Eduardo Radieske

DESENVOLVIMENTO DE UM SISTEMA WEB PARA RESERVA DE SALAS COM INTEGRAÇÃO PARA USO EM FECHADURA ELETRÔNICA

Horizontina - RS

2025

Eduardo Radieske

DESENVOLVIMENTO DE UM SISTEMA WEB PARA RESERVA DE SALAS COM INTEGRAÇÃO PARA USO EM FECHADURA ELETRÔNICA

Trabalho Final de Curso apresentado como requisito parcial para a obtenção do título de bacharel em Engenharia de Controle e Automação na Faculdade Horizontina, sob a orientação do Prof. Dr. Douglas de Castro Karnikowski

Horizontina - RS

2025

FAHOR - FACULDADE HORIZONTINA

CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

A Comissão Examinadora, abaixo assinada, aprova o trabalho final de curso

“DESENVOLVIMENTO DE UM SISTEMA WEB PARA RESERVA DE SALAS COM INTEGRAÇÃO PARA USO EM FECHADURA ELETRÔNICA”

Elaborada por:

EDUARDO RADIESKE

Como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Controle e Automação

Aprovado em: 11/12/2025

Pela Comissão Examinadora

_____

Prof. Dr. Douglas de Castro Karnikowski

Presidente da Comissão Examinadora - Orientador

_____

Prof. Me. Fabrício Desbessel

FAHOR – Faculdade Horizontina

_____

Prof. Me. Rodrigo Bastos

FAHOR – Faculdade Horizontina

Horizontina - RS

2025

Dedico este trabalho à minha família, que sempre acreditou em mim e me apoiou em cada etapa da graduação. Agradeço também a todas as pessoas que, de alguma forma, contribuíram para minha jornada acadêmica, oferecendo apoio, incentivo e compreensão. Sem vocês, esta conquista não seria possível.

Agradeço aos meus pais pelo apoio constante, pela dedicação incondicional e por acreditarem em mim em cada etapa desta jornada. Sem vocês, nada disso seria possível.

RESUMO

O presente trabalho descreve o desenvolvimento de um sistema de reserva de salas para integração com fechadura eletrônica, voltado à automatização do controle de acesso e à solução de dificuldades na gestão de ambientes compartilhados. O problema central envolve organizar reservas, evitar conflitos de uso e garantir acesso autorizado sem processos manuais. O objetivo consiste em criar uma plataforma *web* capaz de cadastrar salas, gerenciar agendamentos e acionar fechaduras inteligentes de forma remota e sincronizada com os horários reservados. A metodologia compreende o levantamento de requisitos, a modelagem da arquitetura e do banco de dados e a implementação utilizando Java 21 com Spring Boot e PostgreSQL no *backend* e Angular no *frontend*, além da integração com a *Tuya Developer Platform* e do uso de Docker para padronização do ambiente. O desenvolvimento evidencia a coordenação entre o *software* desenvolvido e o sistema do fabricante Tuya, contemplando autenticação, gerenciamento de dispositivos e controle de acesso. O trabalho apresenta uma solução tecnicamente consistente e aplicável a diferentes contextos que demandam gestão automatizada e eficiente de espaços físicos.

Palavras-chave: Plataforma *web*. Controle de acesso. Fechadura eletrônica.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

IOT – Internet das coisas

REST - *Representational State Transfer*

HTTP - *HyperText Transfer Protocol*

MQTT - *Message Queuing Telemetry Transport*

IP - *Internet Protocol*

SQL - *Structured Query Language*

HTTPS - *Hypertext Transfer Protocol Secure*

SSL - *Secure Sockets Layer*

TLS - *Transport Layer Security*

HTML - *HyperText Markup Language*

CSS - *Cascading Style Sheets*

JWT - JSON Web Token

API - *Application Programming Interface*

IDE - *Integrated Development Environment*

SUMÁRIO

1 INTRODUÇÃO

A gestão de espaços físicos em ambientes institucionais, como universidades, centros de pesquisa e empresas, demanda cada vez mais soluções tecnológicas que garantam eficiência, segurança e praticidade. Nesse cenário, sistemas de controle de acesso integrados a plataformas digitais ganham destaque por possibilitarem a reserva de espaços de forma remota, com autenticação segura e liberação automatizada por meio de dispositivos eletrônicos. Estas soluções envolvem etapas essenciais da Análise e Desenvolvimento de *Software*, como implementação do processo, análise dos requisitos do sistema, desenho da arquitetura do sistema, análise dos requisitos do *software*, desenho da arquitetura do *software*, desenho detalhado do *software*, codificação e teste do *software* e integração do *software* com o *hardware* envolvido (Alves, 2015).

O avanço das tecnologias de Internet das Coisas (IoT), segundo Alves, Peixoto e Rosa (2021, p. 106), pode ser compreendido como “um ecossistema ciber-físico de sensores e recursos interconectados, que permitem a tomada de decisões inteligentes”. Esse conceito reflete a capacidade da IoT de transformar objetos do cotidiano em dispositivos inteligentes, conectados a sistemas que processam dados em tempo real. Em paralelo, a integração entre *hardware* e *software web* proporciona uma experiência unificada ao usuário. Além de reduzir falhas humanas, esses sistemas contribuem para o uso racional dos espaços, evitando conflitos de agendamento e otimizando os recursos disponíveis.

Nesse sentido, a automação de sistemas de reserva representa um avanço significativo para a modernização da infraestrutura institucional. A aplicação de fechaduras eletrônicas controladas remotamente, integradas a sistemas de autenticação e agendamento online, amplia as possibilidades de gerenciamento e reforça a segurança dos ambientes compartilhados, eliminando a necessidade de chaves físicas e reduzindo custos operacionais.

Diante do exposto, o presente trabalho tem como proposta o desenvolvimento e implementação de um sistema automatizado por meio de fechadura eletrônica, voltado para ambientes institucionais. A proposta contempla a utilização de tecnologias IoT, servidores em nuvem e uma interface *web* para autenticação e gerenciamento, com foco no controle por data e horário, promovendo maior organização, segurança e eficiência no uso dos espaços.

Dessa forma, explorando os conceitos de desenvolvimento de *software*, automação e integração de sistemas de diversas plataformas e tecnologias, através de um projeto multidisciplinar. A utilização de princípios de engenharia de *software*, protocolos de comunicação e práticas de segurança digital foi fundamental para garantir a eficiência, a confiabilidade e a escalabilidade da solução desenvolvida.

1.1 TEMA

Automação de sistemas de controle de acesso e gerenciamento de reservas em ambientes institucionais.

1.2 DELIMITAÇÃO DO TEMA

Desenvolvimento de um sistema automatizado de reserva de salas, para integração e controle de fechadura eletrônica de maneira remota, para aplicação em ambientes institucionais.

1.3 PROBLEMA DE PESQUISA

Como desenvolver um sistema automatizado, seguro e escalável que permita a reserva de salas e controle de acesso em ambientes institucionais, integrando autenticação *web* com dispositivos físicos?

1.4 HIPÓTESES

a) Integrar a fechadura eletrônica ao sistema de reserva de salas permite maior segurança e controle de acesso ao ambiente.

b) Desenvolver um sistema de reserva de salas permite melhorar a gestão de uso das salas em ambientes institucionais.

c) Reduzir falhas operacionais e a necessidade de intervenção manual na gestão de salas com a implementação de um sistema informatizado.

1.5 JUSTIFICATIVA

A crescente demanda por soluções tecnológicas em ambientes institucionais está diretamente relacionada à necessidade de eficiência, segurança e organização no uso de espaços compartilhados, como salas de aula, laboratórios e ambientes corporativos. O gerenciamento manual de reservas e o controle físico de acesso a esses locais é um formato limitado diante da complexidade e da dinamicidade desses contextos. Isso evidencia a importância de adotar sistemas automatizados que otimizem a utilização dos recursos disponíveis e garantam maior controle sobre o acesso.

De acordo com a Associação Brasileira das Empresas de *Software* (ABES), o mercado brasileiro de Tecnologia da Informação (TI) registrou um crescimento expressivo de 13,9% em 2024, mantendo o país entre os dez maiores mercados globais e consolidando sua liderança na América Latina (Abes, 2025). Esse cenário reforça a importância do investimento em soluções digitais nacionais, demonstrando um ambiente favorável à inovação e ao desenvolvimento de sistemas que atendam a demandas reais da sociedade.

A transformação digital tem impactado de forma significativa a maneira como instituições de ensino, empresas e organizações públicas gerenciam seus recursos físicos e operacionais. A integração entre sistemas de informação e dispositivos eletrônicos tem se mostrado uma estratégia eficaz para reduzir falhas humanas, aumentar a rastreabilidade das operações e promover maior transparência nos processos de gestão. Nesse contexto, soluções que unem *software* e *hardware*, como sistemas automatizados de controle de acesso e reserva de espaços, surgem como alternativas viáveis para atender às exigências atuais por maior confiabilidade, agilidade e segurança operacional.

Diante disso, o presente trabalho se justifica por propor o desenvolvimento de uma solução tecnológica que combina automação, segurança de acesso e gestão inteligente de espaços institucionais. Essa iniciativa contribui não apenas para a modernização dos processos internos, mas também para o fortalecimento do conhecimento técnico na área, alinhando-se às demandas reais da sociedade e às tendências do mercado atual.

1.6 OBJETIVOS

A seção de objetivos tem como finalidade apresentar de forma clara e organizada as metas que se pretende alcançar com o desenvolvimento deste trabalho. Aqui são definidos o objetivo geral, que expressa de maneira ampla o propósito central do projeto, e os objetivos específicos, que detalham as etapas e ações necessárias para atingir o resultado proposto.

1.6.1 Objetivo geral

Desenvolver um sistema automatizado de reserva de salas, integrado a uma fechadura eletrônica, visando otimizar o controle de acesso e a gestão de uso de espaços em ambientes institucionais.

1.6.2 Objetivos específicos

- Analisar os requisitos funcionais e não funcionais para o sistema de reservas e controle de acesso.
- Projetar a arquitetura do sistema, incluindo a plataforma *web*, banco de dados e interface de controle da fechadura eletrônica.
- Selecionar os componentes e serviços adequados para a implementação da fechadura eletrônica integrada ao sistema.
- Implementar a comunicação entre o sistema de reservas e o *hardware* de controle da fechadura.
- Desenvolver uma interface *web* que permita o agendamento e gerenciamento das reservas com autenticação de usuários.

2 REVISÃO DA LITERATURA

A seção de revisão da literatura tem como objetivo apresentar e discutir os principais conceitos, teorias e trabalhos já desenvolvidos que servem de base para este estudo. Por meio dessa revisão, busca-se contextualizar o tema, identificar lacunas existentes e fundamentar o desenvolvimento do projeto com o apoio de referências relevantes e atualizadas.

2.1 CONTROLE DE ACESSO ELETRÔNICO

O controle de acesso eletrônico é uma tecnologia que permite gerenciar e restringir o acesso a ambientes físicos de forma automatizada, substituindo métodos tradicionais como chaves mecânicas por sistemas digitais mais seguros e flexíveis. No contexto de automação residencial e domótica, de acordo com Júnior e Farinelli (2018), esse tipo de controle se integra a outros sistemas inteligentes da residência, permitindo a personalização de regras de acesso, monitoramento remoto e integração com dispositivos inteligentes.

Entre as principais tecnologias utilizadas pode-se citar as fechaduras eletrônicas, leitores de cartão RFID, biometria, teclados numéricos e controle via aplicativos móveis. Esses dispositivos são gerenciados por um sistema central, que pode estar integrado a uma plataforma de automação residencial, como *Home Assistant*, entre outros sistemas proprietários.

A autenticação dos usuários pode ser feita por diferentes métodos, como *PIN codes*, biometria (impressão digital, reconhecimento facial) ou dispositivos móveis com Bluetooth ou Wi-Fi. Após a autenticação, o sistema envia um sinal de comando, geralmente via relé eletrônico, contato seco ou protocolo IP, para liberar o acesso ao ambiente. Além do controle físico da abertura de portas, esses sistemas oferecem recursos como registro de logs de acesso, definição de horários permitidos, bloqueios temporários e notificações em tempo real. Na automação residencial, o controle de acesso pode ser integrado a outros cenários, como acendimento automático de luzes, desativação de alarmes, ou acionamento de câmeras de segurança, ampliando a segurança e o conforto dos usuários (Maschietto, Vieira e Torres, 2021)..

