI, J.; KOPÁČIK, A. Positioning in indoor environment using qr codes. **https://www. svf. stuba. sk/buxus/docs/web_katedry/gde/ingeo/TS4-04_Ilkovicova. pdf (27.01. 14)**, 2014.

Jisha e Philip 2016JISHA, S.; PHILIP, M. Rfid based security platform for internet of things in health care environment. p. 1–3, 2016.

Kanan e Elhassan 2015KANAN, R.; ELHASSAN, O. A combined batteryless radio and wifi indoor positioning system. p. 101–107, 2015.

Kok, Hol e Schön 2015KOK, M.; HOL, J. D.; SCHÖN, T. B. Indoor positioning using ultrawideband and inertial measurements. **IEEE Transactions on Vehicular Technology**, IEEE, v. 64, n. 4, p. 1293–1303, 2015.

Li e Huang 2018LI, Z.; HUANG, J. Study on the use of qr codes as landmarks for indoor positioning: Preliminary results. In: IEEE. **2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)**. [S.l.], 2018. p. 1270–1276.

Lin *et al.* 2015LIN, X.-Y.; HO, T.-W.; FANG, C.-C.; YEN, Z.-S.; YANG, B.-J.; LAI, F. A mobile indoor positioning system based on ibeacon technology. p. 4970–4973, 2015.

Mautz 2012MAUTZ, R. Indoor positioning technologies. ETH Zurich, Department of Civil, Environmental and Geomatic Engineering . . . , 2012.

MavenRepository 2021MAVENREPOSITORY. **MavenRepository**. 2021. Disponível em: <https://mvnrepository.com/>. Acesso em: 20 Jun. 2021.

Nakazawa *et al.* 2013NAKAZAWA, Y.; MAKINO, H.; NISHIMORI, K.; WAKATSUKI, D.; KOMAGATA, H. Indoor positioning using a high-speed, fish-eye lens-equipped camera in visible light communication. p. 1–8, 2013.

Nikitović e Mahmutović 2019NIKITOVIć, M.; MAHMUTOVIć, A. Hidden costs of erp implementation. p. 1314–1318, 2019.

Orosz, Selmeci e Orosz 2019OROSZ, I.; SELMECI, A.; OROSZ, T. Software as a service operation model in cloud based erp systems. In: IEEE. **2019 IEEE 17th World Symposium on Applied Machine Intelligence and Informatics (SAMI)**. [S.l.], 2019. p. 345–354.

Pasku *et al.* 2017PASKU, V.; ANGELIS, A. D.; ANGELIS, G. D.; ARUMUGAM, D. D.; DIONIGI, M.; CARBONE, P.; MOSCHITTA, A.; RICKETTS, D. S. Magnetic field-based positioning systems. **IEEE Communications Surveys & Tutorials**, IEEE, v. 19, n. 3, p. 2003–2017, 2017.

Qi e Liu 2017QI, J.; LIU, G.-P. A robust high-accuracy ultrasound indoor positioning system based on a wireless sensor network. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 17, n. 11, p. 2554, 2017.

Rocha e Freixo 2015ROCHA, Á.; FREIXO, J. Information architecture for quality management support in hospitals. **Journal of medical systems**, Springer, v. 39, n. 10, p. 125, 2015.

Saa *et al.* 2017SAA, P.; MOSCOSO-ZEA, O.; COSTALES, A. C.; LUJÁN-MORA, S. Data security issues in cloud-based software-as-a-service erp. In: IEEE. **2017 12th Iberian Conference on Information Systems and Technologies (CISTI)**. [S.l.], 2017. p. 1–7.

SpringDocs 2021SPRINGDOCS. **SpringDocs**. 2021. Disponível em: <https://docs.spring.io/>. Acesso em: 20 Jun. 2021.

Vidhyalakshmi e Kumar 2014VIDHYALAKSHMI, R.; KUMAR, V. Design comparison of traditional application and saas. In: IEEE. **2014 International Conference on Computing for Sustainable Global Development (INDIACom)**. [S.l.], 2014. p. 541–544.

WHO 2020WHO. **Management and Quality**. 2020. Disponível em: <https://www.who.int/hospitals/management-and-quality/en>. Acesso em: 20 fev. 2020.

Xu, Gong e Xu 2018XU, J.; GONG, C.; XU, Z. Experimental indoor visible light positioning systems with centimeter accuracy based on a commercial smartphone camera. **IEEE Photonics Journal**, IEEE, v. 10, n. 6, p. 1–17, 2018.

Yoo *et al.* 2018YOO, S.; KIM, S.; KIM, E.; JUNG, E.; LEE, K.-H.; HWANG, H. Real-time location system-based asset tracking in the healthcare field: lessons learned from a feasibility study. **BMC medical informatics and decision making**, Springer, v. 18, n. 1, p. 1–10, 2018.

APÊNDICE A - DOCUMENTAÇÃO SWAGGER API

Versão: v1.0

AreaController

atualizarArea

PUT

/area/{id}

Usage and SDK Samples

Curl

```
curl -X PUT\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/area/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	nome: string
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}

cadastrarArea

POST

/area

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/area"
```

Parameters

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	nome: string
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}

detalharArea

GET

/area/{id}

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/area/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}

listarAreas

GET

/area

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/area?paginacao="
```

Parameters

Query parameters

Name	Description
paginacao*	PaginacaoDTO Required

Responses

Status: 200 - OK

Schema

▼ {
totalPages: integer (int32)
totalElements: integer (int64)
size: integer (int32)
content: ▼ [
▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
]
number: integer (int32)
sort: ▼ {
empty: boolean
sorted: boolean
unsorted: boolean
}
numberOfElements: integer (int32)
pageable: ▼ {
offset: integer (int64)
sort: ▼ {
empty: boolean
sorted: boolean
unsorted: boolean
}
paged: boolean
pageNumber: integer (int32)
pageSize: integer (int32)
unpaged: boolean
}
first: boolean
last: boolean
empty: boolean
}

removerArea

DELETE

/area/{id}

Usage and SDK Samples

Curl

```
curl -X DELETE\
-H "Accept: */*"\
"http://api.movvo.app.br/area/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

boolean

AutenticacaoController

alterarSenha

POST

/autenticacao/alterar-senha

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/autenticacao/alterar-senha"
```

Parameters

Header parameters

Name	Description
Authorization*	String Required

Body parameters

Name	Description

body *	▼ {
	usuario: string
	senha: string
	novaSenha: string
	}

Responses

Status: 200 - OK

Schema

boolean

cadastrarUsuario

POST

/autenticacao/cadastrar

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/autenticacao/cadastrar"
```

Parameters

Body parameters

Name	Description

body *	▼ {
	usuario: string
	nome: string
	email: string
	senha: string
	confirmacaoSenha: string
	}

Responses

Status: 200 - OK

Schema

▼ {
usuario: string
nome: string
email: string
senha: string
confirmacaoSenha: string
}

confirmarCadastro

POST

/autenticacao/confirmar-cadastro

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/autenticacao/confirmar-cadastro"
```

Parameters

Body parameters

Name	Description

body *	▼ {
	usuario: string
	codigoConfirmacao: string
	}

Responses

Status: 200 - OK

Schema

boolean

confirmarNovaSenha

POST

/autenticacao/confirmar-nova-senha

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/autenticacao/confirmar-nova-senha"
```

