INSTITUTO FEDERAL DE EDUCAÇÃO DA PARAÍBA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

LOHAN YRVINE OLIVEIRA PINHEIRO

KEYCODE MANAGER

Um sistema Web de gerenciamento de chaves através de códigos QR

CAMPINA GRANDE
2025

LOHAN YRVINE OLIVEIRA PINHEIRO

KEYCODE MANAGER:
Um sistema Web de gerenciamento de chaves através de códigos QR

Monografia apresentada ao Instituto Federal de Educação da Paraíba como requisito para obtenção de título Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. David Candeia Medeiros Maia
Coorientador: Prof. Dr. Katyusco de Farias Santos

CAMPINA GRANDE
2025

Catalogação na fonte:
Ficha catalográfica elaborada por Gustavo César Nogueira da Costa - CRB 15/479

P654k Pinheiro, Lohan Yrvine Oliveira
KeyCode Manager: um sistema Web de gerenciamento de chaves através de códigos QR / Lohan Yrvine Oliveira Pinheiro. - Campina Grande, 2025.
77 f. : il.

Trabalho de Conclusão de Curso (Curso Superior de Bacharelado em Engenharia da Computação) - Instituto Federal da Paraíba, 2025.
Orientador: Prof. Dr. David Candeia Medeiros Maia
Coorientador: Prof. Dr. Katyusco de Farias Santos

1. Engenharia de computação 2. Desenvolvimento de sistemas - Sistemas Web 3. Códigos QR - gerenciamento. 4. Segurança da informação. I. Pereira, Fagner de Araújo III.Título.

CDU 004.738.5:005.94

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, Rosangela Pereira de Oliveira e Linduardo Ferreira Pinheiro, por todo o apoio incondicional, incentivo e sacrifícios realizados ao longo desta jornada acadêmica. Aos meus orientadores, Dr. David Candeia Medeiros Maia e Dr. Katyusco de Farias Santos, pela paciência, orientação cuidadosa e confiança depositada neste trabalho. Suas contribuições foram fundamentais para o desenvolvimento e a conclusão deste projeto.

Ao Instituto Federal de Educação da Paraíba pela formação acadêmica de qualidade e pelas oportunidades oferecidas. Agradeço especialmente aos funcionários do setor de chaves, que generosamente compartilharam suas experiências com o processo de gerenciamento atual, contribuindo diretamente para tornar esta solução aplicável à realidade institucional.

À Bianca Nathalya e ao Samuel Lucas, amigos que tornaram esta jornada mais leve e significativa através de conversas, apoio mútuo e companheirismo, seja à distância ou compartilhando o dia a dia. Aos demais amigos que estiveram presentes, compartilhando conhecimentos, desafios e conquistas.

“Caminhante, não há caminho, faz-se caminho ao andar.”

(Antonio Machado)

RESUMO

O gerenciamento manual de chaves físicas em instituições educacionais apresenta desafios significativos relacionados à eficiência operacional e à segurança, incluindo processos baseados em registros em papel suscetíveis a erros, ausência de controle automatizado e dificuldades na rastreabilidade de responsáveis. Este trabalho apresenta o desenvolvimento do *KeyCode Manager*, uma aplicação Web para gerenciamento de chaves físicas utilizando códigos QR como mecanismo de identificação e autenticação. O sistema foi desenvolvido com arquitetura hexagonal utilizando tecnologias *open-source* e gratuitas (Go, HTMX e Alpine.js), oferecendo interfaces para autoatendimento e administração. A arquitetura implementa múltiplas camadas de segurança, incluindo autenticação, proteção contra ataques comuns em aplicações Web e trilha de auditoria imutável. A solução foi validada através de suíte automatizada de 424 casos de teste com cobertura média superior a 80% nas camadas de lógica de negócio e foi apresentada no 6º SIMPIF, conquistando segundo lugar na modalidade Amostra de Projetos. Os resultados demonstram viabilidade técnica e econômica da solução para instituições educacionais, eliminando dependência de processos manuais e melhorando a prestação de contas através de rastreabilidade completa.

Palavras-chave: Gerenciamento de chaves; Códigos QR; Sistemas Web; Controle de acesso físico; Segurança da informação.

ABSTRACT

Manual physical key management in educational institutions presents significant challenges related to operational efficiency and security, including paper-based processes susceptible to errors, absence of automated control, and difficulties in tracking accountability. This work presents the development of *KeyCode Manager*, a Web application for physical key management using QR codes as identification and authentication mechanism. The system was developed with hexagonal architecture using open-source and free technologies (Go, HTMX, and Alpine.js), providing interfaces for self-service and administration. The architecture implements multiple security layers, including authentication, protection against common Web application attacks, and immutable audit trail. The solution was validated through automated test suite with 424 test cases with average coverage exceeding 80% in business logic layers and was presented at the 6th SIMPIF, achieving second place in the Project Showcase category. Results demonstrate technical and economic viability of the solution for educational institutions, eliminating dependency on manual processes and improving accountability through complete traceability.