2.2 INTERNET DAS COISAS (IOT)

A Internet das Coisas (IoT - *Internet of Things*), segundo Morais et al. (2018) é um conceito tecnológico que descreve a interconexão digital de objetos físicos através da internet, permitindo que dispositivos comuniquem, coletem e compartilhem dados de forma autônoma. Este termo é baseado na ideia de fusão do mundo real com o mundo digital representado uma evolução significativa na forma como os sistemas computacionais interagem com o ambiente físico, promovendo a automação, o monitoramento remoto e o controle inteligente de dispositivos.

No contexto de sistemas de controle de acesso e gestão de espaços, a IoT desempenha um papel central, permitindo a integração de sensores, atuadores, sistemas de autenticação e plataformas de gerenciamento em nuvem. Aplicações como fechaduras eletrônicas, sensores de presença, leitores RFID e sistemas de monitoramento remoto tornam-se viáveis e escaláveis com o uso da IoT, proporcionando maior segurança e eficiência na gestão de ambientes institucionais (Alves, Peixoto e Rosa, 2021).

A arquitetura típica de uma solução IoT é composta por três camadas principais: percepção, rede e aplicação. Na camada de percepção, dispositivos como sensores, geolocalizadores e smartphones captam informações do ambiente físico. Esses dispositivos formam a chamada entidade de aplicação, responsável por executar uma ou mais lógicas de serviço identificadas de forma única. A camada de rede realiza a transmissão dos dados capturados, utilizando protocolos como Wi-Fi, MQTT ou ZigBee. Já na camada de aplicação, os dados são processados, armazenados e apresentados ao usuário, geralmente por meio de sistemas web ou aplicativos móveis. Essa camada interage com uma entidade intermediária, conhecida como entidade de serviços comuns, que disponibiliza um conjunto de funções padronizadas por meio de APIs. Essas especificações, inicialmente desenvolvidas em projetos M2M (*Machine-to-Machine*), visam garantir a interoperabilidade entre diferentes redes de sensores, dispositivos e aplicações baseadas em nuvem (Morais, Gonçalves e Ledur., 2018).

2.3 DESENVOLVIMENTO WEB APLICADO À AUTOMAÇÃO

O desenvolvimento *web,* de acordo com Miletto e Bertagnolli (2014), envolve a criação de interfaces, serviços e APIs que permitem a interação entre usuários e dispositivos físicos por meio de navegadores ou aplicativos baseados na *web*. No contexto de sistemas de controle de acesso e reservas, como o projeto de um sistema de reserva de salas com fechadura eletrônica, a camada de aplicação *web* desempenha um papel central na comunicação entre o usuário e os dispositivos embarcados responsáveis pela automação.

A arquitetura *web* moderna geralmente adota o modelo cliente-servidor, no qual o *frontend* e o *backend* são componentes independentes, mas interligados por meio de *APIs RESTful* ou *WebSocket* para comunicação em tempo real. A camada de interface da aplicação, conhecida como *frontend*, é desenvolvida utilizando tecnologias como HTML5, CSS3 e JavaScript, podendo incorporar *frameworks* modernos como React, Angular ou Vue.js para criar interfaces dinâmicas, responsivas e com boa experiência do usuário (UX). Essas interfaces permitem aos usuários realizar operações como agendar reservas, consultar o status de disponibilidade das salas, visualizar logs de acesso e acionar remotamente as fechaduras eletrônicas (Miletto e Bertagnolli, 2014).

No *backend*, é comum a utilização de *frameworks* robustos como Node.js, ASP.NET Core ou Spring Boot, que oferecem suporte a serviços *RESTful* e *WebSocket*, garantindo a escalabilidade, a confiabilidade e a interoperabilidade do sistema. Para aplicações de menor porte ou com menor necessidade de processamento assíncrono, de acordo com Duckett (2024), tecnologias como PHP podem ser utilizadas para o processamento estático do lado do servidor, principalmente em situações de entrega rápida de conteúdo.

A persistência de dados é um aspecto crítico nesse tipo de solução. Em geral, utilizam-se bancos de dados relacionais, como MySQL, PostgreSQL ou SQL *Server*, que oferecem estrutura relacional, integridade referencial e suporte a consultas complexas por meio de SQL. Para cenários que demandam maior flexibilidade na estruturação de dados ou manipulação de grandes volumes de informações não estruturadas, podem ser adotados bancos de dados NoSQL, como MongoDB ou *Firebase Realtime Database* (Duckett, 2024).

2.4 DISPOSITIVOS ELETRÔNICOS INTELIGENTES

Dispositivos eletrônicos inteligentes são equipamentos capazes de processar informações, tomar decisões locais e interagir com outros sistemas por meio de comunicação digital. Esses dispositivos, também conhecidos como "*Smart Devices*", possuem unidades de processamento integradas, sensores, atuadores e interfaces de comunicação, permitindo sua atuação autônoma ou como parte de sistemas maiores baseados em Internet das Coisas (Moura e Neves, 2021).

No contexto de automação e controle de acesso, de acordo com Ideali (2021) os dispositivos inteligentes incluem microcontroladores (como Arduino, ESP32 e Raspberry Pi), controladores de relés, fechaduras eletromecânicas e módulos de comunicação sem fio (Wi-Fi, Bluetooth, Zigbee). Esses elementos permitem o acionamento remoto de mecanismos físicos a partir de comandos digitais, seja por redes locais ou via internet.

Além do controle físico, tais dispositivos podem registrar eventos, armazenar dados temporariamente e responder a comandos externos de maneira segura e eficiente. Essa capacidade de processamento local reduz a dependência de servidores externos para operações críticas, melhorando a confiabilidade do sistema (Ideali, 2021).

2.5 BANCO DE DADOS E GERENCIAMENTO DE INFORMAÇÕES

O banco de dados é um componente essencial em sistemas de informação, responsável por armazenar, organizar e disponibilizar os dados de forma estruturada e segura. Ele permite o gerenciamento eficiente de grandes volumes de informações, garantindo integridade, consistência e disponibilidade. Em aplicações de reserva de salas com controle de acesso eletrônico, o banco de dados armazena informações como: cadastro de usuários, horários de reservas, status de salas, logs de acesso e configurações de dispositivos. Essas informações são fundamentais para validar permissões, controlar horários e gerar relatórios de uso (Alves, 2021).

Dessa forma, os sistemas gerenciadores de banco de dados (SGBDs), como MySQL, PostgreSQL ou SQLite, oferecem recursos como controle de concorrência, integridade referencial e mecanismos de segurança para proteção dos dados sensíveis. A integração entre o banco de dados e a aplicação *web* é feita por meio de

linguagens de consulta estruturada (SQL) e camadas de persistência de dados, que realizam as operações de leitura, inserção, atualização e exclusão. Portanto, o gerenciamento eficiente dessas informações permite que o sistema ofereça respostas rápidas, mantenha o histórico de uso e facilite a manutenção e expansão futura da solução (Pichetti, Vida e Cortes, 2021).

2.6 SEGURANÇA DA INFORMAÇÃO

A segurança da informação refere-se ao conjunto de práticas, políticas e tecnologias aplicadas para proteger os dados contra acessos não autorizados, alterações indevidas, perda ou destruição. Seu principal objetivo é garantir os pilares de confidencialidade, integridade, disponibilidade e autenticidade das informações. Em sistemas de controle de acesso eletrônico, a segurança da informação é fundamental para evitar que usuários não autorizados realizem reservas, acessem ambientes restritos ou modifiquem dados críticos. Para isso, são aplicadas técnicas como autenticação de usuários, controle de permissões, criptografia de dados e registro de logs de acesso (Barreto e Morais, 2018).

Além da proteção dos dados armazenados no banco de dados, de acordo com Moraes e Hayashi (2021) é importante garantir a segurança na comunicação entre o sistema *web* e os dispositivos eletrônicos, utilizando protocolos seguros como HTTPS, SSL/TLS ou autenticação via *tokens*. Dessa forma, o correto gerenciamento da segurança da informação aumenta a confiabilidade do sistema, reduz riscos operacionais e protege a integridade dos recursos físicos e digitais envolvidos no controle de acesso.

2.7 USABILIDADE E EXPERIÊNCIA DO USUÁRIO (UX)

A usabilidade é um atributo de qualidade que avalia o grau de facilidade, eficiência e satisfação com que os usuários interagem com um sistema. Ela envolve princípios como eficácia, eficiência, facilidade de aprendizado, memorização, prevenção de erros e satisfação subjetiva. Interfaces com boa usabilidade minimizam a carga cognitiva, reduzem o tempo de execução de tarefas e diminuem a incidência de erros operacionais (Júnior e Leite, 2024).

A Experiência do Usuário (UX), por sua vez, de acordo com Câmara (2024) abrange uma visão mais ampla e subjetiva, relacionada às percepções, emoções e reações do usuário ao longo de toda a interação com o produto ou serviço trazendo uma experiência positiva do uso da plataforma. Entre os principais fatores de UX estão a usabilidade, acessibilidade, arquitetura da informação, design de interação, tempo de resposta do sistema, *feedback* em tempo real e responsividade em diferentes dispositivos.

Dessa forma, para o desenvolvimento de interfaces eficazes, são aplicadas técnicas como prototipação, testes de usabilidade, análise heurística e avaliação de experiência emocional. Além disso, conceitos como *Design* Centrado no Usuário (DCU), *User Interface Design* (UI Design) e *User-Centered Design* (UCD) são amplamente utilizados durante o processo de criação e validação de interfaces digitais (Câmara, 2024).

2.8 PROTOCOLOS E REDES DE COMUNICAÇÃO

Os protocolos e redes de comunicação são fundamentais para troca de dados entre dispositivos e sistemas computacionais. Um protocolo de comunicação define regras e formatos padronizados que garantem a transmissão, recepção e interpretação correta das informações entre os participantes da rede. Sendo alguns exemplos o TCP/IP, HTTP, MQTT e Modbus, cada um adequado a contextos específicos de velocidade, consumo de energia, confiabilidade e alcance (Lacerda, Suarez e Lenz, 2022).

Sendo que, as redes de comunicação podem ser classificadas conforme sua topologia (como estrela, malha ou barramento), escopo (LAN, WAN, PAN) e meio de transmissão (com ou sem fio). Em aplicações embarcadas e sistemas distribuídos, destaca-se o uso de redes sem fio como Wi-Fi, ZigBee e LoRa, que facilitam a integração entre dispositivos físicos e sistemas de controle. Dessa forma, a escolha dos protocolos e da rede influência diretamente a latência, segurança, robustez e escalabilidade da aplicação. Tecnologias como MQTT sobre TCP/IP, por exemplo, são amplamente utilizadas por sua leveza e suporte a comunicação assíncrona e em tempo real, essenciais para sistemas que operam com sensores e atuadores conectados (Lacerda, Suarez e Lenz, 2022).

2.9 PADRÕES DE PROJETO

Os padrões de projeto (*Design Patterns*) são soluções reutilizáveis para problemas recorrentes no desenvolvimento de *software*. Eles oferecem modelos testados e documentados que auxiliam na organização do código, facilitando a manutenção, reutilização, flexibilidade e escalabilidade das aplicações. Esses padrões são classificados principalmente em três categorias: Padrões Criacionais, que tratam da criação de objetos (ex.: Singleton, Factory Method); Padrões Estruturais, que definem como as classes e objetos se organizam para formar estruturas maiores (ex.: Adapter, Decorator); e Padrões Comportamentais, que descrevem como os objetos interagem e se comunicam, como por exemplo nos padrões *Observer* e *Command* (Zenker, Santos e Couto, 2019).

A aplicação correta dos padrões reduz a complexidade do sistema, facilita a implementação de novas funcionalidades e melhora a legibilidade do código-fonte. Além disso, os padrões promovem o uso de boas práticas de desenvolvimento orientado a objetos, como baixo acoplamento e alta coesão. Sendo que, em projetos modernos, como aplicações *web* e sistemas embarcados, a adoção de padrões de projeto é essencial para garantir a qualidade arquitetural e o desempenho a longo prazo (Zenker, Santos e Couto, 2019).