Parameters

Body parameters

Name	Description

body *	▼ {
	usuario: string
	novaSenha: string
	codigoConfirmacao: string
	}

Responses

Status: 200 - OK

Schema

boolean

login

POST

/autenticacao/login

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/autenticacao/login"
```

Parameters

Body parameters

Name	Description

body *	▼ {
	usuario: string
	senha: string
	}

Responses

Status: 200 - OK

Schema

▼ {
accessToken: string
refreshToken: string
}

recuperarSenha

POST

/autenticacao/recuperar-senha

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
"http://api.movvo.app.br/autenticacao/recuperar-senha?usuario="
```

Parameters

Query parameters

Name	Description
usuario*	String Required

Responses

Status: 200 - OK

Schema

boolean

EquipamentoController

atualizarEquipamento

PUT

/equipamento/{id}

Usage and SDK Samples

Curl

```
curl -X PUT\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/equipamento/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Body parameters

Name	Description

body *

```
▼ {
    id:                integer (int64)
    tipo:              ▼ {
                         id:          integer
                         nome:        string
                         createdAt:   string
                         updatedAt:   string
                       }
    nome:              string
    codigo:            string
    fabricacao:        string (date-time)
    periodoManutencao: string
    ultimaManutencao:  string (date-time)
    transmissor:       ▼ {
                         id:          integer
                         tipo:        ► { }
                         nome:        string
                         codigo:      string
                         statusAtual: ► { }
                         observacao:  string
                         createdAt:   string
                         updatedAt:   string
                       }
    statusAtual:       ▼ {
                         id:          integer
                         status:      string
                         equipamento: integer
                         createdAt:   string
                         updatedAt:   string
                       }
    observacao:        string
    createdAt:         string (date-time)
    updatedAt:         string (date-time)
}
```

Responses

Status: 200 - OK

Schema

```
▼ {
    id:                integer (int64)
    tipo:              ▼ {
                         id:          integer (int64)
                         nome:        string
                         createdAt:   string (date-time)
                         updatedAt:   string (date-time)
                       }
    nome:              string
    codigo:            string
    fabricacao:        string (date-time)
    periodoManutencao: string
    ultimaManutencao:  string (date-time)
    transmissor:       ▼ {
                         id:          integer (int64)
                         tipo:        ▼ {
                                        id:          integer
                                        nome:        string
                                        createdAt:   string
                                        updatedAt:   string
                                      }
                         nome:        string
                         codigo:      string
                         statusAtual: ▼ {
                                        id:          integer
                                        status:      string
                                        transmissor: integer
                                        receptor:    ► { }
                                        createdAt:   string
                                        updatedAt:   string
                                      }
                         observacao:  string
                         createdAt:   string (date-time)
                         updatedAt:   string (date-time)
                       }
    statusAtual:       ▼ {
                         id:          integer (int64)
```

```
                status:         string
                                Enum: Livre , Ocupado , Manutencao , Danificado , Outros
                equipamento:    integer (int64)
                createdAt:      string (date-time)
                updatedAt:      string (date-time)
            }
        observacao:     string
        createdAt:      string (date-time)
        updatedAt:      string (date-time)
}
```

buscarEquipamentos

GET

/equipamento/buscar

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/equipamento/buscar?term=&paginacao="
```

Parameters

Query parameters

Name	Description
term*	String Required
paginacao*	PaginacaoDTO Required

Responses

Status: 200 - OK

Schema

```
▼ {
    totalPages:     integer (int32)
    totalElements:  integer (int64)
    size:           integer (int32)
    content:        ▼ [
                        ▼ {
                            id:                 integer (int64)
                            tipo:               ▼ {
                                                    id:         integer (int64)
                                                    nome:       string
                                                    createdAt:  string (date-time)
                                                    updatedAt:  string (date-time)
                                                }
                            nome:               string
                            codigo:             string
                            fabricacao:         string (date-time)
                            periodoManutencao:  string
                            ultimaManutencao:   string (date-time)
                            transmissor:        ▼ {
                                                    id:         integer (int64)
                                                    tipo:       ▼ {
                                                                    id:         integer (int64)
                                                                    nome:       string
                                                                    createdAt:  string (date-time)
                                                                    updatedAt:  string (date-time)
                                                                }
                                                    nome:       string
                                                    codigo:     string
                                                    statusAtual: ▼ {
                                                                    id:         integer (int64)
                                                                    status:     string
                                                                                Enum: SinalForte , SinalFraco , BateriaFraca , SemSinal
                                                                    transmissor: integer (int64)
                                                                    receptor:   ▼ {
                                                                                    id:         integer (int64)
```

```
                                                tipo:                ▼ {
                                                                       id:          integer (int64)
                                                                       nome:        string
                                                                       createdAt:   string (date-time)
                                                                       updatedAt:   string (date-time)
                                                                     }
                                                nome:                string
                                                codigo:              string
                                                local:               ▼ {
                                                                       id:          integer (int64)
                                                                       nome:        string
                                                                       area:        ▼ {
                                                                                      id:          integer (int64)
                                                                                      nome:        string
                                                                                      createdAt:   string (date-time)
                                                                                      updatedAt:   string (date-time)
                                                                                    }
                                                                       createdAt:   string (date-time)
                                                                       updatedAt:   string (date-time)
                                                                     }
                                                localizacaoX:        number (double)
                                                localizacaoY:        number (double)
                                                statusAtual:         ▼ {
                                                                       id:          integer (int64)
                                                                       status:      string
                                                                                    Enum: SinalForte, BateriaFraca, SemSinal
                                                                       receptor:    integer (int64)
                                                                       createdAt:   string (date-time)
                                                                       updatedAt:   string (date-time)
                                                                     }
                                                observacao:          string
                                                createdAt:           string (date-time)
                                                updatedAt:           string (date-time)
                                              }
                                    createdAt:   string (date-time)
                                    updatedAt:   string (date-time)
                                  }
                        observacao:  string
                        createdAt:   string (date-time)
                        updatedAt:   string (date-time)
                      }
            statusAtual:  ▼ {
                            id:          integer (int64)
                            status:      string
                                         Enum: Livre, Ocupado, Manutencao, Danificado, Outros
                            equipamento: integer (int64)
                            createdAt:   string (date-time)
                            updatedAt:   string (date-time)
                          }
            observacao:   string
            createdAt:    string (date-time)
            updatedAt:    string (date-time)
          }
        ]
  number:           integer (int32)
  sort:             ▼ {
                      empty:     boolean
                      sorted:    boolean
                      unsorted:  boolean
                    }
  numberOfElements: integer (int32)
  pageable:         ▼ {
                      offset:    integer (int64)
                      sort:      ▼ {
                                   empty:     boolean
                                   sorted:    boolean
                                   unsorted:  boolean
                                 }
                      paged:     boolean
                      pageNumber: integer (int32)
                      pageSize:  integer (int32)
                      unpaged:   boolean
                    }
  first:            boolean
  last:             boolean
  empty:            boolean
}
```

cadastrarEquipamento

POST

/equipamento

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/equipamento"
```

Parameters

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	tipo: ▼ {
	id: integer
	nome: string
	createdAt: string
	updatedAt: string
	}
	nome: string
	codigo: string
	fabricacao: string (date-time)
	periodoManutencao: string
	ultimaManutencao: string (date-time)
	transmissor: ▼ {
	id: integer
	tipo: ► { }
	nome: string
	codigo: string
	statusAtual: ► { }
	observacao: string
	createdAt: string
	updatedAt: string
	}
	statusAtual: ▼ {
	id: integer
	status: string
	equipamento: integer
	createdAt: string
	updatedAt: string
	}
	observacao: string
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
tipo: ▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
nome: string
codigo: string
fabricacao: string (date-time)
periodoManutencao: string
ultimaManutencao: string (date-time)
transmissor: ▼ {
id: integer (int64)

