UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

VITOR DANTAS DE OLIVEIRA VALENÇA VAREJÃO

APLICAÇÃO MÓVEL PARA CONTROLE DE ACESSO A AMBIENTES ACADÊMICOS

Recife
2024

VITOR DANTAS DE OLIVEIRA VALENÇA VAREJÃO

APLICAÇÃO MÓVEL PARA CONTROLE DE ACESSO A AMBIENTES ACADÊMICOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Orientador(a): Prof. Dr. Geraldo Leite Maia Jr.

Co-Orientador(a): Prof. Dr. Marcio Rodrigo Santos Carvalho

Recife
2024

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Varejão, Vitor Dantas de Oliveira Valença.
Aplicação móvel para controle de acesso a ambientes acadêmicos / Vitor Dantas de Oliveira Valença Varejão. - Recife, 2024.
62 p. : il., tab.

Orientador(a): Geraldo Leite Maia Júnior
Cooorientador(a): Márcio Rodrigo Santos Carvalho
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, Engenharia de Controle e Automação - Bacharelado, 2024.
Inclui referências.

1. Low-Code. 2. Mendix. 3. ESP8226. 4. IoT. 5. Controle de Acesso. I. Maia Júnior, Geraldo Leite. (Orientação). II. Carvalho, Márcio Rodrigo Santos. (Coorientação). IV. Título.

620 CDD (22.ed.)

VITOR DANTAS DE OLIVEIRA VALENÇA VAREJÃO

APLICAÇÃO MÓVEL PARA CONTROLE DE ACESSO A AMBIENTES ACADÊMICOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Aprovado em: 02/08/2024

BANCA EXAMINADORA

_____

Prof. Dr. Geraldo Leite Maia Júnior (Orientador)
Universidade Federal de Pernambuco

_____

Prof. Dr. Herbert Alberico de Sá Leitão (Examinador Interno)
Universidade Federal de Pernambuco

_____

Eng. MsC. Ericles Mauricio Barbosa (Examinador Interno)
Universidade Federal de Pernambuco

Este trabalho é dedicado a meu pai, meus professores e meus amigos que me apoiaram nessa maratona.

AGRADECIMENTOS

Agradeço a todos que contribuíram para essa aventura: positivamente e negativamente,

Aos meus colegas de curso, que me mostraram que eu não estava sozinho nessa maratona. Os que foram passageiros e os que perseveraram,

Aos professores, que tiveram a maior paciência do mundo,

Aos meus amigos, que mantiveram minha sanidade,

Ao meu orientador, Geraldo Maia, que me considerava capaz de realizar suas ideias cada vez mais mirabolantes,

Ao melhor Diretório Acadêmico de todos os cursos,

E sobretudo ao meu pai, que vibrou e se decepcionou, ficou triste e depois se alegrou, deu sermão e parabenizou, e que mesmo sendo nós dois contra o mundo, nunca deixou de acreditar no meu sucesso.

“Todas as vitórias ocultam uma abdicação”.
Simone de Beauvoir

RESUMO

Os sistemas de controle de acesso desempenham um papel fundamental na segurança e na gestão eficiente de ambientes. Eles são essenciais para garantir que apenas pessoas autorizadas tenham acesso a determinados locais, protegendo as instalações e as pessoas que os frequentam. Além disso, proporcionam um maior controle sobre quem entra e sai, permitindo monitorar e registrar as atividades dos usuários. Isso contribui para a prevenção de acidentes e para a investigação de eventuais ocorrências. Tendo isso em vista, esse trabalho apresenta um sistema de controle de acesso usando ferramentas *low code* e de internet das coisas para acionar fechaduras. O sistema desenvolvido possui uma topologia coesa com outras plataformas de controle de entrada baseadas em internet das coisas: um *software* que se comunica com um dispositivo de controle que, por sua vez, aciona um atuador. Especificamente, o *software* é um aplicativo móvel desenvolvido usando a ferramenta Mendix; o dispositivo de controle é um microcontrolador ESP8226; o atuador é uma tranca elétrica da Intelbras. A comunicação *software-hardware* é feita através do protocolo *Messege Queueing Telemetry Transport* (MQTT), que é amplamente utilizado para aplicações de internet das coisas. A aplicação conta com três diferentes tipos de usuário: o administrador, que tem poder soberano sobre as configurações e dos *dashboards* de monitoramento; o coordenador, que é responsável por um (ou vários) ambientes; o usuário, um aluno ou visitante que usufrui da estrutura do ambiente.

Palavras-chave: Low-Code; Mendix; ESP8226; IoT; Controle de Acesso

ABSTRACT

Access control systems play a fundamental role in the security and efficient management of environments. They are essential for ensuring that only authorized individuals have access to certain areas, thus protecting the facilities and the people who frequent them. Additionally, these systems provide greater control over who enters and exits, allowing for monitoring and recording user activities. This contributes to the prevention of accidents and the investigation of any incidents. With this in mind, this work presents an access control system using low-code tools and Internet of Things to activate locks. The developed system has a cohesive topology with other IoT-based entry control platforms: a software that communicates with a control device, which in turn activates an actuator. Specifically, the software is a mobile application developed using the Mendix tool; the control device is an ESP8266 microcontroller; the actuator is an Intelbras electric lock. The software-hardware communication is done through the Message Queueing Telemetry Transport (MQTT) protocol, which is widely used for IoT applications. The application has three different types of users: the administrator, who has sovereign power over the configurations and monitoring dashboards; the coordinator, who is responsible for one (or several) environments; and the user, a student or visitor who utilizes the environment's infrastructure.

Keywords: Low-Code; Mendix; ESP8226; IoT; Controle de Acesso

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AWS	*Amazon Web Services*
DEE	Departamento de Engenharia Elétrica
HSQLDB	*HyperSQL DataBase*
IoT	*Internet of Things*
IIoT	*Industrial Internet of Things*
JVM	*Java Virtual Machine*
M2M	*Machine-to-Machine*
MD5	*Message-Digest 5*
MQTT	*Message Query Telemetry Transport*
NoSQL	*Not Only Structured Query Languange*
OSI	*Open System Interconnections*
QoS	*Quality of Service*
SQL	*Structured Query Language*
TCP/IP	*Transmission Control Protocol / Internet Protocol*
UI	*User Interface*
UX	*User Experience*

SUMÁRIO

1 INTRODUÇÃO

Atualmente, sistemas de controle de acesso são amplamente empregados devido à necessidade de verificar a identidade de um indivíduo e garantir a segurança de ambientes físicos. Halasz [1] cita que este controle pode ser feito de inúmeras maneiras:

- Cartões de código de barras: são cartões com um código de barras que obedecem a uma certa padronização, são impressos ou colados sobre cartões de PVC, papel-cartão ou outro material;
- Cartões de proximidade: possuem um microchip no seu interior e uma antena para captar ondas eletromagnéticas produzidas pelo leitor de proximidade;
- Leitores de biometria: é um leitor que escaneia a digital do indivíduo.
- Interface com teclado: possui um teclado no qual é possível digitar uma senha.

Para o controle de fluxo, ele cita (dentre outros):

- Fechaduras elétricas: operam com um solenoide que, ao energizada, destrava o fecho;
- Fechaduras eletromagnéticas: são eletroímãs que realizam o destravamento através da interrupção de energia;
- Catracas: são “portões” que permitem a passagem de apenas uma pessoa por vez;
- Cancelas: são “portões” que permitem a passagem de apenas um veículo por vez.

Segundo Souza [2], o controle de acesso “É o sistema que permite ou não a entrada de um indivíduo ou objeto em determinados locais, em determinados horários, mediante sua identificação”.

Ele estabelece três critérios para um sistema de controle de acesso, tais quais:

a) Criar um perímetro de controle: a implementação de um sistema de controle de acesso requer a definição de limites através de barreiras físicas ou virtuais (por exemplo, uma porta com fechadura). Dentro dessas fronteiras, é necessário estabelecer as condições e normas para o acesso;
b) Estabelecer critérios de verificação: é o meio de acesso à fronteira. Pode ser um ou uma combinação dos exemplos que seguem:
   - Algo que só você “tem”: um cartão, um código de barras, um cartão de proximidade;
   - Algo que só você “sabe”: uma senha pessoal e intransferível;
   - Algo que só você “é”: alguma parte do corpo, leitura de impressão digital, íris do olho, geometria da mão, 3D da face.
c) Registrar os eventos: o ato de registrar eventos é essencial para a criação de relatórios futuros destinados à gestão, utilizando indicadores que permitem a elaboração de planos de ações corretivas e preventivas.

Um sistema de controle de acesso eficiente é essencial por dois motivos principais: primeiro, para garantir a segurança e a organização, controlando o fluxo de pessoas; segundo, para proteger os materiais e equipamentos importantes [2].

No contexto do Departamento de Engenharia Elétrica (DEE), torna-se indispensável implementar tal sistema, considerando a presença de laboratórios equipados com ferramentas de alto valor financeiro e científico que requerem proteção adequada. Além disso, é crucial que esse sistema seja projetado para ser intuitivo e de fácil gestão, considerando que professores e alunos serão os principais usuários e mantenedores. Portanto, a simplicidade, a facilidade de manutenção e a capacitação rápida são aspectos fundamentais a serem considerados na sua implementação.

Um aspecto importante para a implementação desses sistemas é a programação, que lida com uma gama diversificada de tecnologias, linguagens e frameworks, representando uma tarefa de grande complexidade. Os profissionais que atuam neste setor necessitam de conhecimentos abrangentes de teorias computacionais e estrutura de dados além de se manterem atualizados conforme a evolução tecnológica, demandando alto custo de mão de obra.

Dito isso, plataformas *low code* têm emergido como uma solução viável para o desenvolvimento rápido e eficaz desses sistemas. Essas plataformas oferecem ferramentas intuitivas e interfaces visuais que permitem aos desenvolvedores criar aplicações de controle de acesso com facilidade, sem a necessidade de habilidades de programação avançadas [3] [4]. Esse tipo de desenvolvimento é extensivamente empregado no âmbito industrial pela agilidade e facilidade na capacitação e manutenção [5], exemplificada por normas internacionais (como, por exemplo a IEC-61131 que difundiu e padronizou a programação *ladder*), que trazem consigo a padronização de linguagens de desenvolvimento *low code* para a programação de dispositivos [6]. Uma dessas plataformas é cerne deste estudo e é apresentada neste trabalho.

1.1 Objetivos

1.1.1 Geral

Implementar um sistema de controle de acesso a salas e laboratórios através ferramentas de IoT e ambientes de desenvolvimento *low-code* para acionar trancas elétricas para garantir uma solução de segurança com baixo custo.

1.1.2 Específicos

- Definir a topologia do sistema a partir das necessidades do departamento.
- Escolher a ferramenta ideal para atender a topologia definida.
- Desenvolver um sistema simples e eficaz para o usuário e para o mantenedor.

1.2 Contribuições

Este trabalho tem como principal contribuição o desenvolvimento de um sistema de controle de acesso funcional e de fácil manutenção e capacitação para o DEE. Além disso, ele pode ser utilizado como ferramenta de ensino da plataforma de

desenvolvimento Mendix e como prova de conceito da eficácia da ferramenta no quesito de transformação digital industrial - um ramo da automação que vem se expandindo na última década.

1.3 Organização do trabalho

Esta monografia está estruturada em três capítulos: “Referencial Teórico”, “Desenvolvimento” e “Resultados”. No referencial teórico, serão introduzidas as ferramentas necessárias para a construção do sistema proposto: o protocolo de comunicação e o banco de dados escolhido. No desenvolvimento, é apresentado como a aplicação foi formulada e executada. Nos resultados, é apresentado o fluxo completo do sistema, com as telas desenvolvidas, funcionalidades e limitações.

2 REFERENCIAL TEÓRICO

Neste capítulo, são apresentados os diferentes ambientes de desenvolvimento, seus prós e contras, os critérios de escolha, a topologia proposta e o estudo de viabilidade. Esses tópicos são de extrema importância, visto que balizam toda a elaboração do sistema.

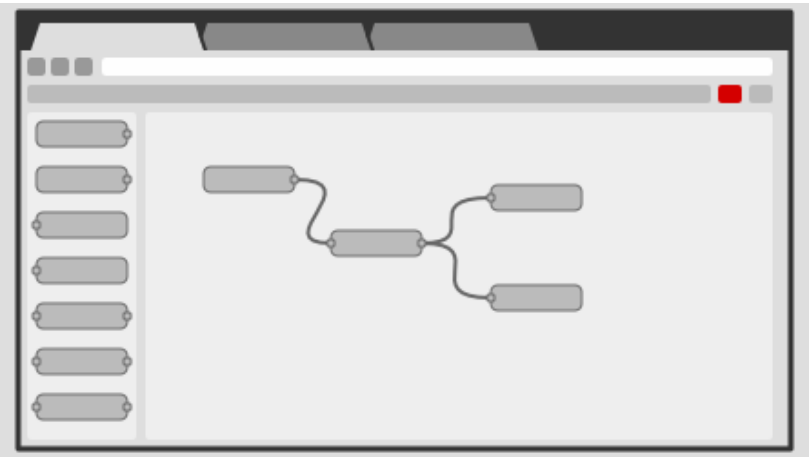
É apresentado também o protocolo de comunicação *Messege Queueing Telemetry Transport* (MQTT): o *broker* utilizado, o nível de qualidade de serviço e a criptografia de segurança.

2.1 Ambientes de Desenvolvimento

2.1.1 Node-RED

Node-RED é uma plataforma de desenvolvimento de código aberto e baseada em nuvem, originalmente criada pela equipe de Tecnologia Emergente da IBM. Seu objetivo principal é simplificar a conexão entre dispositivos de IoT, APIs e serviços online de forma visualmente intuitiva. Utilizando uma abordagem de "programação visual", os usuários podem criar aplicativos ao unir blocos de construção conhecidos como "nós" em uma interface gráfica [7], como ilustrada na Figura 1.

Figura 1 – Ilustração de fluxo feito no Node-red.



Fonte: Retirado de [8].

É construído sobre o ambiente *Node.js*, uma poderosa plataforma de execução de *JavaScript* que se baseia no motor V8 do navegador Google Chrome. Com uma arquitetura de servidor *JavaScript*, o Node-RED destaca-se por sua alta capacidade de expansão e flexibilidade [7]. Além disso, a interface do usuário é desenvolvida utilizando tecnologias web comuns, como HTML, CSS e *JavaScript*, sendo acessível através de um navegador da web.

Ele possui uma comunidade ativa e uma grande quantidade de recursos disponíveis, como tutoriais, exemplos de códigos e fóruns de discussão que ajudam os desenvolvedores a aprenderem e aprimorar suas habilidades com a ferramenta [7].

Esta ferramenta é frequentemente usada para desenvolver uma variedade de aplicações IoT, automação residencial, integração de sistemas, processamento de dados em tempo real e automação de fluxo de trabalho. Alguns exemplos de aplicações incluem controle de dispositivos domésticos inteligentes, integração de dados entre diferentes sistemas e serviços, captura e processamento de dados em tempo real de sensores e automação de processos de negócios [9] [10].