2.10 AUTENTICAÇÃO E AUTORIZAÇÃO EM SISTEMAS WEB

Os processos de autenticação e autorização, de acordo com Araújo e Marinho (2023) são dois pilares fundamentais para a segurança em sistemas *web*. Autenticação é o processo de verificar a identidade do usuário, geralmente através de credenciais de login e senha, *tokens*, certificados digitais ou biometria. Já a autorização consiste em definir os níveis de acesso, determinando quais recursos ou funcionalidades um usuário autenticado pode utilizar.

Entre os métodos de autenticação mais comuns estão JWT (*JSON Web Token*), OAuth 2.0, *OpenID Connect* e SAML, que oferecem formas seguras e escaláveis de validar identidades em aplicações *web* modernas. Para autorização, a aplicação pode utilizar controle baseado em papéis (RBAC – *Role-Based Access Control*) ou controle baseado em atributos (ABAC – *Attribute-Based Access Control*), permitindo granularidade no gerenciamento de permissões. Dessa forma, garantir

autenticação e autorização seguras protege os sistemas contra ameaças como acesso não autorizado, escalonamento de privilégios, ataques de força bruta e sequestro de sessão, sendo um requisito essencial no desenvolvimento de aplicações web robustas e confiáveis (Araújo e Marinho, 2023).

2.11 TECNOLOGIAS DE CONTEINERIZAÇÃO E DOCKER

A conteinerização, de acordo com Erl e Monroy (2024) consolidou-se como uma abordagem fundamental para execução de aplicações modernas, oferecendo isolamento, leveza e padronização. Nesse contexto, o Docker tornou-se a solução mais adotada, organizando-se em servidor, cliente, registro e objetos operacionais. O servidor, responsável pelo *daemon* de contêineres, gerencia a criação, execução e monitoramento das instâncias, aplicando mecanismos como *namespaces* e *cgroups* para manter segurança e utilização adequada dos recursos. O cliente, acessado por CLI ou API REST, possibilita que desenvolvedores e administradores interajam de forma simples e eficiente com o *daemon*, controlando todo o ciclo de vida dos contêineres.

O registro Docker funciona como repositório de imagens, permitindo centralizar diferentes versões utilizadas nas implantações. Pode ser público ou privado, conforme requisitos de segurança, e facilita a padronização e a distribuição ao empregar comandos como *push*, *pull* e *run*. Nesse ambiente, as imagens atuam como modelos imutáveis que geram contêineres reprodutíveis e consistentes em qualquer *host* compatível. Os objetos Docker complementam a solução ao incluir imagens, contêineres, serviços internos e camadas de sistema de arquivos que possibilitam flexibilidade e eficiência. Dessa forma, padronização de imagens e o isolamento de contêineres reduzem falhas e simplificam a manutenção de aplicações distribuídas. Seus recursos de versionamento, replicação e orquestração tornam a plataforma adequada tanto para sistemas corporativos quanto para uso acadêmico, reforçando sua ampla adoção (Erl e Monroy, 2024).

2.12 PESQUISA QUALITATIVA

A pesquisa qualitativa caracteriza-se pela investigação aprofundada de fenômenos, buscando compreender significados, processos e relações, e não apenas quantificá-los. Conforme esse tipo de abordagem, parte-se da definição de objetivos claros, da seleção das informações relevantes e da realização da pesquisa de campo, podendo-se construir hipóteses para explicar o problema identificado. Em seguida, delimita-se o campo de estudo e definem-se os procedimentos necessários para a coleta dos dados, que, após serem obtidos, passam por uma etapa sistemática de análise e interpretação. No contexto deste trabalho, a pesquisa enquadra-se como aplicada e de abordagem qualitativa, pois tem como foco a compreensão e a análise do problema prático estudado, visando propor ou avaliar uma solução concreta. Assim, a investigação não se limita à descrição de dados numéricos, mas prioriza a interpretação dos resultados obtidos a partir da observação, análise documental e/ou estudo de caso, contribuindo diretamente para a aplicação dos conhecimentos teóricos na resolução do problema abordado (Marconi e Lakatos, 2022).

2.13 SISTEMAS DE CONTROLE DE ACESSO

Projetos com temática semelhante ao trabalho proposto já foram desenvolvidos, como o estudo de Rodrigues (2023), cujo objetivo foi criar um sistema de fechadura eletrônica baseado na leitura de *QR Code* para autenticação e controle de acesso de forma segura, eficiente e remota. A proposta permitia a ativação da fechadura por meio da leitura de códigos gerados por um aplicativo, oferecendo uma solução moderna aplicável a residências, empresas e outros ambientes. O sistema também incluía funcionalidades de gerenciamento remoto e registro das ações realizadas, evidenciando a viabilidade da integração entre automação e segurança digital em aplicações práticas do cotidiano.

Sendo que, destaca-se o projeto de Gonçalves (2025), que teve como objetivo desenvolver e implementar um sistema de controle de acesso físico baseado na identificação por RFID e autenticação biométrica por impressão digital. Utilizando um microcontrolador NodeMCU-ESP32 como núcleo da arquitetura, o sistema realiza a comunicação entre os dispositivos de autenticação e um banco de dados remoto. A proposta visa garantir segurança, confiabilidade e eficiência no controle de acesso a

ambientes restritos, armazenando os registros de forma organizada e segura. Além disso, o sistema oferece funcionalidades como atualização remota de permissões de acesso e um modo administrativo para cadastro de novos usuários e ajustes operacionais, demonstrando a viabilidade e aplicabilidade da integração entre automação, segurança física e conectividade em tempo real.

2.14 PROCEDIMENTOS E MÉTODOS

Os procedimentos e métodos de pesquisa, de acordo com Marconi e Lakatos (2022) constituem elementos fundamentais para a organização e condução do trabalho científico. O método monográfico caracteriza-se pela análise aprofundada de uma realidade específica, podendo concentrar-se em aspectos particulares ou abranger o conjunto das atividades de um determinado grupo social, preservando a compreensão da totalidade e das relações entre seus elementos. Essa abordagem permite compreender o fenômeno estudado de forma integrada e contextualizada.

Quanto aos objetivos, as pesquisas podem ser exploratórias, descritivas ou explicativas, sendo a exploratória voltada à familiarização com o problema e à construção inicial de hipóteses, especialmente quando o objeto ainda não está claramente definido. Nesse contexto, a pesquisa aplicada busca gerar conhecimentos com finalidade prática, utilizando procedimentos como a pesquisa bibliográfica, a experimentação e o estudo de caso, que possibilitam analisar, testar e compreender o fenômeno em situações reais, contribuindo de forma objetiva para a solução de problemas concretos (Marconi e Lakatos, 2022).

3 METODOLOGIA

A metodologia adotada neste trabalho tem como finalidade nortear o desenvolvimento de um sistema completo para reserva de salas com controle eletrônico de acesso. A estrutura metodológica contempla a descrição do objeto de estudo, a caracterização do tipo de pesquisa e os procedimentos de coleta e análise de dados empregados ao longo do desenvolvimento.

3.1 MÉTODOS E TÉCNICAS UTILIZADOS

A seção de métodos e técnicas utilizados descreve as abordagens, ferramentas e procedimentos adotados para o desenvolvimento e a execução deste trabalho.

3.1.1 Objeto de estudo

O objeto de estudo deste trabalho é o desenvolvimento de um sistema integrado para reserva de salas e controle de acesso físico, composto por uma plataforma *web* para agendamento e um módulo de acionamento de fechadura eletrônica. O sistema tem como função principal permitir que usuários autorizados realizem reservas de salas por meio de uma interface online, enquanto a autenticação e o controle de acesso às salas são gerenciados automaticamente por meio da comunicação entre o *software* e o *hardware* da fechadura. A solução abrange o desenvolvimento da aplicação *web* com autenticação, o gerenciamento de banco de dados, a integração com dispositivos eletrônicos de controle de acesso e a implementação dos protocolos de comunicação entre os módulos. O escopo contempla tanto os aspectos de *software* quanto os de *hardware*, visando o funcionamento conjunto e automatizado do sistema.

3.1.2 Tipo de pesquisa

Este trabalho caracteriza-se como uma pesquisa aplicada, de abordagem qualitativa e exploratória, com método monográfico e procedimentos de natureza bibliográfica, experimental e de estudo de caso. A pesquisa é aplicada para buscar

uma solução prática e funcional para uma demanda real relacionada à gestão e controle de acesso a salas em ambientes institucionais.

A abordagem qualitativa está presente na análise dos requisitos do sistema, na definição da arquitetura e na seleção dos componentes, enquanto a abordagem quantitativa foi empregada na avaliação do desempenho do sistema, com base em métricas como tempo de resposta, taxa de sucesso de autenticação e confiabilidade da comunicação entre os módulos analisando os logs do servidor da aplicação.

A pesquisa descritiva consiste na revisão de literatura e na documentação dos requisitos, enquanto a parte exploratória envolve o desenvolvimento, a integração e os testes do sistema proposto, permitindo a identificação de limitações e possibilidades de melhoria. Os procedimentos incluem o levantamento de dados técnicos por meio de referências bibliográficas e manuais, a experimentação prática com componentes eletrônicos e o estudo de caso simulado para validação da solução implementada.

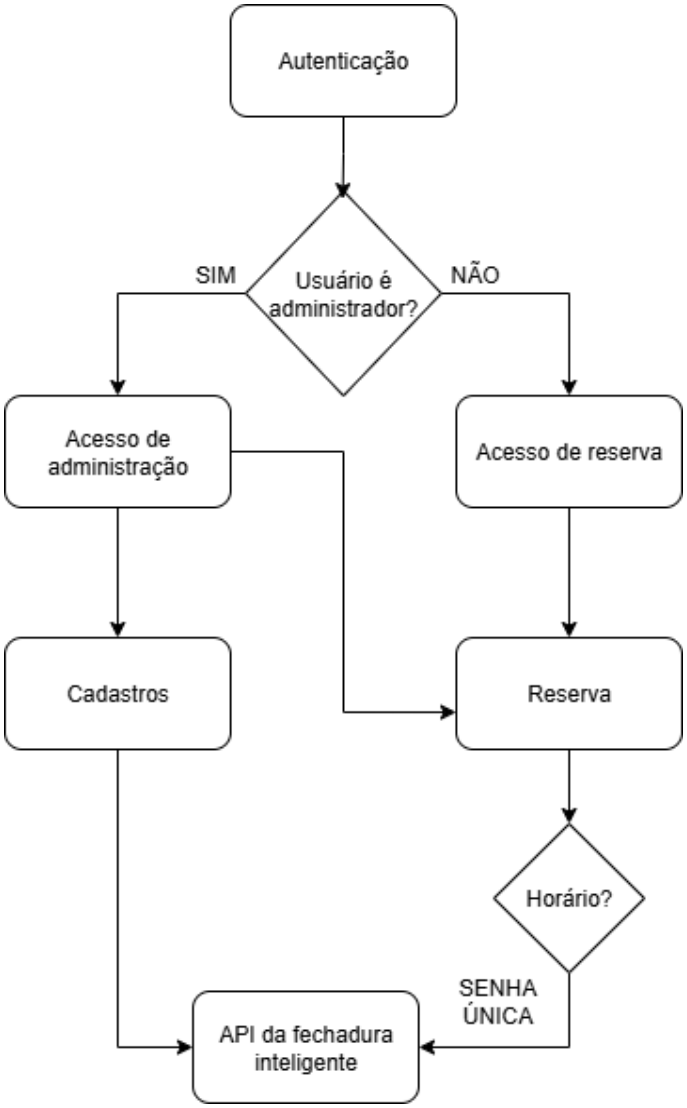
3.1.3 Procedimentos técnicos

Os procedimentos técnicos adotados neste trabalho foram organizados em etapas sequenciais, abrangendo desde o levantamento de requisitos até a validação do sistema proposto, com o objetivo de garantir o desenvolvimento completo da solução integrada de *software* e *hardware*. As etapas adotadas são descritas a seguir:

1. **Levantamento de Requisitos**: Neste procedimento é realizada a identificação dos requisitos funcionais e não funcionais do sistema, com base em pesquisa bibliográfica, normas técnicas e análise de sistemas semelhantes. Essa etapa envolve a definição das funcionalidades necessárias tanto para a plataforma web quanto para o módulo de controle de acesso físico.

2. **Projeto da Arquitetura do Sistema**: Com base nos requisitos identificados, é possível elaborar a arquitetura do sistema, incluindo a estrutura, modelo de banco de dados, definição das rotas de API, e o modelo de comunicação entre o *software* e a fechadura eletrônica. A arquitetura busca garantir modularidade, escalabilidade e segurança conforme o fluxograma da figura 1.