```
                tipo:           ▼ {
                                    id:         integer
                                    nome:       string
                                    createdAt:  string
                                    updatedAt:  string
                                }
                nome:           string
                codigo:         string
                statusAtual:    ▼ {
                                    id:         integer
                                    status:     string
                                    transmissor: integer
                                    receptor:   ► { }
                                    createdAt:  string
                                    updatedAt:  string
                                }
                observacao:     string
                createdAt:      string (date-time)
                updatedAt:      string (date-time)
            }
    statusAtual:        ▼ {
                id:         integer (int64)
                status:     string
                            Enum: Livre , Ocupado , Manutencao , Danificado , Outros
                equipamento: integer (int64)
                createdAt:  string (date-time)
                updatedAt:  string (date-time)
            }
    observacao:         string
    createdAt:          string (date-time)
    updatedAt:          string (date-time)
}
```


desocuparEquipamento

POST

/equipamento/desocupar/{id}

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
"http://api.movvo.app.br/equipamento/desocupar/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema


```
▼ {
    id:             integer (int64)
    tipo:           ▼ {
                        id:         integer (int64)
                        nome:       string
                        createdAt:  string (date-time)
                        updatedAt:  string (date-time)
                    }
    nome:           string
    codigo:         string
    fabricacao:     string (date-time)
    periodoManutencao: string
    ultimaManutencao: string (date-time)
    transmissor:    ▼ {
```

```
                id:             integer (int64)
                tipo:           ▼ {
                                    id:         integer
                                    nome:       string
                                    createdAt:  string
                                    updatedAt:  string
                                }
                nome:           string
                codigo:         string
                statusAtual:    ▼ {
                                    id:             integer
                                    status:         string
                                    transmissor:    integer
                                    receptor:       ► { }
                                    createdAt:      string
                                    updatedAt:      string
                                }
                observacao:     string
                createdAt:      string (date-time)
                updatedAt:      string (date-time)
            }
    statusAtual:        ▼ {
                id:             integer (int64)
                status:         string
                                Enum: Livre , Ocupado , Manutencao , Danificado , Outros
                equipamento:    integer (int64)
                createdAt:      string (date-time)
                updatedAt:      string (date-time)
            }
    observacao:         string
    createdAt:          string (date-time)
    updatedAt:          string (date-time)
}
```

detalharEquipamento

GET

/equipamento/{id}

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/equipamento/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

```
▼ {
    id:                 integer (int64)
    tipo:               ▼ {
                            id:         integer (int64)
                            nome:       string
                            createdAt:  string (date-time)
                            updatedAt:  string (date-time)
                        }
    nome:               string
    codigo:             string
    fabricacao:         string (date-time)
    periodoManutencao:  string
    ultimaManutencao:   string (date-time)
```

```
transmissor:    ▼ {
                    id:          integer (int64)
                    tipo:        ▼ {
                                     id:          integer
                                     nome:        string
                                     createdAt:   string
                                     updatedAt:   string
                                 }
                    nome:        string
                    codigo:      string
                    statusAtual: ▼ {
                                     id:          integer
                                     status:      string
                                     transmissor: integer
                                     receptor:    ► { }
                                     createdAt:   string
                                     updatedAt:   string
                                 }
                    observacao:  string
                    createdAt:   string (date-time)
                    updatedAt:   string (date-time)
                }
statusAtual:    ▼ {
                    id:          integer (int64)
                    status:      string
                                 Enum: Livre, Ocupado, Manutencao, Danificado, Outros
                    equipamento: integer (int64)
                    createdAt:   string (date-time)
                    updatedAt:   string (date-time)
                }
observacao:     string
createdAt:      string (date-time)
updatedAt:      string (date-time)
}
```

listarEquipamentos

GET

/equipamento

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/equipamento?paginacao="
```

Parameters

Query parameters

Name	Description
paginacao*	PaginacaoDTO Required

Responses

Status: 200 - OK

Schema

```
▼ {
totalPages:     integer (int32)
totalElements:  integer (int64)
size:           integer (int32)
content:        ▼ [
                    ▼ {
                        id:          integer (int64)
                        tipo:        ▼ {
                                         id:          integer (int64)
                                         nome:        string
                                         createdAt:   string (date-time)
                                         updatedAt:   string (date-time)
```

```
                    }
                nome: string
                codigo: string
                fabricacao: string (date-time)
                periodoManutencao: string
                ultimaManutencao: string (date-time)
                transmissor: ▼ {
                    id: integer (int64)
                    tipo: ▼ {
                        id: integer (int64)
                        nome: string
                        createdAt: string (date-time)
                        updatedAt: string (date-time)
                    }
                    nome: string
                    codigo: string
                    statusAtual: ▼ {
                        id: integer (int64)
                        status: string
                            Enum: SinalForte , SinalFraco , BateriaFraca , SemSinal
                        transmissor: integer (int64)
                        receptor: ▼ {
                            id: integer (int64)
                            tipo: ▼ {
                                id: integer (int64)
                                nome: string
                                createdAt: string (date-time)
                                updatedAt: string (date-time)
                            }
                            nome: string
                            codigo: string
                            local: ▼ {
                                id: integer (int64)
                                nome: string
                                area: ▼ {
                                    id: integer (int64)
                                    nome: string
                                    createdAt: string (date-time)
                                    updatedAt: string (date-time)
                                }
                                createdAt: string (date-time)
                                updatedAt: string (date-time)
                            }
                            localizacaoX: number (double)
                            localizacaoY: number (double)
                            statusAtual: ▼ {
                                id: integer (int64)
                                status: string
                                    Enum: SinalForte , BateriaFraca , SemSinal
                                receptor: integer (int64)
                                createdAt: string (date-time)
                                updatedAt: string (date-time)
                            }
                            observacao: string
                            createdAt: string (date-time)
                            updatedAt: string (date-time)
                        }
                        createdAt: string (date-time)
                        updatedAt: string (date-time)
                    }
                    observacao: string
                    createdAt: string (date-time)
                    updatedAt: string (date-time)
                }
                statusAtual: ▼ {
                    id: integer (int64)
                    status: string
                        Enum: Livre , Ocupado , Manutencao , Danificado , Outros
                    equipamento: integer (int64)
                    createdAt: string (date-time)
                    updatedAt: string (date-time)
                }
                observacao: string
                createdAt: string (date-time)
                updatedAt: string (date-time)
            }
        ]
    number: integer (int32)
    sort: ▼ {
        empty: boolean
```

```
                sorted:    boolean
                unsorted:  boolean
            }
        numberOfElements:  integer (int32)
        pageable:          ▼ {
                offset:        integer (int64)
                sort:          ▼ {
                                   empty:     boolean
                                   sorted:    boolean
                                   unsorted:  boolean
                               }
                paged:         boolean
                pageNumber:    integer (int32)
                pageSize:      integer (int32)
                unpaged:       boolean
            }
        first:             boolean
        last:              boolean
        empty:             boolean
    }
```


removerEquipamento

DELETE

/equipamento/{id}

Usage and SDK Samples

Curl

```
curl -X DELETE\
-H "Accept: */*"\
"http://api.movvo.app.br/equipamento/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

boolean

utilizarEquipamento

POST

/equipamento/utilizar/{id}

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
"http://api.movvo.app.br/equipamento/utilizar/{id}"
```