Por essas razões, o Node-RED ganhou força também no âmbito industrial, sendo palco de provas de conceito para aplicações de indústria 4.0 [11], sendo utilizado para gerir uma máquina industrial instrumentada com diversos sensores através de um *dashboard* de usuário.

2.1.2 Mendix

Mendix é uma plataforma de desenvolvimento *low code* que permite criar aplicativos empresariais de forma rápida e eficiente, sem a necessidade de habilidades de programação extensivas [12]. Fundada em 2005, o Mendix oferece uma abordagem visual para o desenvolvimento de software, permitindo que os usuários construam aplicativos por meio de uma interface gráfica intuitiva em vez de escrever código manualmente, ilustrada na Figura 2. A plataforma utiliza uma combinação de técnicas de modelagem visual, componentes pré-construídos e automação para simplificar o processo de desenvolvimento de aplicativos. Com o Mendix, o desenvolvimento é feito de forma visual e intuitiva, permitindo que

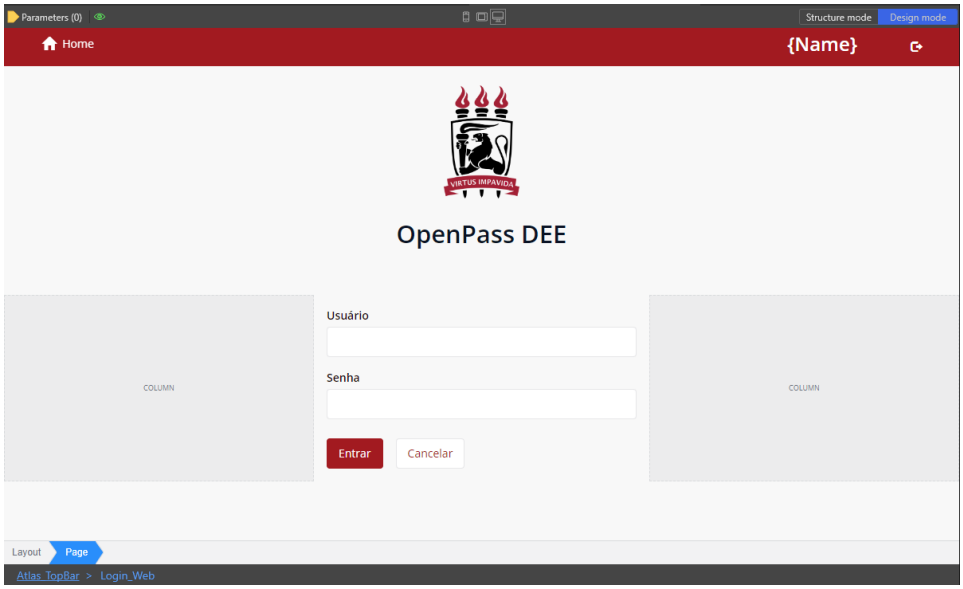
colaboradores com menos experiência em programação possam criar sistemas simples. Isso ajuda a preencher a lacuna entre negócios e TI, promovendo maior colaboração [13].

A plataforma Mendix é construída para a nuvem, o que traz benefícios como agilidade, escalabilidade e menor custo operacional. Isso a torna uma solução atraente para as empresas que buscam inovação e otimização de processos [12].

Os fundamentos do Mendix consistem em tecnologias atuais, como *Java* e *JavaScript*, fornecendo suporte para integração com diversos sistemas e serviços, tais como bancos de dados, APIs externas, dentre outros. Além disso, também oferece uma linguagem de programação própria: o *Mendix Modeler*, que serve para criar a lógica através de fluxos, arrastando e conectando blocos de programação, como demonstrada na Figura 3. Ele é empregado em fluxos chamados de *microflows* – que são fluxos processados na nuvem Mendix; e *nanoflows* – que são fluxos processados no dispositivo.

A plataforma também inclui recursos avançados de colaboração e gerenciamento de ciclo de vida de aplicativos para facilitar o trabalho em equipe e a manutenção contínua dos aplicativos desenvolvidos.

Figura 2 – Exemplo do editor de interface gráfica do Mendix.



Fonte: Imagem do autor.

Figura 3 – Exemplo de lógica feito no *Mendix*.

Fonte: Imagem do autor.

Diversos tipos de aplicativos empresariais podem ser desenvolvidos usando este ambiente. Isso inclui aplicativos de gerenciamento de relacionamento com o cliente, sistemas de gerenciamento de conteúdo, aplicativos de gerenciamento de recursos humanos, sistemas de gestão de inventário, portais de autoatendimento para clientes e funcionários, entre outros. A flexibilidade e a extensibilidade da plataforma permitem adaptar os aplicativos Mendix para atender às necessidades específicas de uma ampla variedade de setores e casos de uso empresariais.

Um exemplo de aplicação que fez grande sucesso durante a pandemia da COVID-19 foi o aplicativo móvel *Crisis Information Translated* (CIT) [14]. Em 2020, as autoridades locais da Bélgica, um país com um alto percentual de imigrantes, enfrentaram problemas graves para disseminar informações importantes sobre novas regulamentações - que sofriam mudanças semanais e, às vezes, diárias. Essas informações eram disponibilizadas primeiramente em holandês para depois ser traduzidas para diversos dialetos o mais rápido possível, mas às vezes não rápido o suficiente. Dessa forma, criou-se um constante estado de *catch-up* entre o governo e a população não nativa. Neste cenário, o CIT, que foi desenvolvido usando o ambiente Mendix, teve papel primordial na tradução imediata dessas informações para mais de 18 dialetos e foi implementado em menos de oito semanas (da concepção inicial até

a entrega final). Essa aplicação demonstra o poder de desenvolvimento ágil que o Mendix oferece.

O produto final para este tipo de desenvolvimento é um aplicativo que pode ser dividido em versão web (para domínios) e nativa (para dispositivos móveis). A plataforma também possui sua própria nuvem: é possível publicar diretamente os aplicativos desenvolvidos sem a necessidade de buscar serviços de terceiros ou mantê-los em uma rede local.

2.2 Protocolo de comunicação MQTT

Segundo Tanenbaum e Bos [15] “O conjunto de regras pelo qual computadores particulares comunicam-se é chamado de protocolo”. O protocolo de comunicação escolhido para a troca de informações entre a aplicação e o hardware é o *Message Queueing Telemetry Transport* (MQTT). Ele é do tipo *Machine-to-Machine* (M2M) que funciona com um mecanismo de publicação e subscrição baseado em *broker*. É executado sob o *Transmission Control Protocol/ Internet Protocol* (TCP/IP) e apresenta características essenciais que o tornam atrativo, tais como sua leveza, segurança, escalabilidade e eficiência, afirma Hillar [16]. Ele também pontua que no MQTT os publicadores e assinantes estão desacoplados: são clientes que apenas estabelecem conexão com o *broker*. Na Figura 4 é exemplificada uma arquitetura que utiliza como meio de comunicação o protocolo MQTT, onde:

- Um cliente publica uma mensagem no tópico “Temperatura” no *broker*;
- O *broker* recebe a mensagem e a envia para todos os clientes assinantes do tópico “Temperatura”.

É relevante destacar que o MQTT é um protocolo amplamente utilizado em aplicações IoT devido a sua capacidade de lidar com grandes volumes de dados, garantir a entrega confiável das mensagens e ser eficiente em termos de largura de banda. Sua natureza leve o torna particularmente adequado para dispositivos com recursos limitados, como microcontroladores.

O MQTT fornece níveis de garantia de entrega de mensagem, denominados *Quality of Service* (QoS). O protocolo provê três tipos de QoS:

- QoS 0: É o nível mais baixo de QoS no MQTT. Emprega um mecanismo de melhor esforço. Nele, o remetente não espera uma confirmação de entrega de mensagem. Isso significa que as mensagens enviadas usando esse QoS são recebidas no máximo uma vez;
- QoS 1: Nesse nível de QoS o foco está em garantir a entrega de mensagem pelo menos uma vez para o destinatário. Quando a mensagem é publicada com QoS 1, o remetente mantém uma cópia da mensagem até receber uma confirmação do recebimento. Caso não haja confirmação, o remetente retransmite a mensagem até que seja recebida;
- QoS 2: Oferece o mais alto nível de serviço no MQTT, garantindo que cada mensagem seja entregue exatamente uma vez. É alcançado através de uma “apresentação de 4 vias”, onde primeiro é solicitado e confirmado uma autenticação, depois o cliente confirma a recepção da chave de criptografia temporária para, em seguida, confirmar a recepção da chave de criptografia. Dessa forma, é criada uma sessão na qual a mensagem pode trafegar.

Nesse tipo de aplicação, é vantajoso trabalhar numa qualidade de serviço 2, garantindo que os dispositivos recebam a mensagem exatamente uma vez, para ser usada como gatilho para *hardwares*.

Figura 4 – Exemplo de topologia que utiliza MQTT

Fonte: Adaptado de [17].

Além disso, o MQTT possui um recurso denominado mensagens de último desejo (*Last Will* e *Testament*). Ele permite que o *broker* publique automaticamente uma mensagem em um tópico específico caso desconectem inesperadamente da rede. Isso é útil para informar clientes sobre o *status* de um dispositivo em específico (no tópico designado). As mensagens de último desejo são transmitidas quando:

- Houver uma falha na rede de comunicação;
- Falha de comunicação com tempo maior do que o configurado;
- Cliente encerra a comunicação abruptamente;
- Broker encerra a conexão de rede devido a um erro de protocolo.

No entanto, o MQTT também possui algumas limitações. Por exemplo, ele não fornece recursos avançados de segurança, como autenticação forte, o que pode ser uma preocupação em ambientes nos quais a segurança é uma prioridade. Além disso, o MQTT não oferece suporte nativo para gerenciamento de dispositivos, o que pode tornar a escalabilidade e o gerenciamento de grandes redes IoT mais desafiadores.

Para o sistema proposto, optou-se por usar o broker *mosquitto*. Ele é amplamente utilizado em aplicações de IoT devido ao seu *design* leve, tornando-o ideal para dispositivos com recursos limitados: microcontroladores como a *Raspberry Pi*. Dessa forma, é eliminado o custo adicional de ter um broker na nuvem de um provedor.

2.3 Banco de Dados

“Um banco de dados é uma coleção de dados relacionados. Por dados, entendemos fatos conhecidos que podem ser registrados e que têm significado implícito” [18][1]. No sistema proposto faz-se necessário utilizar um banco de dados, visto que além de ser capaz de identificar usuários, o sistema também deve manter um registro de eventos para se encaixar no conceito estabelecido por Souza [2].

[1] “A database is a collection of related data. By data, we mean known facts that can be recorded and that have implicit meaning”

Os bancos de dados podem ser do tipo sequencial (SQL) ou não sequencial (NoSQL). Os não sequenciais são amplamente utilizados quando a aplicação precisa armazenar um grande volume de dados não estruturados, tais quais imagens, vídeos e documentos. Já os do tipo SQL são mais indicados para aplicações cujas transações devem ser consistentes e precisas. Visto que a aplicação a ser desenvolvida requer consistência no tráfego de dados para alimentar tanto a parte nativa quanto a parte web, opta-se por um banco de dados do tipo sequencial.

A plataforma Mendix possui um banco de dados interno do tipo relacional (SQL) chamado HyperSQL Database (HSQLDB) escrito em Java. É conhecido por sua portabilidade, visto que pode ser executado em diversas plataformas que suportam máquinas virtuais Java (JVM), incluindo Windows, Linux e MacOS. Uma das suas principais vantagens é a sua natureza embutida, que significa que ele pode ser facilmente incorporado em aplicativos Java [19].

Para manter a aplicação sem custos adicionais, optou-se por utilizá-lo, uma vez que a implementação de um banco de dados externo, como o *PostgreSQL* em uma máquina virtual na *Azure* ou *Amazon Web Services* (AWS), resultaria em despesas mensais indesejadas. Além disso, o HSQLDB atende muito bem às necessidades da aplicação.

Embora esse banco interno seja de difícil acesso e utilizado exclusivamente pela aplicação, o próprio *Mendix Cloud* oferece um serviço de *backup* diário para esse banco. Essa funcionalidade permite restaurar e migrar o banco de dados, caso necessário.

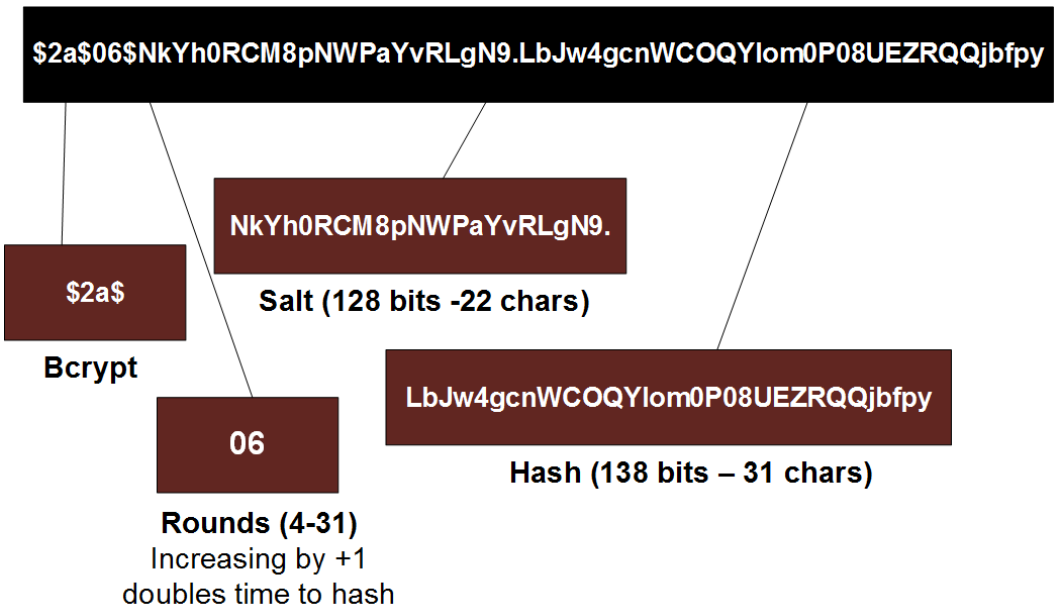
2.4 Criptografia Bcrypt

Uma das camadas de segurança para o sistema de controle de acesso é a segurança no próprio banco de dados. Utilizando o banco de dados nativo do Mendix, as senhas são armazenadas e criptografadas com função de *hash* chamada Bcrypt, criada em 1999 por Niels Povos e David Mazières usando o algoritmo de cifras *Blowfish* como base e é hoje o padrão para assegurar a criptografia de senhas [20].

Esse protocolo é unidirecional, ou seja, não pode ser revertido de volta para a forma original. Para autenticar o usuário, a senha digitada também é criptografada e comparada com a senha anteriormente criptografada armazenada no banco. Durante a criptografia, o algoritmo adiciona uma “impureza” durante o processo de *hash* chamada de “sal”, como ilustrado na Figura 5. Essa impureza é um dado aleatório de 16 bytes adicionado na frente da senha, que será criptografado em conjunto com a senha do usuário para criar uma *string* de 22 caracteres em conjunto à senha criptografada.