Figura 1 - Fluxograma da aplicação



Fonte: Autor (2025)

3. **Seleção dos Componentes de Infraestrutura**: Nesta etapa foram definidos os principais aspectos de infraestrutura necessários para suportar o funcionamento do sistema, incluindo a camada de servidores e os serviços *web* associados. Optou-se pelo uso de Docker como tecnologia de virtualização baseada em containers, devido à sua capacidade de padronizar ambientes, simplificar o processo de implantação e garantir maior portabilidade e escalabilidade das aplicações. A seleção da infraestrutura levou em conta critérios como desempenho, custo, segurança e viabilidade de implementação.

4. **Desenvolvimento da Plataforma Web**: Será implementado o sistema *web* para agendamento e gerenciamento de reservas, com funcionalidades como autenticação de usuários, interface para visualização do calendário de uso, controle de permissões e histórico de acessos. A aplicação será desenvolvida utilizando tecnologias como HTML, CSS, JavaScript e um *backend* em linguagem compatível com bancos de dados relacionais.

5. **Integração Software-Hardware**: A comunicação entre o sistema *web* e o módulo IoT responsável pela liberação da fechadura será realizada por meio da *Tuya Developer Platform*, uma plataforma do fabricante Tuya que permite integrar, gerenciar e controlar dispositivos inteligentes compatíveis com o ecossistema *Tuya Smart*. Utilizando suas APIs REST e recursos de comunicação segura, será possível enviar comandos ao dispositivo, consultar seu estado e sincronizar em tempo real as informações de agendamento. A integração ocorre via HTTPS, garantindo autenticação adequada, integridade dos dados e resposta imediata às solicitações do sistema.

6. **Testes e Validação do Sistema**: Após a integração dos módulos, serão realizados testes unitários e funcionais para verificar o correto funcionamento da reserva, autenticação e acionamento da fechadura. A validação do sistema será feita por meio de simulação de uso em ambiente controlado.

Esses procedimentos técnicos permitem o desenvolvimento de um sistema funcional, modular e com potencial de aplicação prática em ambientes institucionais, alinhando as etapas de engenharia de *software* com práticas de automação e controle.

3.1.4 Técnica de coleta de dados

A coleta de dados neste trabalho foi realizada por meio de técnicas sistemáticas e adequadas à natureza aplicada e experimental da pesquisa. Inicialmente, foram conduzidos testes funcionais e de desempenho do sistema em ambiente de homologação, com o objetivo de validar a comunicação entre a plataforma *web* e o

módulo de controle da fechadura eletrônica. Estes testes permitiram coletar dados quantitativos, a efetividade na autenticação de usuários e a estabilidade da conexão entre os componentes.

Além disso, foi empregada a observação direta controlada, em ambiente de simulação, para acompanhar o comportamento do sistema frente a diferentes cenários de uso, tais como múltiplas reservas, tentativas de acesso não autorizadas e conflitos de agendamento. Durante esse processo, foram registrados eventuais erros de funcionamento, falhas de segurança, atrasos na execução e inconsistências operacionais. A combinação dessas técnicas possibilitará uma análise abrangente da solução desenvolvida, considerando aspectos técnicos e de experiência do usuário, e subsidiará ajustes e melhorias antes da aplicação em ambiente real.

3.1.5 Técnica de análise de dados

A análise dos dados obtidos ao longo da execução do trabalho foi conduzida por meio de abordagens qualitativas e quantitativas, conforme a natureza das informações coletadas. Os dados observados e coletados, provenientes dos testes funcionais e de desempenho realizados em ambiente de homologação, foram analisados por meio de estatísticas descritivas, contemplando métricas como, observações no sucesso das operações de autenticação e acionamento da fechadura, frequência de falhas e disponibilidade dos serviços.

Os dados qualitativos, coletados por meio da observação direta, foram analisados utilizando a técnica de análise de conteúdo, permitindo a identificação de padrões de comportamento, dificuldades de usabilidade e demais elementos subjetivos que influenciam a experiência de uso do sistema. A utilização dessas técnicas de análise permitiu uma avaliação crítica e abrangente do sistema, subsidiando a identificação de eventuais limitações, bem como a proposição de ajustes e aprimoramentos antes de uma possível aplicação em ambiente real.

3.2 MATERIAIS E EQUIPAMENTOS

Para a realização do presente trabalho, foram necessários recursos materiais e tecnológicos compatíveis com o desenvolvimento de sistemas computacionais integrados com dispositivos eletrônicos. Os materiais previstos incluem:

a) Computador pessoal;

b) Acesso à internet;

c) Servidor para hospedagem web;

d) Documentação necessária para operação e integração das APIs;

e) Licenças de homologação para realização dos testes;

f) Editor de código e ambiente de desenvolvimento (IDE).

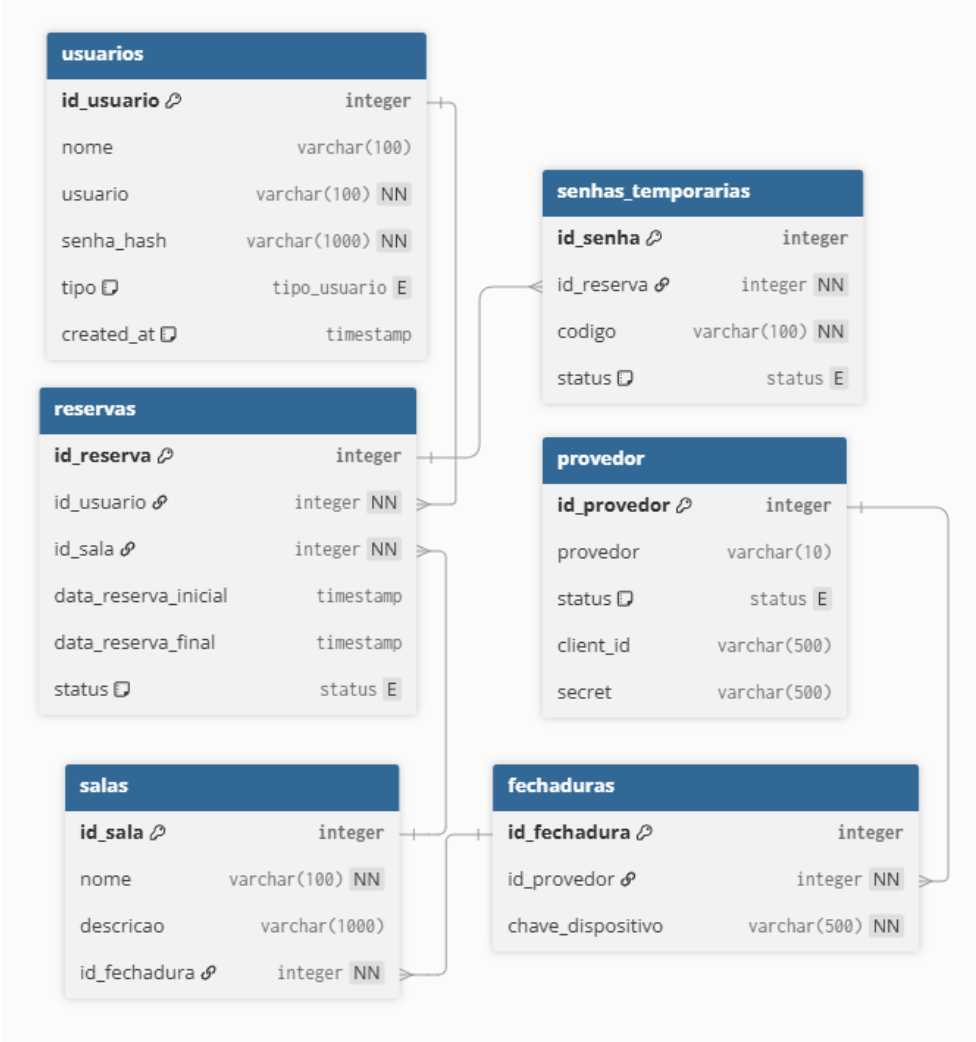
4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Com os requisitos e o escopo claramente estabelecidos, deu-se início à etapa de desenvolvimento do trabalho, na qual foram definidas as estratégias, ferramentas e as etapas necessárias para a implementação da solução proposta, além do desenvolvimento do código-fonte do sistema.

4.1 DESENVOLVIMENTO DO BANCO DE DADOS

Na primeira etapa do planejamento foi construído o diagrama do banco de dados, conforme figura 2, para o sistema de reservas, evidenciando suas principais tabelas e os relacionamentos existentes entre elas. A modelagem foi estruturada de forma a garantir integridade referencial, organização lógica das informações e suporte às funcionalidades essenciais do sistema. No diagrama, é possível visualizar as entidades responsáveis pelo cadastro de usuários, salas, provedores e reservas, todas interligadas por relações bem definidas que asseguram consistência dos dados e facilitam consultas e operações transacionais. Esse esquema fornece a base estrutural para o funcionamento do sistema e orienta tanto o desenvolvimento *backend* quanto futuras manutenções e evoluções.

Figura 2 - Tabelas do banco de dados



Fonte: Autor (2025)

4.2 DESENVOLVIMENTO DO *BACKEND*

Ainda durante a etapa de planejamento do projeto definiu-se a utilização de tecnologias modernas e amplamente adotadas no mercado, de forma a garantir desempenho, manutenção facilitada e escalabilidade. No *backend*, optou-se por desenvolver a aplicação utilizando Java 21 em conjunto com o *framework* Spring, devido à sua maturidade, robustez e amplo ecossistema de bibliotecas que simplificam a criação de APIs REST utilizando protocolo HTTP, segurança e persistência de dados. Para o armazenamento das informações, foi escolhido o banco de dados PostgreSQL, que oferece alta confiabilidade, performance e suporte avançado a operações transacionais. Já no *frontend*, decidiu-se pela utilização do

Angular, um *framework* moderno baseado em TypeScript que permite a criação de interfaces organizadas, reativas e de fácil manutenção, proporcionando uma experiência fluida ao usuário final. Essas escolhas tecnológicas foram feitas com base nos requisitos levantados no planejamento e na necessidade de construir uma solução estável, escalável e alinhada às boas práticas de desenvolvimento.

Na primeira etapa do desenvolvimento foi construído o *backend* da aplicação de acordo com as tecnologias escolhidas no planejamento, onde cada tabela foi mapeada para uma classe Java conforme o exemplo representado na figura 3. Utilizando as anotações *@Table*, *@Entity* e *@Column* do *framework* Spring para indicar os campos e tabelas do banco de dados.