Parameters

Path parameters

Name	Description

id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
tipo: ▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
nome: string
codigo: string
fabricacao: string (date-time)
periodoManutencao: string
ultimaManutencao: string (date-time)
transmissor: ▼ {
id: integer (int64)
tipo: ▼ {
id: integer
nome: string
createdAt: string
updatedAt: string
}
nome: string
codigo: string
statusAtual: ▼ {
id: integer
status: string
transmissor: integer
receptor: ► { }
createdAt: string
updatedAt: string
}
observacao: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
statusAtual: ▼ {
id: integer (int64)
status: string
Enum: Livre , Ocupado , Manutencao , Danificado , Outros
equipamento: integer (int64)
createdAt: string (date-time)
updatedAt: string (date-time)
}
observacao: string
createdAt: string (date-time)
updatedAt: string (date-time)
}

EquipamentoStatusController

atualizarStatusEquipamento

PUT

/equipamento-status/{id}

Usage and SDK Samples

Curl

```
curl -X PUT\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/equipamento-status/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	status: string
	Enum: `Livre` , `Ocupado` , `Manutencao` , `Danificado` , `Outros`
	equipamento: integer (int64)
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
status: string
Enum: `Livre` , `Ocupado` , `Manutencao` , `Danificado` , `Outros`
equipamento: integer (int64)
createdAt: string (date-time)
updatedAt: string (date-time)
}

cadastrarStatusEquipamento

POST

/equipamento-status

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/equipamento-status"
```

Parameters

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	status: string
	Enum: `Livre` , `Ocupado` , `Manutencao` , `Danificado` , `Outros`
	equipamento: integer (int64)
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
status: string
Enum: `Livre` , `Ocupado` , `Manutencao` , `Danificado` , `Outros`
equipamento: integer (int64)

createdAt: string (date-time)
updatedAt: string (date-time)
}

detalharStatusEquipamento

GET

/equipamento-status/{id}

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/equipamento-status/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
status: string
Enum: `Livre` , `Ocupado` , `Manutencao` , `Danificado` , `Outros`
equipamento: integer (int64)
createdAt: string (date-time)
updatedAt: string (date-time)
}

listarStatusEquipamentos

GET

/equipamento-status

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/equipamento-status?paginacao="
```

Parameters

Query parameters

Name	Description
paginacao*	PaginacaoDTO Required

Responses

Status: 200 - OK

Schema

▼ {

```
totalPages:      integer (int32)
totalElements:   integer (int64)
size:            integer (int32)
content:         ▼ [
                   ▼ {
                       id:           integer (int64)
                       status:       string
                                       Enum: Livre , Ocupado , Manutencao , Danificado , Outros
                       equipamento:  integer (int64)
                       createdAt:    string (date-time)
                       updatedAt:    string (date-time)
                     }
                 ]
number:          integer (int32)
sort:            ▼ {
                     empty:    boolean
                     sorted:   boolean
                     unsorted: boolean
                 }
numberOfElements: integer (int32)
pageable:        ▼ {
                     offset:   integer (int64)
                     sort:     ▼ {
                                   empty:    boolean
                                   sorted:   boolean
                                   unsorted: boolean
                               }
                     paged:    boolean
                     pageNumber: integer (int32)
                     pageSize: integer (int32)
                     unpaged:  boolean
                 }
first:           boolean
last:            boolean
empty:           boolean
}
```

removerStatusEquipamento

DELETE

/equipamento-status/{id}

Usage and SDK Samples

Curl

```
curl -X DELETE\
-H "Accept: */*"\
"http://api.movvo.app.br/equipamento-status/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

boolean

EquipamentoTipoController

atualizarTipoEquipamento

PUT

/equipamento-tipo/{id}

Usage and SDK Samples

Curl

```
curl -X PUT\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/equipamento-tipo/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	nome: string
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}

cadastrarTipoEquipamento

POST

/equipamento-tipo

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/equipamento-tipo"
```

Parameters

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	nome: string
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}

detalharTipoEquipamento

GET

/equipamento-tipo/{id}

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/equipamento-tipo/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}

listarTiposEquipamentos

GET

/equipamento-tipo

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/equipamento-tipo?paginacao="
```

Parameters

Query parameters

Name	Description
paginacao*	PaginacaoDTO Required

Responses

Status: 200 - OK

Schema


```
▼ {
    totalPages:          integer (int32)
    totalElements:       integer (int64)
    size:                integer (int32)
    content:             ▼ [
                           ▼ {
                               id:          integer (int64)
                               nome:        string
                               createdAt:   string (date-time)
                               updatedAt:   string (date-time)
                             }
                         ]
    number:              integer (int32)
    sort:                ▼ {
                             empty:     boolean
                             sorted:    boolean
                             unsorted:  boolean
                         }
    numberOfElements:    integer (int32)
    pageable:            ▼ {
                             offset:        integer (int64)
                             sort:          ▼ {
                                               empty:     boolean
                                               sorted:    boolean
                                               unsorted:  boolean
                                             }
                             paged:         boolean
                             pageNumber:    integer (int32)
                             pageSize:      integer (int32)
                             unpaged:       boolean
                         }
    first:               boolean
    last:                boolean
    empty:               boolean
}
```


removerTipoEquipamento

DELETE

/equipamento-tipo/{id}

Usage and SDK Samples

Curl

```
curl -X DELETE\
-H "Accept: */*"\
"http://api.movvo.app.br/equipamento-tipo/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

boolean

LocalController

atualizarLocal

PUT

/local/{id}

Usage and SDK Samples

Curl

```
curl -X PUT\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/local/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	nome: string
	area: ▼ {
	id: integer
	nome: string
	createdAt: string
	updatedAt: string
	}
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
nome: string
area: ▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
createdAt: string (date-time)
updatedAt: string (date-time)
}

buscarLocais

GET

/local/buscar

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/local/buscar?term=&paginacao="
```

Parameters

Query parameters

Name	Description
term*	String Required
paginacao*	PaginacaoDTO Required

Responses

Status: 200 - OK

Schema

▼ {
totalPages: integer (int32)
totalElements: integer (int64)
size: integer (int32)
content: ▼ [
▼ {
id: integer (int64)
nome: string
area: ▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
createdAt: string (date-time)
updatedAt: string (date-time)
}
]
number: integer (int32)
sort: ▼ {
empty: boolean
sorted: boolean
unsorted: boolean
}
numberOfElements: integer (int32)
pageable: ▼ {
offset: integer (int64)
sort: ▼ {
empty: boolean
sorted: boolean
unsorted: boolean
}
paged: boolean
pageNumber: integer (int32)
pageSize: integer (int32)
unpaged: boolean
}
first: boolean
last: boolean
empty: boolean
}

cadastrarLocal

POST

/local

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/local"
```

Parameters

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	nome: string
	area: ▼ {
	id: integer
	nome: string
	createdAt: string
	updatedAt: string
	}
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
nome: string
area: ▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
createdAt: string (date-time)
updatedAt: string (date-time)
}

detalharLocal

GET

/local/{id}

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/local/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
nome: string
area: ▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
createdAt: string (date-time)
updatedAt: string (date-time)
}

listarLocais

GET

/local

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/local?paginacao="
```