Além disso, o *bcrypt* é um protocolo extremamente lento – enquanto a criptografia do tipo MD5 criptografa a senha “yaaa” em menos de um microssegundo, o *bcrypt* precisa de um terço de segundo. Embora possa parecer uma desvantagem em termos de desempenho, essa característica torna-o altamente resistente a ataques de força bruta e de tabela arco-íris [20].

Figura 5 – Exemplo de criptografia *bcrypt*



Fonte: Retirado de [21]

Em suma, o *bcrypt* é considerado uma das melhores práticas para armazenar senhas de forma segura em sistemas de software. Sua capacidade de produzir *hashes* de senhas com custo adaptativo e sua resistência a ataques de força bruta o

tornam uma escolha popular para proteger informações confidenciais de usuários em aplicativos e sistemas online.

2.5 O protocolo TLS

A aplicação desenvolvida se trata de um sistema que se comunica com um servidor de envio de mensagens. Dessa forma, é importante protege-la contra ataques externos, garantindo a segurança no tráfego de mensagens. Para este fim, é necessário um protocolo de segurança de tráfego.

O protocolo *Transport Layer Security* (TLS) é uma evolução do antigo *Secure Sockets Layer* (SSL) e é utilizado para esse tipo de segurança. Tem como objetivo fornecer segurança incluindo integridade e autenticidade de dados por meio do uso de criptografias entre duas ou mais aplicações de computador que se comunicam. Este protocolo é amplamente utilizado numa variedade de aplicações e serviços na Internet, incluindo a navegação segura na web (HTTPS), e-mail seguro (SMTPS, IMAPS, POP3S), transferência de arquivos (FTPS) e muitos outros [22].

Ele opera na camada de apresentação do modelo *Open System Interconnections* e é composto de duas subcamadas principais: o protocolo TLS de registro e o protocolo TLS de *handshake*. O de registro é responsável por encapsular os dados trafegados e aplicar técnica de criptografia, enquanto o de *handshake* é responsável pela negociação dos parâmetros de segurança de conexão, autenticação das partes envolvidas e estabelecimento de chaves de criptografia [22].

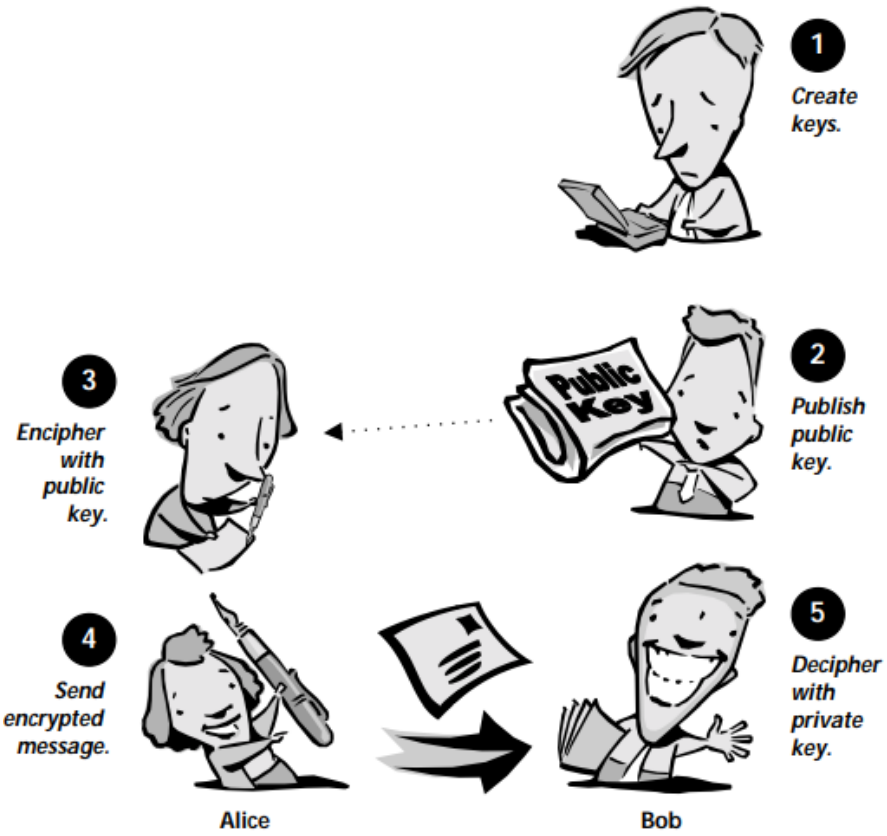
Uma de suas características principais é a sua capacidade de autenticar participantes da comunicação e garantir a integridade dos dados transmitidos. Isso se dá a partir do uso de certificados digitais, que são emitidos pelas autoridades de certificação confiáveis e usados para verificar a identidade de todas as partes envolvidas.

Segundo Thomas [22], as técnicas de criptografia podem ser classificadas de duas maneiras: a de chave secreta, onde ambas as partes possuem suas chaves e as mantém em segredo, e a de chave pública, onde uma das chaves é pública e cedida. A Figura 6 exemplifica a criptografia usando uma chave pública. Neste caso,

a chave privada é mantida para decodificação, enquanto a pública é cedida para codificação. Essa é uma das muitas topologias de criptografia por chave pública.

Dentro do servidor MQTT, é possível cadastrar certificados (que encapsulam chaves de criptografia) para habilitar o protocolo de segurança TLS. Neste caso, para estabelecer uma conexão válida, é necessário iniciá-la com o certificado e, caso seja verificado positivamente, as chaves são utilizadas pelo cliente para trafegar os dados criptografados.

Figura 6 – Exemplo de criptografia de chave pública



Fonte: Thomas (2000, p 25).

3 DESENVOLVIMENTO

Este capítulo apresenta as definições da topologia do sistema. É enfatizado a escolha do tipo de banco de dados que deverá atendê-la, assim como sua criptografia de segurança. Também é analisado, minunciosamente, a modelagem das entidades e seus relacionamentos, e o plano de funcionamento do sistema utilizando a modelagem proposta.

3.1 Formulação do problema

O primeiro passo consiste em analisar quem utilizará o sistema. Os ambientes pertencentes ao DEE serão frequentemente acessados por professores e alunos através da internet. Cada ambiente tem um responsável que é, em sua totalidade, professor. Nesse contexto, foi planejado três tipos de usuário: o administrador, que gerencia o aplicativo em si; o responsável pelos ambientes, encarregado de gerenciar o seu espaço; e o usuário comum, que apenas usufrui daquele ambiente.

Visando a simplicidade, independentemente de quem acessar, o sistema deve ser conciso e intuitivo – poucas páginas para quaisquer tipos de usuário e ações de fácil compreensão. Deve também ser capaz de prover a informação de quem entrou, quem saiu e em qual horário aconteceu [1].

3.2 Escolha da ferramenta

As principais características das duas plataformas, Node-RED e Mendix, estão apresentadas na Tabela 1. Observa-se que o Mendix oferece muito mais em ferramentas gráficas e facilidade de publicação que o Node-RED: a *Mendix Cloud* possui um plano sem custo que atende às especificações do sistema proposto, sem necessidade de pagar um plano extra, eliminando o custo da aplicação. Conclui-se que a ferramenta que mais atende aos requisitos do sistema é o Mendix.

Tabela 1 – Características gerais das plataformas.

	Node-red	***Mendix***
Linguagem de Programação	*Javascript*	*Mendix Modeler*
Produto Final	Serviço com interface de usuário	Aplicativos web e nativo
Implementação em rede pública	Necessário serviços de cloud externos, como *AWS* e *Azure*	*Mendix Cloud, AWS* ou *Azure*
Implementação em rede privada	Qualquer dispositivo que suporte o *Node-red*	Qualquer dispositivo que suporte o *Docker*
Banco de Dados	Externo	Interno ou Externo
UI/UX	Carece de ferramentas gráficas para *low-code*: é necessário programar para criar uma interface rica	É rico em ferramentas gráficas para *low-code*: não é necessário programar para criar uma interface rica
Comunidade	É fácil encontrar conteúdo e suporte	Não é tão fácil encontrar conteúdo e suporte além dos conteúdos oficiais (Fórum Mendix e *Mendix Academy*)
Aprendizado	É de fácil aprendizado	Nível moderado
Usuários da plataforma	Qualquer um pode usar	É necessário e-mail corporativo

Ao analisar o plano gratuito do *Mendix Cloud* [23], nota-se que há limitações de tempo contínuo de execução e quantidade de usuários logados: o aplicativo nesse plano terá um tempo de execução sem requisições de uma hora, após o qual entrará num estado de hibernação e “acorda” após alguma ação. O plano suporta até três usuários simultâneos, embora vários dispositivos possam estar logados no mesmo usuário, o que é uma limitação preocupante, dado que o departamento possui muito mais que três usuários.

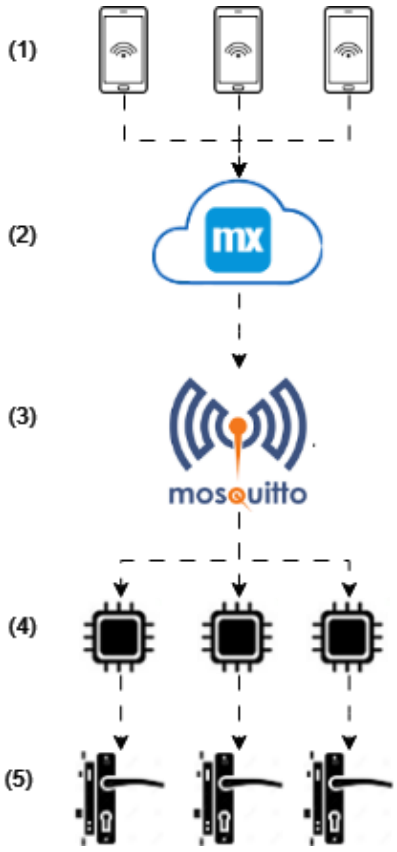
Apesar dessa limitação, o Mendix ainda é um ambiente de desenvolvimento mais atraente pela gama de ferramentas de construção gráfica e o banco de dados embutido. Para contornar essa limitação, um indivíduo pode ser diferenciado não pelo seu login e senha pessoal, mas pela identificação do seu dispositivo móvel, enquanto usa senhas para usuários compartilhados.

3.3 Topologia do sistema

No sistema proposto, podem ser identificados elementos que correspondem aos princípios de controle de acesso [1] e com IoT [24]. Na Figura 7, os seguintes passos podem ser localizados:

1) Dispositivos móveis com a aplicação nativa desenvolvida;
2) Ambiente do *Mendix Cloud*;
3) *Broker* MQTT utilizado no sistema (*mosquitto*);
4) Microcontrolador (*ESP*) utilizado para acionar a fechadura elétrica;
5) Fechadura elétrica.

Figura 7 – Ilustração da arquitetura do sistema



Fonte: Imagem do autor.

Dessa forma, a topologia proposta consiste em: um cliente da periferia acessa um *software*, se comunica com um controlador que aciona um atuador. Ela pode ser encontrada em diversas aplicações de controle de acesso baseadas em IoT, como a desenvolvida por Gupta, Kris, *et al* [25], onde uma interface que recebe mensagens de um celular se comunica com uma placa *Field Programmable Gate Array* (FPGA) para acionar as travas.

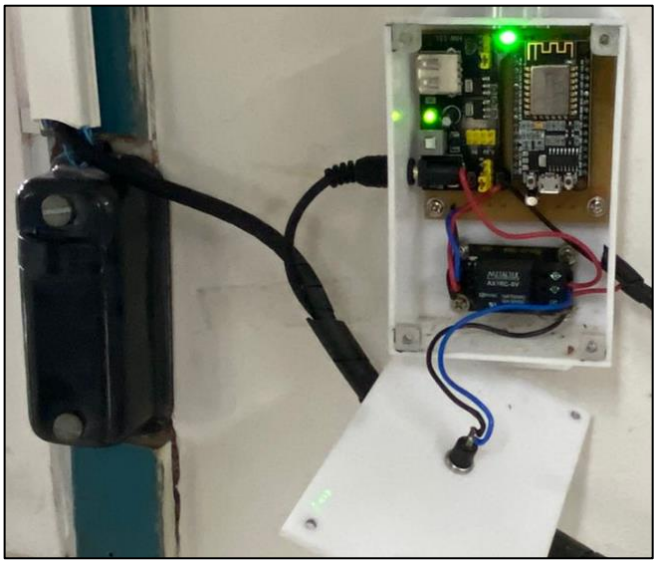
3.4 *Hardware* do controle de acesso

Após determinar os tipos de usuário, pode-se definir o plano de ação para o baixo nível, em que se designa o que o alto nível deve entregar.

Alguns ambientes do DEE já possuem sistema eletrônico de acesso, que consiste em: um microcontrolador (ESP8226), um relé e uma tranca elétrica da Intelbras, ilustrado na Figura 8.

Essa estrutura encontra-se em bom estado e opera há um período considerável de tempo (superior a três anos) sem apresentar falhas e, portanto, será mantida.

Figura 8 – Sistema eletrônico instalado em alguns ambientes do departamento



Fonte: Imagem do autor.

3.5 Modelo de domínio do banco de dados

Com a arquitetura da aplicação e suas funcionalidades definidas, o próximo passo é trabalhar no modelo de domínio do banco de dados. Primeiramente, se faz necessária uma entidade para armazenar as informações pessoais de cada usuário do sistema. Em seguida, outra entidade é criada para guardar as informações dos ambientes. Posteriormente, uma terceira entidade é configurada para caracterizar a

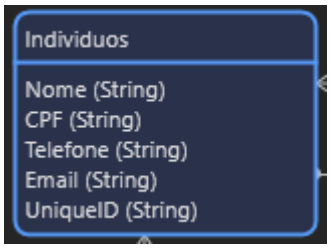
relação entre as pessoas e os ambientes. Por fim, uma última entidade é estabelecida para armazenar o histórico de acesso aos ambientes.

O modelo de domínio foi concebido principalmente para aderir à ideia de manter o sistema sem custos financeiros adicionais, adotando uma abordagem alternativa ao uso do login pessoal para identificar cada usuário e suas respectivas funções no ambiente em questão.

3.5.1 Entidade "Individuos"

Aqui é onde as informações pessoais dos usuários são armazenadas. Ela é caracterizada por cinco atributos (ilustrados na Figura 9): "Nome", "CPF", "Telefone", "Email", que descrevem informações do usuário; e "UniqueID", que identifica seu dispositivo móvel. Pode ser preenchido pelo próprio usuário através da aplicação nativa, a partir do formulário de "Pedir acesso" ou pelo administrador no *dashboard* da aplicação web.

Figura 9 – Entidade "Individuos"



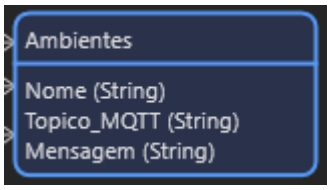
Fonte: Imagem do autor.

3.5.2 Entidade "Ambientes"

Esta entidade atua como o repositório principal para o cadastro de ambientes, armazenando configurações essenciais para o sistema de abertura de trancas. É caracterizada por três atributos principais (que estão representados na Figura 10): "Nome", que identifica de maneira única cada ambiente; "Topico_MQTT" e "Mensagem", que descreve em qual tópico e qual mensagem será enviado para o

broker. Essa entidade pode ser editada apenas pelo administrador do sistema, visto que é uma configuração da aplicação.