Figura 3 - Classe java de fechadura

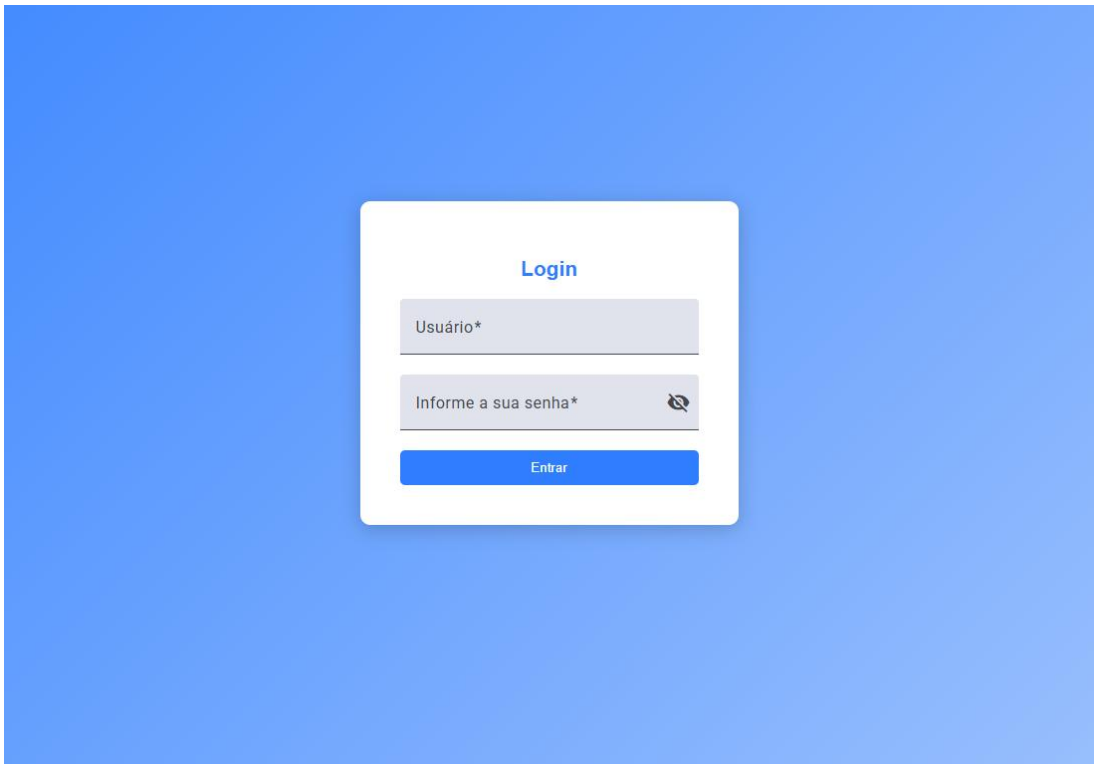
```
13    @Data
14    @Entity
15    @Table(name = "fechaduras")
16    public class Fechadura
17    {
18        @Id
19        @GeneratedValue(strategy = GenerationType.IDENTITY)
20        @Column(name = "id_fechadura")
21        private Integer idFechadura;
22
23        @ManyToOne
24        @JoinColumn(name = "id_provedor", nullable = false)
25        private Provedor provedor;
26
27        @Column(name = "chave_dispositivo", length = 500, nullable = false)
28        private String chaveDispositivo;
29    }
```

Fonte: Autor (2025)

4.3 DESENVOLVIMENTO DO *FRONTEND*

Na segunda etapa do desenvolvimento foi implementado o *frontend* começando pela tela de *login*, apresentada conforme a figura 4, representando o ponto de entrada do usuário no sistema, permitindo a autenticação de forma simples e objetiva, sendo implementado o código conforme apêndice A utilizando o padrão de componentes *@Component* do Angular, padrão este utilizado por todas as demais telas do sistema. Nessa interface, o usuário informa suas credenciais para acessar as funcionalidades de acordo com seu perfil, garantindo segurança JWT e controle de acesso. O design prioriza clareza e facilidade de uso, de modo a oferecer uma experiência intuitiva tanto em dispositivos móveis quanto em computadores, contribuindo para a usabilidade geral do sistema.

Figura 4 - Tela de login

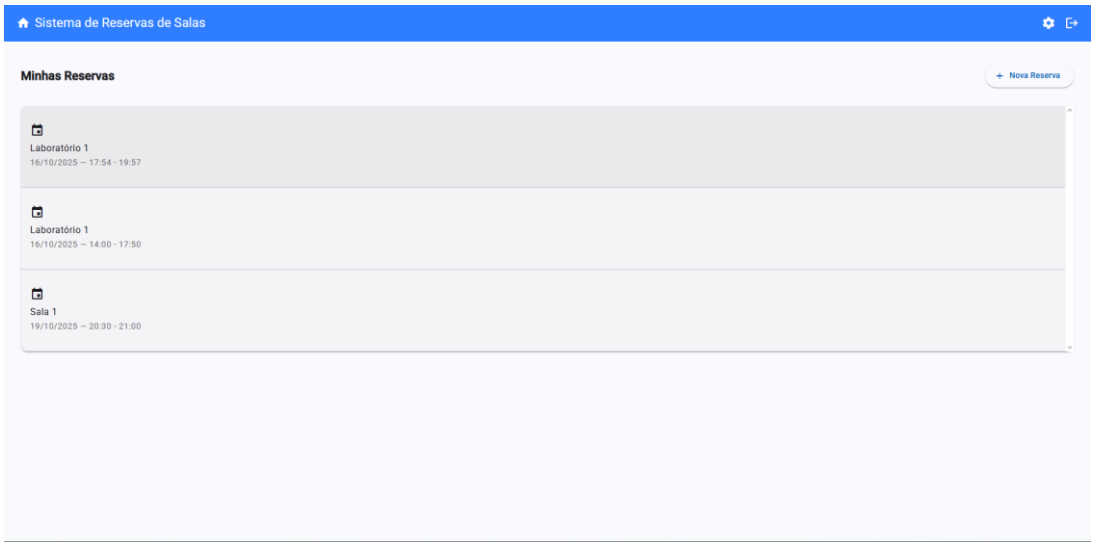


Fonte: Autor (2025)

Após o login o usuário é redirecionado ao componente da tela principal do sistema, exibida conforme a figura 5, apresenta ao usuário a lista das reservas já realizadas utilizando a API */reservas* implementada conforme apêndice B, permitindo a visualização rápida das informações mais relevantes, como data, horário e sala correspondente. Além disso, essa interface reúne os menus de acesso às

configurações, disponibilizados apenas para administradores, e a opção de *logout*, proporcionando navegação clara e organizada. Dessa forma, a tela principal funciona como um painel central que facilita o gerenciamento das reservas e o acesso às funcionalidades essenciais do sistema.

Figura 5 - Tela principal



Fonte: Autor (2025)

A interface de detalhes da reserva, mostrada na figura 6 e acessível ao clicar sobre uma reserva, permite que o usuário visualize informações completas sobre uma reserva específica, incluindo sala selecionada, data, horário e demais dados associados. Essa visualização detalhada facilita a conferência das informações registradas, apresentando os elementos de forma clara e organizada para que o usuário compreenda facilmente todas as informações relacionadas à reserva.

Figura 6 - Detalhes da reserva

Fonte: Autor (2025)

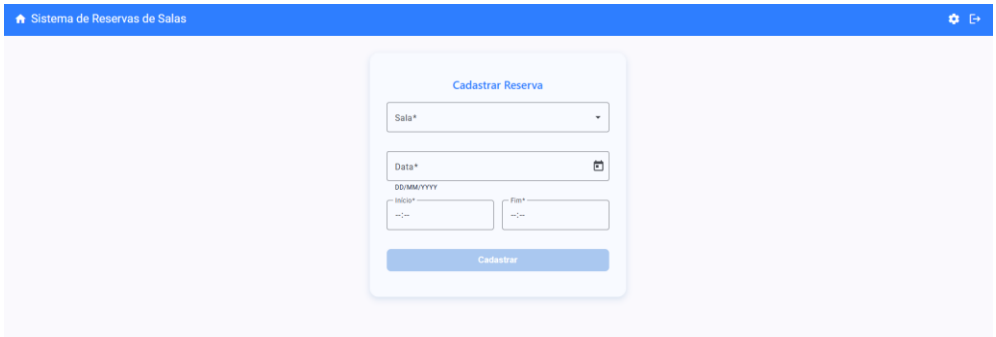
Ao acessar a opção de visualizar uma reserva, o usuário pode consultar informações adicionais, incluindo a senha temporária gerada para acesso à sala, conforme ilustrado na figura 7. Essa tela complementar exibe a senha de forma clara e objetiva, garantindo que o usuário tenha em mãos todos os dados necessários para utilizar a sala no horário reservado, mantendo a praticidade e a segurança do processo.

Figura 7 - Visualização da senha temporária

Fonte: Autor (2025)

A tela de cadastro de reserva, apresentada conforme a figura 8, permite que o usuário selecione a sala desejada, a data e o horário para efetivar uma nova reserva no sistema. A tela serve como ponto inicial para criação de novas reservas, assegurando que todas as informações necessárias sejam registradas de maneira clara e estruturada. Sendo que, ao registrar uma nova reserva são disparados os fluxos de automação assíncronos dos apêndices D e E, onde é gerado uma senha aleatória e a mesma é enviada a *Tuya Developer Platform* para o dispositivo vinculado, registrando no horário vinculado. Dessa forma, a plataforma da Tuya irá gerenciar o dispositivo e permitir o uso desta senha no horário atribuído.

Figura 8 - Cadastro de reserva



Fonte: Autor (2025)

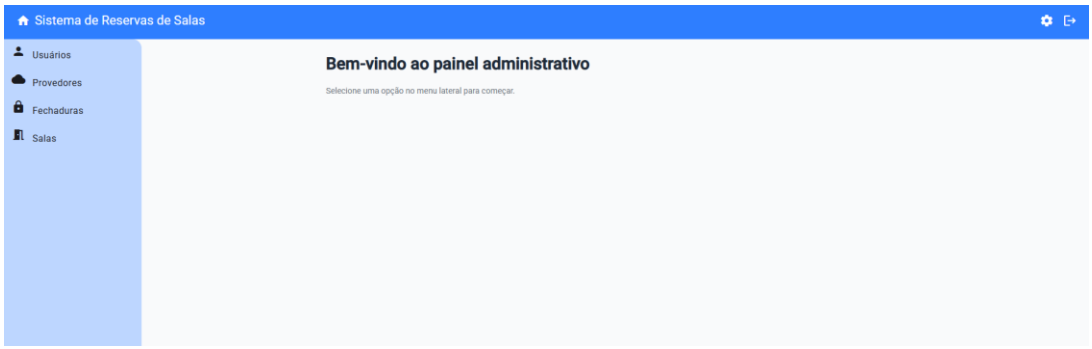
Caso o usuário selecione um horário que já esteja reservado, foi implementado no código do apêndice C uma validação em que o sistema exibirá um aviso de impedimento, conforme ilustrado na figura 9. Essa tela informa de maneira clara que o horário escolhido não está disponível, evitando conflitos de agendamento e garantindo a integridade do processo de reserva. Com isso, o usuário é orientado a escolher outro horário ou ajustar sua solicitação antes de prosseguir.

Figura 9 - Impedimento de reserva

Fonte: Autor (2025)

Usuários com perfil administrativo têm acesso a um painel exclusivo destinado às configurações e ao gerenciamento do sistema, conforme apresentado na figura 10. Esse painel reúne as funcionalidades necessárias para administrar salas, usuários e demais parâmetros operacionais, oferecendo uma visão centralizada e organizada dos recursos. Dessa forma, o administrador consegue manter o sistema atualizado e em pleno funcionamento, garantindo uma operação eficiente e segura.

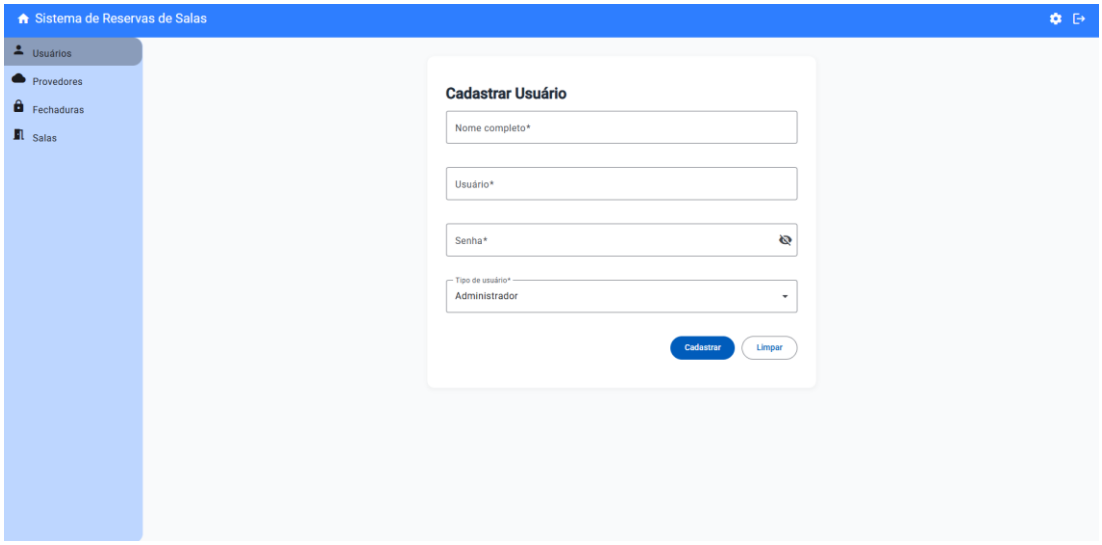
Figura 10 - Painel administrativo



Fonte: Autor (2025)

Dentro do painel administrativo, conforme ilustrado na figura 11, encontra-se a tela de cadastro de usuário, onde o administrador pode inserir informações essenciais como nome, usuário, tipo de acesso e senha. Essa interface permite gerenciar os perfis que terão acesso ao sistema, garantindo controle e segurança na utilização das funcionalidades. A disposição clara dos campos facilita o preenchimento e contribui para um processo de cadastro eficiente e organizado.