Parameters

Query parameters

Name	Description
paginacao*	PaginacaoDTO Required

Responses

Status: 200 - OK

Schema


```
▼ {
    totalPages:         integer (int32)
    totalElements:      integer (int64)
    size:               integer (int32)
    content:            ▼ [
                          ▼ {
                              id:          integer (int64)
                              nome:        string
                              area:        ▼ {
                                             id:          integer (int64)
                                             nome:        string
                                             createdAt:   string (date-time)
                                             updatedAt:   string (date-time)
                                           }
                              createdAt:   string (date-time)
                              updatedAt:   string (date-time)
                            }
                        ]
    number:             integer (int32)
    sort:               ▼ {
                            empty:    boolean
                            sorted:   boolean
                            unsorted: boolean
                        }
    numberOfElements:   integer (int32)
    pageable:           ▼ {
                            offset:     integer (int64)
                            sort:       ▼ {
                                          empty:    boolean
                                          sorted:   boolean
                                          unsorted: boolean
                                        }
                            paged:      boolean
                            pageNumber: integer (int32)
                            pageSize:   integer (int32)
                            unpaged:    boolean
                        }
    first:              boolean
    last:               boolean
    empty:              boolean
}
```


removerLocal

DELETE

/local/{id}

Usage and SDK Samples

Curl

```
curl -X DELETE\
-H "Accept: */*"\
"http://api.movvo.app.br/local/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

boolean

ReceptorController

atualizarReceptor

PUT

/receptor/{id}

Usage and SDK Samples

Curl

```
curl -X PUT\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/receptor/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	tipo: ▼ {
	id: integer
	nome: string
	createdAt: string
	updatedAt: string
	}
	nome: string
	codigo: string
	local: ▼ {
	id: integer
	nome: string
	area: ► { }
	createdAt: string
	updatedAt: string
	}
	localizacaoX: number (double)
	localizacaoY: number (double)
	statusAtual: ▼ {
	id: integer
	status: string
	receptor: integer
	createdAt: string
	updatedAt: string
	}
	observacao: string
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
tipo: ▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
nome: string
codigo: string
local: ▼ {
id: integer (int64)
nome: string
area: ▼ {
id: integer
nome: string
createdAt: string
updatedAt: string
}
createdAt: string (date-time)
updatedAt: string (date-time)
}
localizacaoX: number (double)
localizacaoY: number (double)
statusAtual: ▼ {
id: integer (int64)
status: string
Enum: `SinalForte`, `BateriaFraca`, `SemSinal`
receptor: integer (int64)
createdAt: string (date-time)
updatedAt: string (date-time)
}
observacao: string
createdAt: string (date-time)
updatedAt: string (date-time)
}

buscarReceptores

GET

/receptor/buscar

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/receptor/buscar?term=&paginacao="
```

Parameters

Query parameters

Name	Description
term*	String Required
paginacao*	PaginacaoDTO Required

Responses

Status: 200 - OK

Schema

```
▼ {
  totalPages: integer (int32)
  totalElements: integer (int64)
  size: integer (int32)
  content: ▼ [
    ▼ {
      id: integer (int64)
      tipo: ▼ {
        id: integer (int64)
        nome: string
        createdAt: string (date-time)
        updatedAt: string (date-time)
      }
      nome: string
      codigo: string
      local: ▼ {
        id: integer (int64)
        nome: string
        area: ▼ {
          id: integer (int64)
          nome: string
          createdAt: string (date-time)
          updatedAt: string (date-time)
        }
        createdAt: string (date-time)
        updatedAt: string (date-time)
      }
      localizacaoX: number (double)
      localizacaoY: number (double)
      statusAtual: ▼ {
        id: integer (int64)
        status: string
          Enum: SinalForte , BateriaFraca , SemSinal
        receptor: integer (int64)
        createdAt: string (date-time)
        updatedAt: string (date-time)
      }
      observacao: string
      createdAt: string (date-time)
      updatedAt: string (date-time)
    }
  ]
  number: integer (int32)
  sort: ▼ {
    empty: boolean
    sorted: boolean
    unsorted: boolean
  }
  numberOfElements: integer (int32)
  pageable: ▼ {
```

```
                offset:          integer (int64)
                sort:            ▼ {
                                   empty:    boolean
                                   sorted:   boolean
                                   unsorted: boolean
                                 }
                paged:           boolean
                pageNumber:      integer (int32)
                pageSize:        integer (int32)
                unpaged:         boolean
              }
    first:    boolean
    last:     boolean
    empty:    boolean
}
```

cadastrarReceptor

POST

/receptor

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/receptor"
```

Parameters

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	tipo: ▼ {
	id: integer
	nome: string
	createdAt: string
	updatedAt: string
	}
	nome: string
	codigo: string
	local: ▼ {
	id: integer
	nome: string
	area: ► { }
	createdAt: string
	updatedAt: string
	}
	localizacaoX: number (double)
	localizacaoY: number (double)
	statusAtual: ▼ {
	id: integer
	status: string
	receptor: integer
	createdAt: string
	updatedAt: string
	}
	observacao: string
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

```
▼ {
  id:    integer (int64)
```

```
    tipo:           ▼ {
                      id:           integer (int64)
                      nome:         string
                      createdAt:    string (date-time)
                      updatedAt:    string (date-time)
                    }
    nome:           string
    codigo:         string
    local:          ▼ {
                      id:           integer (int64)
                      nome:         string
                      area:         ▼ {
                                      id:           integer
                                      nome:         string
                                      createdAt:    string
                                      updatedAt:    string
                                    }
                      createdAt:    string (date-time)
                      updatedAt:    string (date-time)
                    }
    localizacaoX:   number (double)
    localizacaoY:   number (double)
    statusAtual:    ▼ {
                      id:           integer (int64)
                      status:       string
                                    Enum:   SinalForte , BateriaFraca , SemSinal
                      receptor:     integer (int64)
                      createdAt:    string (date-time)
                      updatedAt:    string (date-time)
                    }
    observacao:     string
    createdAt:      string (date-time)
    updatedAt:      string (date-time)
}
```

detalharReceptor

GET

/receptor/{id}

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/receptor/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

```
▼ {
    id:           integer (int64)
    tipo:         ▼ {
                    id:           integer (int64)
                    nome:         string
                    createdAt:    string (date-time)
                    updatedAt:    string (date-time)
                  }
    nome:         string
    codigo:       string
    local:        ▼ {
                    id:           integer (int64)
```

```
                nome:       string
                area:       ▼ {
                              id:         integer
                              nome:       string
                              createdAt:  string
                              updatedAt:  string
                            }
                createdAt:  string (date-time)
                updatedAt:  string (date-time)
            }
    localizacaoX:  number (double)
    localizacaoY:  number (double)
    statusAtual:   ▼ {
                id:         integer (int64)
                status:     string
                              Enum:  SinalForte , BateriaFraca , SemSinal
                receptor:   integer (int64)
                createdAt:  string (date-time)
                updatedAt:  string (date-time)
            }
    observacao:    string
    createdAt:     string (date-time)
    updatedAt:     string (date-time)
}
```

listarReceptores

GET

/receptor

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/receptor?paginacao="
```

Parameters

Query parameters

Name	Description
paginacao*	PaginacaoDTO Required

Responses

Status: 200 - OK

Schema

```
▼ {
    totalPages:     integer (int32)
    totalElements:  integer (int64)
    size:           integer (int32)
    content:        ▼ [
                      ▼ {
                          id:     integer (int64)
                          tipo:   ▼ {
                                    id:         integer (int64)
                                    nome:       string
                                    createdAt:  string (date-time)
                                    updatedAt:  string (date-time)
                                  }
                          nome:   string
                          codigo: string
                          local:  ▼ {
                                    id:     integer (int64)
                                    nome:   string
                                    area:   ▼ {
                                              id:         integer (int64)
                                              nome:       string
                                              createdAt:  string (date-time)
```