Figura 10 – Entidade “Ambientes”



Fonte: Imagem do autor.

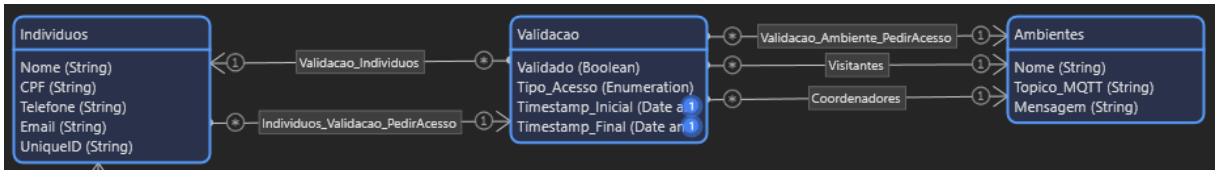
3.5.3 Entidade “Validacao”

A entidade "Validacao” caracteriza a relação entre “Individuos” e “Ambientes”, ilustrado na Figura 11. Ela serve como entidade intermediária, tendo associações com ambas entidades anteriores. Possui como atributos: “Validado”, que valida um usuário num ambiente; “Tipo_Acesso”, que pode ser limitado ou ilimitado; “Timestamp_Inicial” e “Timestamp_Final”, que determinam um intervalo de tempo para o acesso limitado.

Além disso, “Validacao” possui duas associações com “Ambientes (de muitos para um) extremamente importantes: “Visitantes” e “Coordenadores”, que têm como objetivo separar os usuários em suas respectivas hierarquias do ambiente. Vale notar que, apesar de serem do tipo “um para muitos” no modelo de domínio, a lógica por trás da aplicação garante que haja apenas um único objeto do tipo “Validacao” para um único “Ambiente”. Essa restrição foi feita devido a uma limitação do ambiente de desenvolvimento nativo, onde não são aceitas associações bilaterais - do tipo “um para um” ou “muitos para muitos”.

Objetos nessa entidade são criados pelos visitantes a partir do formulário na aplicação nativa e editados pelo coordenador do ambiente na área de gestão, validando-os (ou não) e definindo seu tipo de acesso. Os administradores do sistema conseguem cadastrar através do *dashboard* os coordenadores de cada ambiente.

Figura 11 – Relação entre “Individuos”, “Validacao” e “Ambientes”

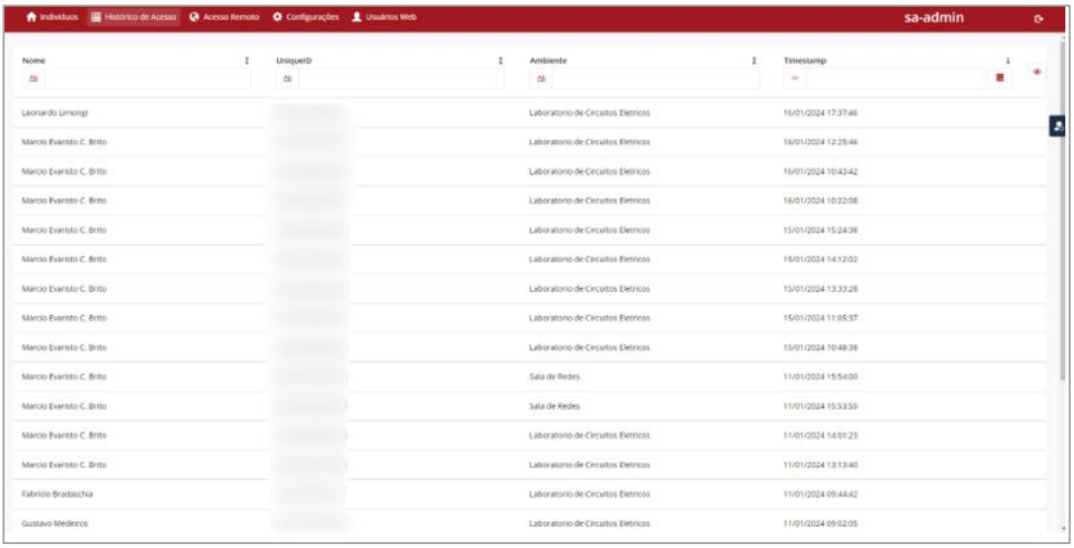


Fonte: Imagem do autor.

3.5.4 Entidade “Historico_Acesso”

É o objetivo final do sistema de controle de acesso. Obter êxito em acessar o ambiente cria um objeto na entidade “Historico_Acesso”, salvando a identificação do dispositivo, o nome do usuário, o ambiente acessado e a data/hora. Dessa forma, os administradores conseguem rastrear as atividades dos ambientes cadastrados, como demonstra a Figura 12.

Figura 12 – Página de histórico de acesso no *dashboard* web



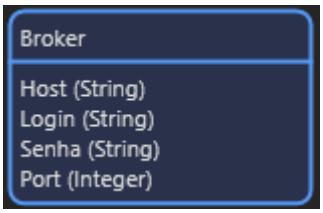
Fonte: Imagem do autor.

3.5.5 Entidade “Broker”

Guarda as informações de configuração do broker MQTT. Na Figura 13 é possível visualizar a entidade e seus atributos. São eles: “Host”, que é o endereço do servidor; “Port”, que é a porta do serviço MQTT; “Login” e “Senha” para autenticação.

É acessível apenas pelos administradores do sistema através do *dashboard*, na página de configurações.

Figura 13 – Entidade “Broker”



Fonte: Imagem do autor.

3.5.6 Entidades não-persistentes de ajuda

Estas são entidades temporárias usadas pelo Mendix para servir como variáveis do sistema. Todo campo de entrada de texto necessita de um objeto, mas nem sempre é preciso salvá-lo, como, por exemplo, para os campos de login e de filtrar. A consulta de ambientes associados a um usuário também gera uma lista não-persistente que é usada diretamente na página inicial da aplicação nativa. São elas: “Helper_Login”, utilizada no campo de login nativo; “Helper_Gestão”, que serve como filtro para a página de gestão do coordenador; e “Helper_Ambientes”, que representa a saída de uma consulta de ambientes validados para o usuário.

3.6 Hierarquia de permissões

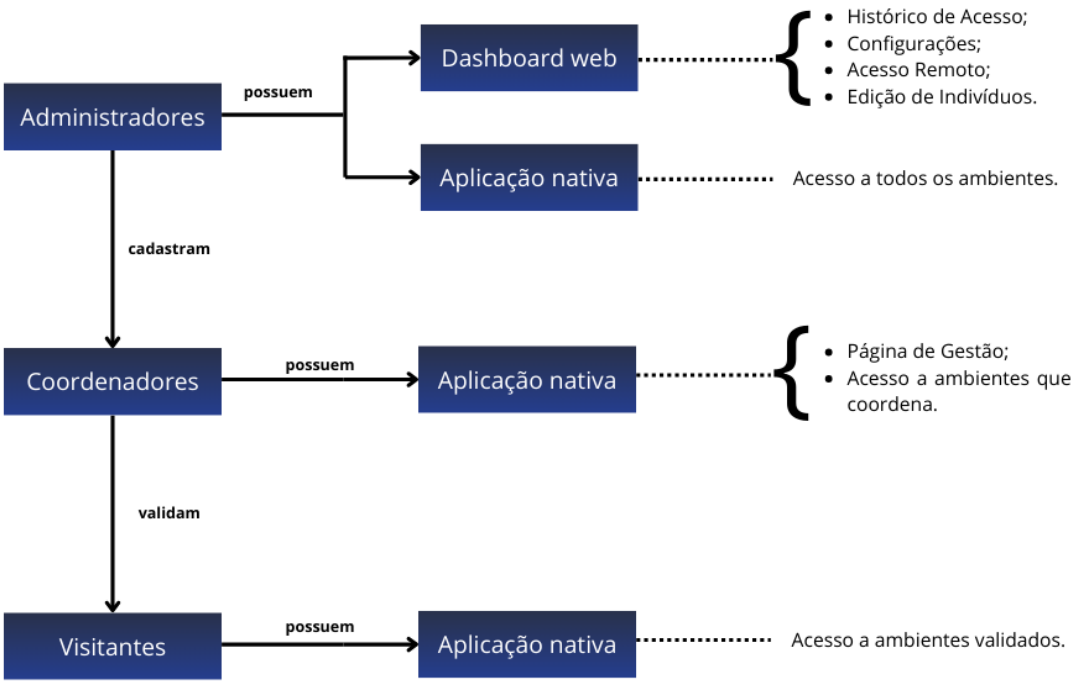
No sistema de controle de acesso, uma hierarquia bem definida organiza as funções dos usuários. Os administradores desempenham um papel crucial ao cadastrar coordenadores e têm acesso privilegiado ao *dashboard* web, fornecendo controle abrangente sobre o sistema.

Por sua vez, os coordenadores são responsáveis por validar visitantes através do dashboard de gestão de ambientes, onde podem gerenciar a validação e definir os tipos de acesso.

Os visitantes, por fim, possuem funções mais restritas, limitando-se a acessar diretamente os ambientes por meio da página inicial, sem a capacidade de modificar configurações mais amplas no sistema.

Essa hierarquia de funções assegura uma distribuição eficiente de responsabilidades e acesso, contribuindo para a gestão eficaz do controle de acesso ao longo do sistema. Na Figura 14 é ilustrada essa hierarquia através de um fluxograma.

Figura 14 – Fluxograma da hierarquia de permissões



Fonte: Imagem do autor.

3.7 Estrutura da aplicação

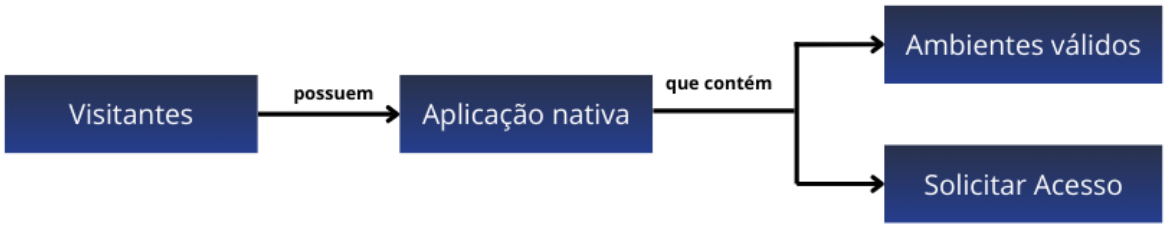
Seguindo a hierarquia do sistema, cada tipo de usuário desempenha funções distintas e, consequentemente, tem páginas específicas para acessar essas funcionalidades. As telas mais básicas da aplicação são compartilhadas por todos os tipos de usuário: tela de login, formulário de requisição de acesso e página inicial com botões para acessar os ambientes.

3.7.1 Estrutura para o usuário do tipo visitante

A estrutura para o usuário do tipo visitante apresenta as opções ilustradas na Figura 15, de modo que essas opções sejam todas acessíveis na aplicação nativa através da página inicial:

- Botões para acessar apenas os ambientes cujos cadastros e tipo de acesso são válidos;
- Botão para solicitar um novo acesso. Quando acessado através da página inicial (ao invés da página de login), o sistema já preenche automaticamente os campos de informações pessoais cadastradas com aquela identificação de dispositivo. Este formulário serve para solicitar acesso do tipo “Visitante”.

Figura 15 – Estrutura para o usuário do tipo Visitante



Fonte: Imagem do autor.

3.7.2 Estrutura para o usuário do tipo Coordenador

A estrutura para o usuário do tipo coordenador apresenta as opções ilustradas na Figura 16, de modo que essas opções sejam todas acessíveis na aplicação nativa, também através da página inicial:

- Botões para acessar apenas os ambientes que está responsável;
- Botões para acessar a página de gestão de cada ambiente;
- Botão para solicitar um novo acesso. Quando acessado através da página inicial (ao invés da página de login), o sistema já preenche automaticamente os campos de informações pessoais cadastradas com aquela identificação de dispositivo. Este formulário serve para solicitar acesso do tipo “Visitante”.

Figura 16 – Estrutura para o usuário do tipo Coordenador

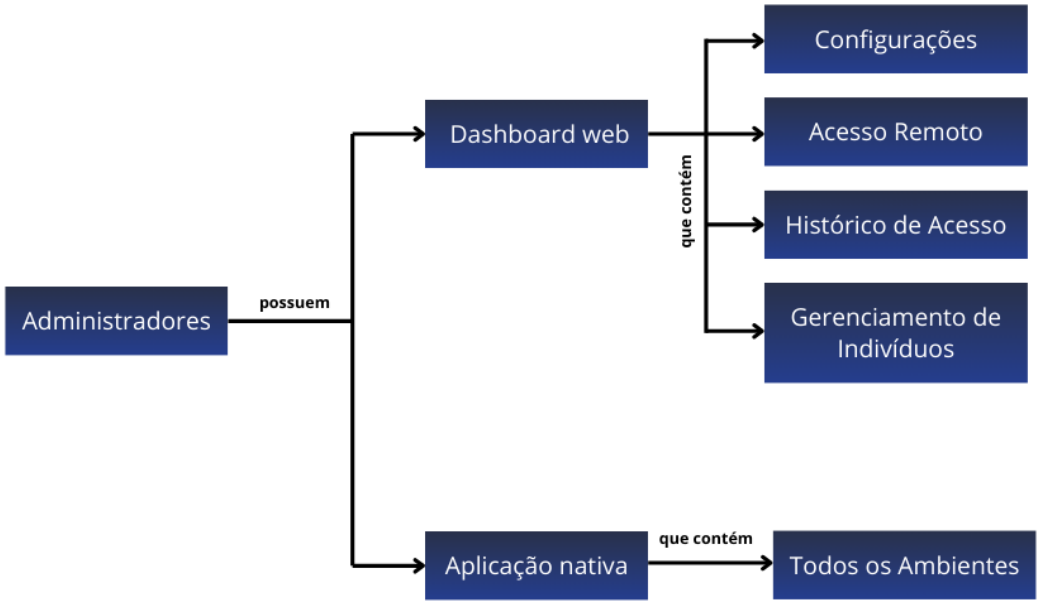
Fonte: Imagem do autor.

3.7.3 Administrador

A estrutura para o usuário do tipo administrador apresenta as opções ilustradas na Figura 17, de modo que essas opções sejam acessíveis na aplicação nativa pela página inicial e no *dashboard* web pelo menu de navegação:

- Botões para acessar todos os ambientes por meio do aplicativo;
- Página de configurações, que deverá conter as configurações do *broker* e o cadastro dos ambientes;
- Página de acesso remoto, proporcionando acesso a todos os ambientes sem a necessidade do aplicativo;
- Página de gerenciamento de indivíduos, permitindo consultar, editar e cadastrar pessoas no sistema (visitantes, coordenadores ou administradores);
- Página de histórico de acesso, que deve mostrar o histórico de acesso aos ambientes.