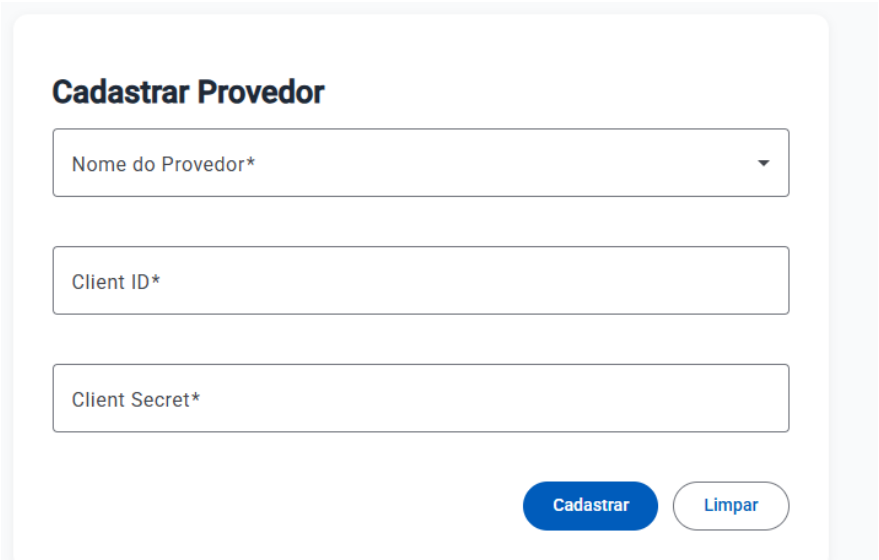
Figura 11 - Cadastro de usuário



Fonte: Autor (2025)

Dentro do painel administrativo, conforme apresentado na figura 12, encontra-se a tela dedicada ao cadastro de provedores, onde o administrador pode configurar integrações externas por meio das APIs disponíveis. Nessa interface, são inseridos dados como o provedor selecionado, o *client id* e o *client secret*, permitindo estabelecer a comunicação segura entre o sistema e serviços externos. A organização dos campos facilita o processo de configuração, garantindo que as integrações sejam efetuadas de forma clara, controlada e padronizada.

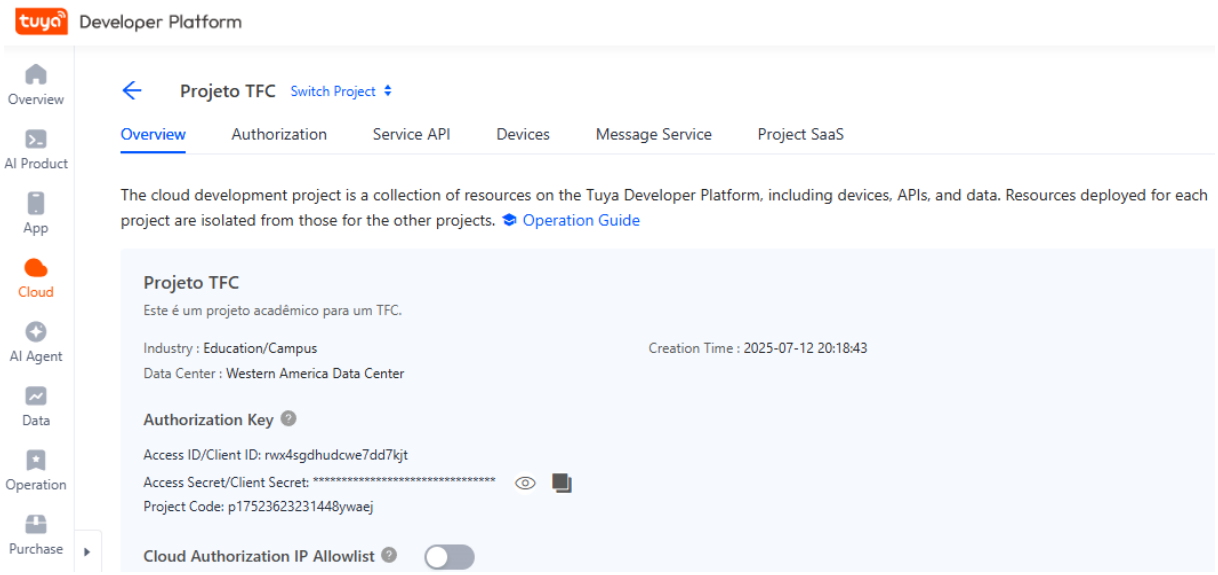
Figura 12 - Cadastro de provedor



Fonte: Autor (2025)

A figura 13 representa a tela da *Tuya Developer Platform* em que estão disponíveis as credenciais *client* vinculadas ao projeto do usuário. Nessa interface, é possível visualizar e copiar informações essenciais, como *Client ID* e *Client Secret*, que serão utilizadas para integrar o sistema com os serviços da Tuya. A tela centraliza esses dados de forma clara e acessível, permitindo que o usuário recupere rapidamente as credenciais necessárias para configurar a comunicação entre as plataformas.

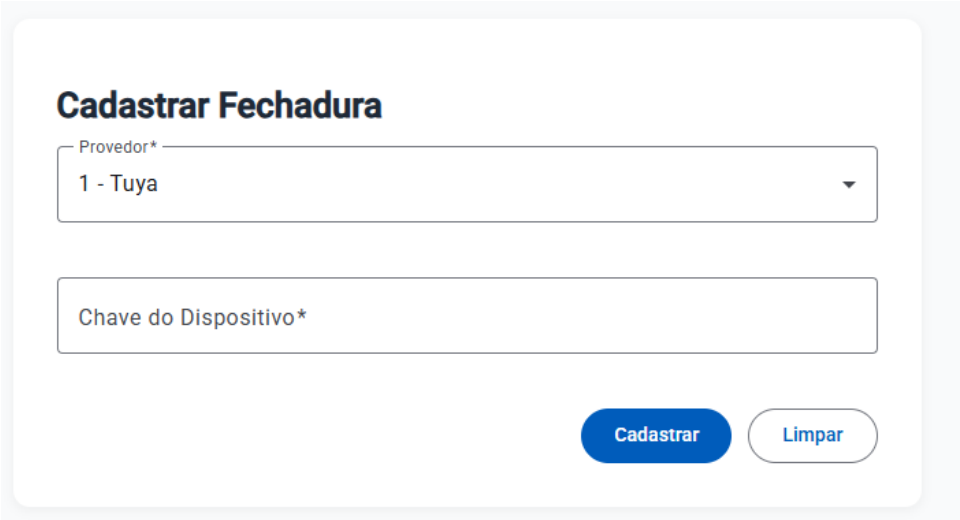
Figura 13 - Credenciais da Tuya Developer Platform



Fonte: Autor (2025)

A figura 14 apresenta a tela de cadastro de fechadura, onde o administrador pode configurar novos dispositivos a serem utilizados no sistema. Nessa interface, é disponibilizado um campo para seleção do provedor previamente cadastrado e outro para inserção do código do dispositivo registrado na plataforma Tuya. A organização dos campos facilita o processo de vinculação da fechadura ao sistema, garantindo que o dispositivo seja corretamente identificado e integrado às funcionalidades de controle de acesso.

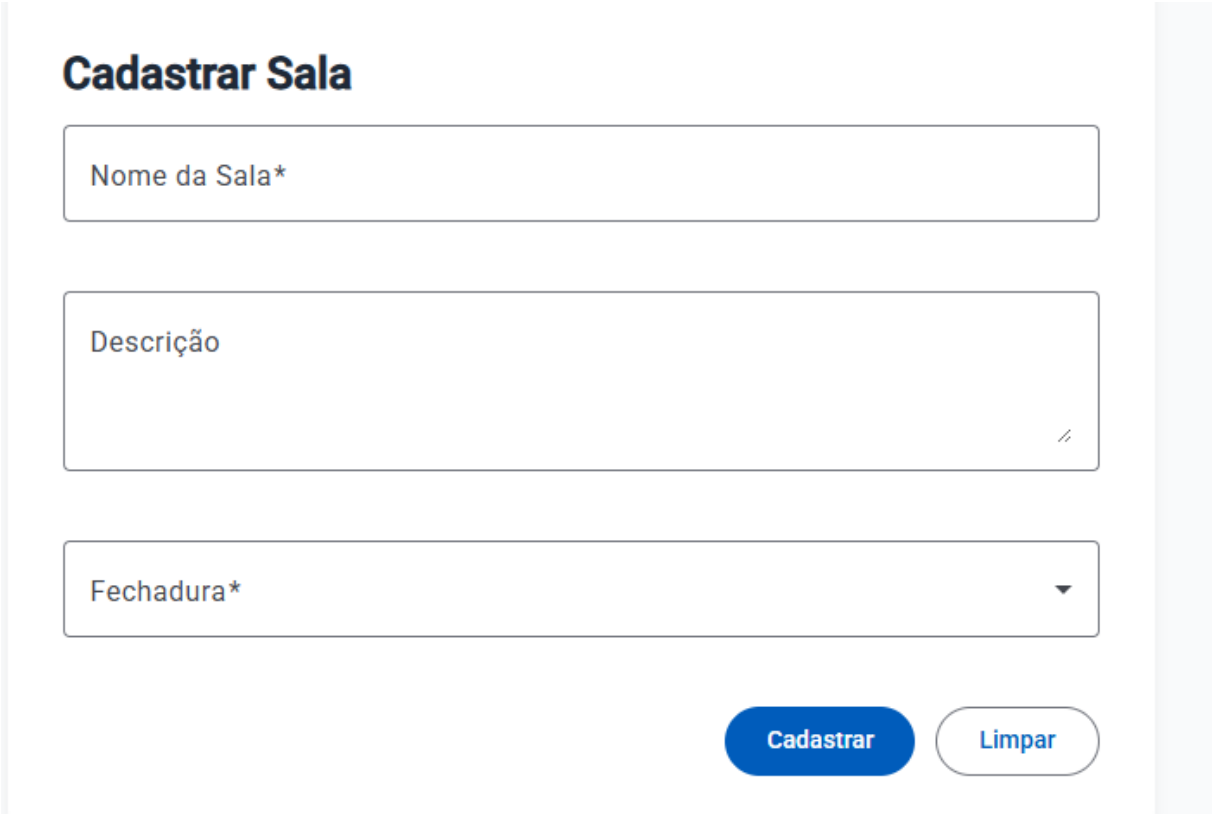
Figura 14 - Cadastro de fechadura



Fonte: Autor (2025)

No painel administrativo também é possível realizar o cadastro de salas, conforme ilustrado na figura 15. Nessa tela, o administrador pode inserir informações como nome da sala, descrição e a fechadura associada, permitindo vincular cada ambiente a um dispositivo de controle de acesso. A estrutura dos campos foi organizada para tornar o processo de cadastro simples e claro, garantindo que cada sala seja registrada de forma consistente e totalmente integrada ao restante do sistema.

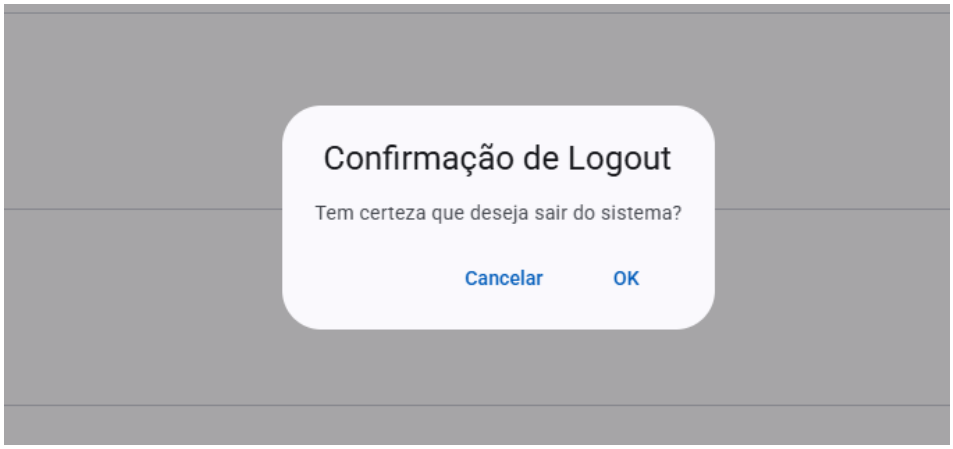
Figura 15 - Cadastro de sala



Fonte: Autor (2025)

Ao solicitar a saída do sistema, é exibida uma tela de confirmação, conforme mostrado na figura 16. Essa etapa adicional garante que o usuário não encerre sua sessão por engano, sendo implementada no código do apêndice F. Oferecendo assim uma camada extra de segurança e evitando interrupções acidentais no uso da plataforma.