Keywords: Key management; QR codes; Web systems; Physical access control; Information security.

LISTA DE ILUSTRAÇÕES

FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
API	*Application Programming Interface*
CORS	*Cross-Origin Resource Sharing*
CPF	Cadastro de Pessoas Físicas
CRUD	*Create, Read, Update, Delete*
CSRF	*Cross-Site Request Forgery*
FI	*Forward Integrity*
FSL	*Facility Security Levels*
GPIO	*General Purpose Input/Output*
GSI/PR	Gabinete de Segurança Institucional da Presidência da República
HDA	*Hypermedia-Driven Application*
HMAC	*Hash-based Message Authentication Code*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
IFPB	Instituto Federal de Educação da Paraíba
IoT	*Internet of Things*
ISO	*International Organization for Standardization*
JSON	*JavaScript Object Notation*
LAC	*Logical Access Control*
LGPD	Lei Geral de Proteção de Dados
MAC	*Message Authentication Code*
MMI	*Man-Machine Interface*
MPA	*Multi-Page Application*
NBR	Norma Brasileira
NIST	*National Institute of Standards and Technology*
ORM	*Object-Relational Mapping*
PAC	*Physical Access Control*
PACS	*Physical Access Control System*
PDF	*Portable Document Format*
PIN	*Personal Identification Number*

PIV	*Personal Identity Verification*
PME	Pequenas e Médias Empresas
QR Code	*Quick Response Code*
RBAC	*Role-Based Access Control*
SIMPIF	Simpósio de Pesquisa, Inovação e Pós-Graduação do IFPB
SPA	*Single Page Application*
SQL	*Structured Query Language*
SSE	*Server-Sent Events*
SUAP	Sistema Unificado de Administração Pública
XSS	*Cross-Site Scripting*

SUMÁRIO

1 INTRODUÇÃO

Com o avanço das tecnologias e a crescente urbanização, cidades ao redor do mundo enfrentam desafios sem precedentes, como o aumento do consumo de energia, congestionamento de tráfego, preocupações com segurança e privacidade e dificuldades gerais na provisão eficiente de serviços a uma população em crescimento (Zaman *et al*., 2024). Para enfrentar esses desafios, tem-se discutido intensamente ao longo dos últimos anos a criação e o desenvolvimento de ambientes inteligentes baseados na Internet das Coisas (do inglês, *Internet of Things* — IoT), tais como cidades inteligentes (do inglês, *Smart Cities*), residências inteligentes (do inglês, *Smart Homes*) e campi inteligentes (do inglês, *Smart Campus*) (Zaman *et al*., 2024).

O gerenciamento manual de chaves apresenta diversos desafios que comprometem a eficiência operacional e a segurança institucional. Os principais problemas incluem a perda ou roubo de chaves, ausência de controle sobre quem possui acesso, uso não autorizado e perda de tempo em processos administrativos (GENESIS RESOURCE, [s.d.]). Esses riscos geram custos desnecessários e vulnerabilidades que poderiam ser evitados com a implementação de sistemas automatizados.

Esses ambientes inteligentes buscam integrar tecnologias de informação e comunicação, especialmente IoT, para coletar e analisar dados em tempo real, otimizar a alocação de recursos, melhorar a tomada de decisões e aumentar a qualidade de vida dos residentes. Um componente fundamental desses ambientes é o controle de acesso físico automatizado, que permite gerenciar quem pode acessar determinados espaços e quando, garantindo segurança sem comprometer a eficiência operacional. Nesse contexto de automação de controle de acesso, foi desenvolvido anteriormente por estudantes do campus um protótipo de sistema para gerenciamento de chaves físicas.[1]

Este trabalho propôs a reimplementação completa desse sistema, originalmente denominado *KeyCode Manager*[2], utilizando tecnologias atuais como Go, HTMX e Alpine.js (descritas em detalhes na Seção 4.3) e incorporando funcionalidades ausentes na versão original, como geração de relatórios e controle de acesso baseado em perfis. A nova implementação mantém o objetivo de otimizar o controle e rastreamento de chaves físicas por meio de automação e identificação via códigos QR, oferecendo uma plataforma de fácil manutenção, utilizando

[1]Código-fonte do protótipo original disponível em: https://github.com/edoardap/KeyCodeManager. Acesso em: 16 dez. 2025

[2]Código-fonte disponível em: https://codeberg.org/leakedmemory/keycode-manager. Acesso em: 15 dez. 2025

exclusivamente tecnologias *open-source* e arquitetura modular, tornando-a economicamente viável para organizações de pequeno porte.