Figura 17 – Estrutura para o usuário do tipo Administrador



Fonte: Imagem do autor.

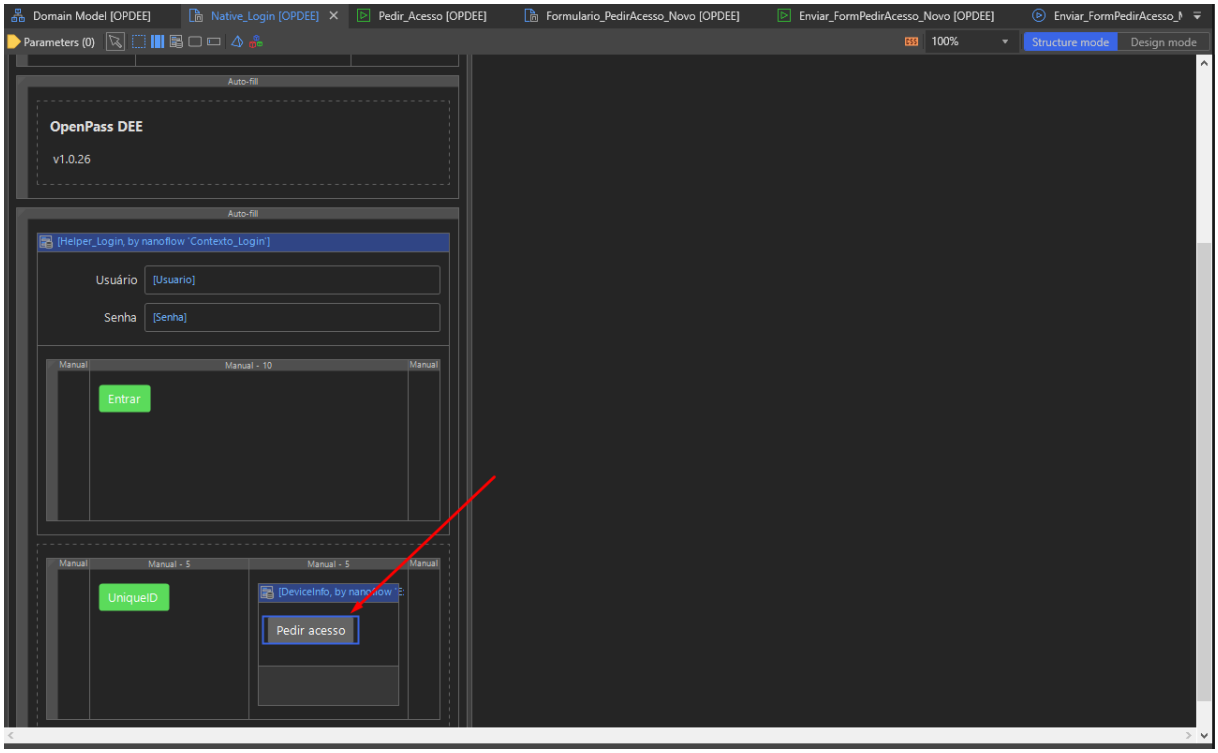
3.8 Fluxo para criação de um formulário de acesso

Nesta seção será detalhado como foi codificada a lógica para o fluxo de criação de um formulário de acesso usando o Mendix (em nível intermediário).

Primeiramente, um botão foi adicionado à(s) tela(s) na qual será possível pedir um acesso (exemplificada na Figura 18). Neste caso, foram adicionados botões tanto na tela de login quanto na tela inicial. Dado que essas telas são nativas, o botão deve acessar um *nanoflow*.

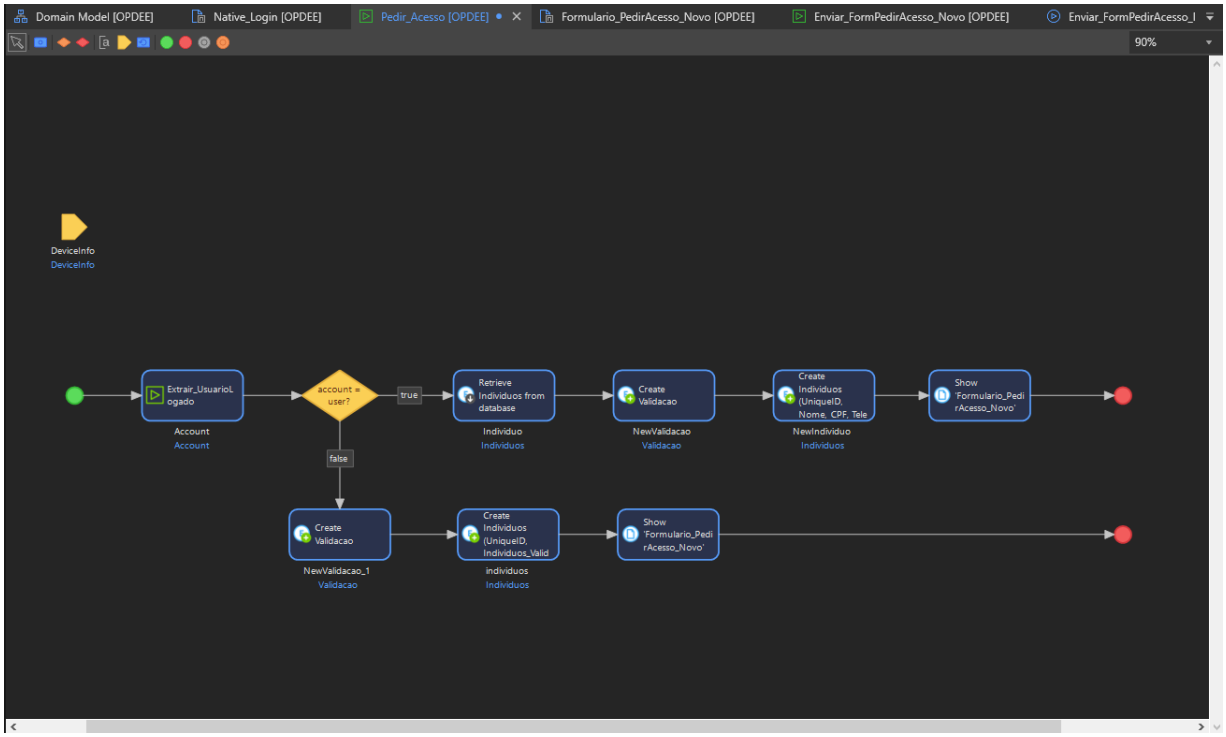
Esse *nanoflow*, por sua vez, é o encarregado de criar objetos vazios do tipo “Individuos” e “Validacao” (ou caso já tenha feito o login, consultar o objeto do tipo “Individuos” para trazer seus dados já cadastrados e criar um novo objeto do tipo “Validacao”), demonstrada pela Figura 19.

Figura 18 – Tela de login no ambiente Mendix



Fonte: Imagem do autor.

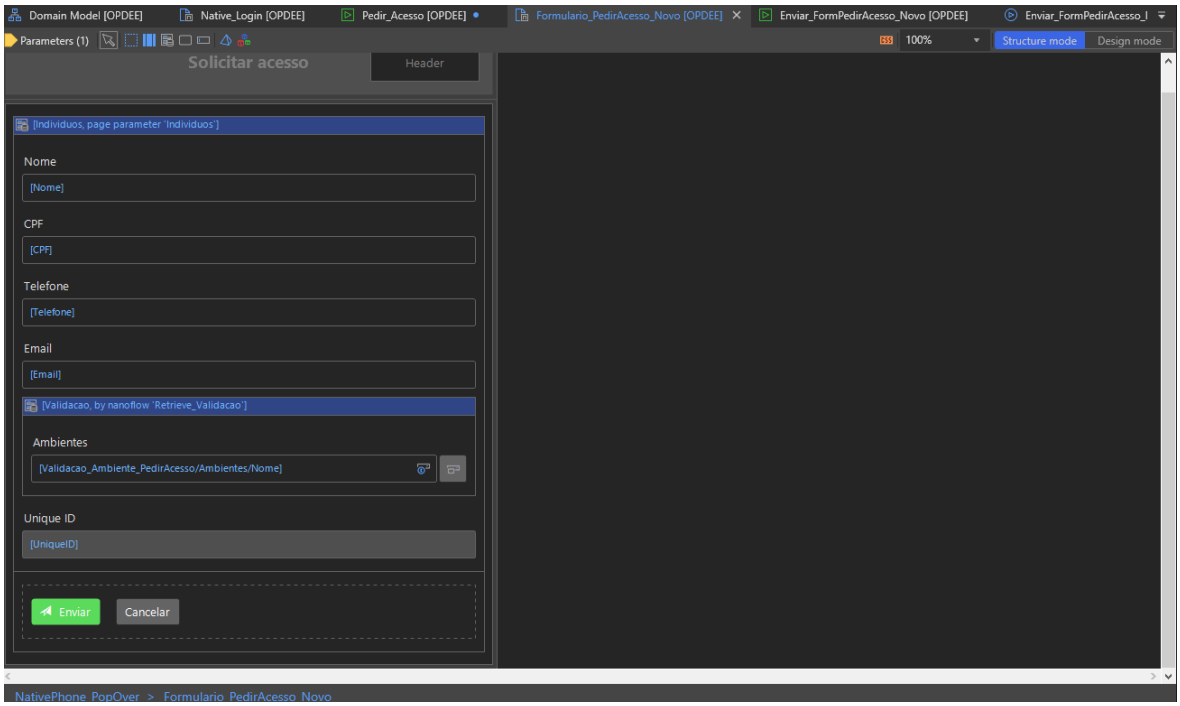
Figura 19 – *Nanoflow* do botão de pedir acesso



Fonte: Imagem do autor.

Após criados, são enviados como parâmetro para a página de formulário (Figura 20), onde o usuário deve preencher todos os campos antes de enviar.

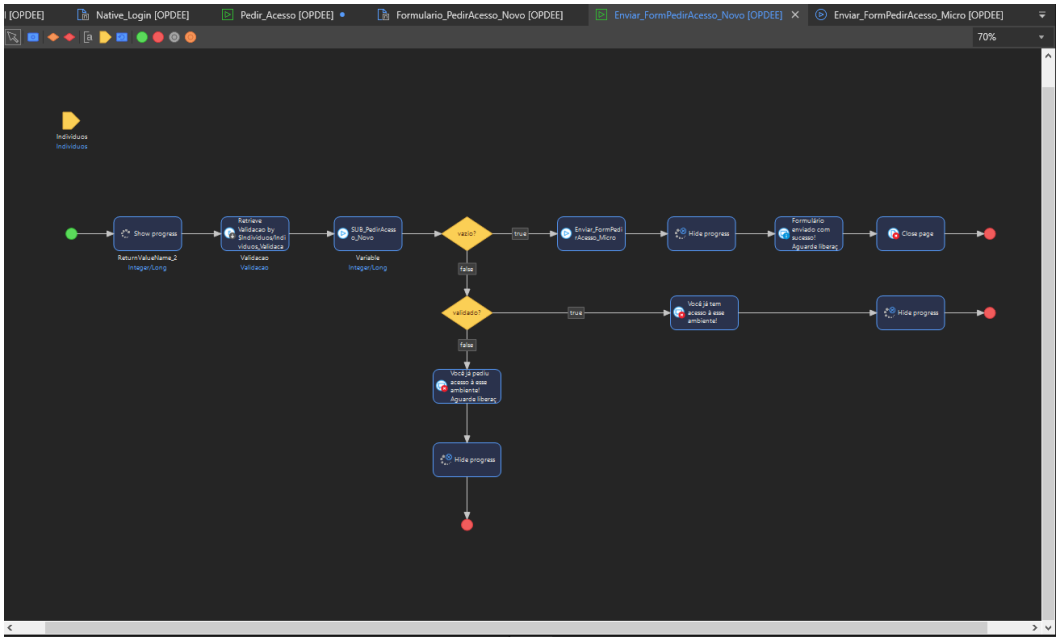
Figura 20 – Página de requisição de acesso no Mendix



Fonte: Imagem do autor.

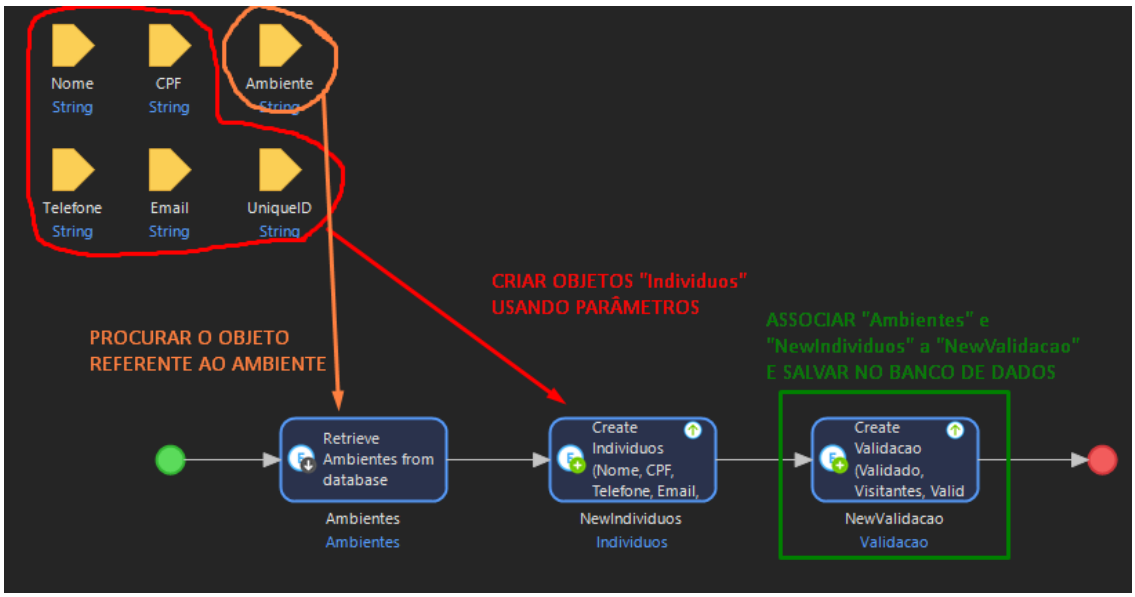
Clicar em enviar deve acionar um *nanoflow* (Figura 21) que busca se há requisições de acesso pendentes no ambiente escolhido antes de submeter o novo cadastro. Isso previne os coordenadores de terem diversos cadastros de um mesmo usuário naquele ambiente ou pedir novos acessos à ambientes em que já é um visitante validado. Caso seja o primeiro cadastro no ambiente, ele será submetido ao banco de dados na nuvem através de um *microflow*.

Figura 21 – *Nanoflow* de enviar formulário de requisição de acesso



Fonte: Imagem do autor.

Figura 22 – *Sub-microflow* de enviar formulário de requisição de acesso



Fonte: Imagem do autor.

3.9 Fluxo de comunicação com o servidor MQTT

Nesta seção será detalhado como foi codificada a lógica para o fluxo de comunicação com o *broker* usando o Mendix (nível intermediário).