Figura 16 - Confirmação de logout



Fonte: Autor (2025)

4.4 CONFIGURAÇÃO DO AMBIENTE E CONTEINERIZAÇÃO COM DOCKER

Após a conclusão da implementação dos módulos de *backend* e *frontend*, foi conduzida a etapa de preparação do ambiente de execução da solução. Nesse momento, definiu-se a utilização do Docker como ferramenta para a orquestração dos serviços necessários ao funcionamento do sistema, conforme o código representado na figura 17. Essa escolha, prevista ainda na fase de planejamento, teve como objetivo padronizar o ambiente, reduzir discrepâncias entre máquinas de desenvolvimento e facilitar o processo de implantação.

Figura 17 - Dockerfile do backend

```
# Builder
FROM bellsoft/liberica-openjre-debian:24-cds AS builder
WORKDIR /builder

ARG JAR_FILE=target/*.jar

COPY ${JAR_FILE} application.jar

RUN java -Djarmode=tools -jar application.jar extract --layers --destination extracted

# Final container
FROM bellsoft/liberica-openjre-debian:24-cds
WORKDIR /application

COPY --from=builder /builder/extracted/dependencies/ ./
COPY --from=builder /builder/extracted/spring-boot-loader/ ./
COPY --from=builder /builder/extracted/snapshot-dependencies/ ./
COPY --from=builder /builder/extracted/application/ ./

ENTRYPOINT ["java", "-jar", "application.jar"]
```

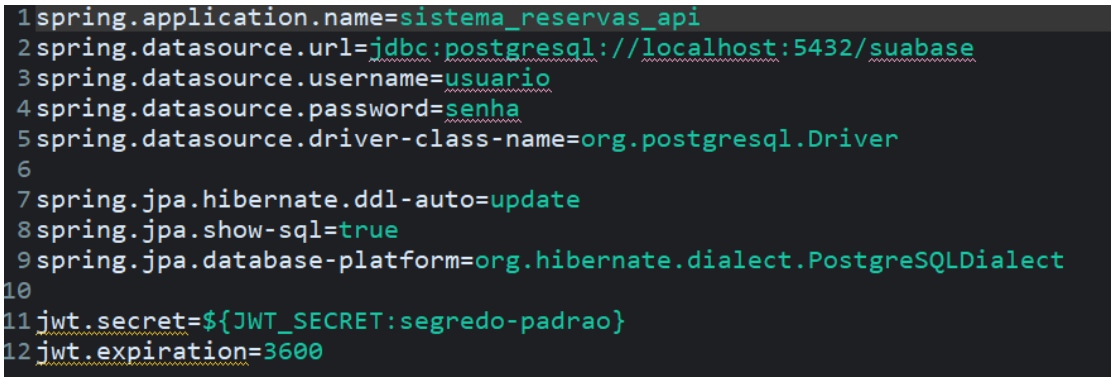
Fonte: Autor (2025)

Por meio do Docker, foram configurados os contêineres responsáveis pelo servidor da aplicação e pelos serviços auxiliares essenciais à operação do sistema. A utilização dessa abordagem permitiu estruturar um ambiente escalável e automatizado, garantindo isolamento entre componentes, maior controle sobre dependências e uma manutenção mais eficiente ao longo do ciclo de vida do projeto. Além disso, a definição centralizada da infraestrutura contribuiu para a escalabilidade futura e para a simplificação do processo de execução durante os testes integrados e validações finais.

Outro ponto importante é o uso de propriedades e variáveis de ambiente, conforme figura 18, para separar claramente as configurações sensíveis do código da

aplicação, permitindo que ajustes sejam feitos sem necessidade de recompilar ou modificar os arquivos do projeto.

Figura 18 - Propriedades Java do backend



Fonte: Autor (2025)

Essa abordagem reduz riscos de exposição acidental, melhora a organização das configurações e garante que diferentes ambientes, como desenvolvimento, homologação e produção, possam operar com parâmetros específicos de forma segura e padronizada.

CONCLUSÃO

A realização deste trabalho permitiu demonstrar, de forma prática e objetiva, a viabilidade do desenvolvimento de um sistema de reserva de salas integrado a fechaduras eletrônicas inteligentes. A construção iniciou-se com a contextualização da necessidade de tornar mais eficiente o controle de acesso a ambientes compartilhados. Em seguida, avançou-se para a definição dos requisitos funcionais, modelagem do sistema, implementação da arquitetura e, por fim, a análise dos resultados obtidos. Cada etapa foi estruturada de modo a destacar a relação direta entre o problema identificado, as decisões técnicas adotadas e o desempenho alcançado pela solução desenvolvida.

Durante o desenvolvimento, verificou-se que os objetivos estabelecidos inicialmente foram plenamente atingidos. O *backend*, implementado em Java 21 com Spring Boot e PostgreSQL, mostrou-se adequado para tratar autenticação, persistência de dados, regras de reserva e comunicação com serviços externos. O *frontend*, desenvolvido em Angular, apresentou uma interface clara, organizada e funcional, permitindo a administração de salas, fechaduras e usuários de forma intuitiva. A integração com a plataforma Tuya ocorreu conforme esperado, possibilitando a interação com a fechadura e garantindo o registro preciso das operações realizadas.

A avaliação do sistema evidenciou que a solução atende aos requisitos de praticidade, segurança e confiabilidade esperados em ambientes acadêmicos e corporativos. Os testes unitários de cada módulo, efetuados de forma manual utilizando o sistema, confirmaram que o fluxo de reserva, autenticação e registro da senha de acesso acontece de forma contínua e coerente com o modelo proposto, demonstrando a consistência e eficácia da abordagem adotada.

Dessa forma, como sugestão para trabalhos futuros, destaca-se a possibilidade de expandir o sistema por meio da integração com outros provedores além da Tuya, permitindo maior flexibilidade na escolha de modelos de fechaduras e aumentando a escalabilidade da plataforma. Outras evoluções potenciais incluem a criação de dashboards analíticos, implementação de notificações em tempo real e a adoção de novos métodos de autenticação que tornem o processo ainda mais seguro e acessível.

REFERÊNCIAS

ALVES, William P. **Projetos de Sistemas Web Conceitos, Estruturas, Criação de Banco de dados e Ferramentas de Desenvolvimento**. Rio de Janeiro: Érica, 2015. E-book. p.43. ISBN 9788536532462. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9788536532462/. Acesso em: 26 mai. 2025.

ALVES, David; PEIXOTO, Mario; ROSA, Thiago. **Internet das Coisas (IoT): Segurança e privacidade de dados pessoais**. Rio de Janeiro: Editora Alta Books, 2021. E-book. p.106. ISBN 9786555202793. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9786555202793/. Acesso em: 26 mai. 2025.

ALVES, William P. **Banco de Dados: teoria e desenvolvimento**. 2. ed. Rio de Janeiro: Érica, 2021. E-book. p.22. ISBN 9788536533759. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788536533759/. Acesso em: 16 jun. 2025.

ARAÚJO, Lis; MARINHO, Edwin. **Desafios da Autenticação e Autorização na Comunicação entre Serviços em Arquiteturas de Microsserviços**. In: WORKSHOP DE TRABALHOS DE INICIAÇÃO CIENTÍFICA E DE GRADUAÇÃO - SIMPÓSIO BRASILEIRO DE SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS (SBSEG), 23. , 2023, Juiz de Fora/MG. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2023. p. 153-164. DOI: https://doi.org/10.5753/sbseg_estendido.2023.234292. Acesso em: 16 jun. 2025.

Brasil acelera crescimento no setor de TI, mantém posição no Top 10 global e se destaca na América Latina, aponta novo estudo da ABES. [S. l.]: ABES, 21 mar. 2025. Disponível em: https://abes.com.br/brasil-acelera-crescimento-no-setor-de-ti-mantem-posicao-no-top-10-global-e-se-destaca-na-america-latina-aponta-novo-estudo-da-abes/. Acesso em: 26 maio 2025.

BARRETO, Jeanine S.; ZANIN, Aline; MORAIS, Izabelly S.; et al. **Fundamentos de segurança da informação**. Porto Alegre: SAGAH, 2018. E-book. p.13. ISBN 9788595025875. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788595025875/. Acesso em: 16 jun. 2025.

CÂMARA, Ana Beatriz Araújo. **A IMPORTÂNCIA DO UX/UI DESIGN NO DESENVOLVIMENTO DE PLATAFORMAS DE REGULAÇÃO DO ACESSO AOS SERVIÇOS DE SAÚDE: CASO DE ESTUDO NA PLATAFORMA DO REGULA RN**. Orientador: Juciano de Sousa Lacerda. 2024. Trabalho de Conclusão de Curso - TCC (Bacharel no Curso de Publicidade e Propaganda) - Universidade Federal do Rio Grande do Norte (UFRN), [S. l.], 2024. Disponível em: https://repositorio.ufrn.br/server/api/core/bitstreams/3d1cdfde-a5cd-4d11-b086-f2e2b39aadca/content. Acesso em: 16 junho 2025.

DUCKETT, Jon. **PHP&MYSQL: desenvolvimento web no lado do servidor**. Rio de Janeiro: Editora Alta Books, 2024. E-book. p.1. ISBN 9786555205930. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9786555205930/. Acesso em: 14 jun. 2025.

ERL, Thomas; MONROY, Eric B. **Computação em Nuvem: Conceitos, Tecnologia, Segurança e Arquitetura**. 2. ed. Porto Alegre: Bookman, 2024. E-book. p.531. ISBN 9788582606599. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788582606599/. Acesso em: 29 nov. 2025.

GONÇALVES, Luiz Carlos. **Sistema de controle de acesso físico por dispositivos com identificação por RFID e autenticação por biometria de impressão digital**. Orientador: Dra. Adrielle de Carvalho Santana. 2025. 83 p. Monografia (Graduação em Engenharia de Controle de Automação) - Escola de Minas, Universidade Federal de Ouro Preto, Ouro Preto, 2025. Disponível em: https://monografias.ufop.br/handle/35400000/7735. Acesso em: 02 junho 2025.

IDEALI, Wagner. **Conectividade em Automação e IoT: Protocolos I2C, SPI, USB, TCP-IP entre outros. Funcionalidade e interligação para automação e IoT**. Rio de Janeiro: Editora Alta Books, 2021. E-book. p.191. ISBN 9786555202564. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9786555202564/. Acesso em: 16 jun. 2025.

JÚNIOR, Sérgio Luiz S.; FARINELLI, Felipe A. **DOMÓTICA - AUTOMAÇÃO RESIDENCIAL E CASAS INTELIGENTES COM ARDUINO E ESP826**. Rio de Janeiro: Érica, 2018. E-book. ISBN 9788536530055. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9788536530055/. Acesso em: 14 jun. 2025.

JÚNIOR, Sinval Ferreira Resende; LEITE, Leticia Lopes. ***Um estudo sobre problemas de usabilidade no software de processos administrativos eletrônicos do Governo Federal do Brasil***. *iSys – Brazilian Journal of Information Systems*, v. 17, n. 1, jun. 2024. DOI:10.5753/isys.2024.4141. Acesso em: 16 jun. 2025.

LACERDA, Paulo S. Pádua de; SOARES, Juliane A.; LENZ, Maikon L.; et al. **Projeto de Redes de Computadores**. Porto Alegre: SAGAH, 2022. E-book. p.84. ISBN 9786556902074. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9786556902074/. Acesso em: 16 jun. 2025.

MARCONI, Marina de A.; LAKATOS, Eva M. **Metodologia Científica**. 8. ed. Rio de Janeiro: Atlas, 2022. E-book. p.295. ISBN 9786559770670. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9786559770670/. Acesso em: 16 dez. 2025.

MORAIS, Izabelly Soares de; GONÇALVES, Priscila de F.; LEDUR, Cleverson L.; et al. **Introdução a Big Data e Internet das Coisas (IoT)**. Porto Alegre: SAGAH, 2018. E-book. p.18. ISBN 9788595027640. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9788595027640/. Acesso em: 14 jun. 2025.