```
                                        updatedAt: string (date-time)
                                    }
                                createdAt: string (date-time)
                                updatedAt: string (date-time)
                            }
                        localizacaoX: number (double)
                        localizacaoY: number (double)
                        statusAtual: ▼ {
                                id: integer (int64)
                                status: string
                                    Enum: SinalForte , BateriaFraca , SemSinal
                                receptor: integer (int64)
                                createdAt: string (date-time)
                                updatedAt: string (date-time)
                            }
                        observacao: string
                        createdAt: string (date-time)
                        updatedAt: string (date-time)
                    }
                ]
    number: integer (int32)
    sort: ▼ {
            empty: boolean
            sorted: boolean
            unsorted: boolean
        }
    numberOfElements: integer (int32)
    pageable: ▼ {
            offset: integer (int64)
            sort: ▼ {
                    empty: boolean
                    sorted: boolean
                    unsorted: boolean
                }
            paged: boolean
            pageNumber: integer (int32)
            pageSize: integer (int32)
            unpaged: boolean
        }
    first: boolean
    last: boolean
    empty: boolean
}
```


removerReceptor

DELETE

/receptor/{id}

Usage and SDK Samples

Curl

```
curl -X DELETE\
-H "Accept: */*"\
"http://api.movvo.app.br/receptor/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

boolean

ReceptorStatusController

atualizarStatusReceptor

PUT

/receptor-status/{id}

Usage and SDK Samples

Curl

```
curl -X PUT\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/receptor-status/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	status: string
	Enum: `SinalForte` , `BateriaFraca` , `SemSinal`
	receptor: integer (int64)
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
status: string
Enum: `SinalForte` , `BateriaFraca` , `SemSinal`
receptor: integer (int64)
createdAt: string (date-time)
updatedAt: string (date-time)
}

cadastrarStatusReceptor

POST

/receptor-status

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/receptor-status"
```

Parameters

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	status: string
	Enum: `SinalForte` , `BateriaFraca` , `SemSinal`
	receptor: integer (int64)
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
status: string
Enum: `SinalForte` , `BateriaFraca` , `SemSinal`
receptor: integer (int64)
createdAt: string (date-time)
updatedAt: string (date-time)
}

detalharStatusReceptor

GET

/receptor-status/{id}

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/receptor-status/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
status: string
Enum: `SinalForte` , `BateriaFraca` , `SemSinal`
receptor: integer (int64)
createdAt: string (date-time)
updatedAt: string (date-time)
}

listarStatusReceptores

GET

/receptor-status

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/receptor-status?paginacao="
```

Parameters

Query parameters

Name	Description
paginacao*	PaginacaoDTO Required

Responses

Status: 200 - OK

Schema

▼ {
totalPages: integer (int32)
totalElements: integer (int64)
size: integer (int32)
content: ▼ [
▼ {
id: integer (int64)
status: string
Enum: `SinalForte` , `BateriaFraca` , `SemSinal`
receptor: integer (int64)
createdAt: string (date-time)
updatedAt: string (date-time)
}
]
number: integer (int32)
sort: ▼ {
empty: boolean
sorted: boolean
unsorted: boolean
}
numberOfElements: integer (int32)
pageable: ▼ {
offset: integer (int64)
sort: ▼ {
empty: boolean
sorted: boolean
unsorted: boolean
}
paged: boolean
pageNumber: integer (int32)
pageSize: integer (int32)
unpaged: boolean
}
first: boolean
last: boolean
empty: boolean
}

removerStatusReceptor

DELETE

/receptor-status/{id}

Usage and SDK Samples

Curl

```
curl -X DELETE\
-H "Accept: */*"\
"http://api.movvo.app.br/receptor-status/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

boolean

ReceptorTipoController

atualizarTipoReceptor

PUT

/receptor-tipo/{id}

Usage and SDK Samples

Curl

```
curl -X PUT\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/receptor-tipo/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	nome: string
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}

cadastrarTipoReceptor

POST

/receptor-tipo

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/receptor-tipo"
```

Parameters

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	nome: string
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}

detalharTipoReceptor

GET

/receptor-tipo/{id}

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/receptor-tipo/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}

listarTiposReceptores

GET

/receptor-tipo

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/receptor-tipo?paginacao="
```

Parameters

Query parameters

Name	Description
paginacao*	PaginacaoDTO Required

Responses

Status: 200 - OK

Schema


```
▼ {
  totalPages: integer (int32)
  totalElements: integer (int64)
  size: integer (int32)
  content: ▼ [
    ▼ {
      id: integer (int64)
      nome: string
      createdAt: string (date-time)
      updatedAt: string (date-time)
    }
  ]
  number: integer (int32)
  sort: ▼ {
    empty: boolean
    sorted: boolean
    unsorted: boolean
  }
  numberOfElements: integer (int32)
  pageable: ▼ {
    offset: integer (int64)
    sort: ▼ {
      empty: boolean
      sorted: boolean
      unsorted: boolean
    }
    paged: boolean
    pageNumber: integer (int32)
    pageSize: integer (int32)
    unpaged: boolean
  }
  first: boolean
  last: boolean
  empty: boolean
}
```


removerTipoReceptor

DELETE

/receptor-tipo/{id}

Usage and SDK Samples

Curl

```
curl -X DELETE\
-H "Accept: */*"\
"http://api.movvo.app.br/receptor-tipo/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

boolean

TokenController

atualizarToken

GET

/token/atualizar

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/token/atualizar?refreshToken="
```

Parameters

Query parameters

Name	Description
refreshToken*	String Required

Responses

Status: 200 - OK

Schema

```
▼ {
    accessToken: string
    refreshToken: string
}
```

revogarToken

GET

/token/revogar

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/token/revogar?refreshToken="
```

Parameters

Query parameters

Name	Description
refreshToken*	String Required

Responses

Status: 200 - OK

Schema

boolean

validarToken

GET

/token/validar

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/token/validar"
```

Parameters

Header parameters

Name	Description
Authorization*	String Required

Responses

Status: 200 - OK

Schema

boolean

TransmissorController

atualizarTransmissor

PUT

/transmissor/{id}

Usage and SDK Samples

Curl

```
curl -X PUT\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/transmissor/{id}"
```

Parameters

Path parameters

Name	Description

id*	Long (int64) Required

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	tipo: ▼ {
	id: integer
	nome: string
	createdAt: string
	updatedAt: string
	}
	nome: string
	codigo: string
	statusAtual: ▼ {
	id: integer
	status: string
	transmissor: integer
	receptor: ► { }
	createdAt: string
	updatedAt: string
	}
	observacao: string
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
tipo: ▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
nome: string
codigo: string
statusAtual: ▼ {
id: integer (int64)
status: string
Enum: `SinalForte` , `SinalFraco` , `BateriaFraca` , `SemSinal`
transmissor: integer (int64)
receptor: ▼ {
id: integer
tipo: ► { }
nome: string
codigo: string
local: ► { }
localizacaoX: number
localizacaoY: number
statusAtual: ► { }
observacao: string
createdAt: string
updatedAt: string
}
createdAt: string (date-time)
updatedAt: string (date-time)
}
observacao: string
createdAt: string (date-time)
updatedAt: string (date-time)
}

buscarTransmissores

GET

/transmissor/buscar

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/transmissor/buscar?term=&paginacao="
```