Primeiramente, é de suma importância trazer todos os ambientes válidos para acesso. Isso se dá a partir de uma *list view* simples. Ele precisa de uma lista de objetos como contexto, que será fornecida através do *microflow* da Figura 23.

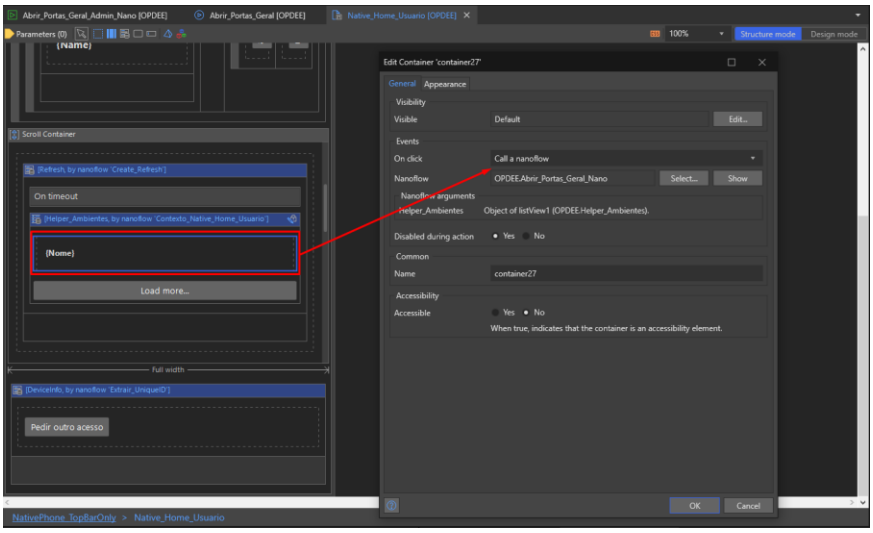
Figura 23 – *Microflow* de contexto para a *list view* do Home

Fonte: Imagem do autor.

Nessa *list view*, o container do nome recebe um evento de click (Figura 24), onde um *nanoflow* aciona o *microflow* de comunicação (Figura 24). Esse *microflow* herda o UniqueID proveniente do dispositivo bem como o objeto da lista da página inicial.

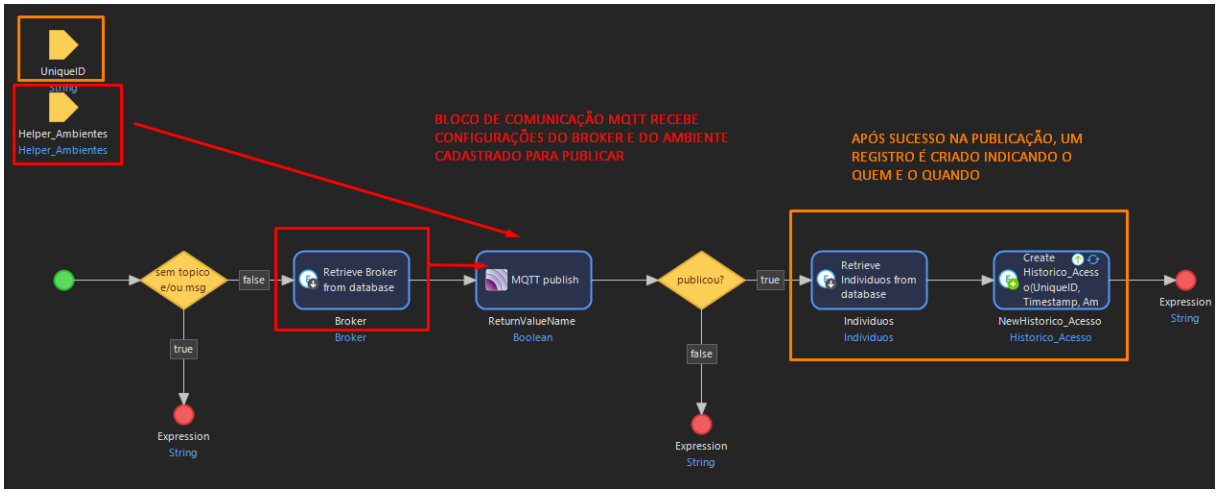
Por fim, um novo registro na entidade de histórico é criado após uma resposta positiva do bloco de publicação.

Figura 24 – Evento de click do container da *list view*



Fonte: Imagem do autor.

Figura 25 – *Microflow* de comunicação



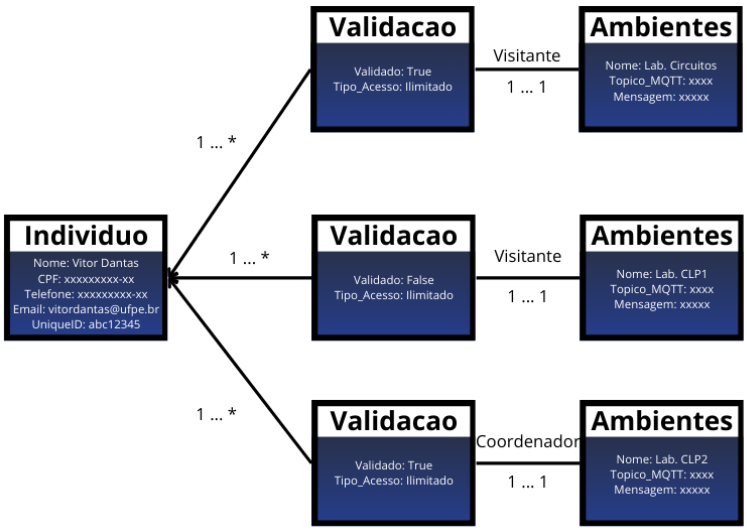
Fonte: Imagem do autor.

3.10 Exemplo de uso

A Figura 26 exemplifica como as entidades principais se relacionam. Para construir esse relacionamento, é necessário que (apenas para os cadastros do tipo “visitante”):

1. O usuário realize’ seu cadastro através do formulário de requisição de acesso;
2. O sistema crie um objeto do tipo “Individuo” para guardar suas informações pessoais;
3. Em seguida, um objeto do tipo “Validacao” foi criado e associado, a fim de caracterizar o ambiente escolhido no formulário de requisição de acesso;
4. Por sua vez, foi associado ao objeto do tipo “Ambiente”, que já estava cadastrado previamente por um administrador;
5. Por fim, o coordenador do ambiente em que o acesso foi solicitado validou (ou não) alterando o atributo “Validado” da entidade intermediária.

Figura 26 – Exemplo de indivíduo



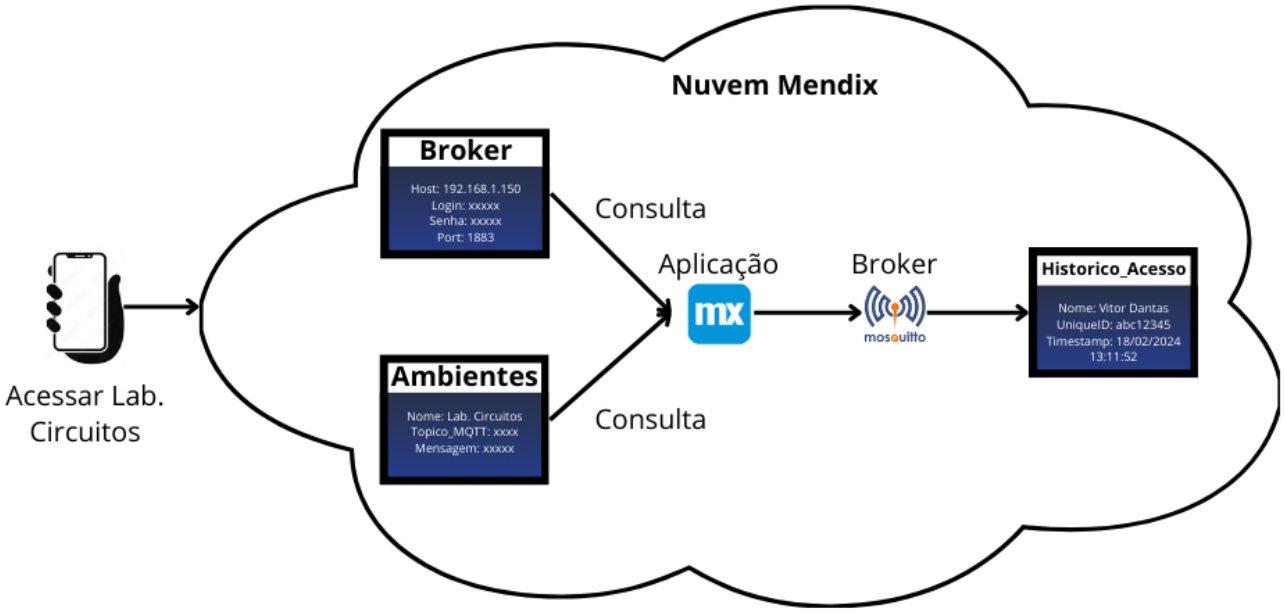
Fonte: Imagem do autor.

Neste exemplo, o acesso do indivíduo é visitante ao “Lab. Circuitos” foi negado pelo coordenador do “Lab. CLP1” e é coordenador do “Lab. CLP2”.

Já a Figura 27 demonstra os passos executados pelo sistema ao acessar um ambiente:

1. O usuário deseja acessar um ambiente válido;
2. O sistema consulta as configurações fornecidas pelos administradores através das entidades “Broker” e “Ambientes”;
3. Com todas as informações necessárias, uma conexão é estabelecida com o servidor e a mensagem é publicada no tópico correspondente;
4. Em caso de sucesso, um novo registro é criado na entidade “Histórico_Acesso”, contendo a identificação de quem e quando acessou aquele ambiente.

Figura 27 – Exemplo de acesso



Fonte: Imagem do autor.

4 RESULTADOS

Neste capítulo, é apresentado o fluxo completo do sistema, incluindo as telas desenvolvidas e as respostas da plataforma aos diversos tipos de acesso, a versão final do sistema, incluindo suas funcionalidades, limitações e segurança. Também são exibidas as telas da aplicação nativa e o *dashboard* dos administradores.

4.1 Telas da aplicação

Nesta seção, são apresentados detalhes do funcionamento do sistema desenvolvido, abrangendo seus fluxos operacionais tanto em ambientes nativos quanto na web. São mostradas as telas criadas, fornecendo uma visão abrangente das interações entre os usuários e a plataforma, juntamente com as respostas correspondentes da aplicação.

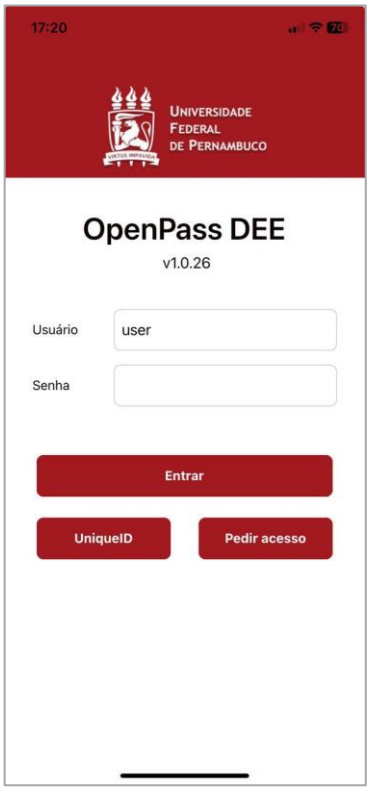
4.1.1 Parte nativa

A página comum a todos os tipos de usuários é a tela de login. Esta tela, demonstrada na Figura 28, possui campos para autenticação do usuário, um botão para solicitar acesso a um ambiente e outro para exibir o UniqueID do dispositivo por meio de uma janela pop-up.

Ao requisitar um acesso, uma nova janela é aberta com um formulário em branco (ou preenchido, caso acessado pelo botão de pedir acesso quando já autenticado no sistema) que contém os dados do usuário e um *combo box* com todos os ambientes cadastrados no sistema, ilustrada na Figura 29.

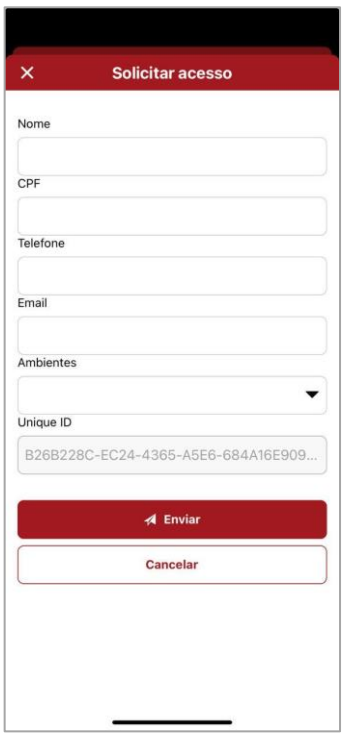
Ao autenticar-se como usuário do tipo visitante, é exibida a tela inicial (Figura 30), que contém um botão de logout, botões para acesso a ambientes válidos e outro para acessar o formulário de requisição de acesso.

Figura 28 – Tela de login do aplicativo nativo



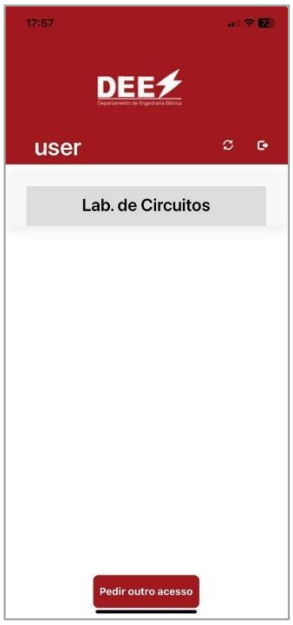
Fonte: Imagem do autor.

Figura 29 – Formulário de requisição de acesso



Fonte: Imagem do autor.

Figura 30 – Tela inicial do aplicativo nativo para visitantes e administradores

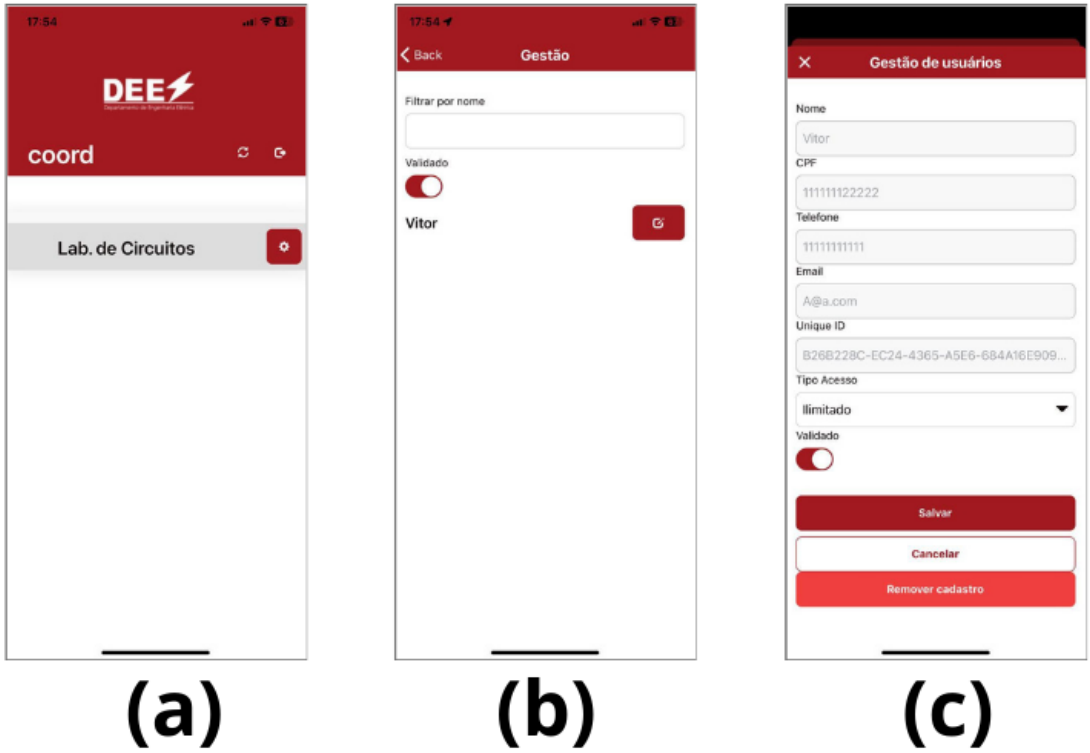


Fonte: Imagem do autor.