MILETTO, Evandro M.; BERTAGNOLLI, Silvia C. **Desenvolvimento de software II: introdução ao desenvolvimento web com HTML, CSS, javascript e PHP. (Tekne)**. Porto Alegre: Bookman, 2014. E-book. p.71. ISBN 9788582601969. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9788582601969/. Acesso em: 14 jun. 2025.

MASCHIETTO, Luís G.; VIEIRA, Anderson L N.; TORRES, Fernando E.; et al. **Arquitetura e Infraestrutura de IoT**. Porto Alegre: SAGAH, 2021. E-book. p.206. ISBN 9786556901947. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9786556901947/. Acesso em: 14 jun. 2025.

MOURA, Thiago Martiusi; NEVES, João Emmanuel D' Alkmin. **ANÁLISE DE SEGURANÇA EM DISPOSITIVOS INTERNET DAS COISAS**. Revista Interface Tecnológica, Taquaritinga, SP, v. 18, n. 2, p. 15–27, 2021. DOI: 10.31510/infa.v18i2.1174. Disponível em: https://revista.fatectq.edu.br/interfacetecnologica/article/view/1174. Acesso em: 16 jun. 2025.

MORAES, Alexandre de; HAYASHI, Victor T. **Segurança em IoT**. Rio de Janeiro: Editora Alta Books, 2021. E-book. p.58. ISBN 9788550816548. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788550816548/. Acesso em: 16 jun. 2025.

RODRIGUES, Kenny Robert. **DESENVOLVIMENTO DE UMA FECHADURA ELETRÔNICA PARA APLICATIVO DE ALUGUEIS COM AUTENTICAÇÃO POR LEITURA DE QR CODE**. Orientador: Sergio Coral. 2023. 15 p. Trabalho de Conclusão de Curso - TCC (Bacharel no Curso de Ciência da Computação) - Universidade do Extremo Sul Catarinense, [S. l.], 2023. Disponível em: http://repositorio.unesc.net/handle/1/10351. Acesso em: 19 maio 2025.

PICHETTI, Roni F.; VIDA, Edinilson S.; CORTES, Vanessa S. M P. **Banco de dados**. Porto Alegre: SAGAH, 2021. E-book. p.26. ISBN 9786556900186. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9786556900186/. Acesso em: 16 jun. 2025.

ZENKER, Aline M.; SANTOS, Jailson Costa dos; COUTO, Júlia M C.; et al. **Arquitetura de sistemas**. Porto Alegre: SAGAH, 2019. E-book. p.75. ISBN 9788595029767. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788595029767/. Acesso em: 16 jun. 2025.

APÊNDICE

APÊNDICE A – Código da tela de login

```typescript
import { Component, inject, signal } from '@angular/core';
import { FormControl, FormGroup, ReactiveFormsModule, Validators } from '@angular/forms';
import { AuthService } from '../../core/services/auth.service';
import { TokenService } from '../../core/services/token.service';
import { Router } from '@angular/router';
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatInputModule } from '@angular/material/input';
import { MatButtonModule } from '@angular/material/button';
import { MatIconModule } from '@angular/material/icon';

@Component({
  selector: 'app-login',
  templateUrl: './login.html',
  styleUrl: './login.css',
  imports: [ReactiveFormsModule, MatFormFieldModule, MatInputModule, MatButtonModule, MatIconModule]
})
export class Login {
  private loginService = inject(AuthService);
  private tokenService = inject(TokenService);
  private router = inject(Router);

  loginForm = new FormGroup({
    usuario: new FormControl('', Validators.required),
    senha: new FormControl('', Validators.required),
  });

  falhaLogin: string = '';

  hide = signal(true);

  exibirSenha(event: MouseEvent) {
    this.hide.set(!this.hide());
    event.stopPropagation();
  }

  async fazerLogin() {
    try
    {
      let token = await this.loginService.processarLogin({
        usuario: (this.loginForm.value.usuario || ''),
        senha: (this.loginForm.value.senha  || ''),
      });

      this.tokenService.saveToken(token);
      this.router.navigate(['/home']);
    } catch (erro: any) {
      if (erro.message.toLowerCase().startsWith('failed to fetch')) {
        this.falhaLogin = 'Sem conexão com o servidor';
      } else {
        this.falhaLogin = erro.message || 'Erro ao conectar com o servidor';
      }
    }

  }
}
```

Fonte: Autor (2025)

APÊNDICE B – Código das reservas

```java

@RestController
@RequestMapping("reservas")
public class ReservaController
{
    @Autowired
    private ReservaService reservaService;

    @Autowired
    private SenhaTemporariaService senhaService;

    @Autowired
    private AuthenticatedUserService authUserService;

    @GetMapping
    public ResponseEntity<List<ReservaDTO>> findActive()
    {
        return ResponseEntity.status(HttpStatus.OK)
                .body(reservaService.findActive(authUserService.getAuthenticatedUser()));
    }

    @GetMapping("/{id}")
    public ResponseEntity<Optional<ReservaDTO>> findById(@PathVariable Integer id)
    {
        return ResponseEntity.status(HttpStatus.OK).body(reservaService.findById(id));
    }

    @PostMapping
    public ResponseEntity<ReservaDTO> create(@RequestBody Reserva usuario)
    {
        return ResponseEntity.status(HttpStatus.CREATED).body(reservaService.save(usuario));
    }

    @PutMapping
    public ResponseEntity<ReservaDTO> update(@RequestBody Reserva usuario)
    {
        return ResponseEntity.status(HttpStatus.OK).body(reservaService.update(usuario));
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<?> delete(@PathVariable Integer id)
    {
        reservaService.deleteById(id);
        return ResponseEntity.status(HttpStatus.OK).build();
    }

    @GetMapping("/{id}/senha")
    public ResponseEntity<SenhaTempDTO> findPasswordById(@PathVariable Integer id)
    {
        return ResponseEntity.status(HttpStatus.OK).body(senhaService.findByIdReserva(id));
    }
}
```

Fonte: Autor (2025)

APÊNDICE C – Código de validação de reservas

```java
@Service
public class ReservaServiceImpl implements ReservaService
{
    @Autowired
    private ReservaRepository reservaRepository;

    @Autowired
    private UsuarioRepository usuarioRepository;

    @Autowired
    private SalaRepository salaRepository;

    @Autowired
    private SenhaTemporariaService senhaService;

    @Override
    public ReservaDTO save(Reserva reserva)
    {
        validateReserva(reserva);

        Reserva novaReserva = reservaRepository.save(reserva);

        // Gera senha automática vinculada à reserva
        senhaService.generatePasswordForReservation(novaReserva);

        return ReservaDTO.fromEntity(novaReserva);
    }

    private void validateReserva(Reserva reserva)
    {
        Usuario usuario = usuarioRepository.findById(reserva.getUsuario().getIdUsuario())
                .orElseThrow(() -> new RuntimeException("Usuário " + reserva.getUsuario().getIdUsuario() + " não encontrado"));

        Sala sala = salaRepository.findById(reserva.getSala().getIdSala())
                .orElseThrow(() -> new RuntimeException("Sala " + reserva.getSala().getIdSala() + " não encontrada"));

        reserva.setUsuario(usuario);
        reserva.setSala(sala);

        Optional<Reserva> tempReserva = reservaRepository.findByBetweenDate(reserva.getSala().getIdSala(), reserva.getDataReservaInicial(), reserva.getDataReservaFinal());

        if (tempReserva.isPresent())
        {
            throw new RuntimeException("Já existe uma reserva para este horário!");
        }
    }

    @Override
    public List<ReservaDTO> findAll()
    {
        return reservaRepository.findAll()
                .stream()
                .map(ReservaDTO::fromEntity)
                .toList();
    }
```

Fonte: Autor (2025)

APÊNDICE D – Código de automação da integração

```java
package com.radieske.reservasapi.integration;

import java.util.Objects;

import org.springframework.scheduling.annotation.Async;

import com.radieske.reservasapi.model.Fechadura;
import com.radieske.reservasapi.model.Provedor;
import com.radieske.reservasapi.model.Reserva;
import com.radieske.reservasapi.model.SenhaTemporaria;

public class ProcessIntegration
{
    @Async
    public static void processAutomation(SenhaTemporaria senha)
    {
        Reserva reserva = senha.getReserva();

        Fechadura fechadura = reserva.getSala().getFechadura();
        Provedor provedor = fechadura.getProvedor();

        Integration integracao = IntegrationFactory.getIntegration(provedor);

        if (Objects.isNull(provedor.getSecret()) || Objects.isNull(provedor.getClientId()))
        {
            return;
        }

        integracao.registerPassword(senha);
    }

}
```

Fonte: Autor (2025)

APÊNDICE E – Código de integração Tuya

```java
public class Tuya implements Integration
{
    private Provedor provedor;
    private AuthTuya auth;

    private final String URL_API = "https://openapi.tuyaus.com";
    private final String SIGN_METHOD = "HMAC-SHA256";

    public Tuya(Provedor provedor)
    {
        this.provedor = provedor;
        this.auth = new AuthTuya(provedor);
    }

    @Override
    public void registerPassword(SenhaTemporaria senha)
    {
        Reserva reserva = senha.getReserva();
        Fechadura fechadura = reserva.getSala().getFechadura();

        try
        {
            String strTicket = sendPost("/v1.0/devices/" + fechadura.getChaveDispositivo()
                + "/door-lock/password-ticket", "{}");

            JSONObject jsonTicket = new JSONObject(strTicket);

            String passwordEncrypted = PasswordEncryptUtil.encryptPassword(senha.getCodigo(), provedor.getSecret(), jsonTicket.getString("ticket_key"));

            LocalDateTime dataReservaInicial = reserva.getDataReservaInicial();
            LocalDateTime dataReservaFinal   = reserva.getDataReservaFinal();

            long effectiveTime = dataReservaInicial.toEpochSecond(ZoneOffset.UTC);
            long invalidTime   = dataReservaFinal.toEpochSecond(ZoneOffset.UTC);

            JSONObject tempPassword = new JSONObject();
            tempPassword.put("device_id", fechadura.getChaveDispositivo());
            tempPassword.put("name", "SenhaTemporariaExemplo");
            tempPassword.put("password", passwordEncrypted);
            tempPassword.put("password_type", "ticket");
            tempPassword.put("ticket_id", jsonTicket.getString("ticket_id"));
            tempPassword.put("effective_time", effectiveTime);
            tempPassword.put("invalid_time", invalidTime);

            sendPost("/v1.0/devices/" + fechadura.getChaveDispositivo()
                + "/door-lock/temp-password", tempPassword.toString());
        }
        catch (RuntimeException ex) {
            throw ex;
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
```

Fonte: Autor (2025)

APÊNDICE F – Código do cabeçalho do sistema

```
import { Component, inject } from '@angular/core';
import { MatButtonModule } from '@angular/material/button';
import { MatIconModule } from '@angular/material/icon';
import { MatMenuModule } from '@angular/material/menu';
import { MatToolbarModule } from '@angular/material/toolbar';
import { Router } from '@angular/router';
import { TokenService } from '../../services/token.service';
import { MatDialog } from '@angular/material/dialog';
import { DialogComponent } from '../dialog/dialog.component';

@Component({
  selector: 'app-header',
  imports: [MatToolbarModule, MatButtonModule, MatIconModule, MatMenuModule],
  templateUrl: './header.component.html',
  styleUrl: './header.component.css'
})
export class HeaderComponent {
  private router = inject(Router);
  private tokenService = inject(TokenService);
  private dialog = inject(MatDialog);

  userRole = this.tokenService.getDecodedPayload() ? this.tokenService.getDecodedPayload().ROLE : 'comum';

  goToHome() {
    this.router.navigate(['/home']);
  }

  logout() {
    const dialogRef = this.dialog.open(DialogComponent, {
      enterAnimationDuration: '250ms',
      data: {
        titulo: 'Confirmação de Logout',
        conteudo: 'Tem certeza que deseja sair do sistema?'
      }
    });

    dialogRef.afterClosed().subscribe(result => {
      if (result) {
        this.tokenService.removeToken();
        this.router.navigate(['/login']);
      }
    });
  }

  goToConfigs() {
    this.router.navigate(['/config']);
  }
}
```

Fonte: Autor (2025)