Parameters

Query parameters

Name	Description
term*	String Required
paginacao*	PaginacaoDTO Required

Responses

Status: 200 - OK

Schema

▼ {
totalPages: integer (int32)
totalElements: integer (int64)
size: integer (int32)
content: ▼ [
▼ {
id: integer (int64)
tipo: ▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
nome: string
codigo: string
statusAtual: ▼ {
id: integer (int64)
status: string
Enum: `SinalForte` , `SinalFraco` , `BateriaFraca` , `SemSinal`
transmissor: integer (int64)
receptor: ▼ {
id: integer (int64)
tipo: ▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
nome: string
codigo: string
local: ▼ {
id: integer (int64)
nome: string
area: ▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
createdAt: string (date-time)
updatedAt: string (date-time)
}
localizacaoX: number (double)
localizacaoY: number (double)
statusAtual: ▼ {
id: integer (int64)
status: string
Enum: `SinalForte` , `BateriaFraca` , `SemSinal`
receptor: integer (int64)
createdAt: string (date-time)
updatedAt: string (date-time)
}
observacao: string
createdAt: string (date-time)

```
                                        updatedAt:    string (date-time)
                                    }
                                createdAt:    string (date-time)
                                updatedAt:    string (date-time)
                            }
                        observacao:  string
                        createdAt:   string (date-time)
                        updatedAt:   string (date-time)
                    }
                ]
    number:             integer (int32)
    sort:               ▼ {
                            empty:    boolean
                            sorted:   boolean
                            unsorted: boolean
                        }
    numberOfElements:   integer (int32)
    pageable:           ▼ {
                            offset:   integer (int64)
                            sort:     ▼ {
                                        empty:    boolean
                                        sorted:   boolean
                                        unsorted: boolean
                                      }
                            paged:    boolean
                            pageNumber: integer (int32)
                            pageSize: integer (int32)
                            unpaged:  boolean
                        }
    first:              boolean
    last:               boolean
    empty:              boolean
}
```


cadastrarTransmissor

POST

/transmissor

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/transmissor"
```

Parameters

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	tipo: ▼ {
	id: integer
	nome: string
	createdAt: string
	updatedAt: string
	}
	nome: string
	codigo: string
	statusAtual: ▼ {
	id: integer
	status: string
	transmissor: integer
	receptor: ► { }
	createdAt: string
	updatedAt: string
	}
	observacao: string
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema


```
▼ {
  id:           integer (int64)
  tipo:         ▼ {
                  id:           integer (int64)
                  nome:         string
                  createdAt:    string (date-time)
                  updatedAt:    string (date-time)
                }
  nome:         string
  codigo:       string
  statusAtual:  ▼ {
                  id:           integer (int64)
                  status:       string
                                Enum: SinalForte, SinalFraco, BateriaFraca, SemSinal
                  transmissor:  integer (int64)
                  receptor:     ▼ {
                                  id:           integer
                                  tipo:         ► { }
                                  nome:         string
                                  codigo:       string
                                  local:        ► { }
                                  localizacaoX: number
                                  localizacaoY: number
                                  statusAtual:  ► { }
                                  observacao:   string
                                  createdAt:    string
                                  updatedAt:    string
                                }
                  createdAt:    string (date-time)
                  updatedAt:    string (date-time)
                }
  observacao:   string
  createdAt:    string (date-time)
  updatedAt:    string (date-time)
}
```


detalharTransmissor

GET

/transmissor/{id}

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/transmissor/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema


```
▼ {
  id:           integer (int64)
  tipo:         ▼ {
                  id:           integer (int64)
```

```
                nome:       string
                createdAt:  string (date-time)
                updatedAt:  string (date-time)
            }
    nome:       string
    codigo:     string
    statusAtual:  ▼ {
                id:           integer (int64)
                status:       string
                              Enum:  SinalForte , SinalFraco , BateriaFraca , SemSinal
                transmissor:  integer (int64)
                receptor:     ▼ {
                                id:           integer
                                tipo:         ► { }
                                nome:         string
                                codigo:       string
                                local:        ► { }
                                localizacaoX: number
                                localizacaoY: number
                                statusAtual:  ► { }
                                observacao:   string
                                createdAt:    string
                                updatedAt:    string
                              }
                createdAt:    string (date-time)
                updatedAt:    string (date-time)
            }
    observacao: string
    createdAt:  string (date-time)
    updatedAt:  string (date-time)
}
```

listarTransmissores

GET

/transmissor

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/transmissor?paginacao="
```

Parameters

Query parameters

Name	Description
paginacao*	PaginacaoDTO Required

Responses

Status: 200 - OK

Schema

```
▼ {
    totalPages:     integer (int32)
    totalElements:  integer (int64)
    size:           integer (int32)
    content:        ▼ [
                      ▼ {
                          id:     integer (int64)
                          tipo:   ▼ {
                                    id:         integer (int64)
                                    nome:       string
                                    createdAt:  string (date-time)
                                    updatedAt:  string (date-time)
                                  }
                          nome:   string
                          codigo: string
```

```
                statusAtual: ▼ {
                               id: integer (int64)
                               status: string
                                  Enum: SinalForte , SinalFraco , BateriaFraca , SemSinal
                               transmissor: integer (int64)
                               receptor: ▼ {
                                             id: integer (int64)
                                             tipo: ▼ {
                                                       id: integer (int64)
                                                       nome: string
                                                       createdAt: string (date-time)
                                                       updatedAt: string (date-time)
                                                     }
                                             nome: string
                                             codigo: string
                                             local: ▼ {
                                                        id: integer (int64)
                                                        nome: string
                                                        area: ▼ {
                                                                  id: integer (int64)
                                                                  nome: string
                                                                  createdAt: string (date-time)
                                                                  updatedAt: string (date-time)
                                                                }
                                                        createdAt: string (date-time)
                                                        updatedAt: string (date-time)
                                                      }
                                             localizacaoX: number (double)
                                             localizacaoY: number (double)
                                             statusAtual: ▼ {
                                                            id: integer (int64)
                                                            status: string
                                                               Enum: SinalForte , BateriaFraca , SemSinal
                                                            receptor: integer (int64)
                                                            createdAt: string (date-time)
                                                            updatedAt: string (date-time)
                                                          }
                                             observacao: string
                                             createdAt: string (date-time)
                                             updatedAt: string (date-time)
                                           }
                               createdAt: string (date-time)
                               updatedAt: string (date-time)
                             }
                observacao: string
                createdAt: string (date-time)
                updatedAt: string (date-time)
              }
            ]
  number: integer (int32)
  sort: ▼ {
          empty: boolean
          sorted: boolean
          unsorted: boolean
        }
  numberOfElements: integer (int32)
  pageable: ▼ {
              offset: integer (int64)
              sort: ▼ {
                        empty: boolean
                        sorted: boolean
                        unsorted: boolean
                      }
              paged: boolean
              pageNumber: integer (int32)
              pageSize: integer (int32)
              unpaged: boolean
            }
  first: boolean
  last: boolean
  empty: boolean
}
```


removerTransmissor

DELETE

/transmissor/{id}

Usage and SDK Samples

Curl

```
curl -X DELETE\
-H "Accept: */*"\
"http://api.movvo.app.br/transmissor/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

boolean

TransmissorStatusController

atualizarStatusTransmissor

PUT

/transmissor-status/{id}

Usage and SDK Samples

Curl