Para uma autenticação de usuário do tipo coordenador, é exibida uma tela inicial semelhante à do visitante (Figura 31a), porém com botões de gestão de ambientes além dos de acesso. Na página de gestão, é possível visualizar uma lista de visitantes daquele ambiente, com opções de filtragem por nome e validação de cadastro (Figura 31b), juntamente com um botão de edição de cadastro. Na janela de edição de cadastro, é possível validar e modificar o tipo de acesso do visitante (Figura 31c).

Já para uma autenticação de usuário do tipo administrador, a tela inicial é a tela de visitantes, com a diferença de que todos os ambientes cadastrados são considerados válidos por padrão. Desta forma, caso o administrador do sistema queira fazer alguma mudança de configuração ou cadastro de novos coordenadores, deverá ser feito pela parte web da aplicação.

Figura 31 – Aplicação nativa para coordenadores



(a) Tela inicial da aplicação nativa

(b) Página de gestão de ambientes

(c) Edição de cadastro

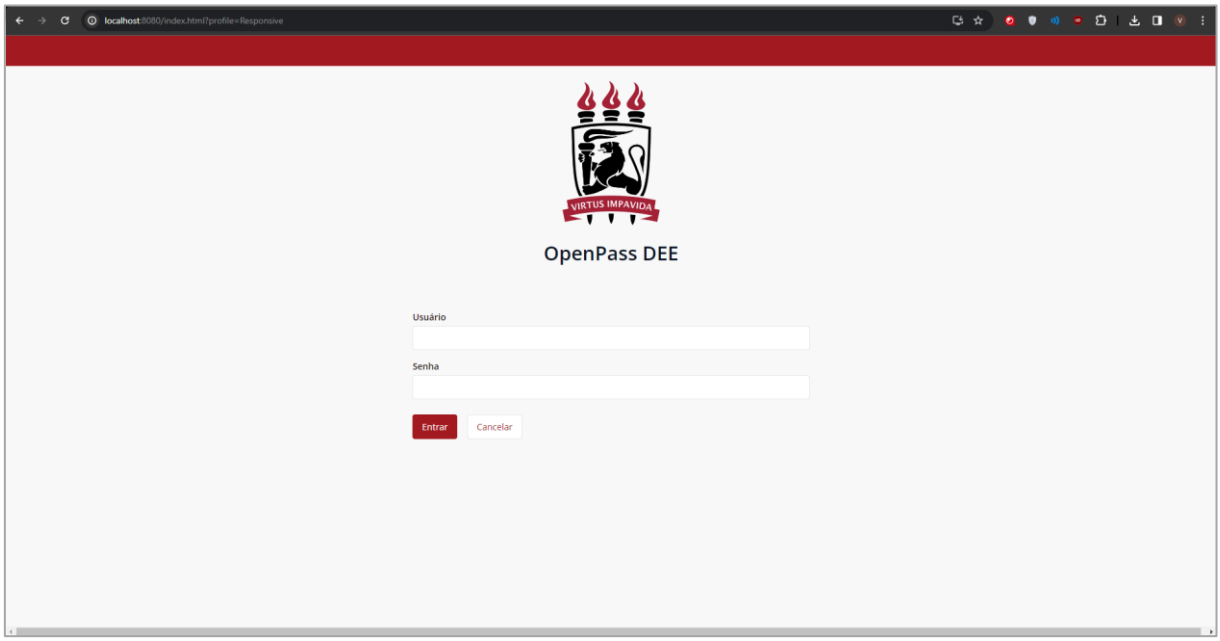
Fonte: Imagem do autor.

4.1.2 Parte web

A parte web é a parte reservada para os administradores do sistema. Nela são feitas as configurações do *broker*, cadastro de indivíduos, cadastro de ambientes, acesso remoto e monitoramento de acessos através do histórico.

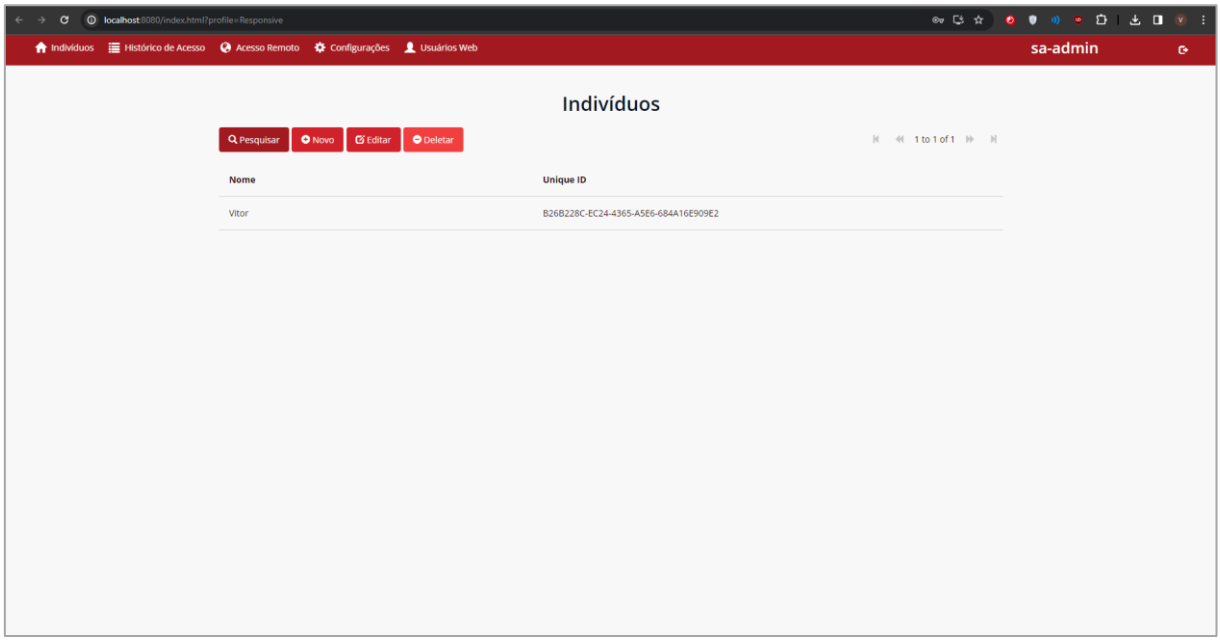
A tela de login (Figura 32) possui apenas os campos de credenciais para autenticação. Após a autenticação, a tela inicial é a de gestão de indivíduos, conforme mostrado na Figura 33. Ela tem a finalidade de gerenciar os dados cadastrais dos usuários do sistema. Ao clicar no botão de edição, uma janela *pop-up* é exibida, contendo todas as informações disponíveis para edição, exemplificada na Figura 34. É nessa interface que os administradores podem cadastrar visitantes ou coordenadores para um determinado ambiente.

Figura 32 – Página de login da aplicação web



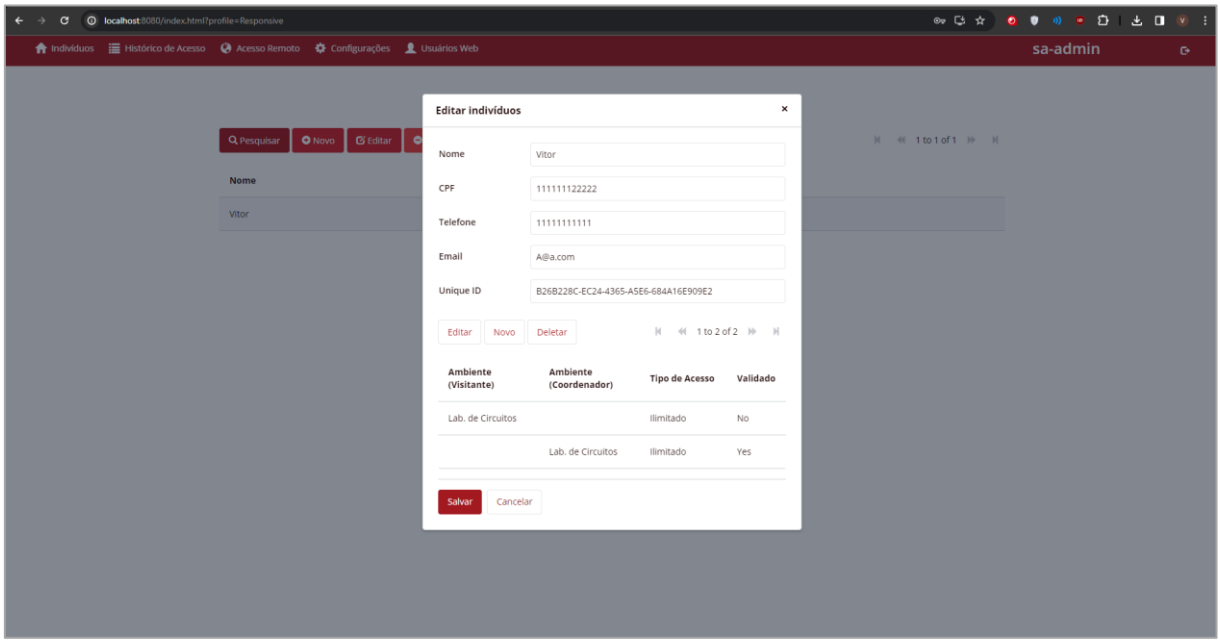
Fonte: Imagem do autor.

Figura 33 – Página de gestão de indivíduos da aplicação web



Fonte: Imagem do autor.

Figura 34 – Página de edição de indivíduos

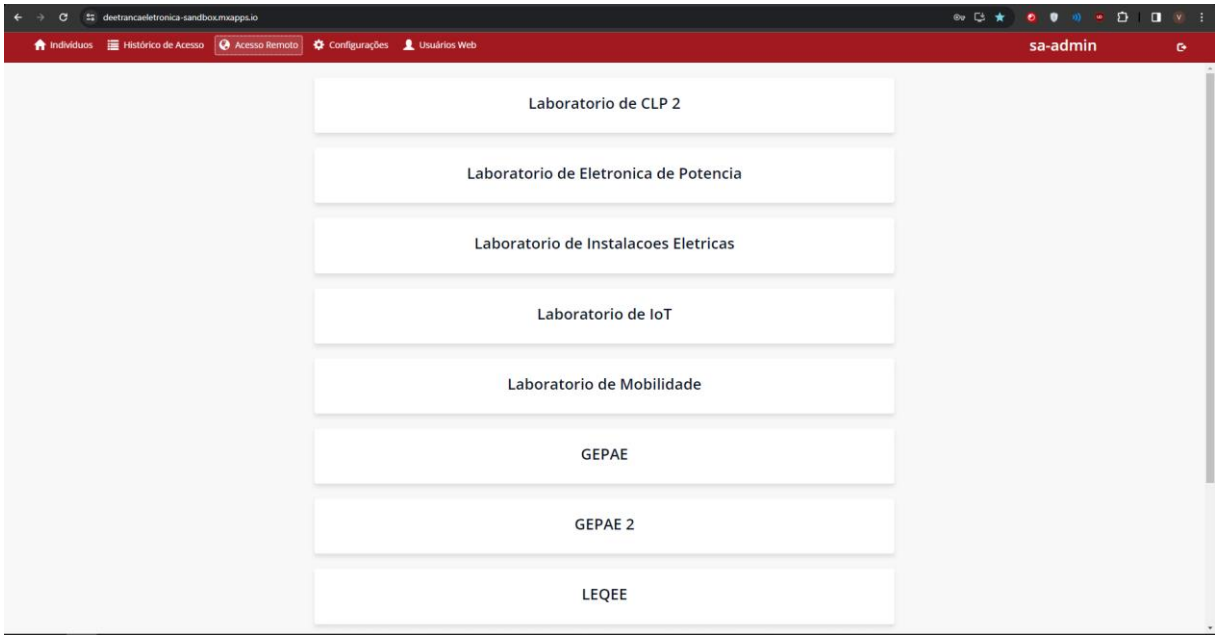


Fonte: Imagem do autor.

A página de histórico de acesso foi previamente demonstrada na Figura 12. Este recurso destina-se apenas à visualização, uma vez que esse tipo de informação não deve ser editada nem excluída para preservar a integridade do sistema.

Já a página de acesso remoto é um paliativo criado para os administradores do sistema que não possuem acesso à aplicação nativa, tendo em vista que ainda não foi possível publicar nas lojas de dispositivos com sistema operacional do tipo iOS. Ela serve exclusivamente para acessar os ambientes de forma remota e possui botões para todos os ambientes cadastrados, como ilustrado na Figura 35.

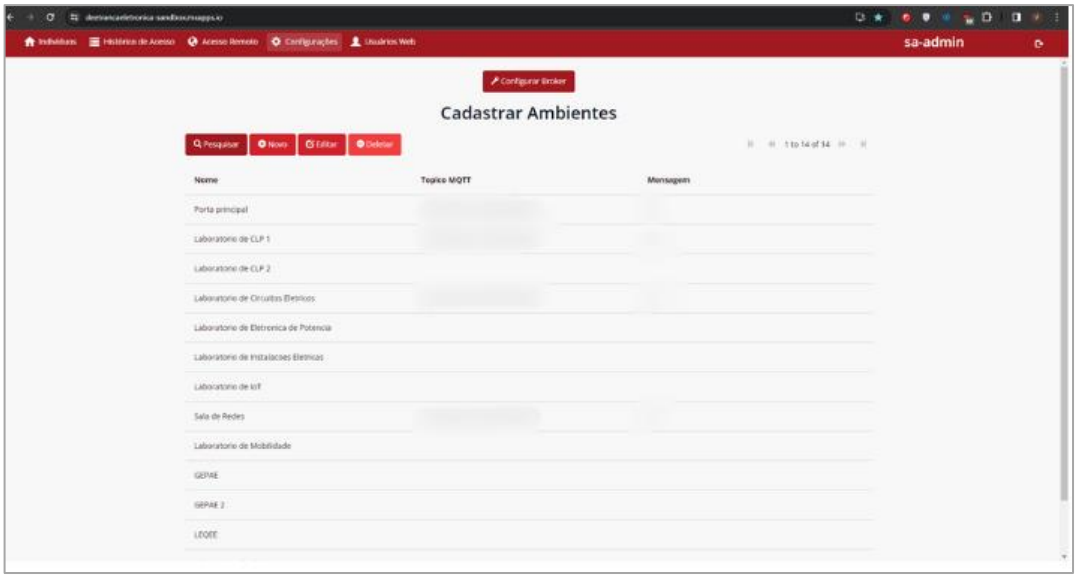
Figura 35 – Página de acesso remoto



Fonte: Imagem do autor.

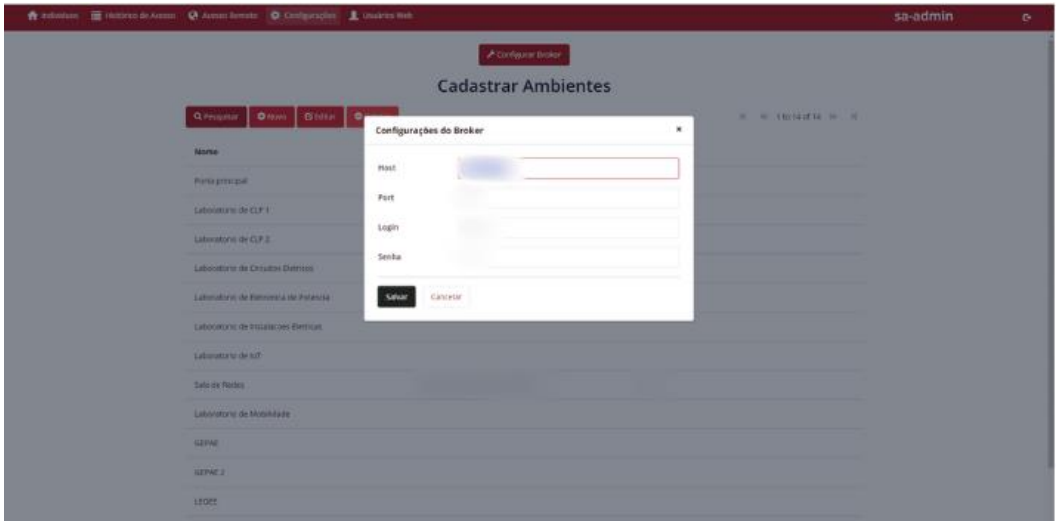
A opção de configurações é uma página capaz de definir os parâmetros do *broker* e cadastrar os ambientes, conforme demonstrado na Figura 36 e na Figura 37. Em ambos os casos, janelas *pop-ups* são exibidas para salvar os parâmetros de configuração.

Figura 36 – Página de configurações



Fonte: Imagem do autor.

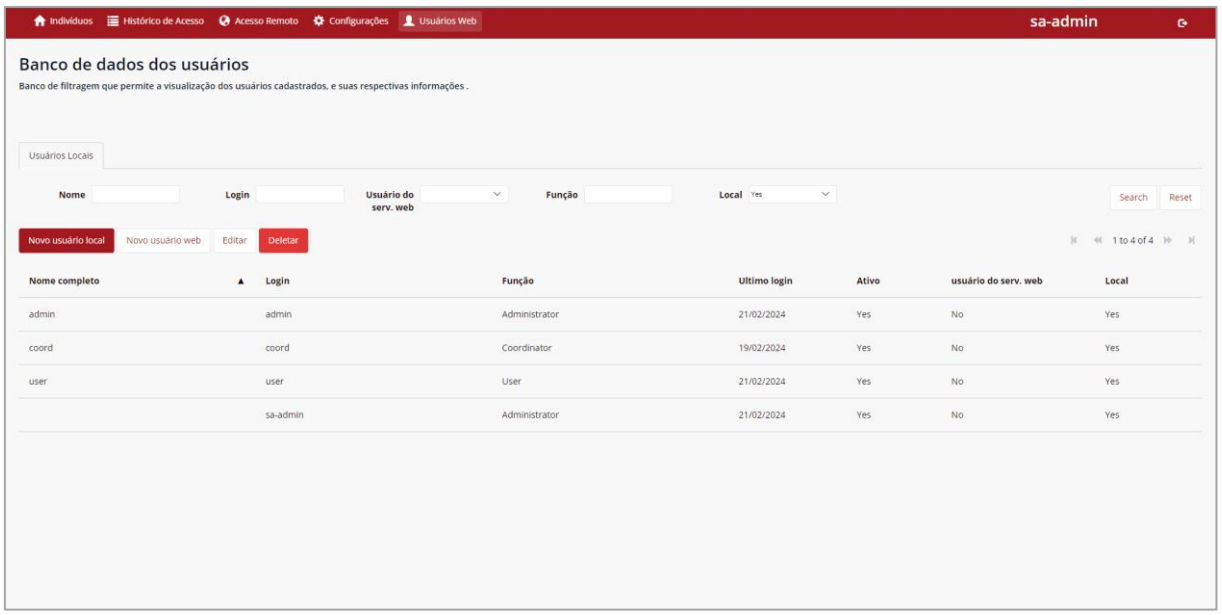
Figura 37 – Pop-up de configurações do *broker*



Fonte: Imagem do autor.

Por fim, na página de usuários web, ilustrada na Figura 38, tem a finalidade de gerenciar as configurações dos usuários web. Estes são os logins e senhas compartilhados para autenticar na aplicação web e na aplicação nativa.

Figura 38 – Página de usuários web



Fonte: Imagem do autor.

4.2 Status de Acesso

Nesta seção, são apresentadas situações de uso da aplicação e as correspondentes respostas da plataforma. São elas:

- Acesso permitido: o usuário tem acesso ao ambiente;
- Acesso negado: o usuário não tem acesso ao ambiente.

Ter acesso ao ambiente está ligado diretamente com a visibilidade dos botões de acesso na página inicial do aplicativo (Figura 30). Ao garantir o acesso àquele ambiente, seu botão estará visível e interagível. Quando o acesso é negado, seu botão estará invisível. A página inicial tem uma taxa de atualização de um minuto e, se o usuário perder o acesso, o botão será removido no próximo ciclo de atualização.

4.2.1 Acesso permitido

Nessa situação, o usuário possui acesso ao ambiente. Ainda no exemplo da Figura 30, apenas o laboratório de circuitos está acessível. Os casos caracterizados por um acesso permitido são:

- Usuários validados de acesso ilimitado: são visitantes, coordenadores ou administradores que possuem acesso ao ambiente de forma ilimitada e, enquanto mantiverem esse tipo de acesso, o botão permanece visível;
- Usuários validados de acesso limitado: são visitantes cuja forma de acesso é limitada. Neste caso, o botão é visível apenas durante um certo período, por exemplo: entre o dia 01/01/2024 às 13:00 e o dia 05/01/2024 às 14:00.

4.2.2 Acesso negado

Nessa situação, o usuário não possui acesso ao ambiente. Ainda no exemplo da Figura 30, a sala de redes não está acessível, visto que não possui botão de acesso. Os casos caracterizados por um acesso negado são:

- Usuários não validados: visitantes e coordenadores que não foram validados no ambiente;
- Usuários validados de acesso limitado: são visitantes cuja forma de acesso é limitada. Neste caso, ao estar fora do período cadastrado no sistema, seu acesso é negado.

5 CONCLUSÃO

O sistema desenvolvido contempla os conceitos de controle de acesso apresentados por Souza [2] e Halasz [1] e atingiu todos os objetivos propostos no escopo inicial, atendendo aos seguintes aspectos:

- Segurança: acessível apenas através do dispositivo cadastrado no sistema, sem possibilidade de vazamento de senhas – visto que seus usuários são compartilhados;
- Simplicidade e intuitividade: é limpo e só faz o necessário;
- Fácil manutenção: o sistema foi feito numa plataforma *low-code*;
- Baixo custo operacional: para distribuição, é necessário apenas pagar as licenças de desenvolvedor para publicar nas lojas de aplicativos móveis (custo único).

Um grande desafio que foi superado no desenvolvimento da aplicação foi a restrição de usuários do plano *free* do Mendix. Para identificar cada dispositivo (vinculado a seus usuários), o *UniqueID* serviu muito bem. As associações criadas para coordenador e visitante serviram como substitutas para os cargos de usuários e desempenham um papel fundamental para o bom funcionamento do sistema desenvolvido, tornando o modelo de entidades e relacionamento proposto uma estrutura coesa e eficaz.

O *design* das telas de uso evidencia a preocupação com UI/UX, tendo em vista a sua simplicidade e intuitividade. A paleta de cores escolhida segue o padrão do DEE.

A aplicação está disponível para usuários de dispositivos Android e está na fase de coleta de feedbacks, que fazem um papel indispensável para correção de bugs, usabilidade, experiência de usuário, melhorias de funcionalidades, desempenho e integração com outros sistemas.

As limitações impostas pelo plano *Mendix Free* geraram dificuldades para a implementação do sistema. Portanto, as propostas de continuidade do trabalho incluem principalmente maneiras de refatorar o sistema de controle de acesso para criação de usuários pessoais.

Além disso, o *broker* aceita criptografia TLS mas ainda não foi implementada. Uma ação futura seria a utilização de certificados para aumentar a segurança na comunicação entre o *broker* e o *hardware*.

Por fim, os entraves postos pelo sistema operacional iOS não dão alternativas para instalações não seguras de aplicativos, tornando necessário adquirir uma licença de desenvolvedor.

REFERÊNCIAS

1. HALASZ, R. S. **SISTEMAS INTEGRADOS DE SEGURANÇA**. [S.l.]: Sicurezza, 2005.
2. SOUZA, M. B. D. **Controle de Acesso:** CONCEITOS, TECNOLOGIAS E BENEFÍCIOS. [S.l.]: SICUREZZA, 2010.
3. PROCENGE. SISTEMAS LOW CODE: FIQUE POR DENTRO DOS PRINCIPAIS IMPACTOS PARA SEU NEGÓCIO. **PROCENGE**, 2024. Disponivel em: <https://procenge.com.br/blog/sistemas-low-code-fique-por-dentro-dos-principais-impactos-para-seu-negocio/>. Acesso em: 04 maio 2024.
4. KOVACS, L. O que são plataformas low code? **Tecnoblog**, 2022. Disponivel em: <https://tecnoblog.net/responde/o-que-sao-plataformas-low-code/>.
5. FADDEN, M. M. Low-code é solução para tudo? **tiinside**, 2021. Disponivel em: <https://tiinside.com.br/15/10/2021/low-code-e-solucao-para-tudo/>. Acesso em: 04 maio 2024.
6. FONSECA, M. IEC 61131-3: a norma para programação. **PLCopen**, 2004. Disponivel em: <https://plcopen.org/sites/default/files/downloads/intro_iec_march04_portuguese.pdf>. Acesso em: 04 maio 2024.
7. AWARI. Aprenda A Desenvolver Aplicações Front-End Com Node-Red. **Awari**, 2023. Disponivel em: <https://awari.com.br/aprenda-a-desenvolver-aplicacoes-front-end-com-node-red/>. Acesso em: 04 maio 2024.
8. NODE-RED. **Node-red**. Disponivel em: <https://nodered.org/>. Acesso em: 27 set. 2023.
9. SANTOS, H. J. D. S. et al. INTERNET OF THINGS (IOT) FOR HOME MONITORING AND AUTOMATION. **Revistaft**, v. 27, Maio 2023. ISSN 10.5281/zenodo.7977340.
10. REIS, E. D. C. L. AUTOMAÇÃO RESIDENCIAL COM A UTILIZAÇÃO DE RASPBERRY PI. **PUC Goiás**, Goiás, 2021.
11. FERENCZ, K.; DOMOKOS, J. Using Node-RED platform in an industrial environment. **XXXV. Jubileumi Kandó Konferencia**, Budapest, 2019. 52-63.
12. ANKIX. Mendix - Desenvolvimento Low Code. **ankix**, 2024. Disponivel em: <https://www.ankix.com/desenvolvimento-mendix>. Acesso em: 04 maio 2024.
13. TRUECHANGE. truechange. **Mendix low-code na otimização e desenvolvimento de negócios**, 2024. Disponivel em: <https://truechange.com.br/blog/mendix-low-code-na-otimizacao-de-negocios/>. Acesso em: 04 maio 2024.
14. APVINE. Customer Case AGII | CIT - A Corona App. **apvine**, 2020. Disponivel em: <https://apvine.com/cases/cit-corona-app>. Acesso em: 22 fev. 2024.
15. TANENBAUM, A. S.; BOS, H. **Sistemas Operacionais Modernos**. 4. ed. [S.l.]: PEARSON, 2015.
16. HILLAR, G. C. **MQTT Essentials - A Lightweight IoT Protocol**. [S.l.]: Packt, 2017.
17. MQTT. **mqtt.org**. Disponivel em: <https://mqtt.org/>. Acesso em: 27 set. 2023.

18. ELMASRI, R.; NAVATHE, S. B. **Fundamentals of Database Systems**. 7. ed. [S.l.]: Pearson, 2015.

19. HSQLDB. **HyperSQL**, 2024. Disponivel em: <https://hsqldb.org/>. Acesso em: 04 fev. 2024.

20. SRIRAMYA, P.; KARTHIKA, R. A. PROVIDING PASSWORD SECURITY BY SALTED PASSWORD HASHING. **ARPN journal of engineering and applied sciences**, jul. 2015. 5551-5556.

21. BUCHANAN, W. J. BCrypt. **asecuritysite**, 2024. Disponivel em: <https://asecuritysite.com/hash/bcrypt>. Acesso em: 21 fev. 2024.

22. THOMAS, S. **SSL and TLS Essentials**. [S.l.]: WILEY, 2000.

23. MENDIX - Start for Free. **Mendix - Start for Free**, 2023. Disponivel em: <https://www.mendix.com/pricing/start-for-free/>. Acesso em: 27 set. 2023.

24. ORACLE, 2023. Disponivel em: <https://www.oracle.com/internet-of-things/what-is-iot/>. Acesso em: 27 set. 2023.

25. GUPTA, R. K. et al. IoT Based Door Entry System. **Indian Journal of Science and Technology**, out. 2016.

26. SIEMENS to acquire Mendix. **Siemens to acquire Mendix**. Disponivel em: <https://www.mendix.com/blog/siemens-to-acquire-mendix/>. Acesso em: 04 fev. 2024.

27. THIS is Mindsphere. **This is Mindsphere**. Disponivel em: <https://www.youtube.com/watch?v=y_kN1SExehI>. Acesso em: 04 fev. 2024.

28. MENDIX Academy - Become a Rapid Developer. **Mendix Academy - Become a Rapid Developer**. Disponivel em: <https://academy.mendix.com/link/paths/31/Become-a-Rapid-Developer>. Acesso em: 17 set. 2023.

29. WAL, E. V. D. PLCopen. **Structuring with IEC 61131-3:** 7 steps to success, 2021. Disponivel em: <https://www.controleng.com/articles/structuring-with-iec-61131-3-7-steps-to-success/>. Acesso em: 04 maio 2024.