```
curl -X PUT\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/transmissor-status/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	status: string
	Enum: `SinalForte` , `SinalFraco` , `BateriaFraca` , `SemSinal`
	transmissor: integer (int64)
	receptor: ▼ {
	id: integer
	tipo: ► { }
	nome: string
	codigo: string
	local: ► { }
	localizacaoX: number
	localizacaoY: number
	statusAtual: ► { }
	observacao: string
	createdAt: string
	updatedAt: string
	}
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
status: string
Enum: SinalForte, SinalFraco, BateriaFraca, SemSinal
transmissor: integer (int64)
receptor: ▼ {
id: integer (int64)
tipo: ▼ {
id: integer
nome: string
createdAt: string
updatedAt: string
}
nome: string
codigo: string
local: ▼ {
id: integer
nome: string
area: ► { }
createdAt: string
updatedAt: string
}
localizacaoX: number (double)
localizacaoY: number (double)
statusAtual: ▼ {
id: integer
status: string
receptor: integer
createdAt: string
updatedAt: string
}
observacao: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
createdAt: string (date-time)
updatedAt: string (date-time)
}

cadastrarStatusTransmissor

POST

/transmissor-status

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/transmissor-status"
```

Parameters

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	status: string
	Enum: `SinalForte` , `SinalFraco` , `BateriaFraca` , `SemSinal`
	transmissor: integer (int64)
	receptor: ▼ {
	id: integer
	tipo: ► { }
	nome: string
	codigo: string
	local: ► { }
	localizacaoX: number
	localizacaoY: number
	statusAtual: ► { }
	observacao: string
	createdAt: string
	updatedAt: string
	}
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
status: string
Enum: `SinalForte` , `SinalFraco` , `BateriaFraca` , `SemSinal`
transmissor: integer (int64)
receptor: ▼ {
id: integer (int64)
tipo: ▼ {
id: integer
nome: string
createdAt: string
updatedAt: string
}
nome: string
codigo: string
local: ▼ {
id: integer
nome: string
area: ► { }
createdAt: string
updatedAt: string
}
localizacaoX: number (double)
localizacaoY: number (double)
statusAtual: ▼ {
id: integer
status: string
receptor: integer
createdAt: string
updatedAt: string
}
observacao: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
createdAt: string (date-time)
updatedAt: string (date-time)
}

detalharStatusTransmissor

GET

/transmissor-status/{id}

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/transmissor-status/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
status: string
Enum: SinalForte , SinalFraco , BateriaFraca , SemSinal
transmissor: integer (int64)
receptor: ▼ {
id: integer (int64)
tipo: ▼ {
id: integer
nome: string
createdAt: string
updatedAt: string
}
nome: string
codigo: string
local: ▼ {
id: integer
nome: string
area: ► { }
createdAt: string
updatedAt: string
}
localizacaoX: number (double)
localizacaoY: number (double)
statusAtual: ▼ {
id: integer
status: string
receptor: integer
createdAt: string
updatedAt: string
}
observacao: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
createdAt: string (date-time)
updatedAt: string (date-time)
}

listarStatusTransmissores

GET

/transmissor-status

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/transmissor-status?paginacao="
```

Parameters

Query parameters

Name	Description
paginacao*	PaginacaoDTO Required

Responses

Status: 200 - OK

Schema

```
▼ {
  totalPages: integer (int32)
  totalElements: integer (int64)
  size: integer (int32)
  content: ▼ [
    ▼ {
      id: integer (int64)
      status: string
        Enum: SinalForte, SinalFraco, BateriaFraca, SemSinal
      transmissor: integer (int64)
      receptor: ▼ {
        id: integer (int64)
        tipo: ▼ {
          id: integer (int64)
          nome: string
          createdAt: string (date-time)
          updatedAt: string (date-time)
        }
        nome: string
        codigo: string
        local: ▼ {
          id: integer (int64)
          nome: string
          area: ▼ {
            id: integer (int64)
            nome: string
            createdAt: string (date-time)
            updatedAt: string (date-time)
          }
          createdAt: string (date-time)
          updatedAt: string (date-time)
        }
        localizacaoX: number (double)
        localizacaoY: number (double)
        statusAtual: ▼ {
          id: integer (int64)
          status: string
            Enum: SinalForte, BateriaFraca, SemSinal
          receptor: integer (int64)
          createdAt: string (date-time)
          updatedAt: string (date-time)
        }
        observacao: string
        createdAt: string (date-time)
        updatedAt: string (date-time)
      }
      createdAt: string (date-time)
      updatedAt: string (date-time)
    }
  ]
  number: integer (int32)
  sort: ▼ {
    empty: boolean
    sorted: boolean
    unsorted: boolean
  }
  numberOfElements: integer (int32)
  pageable: ▼ {
    offset: integer (int64)
    sort: ▼ {
      empty: boolean
      sorted: boolean
      unsorted: boolean
    }
    paged: boolean
    pageNumber: integer (int32)
```

```
                pageSize:    integer (int32)
                unpaged:     boolean
            }
        first:               boolean
        last:                boolean
        empty:               boolean
    }
```

removerStatusTransmissor

DELETE

/transmissor-status/{id}

Usage and SDK Samples

Curl

```
curl -X DELETE\
-H "Accept: */*"\
"http://api.movvo.app.br/transmissor-status/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

```
boolean
```

TransmissorTipoController

atualizarTipoTransmissor

PUT

/transmissor-tipo/{id}

Usage and SDK Samples

Curl

```
curl -X PUT\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/transmissor-tipo/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	nome: string
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}

cadastrarTipoTransmissor

POST

/transmissor-tipo

Usage and SDK Samples

Curl

```
curl -X POST\
-H "Accept: */*"\
-H "Content-Type: application/json"\
"http://api.movvo.app.br/transmissor-tipo"
```

Parameters

Body parameters

Name	Description

body *	▼ {
	id: integer (int64)
	nome: string
	createdAt: string (date-time)
	updatedAt: string (date-time)
	}

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}

detalharTipoTransmissor

GET

/transmissor-tipo/{id}

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/transmissor-tipo/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}

listarTiposTransmissores

GET

/transmissor-tipo

Usage and SDK Samples

Curl

```
curl -X GET\
-H "Accept: */*"\
"http://api.movvo.app.br/transmissor-tipo?paginacao="
```

Parameters

Query parameters

Name	Description
paginacao*	PaginacaoDTO Required

Responses

Status: 200 - OK

Schema

▼ {
totalPages: integer (int32)
totalElements: integer (int64)
size: integer (int32)
content: ▼ [
▼ {
id: integer (int64)
nome: string
createdAt: string (date-time)
updatedAt: string (date-time)
}
]
number: integer (int32)
sort: ▼ {
empty: boolean
sorted: boolean
unsorted: boolean
}
numberOfElements: integer (int32)
pageable: ▼ {
offset: integer (int64)
sort: ▼ {
empty: boolean
sorted: boolean
unsorted: boolean
}

```
                paged:          boolean
                pageNumber:     integer (int32)
                pageSize:       integer (int32)
                unpaged:        boolean
            }
        first:          boolean
        last:           boolean
        empty:          boolean
    }
```

removerTipoTransmissor

DELETE

/transmissor-tipo/{id}

Usage and SDK Samples

Curl

```
curl -X DELETE\
-H "Accept: */*"\
"http://api.movvo.app.br/transmissor-tipo/{id}"
```

Parameters

Path parameters

Name	Description
id*	Long (int64) Required

Responses

Status: 200 - OK

Schema

boolean