1.1 Problematização

No que diz respeito à gestão de chaves físicas, organizações enfrentam desafios recorrentes para garantir controle eficaz e segurança no acesso a espaços restritos. Muitos desses ambientes ainda dependem de métodos manuais, como planilhas ou registros em papel, o que dificulta a rastreabilidade e aumenta a suscetibilidade a falhas humanas. Processos manuais apresentam taxa de erro significativa, com pessoas cometendo pelo menos 10 erros a cada 100 etapas em processos manuais, mesmo em trabalhos repetitivos, e probabilidade de erro entre 18% e 40% ao inserir informações em planilhas simples (UNQORK, 2021).

O Instituto Federal de Educação da Paraíba (IFPB) — campus Campina Grande, especificamente — passa por este problema corriqueiramente seguindo o seguinte fluxo nas retiradas e devoluções de chaves:

1. A chave é solicitada para um funcionário do setor de chaves;
2. O funcionário, então, pega a chave e indica ao solicitante qual livro, dentre os diferentes existentes, deve ser assinado;
3. A assinatura é feita e consiste em: nome do solicitante, a chave que está sendo devolvida/retirada, data e horário da operação.

Além dos problemas mencionados anteriormente no controle manual de espaços, dentro deste contexto específico do campus, outros também se fazem presentes, como: eventuais esquecimentos na devolução de chaves após término de aula, monitoria ou limpeza, necessitando uma custosa busca pelo atual responsável nas assinaturas nos livros; gasto de tempo para assinar os vários campos solicitados e falta de controle de permissões de acesso a determinadas chaves.

Uma implementação inicial de solução foi feita previamente por estudantes do campus no âmbito de disciplinas regulares, porém este protótipo original, embora funcional para fins de prova de conceito, apresentava limitações significativas em sua arquitetura, prejudicando tanto a escalabilidade quanto a manutenção a longo prazo. Além da ausência de recursos como controle de sessões e gerenciamento de permissões, o protótipo apresentava vulnerabilidades de segurança, incluindo armazenamento de senhas em texto plano e ausência de validação criptográfica nos códigos QR. Estas limitações, combinadas com a falta de testes automatizados e de um sólido modelo arquitetônico, evidenciaram a necessidade de uma reescrita integral da aplicação, baseada em uma arquitetura mais robusta.

1.2 Objetivos

1.2.1 Objetivo geral

Desenvolver um sistema Web para gerenciamento de chaves físicas com uso de códigos QR, oferecendo proteção aos dados do usuário, buscando utilizar materiais de baixo custo e facilitando a rastreabilidade da chave. A proposta prioriza também a simplicidade operacional e a solidez arquitetural, aspectos essenciais para a adoção do sistema por organizações empresariais.

1.2.2 Objetivos específicos

Dentre os objetivos específicos, destacam-se:

- Implementar identificação de chaves físicas por meio de códigos QR únicos;
- Desenvolver uma interface Web para cadastro, consulta e gerenciamento de chaves;
- Criar um sistema abrangente de *logging* para auditoria das operações realizadas;
- Estabelecer controle de acesso baseado em perfis de usuários com diferentes níveis de permissão;
- Construir um módulo de relatórios para análise de uso e identificação de padrões;
- Garantir a segurança dos dados do usuário por meio de criptografia e boas práticas de desenvolvimento;
- Utilizar uma arquitetura extensível e de fácil manutenção, preparada para evolução contínua.

1.3 Justificativa e Contribuições

A relevância deste trabalho reside na crescente demanda por soluções digitais que otimizem processos organizacionais com eficiência e segurança. A reescrita do sistema original representa um avanço necessário, não apenas pela introdução de funcionalidades ausentes na primeira versão, mas também pela adoção de uma nova base arquitetural. O uso de códigos QR, por ser uma tecnologia acessível e consolidada, torna o sistema viável para diferentes perfis de organização.

O projeto visa auxiliar na solução do problema logístico do IFPB relacionado ao gerenciamento manual de chaves, que atualmente demanda controle presencial, registros em papel e verificações constantes. O sistema proposto automatiza esse processo, reduzindo o tempo gasto em tarefas administrativas, minimizando erros humanos e proporcionando maior controle sobre o acesso aos ambientes institucionais. O sistema desenvolvido demonstra viabilidade prática e

pode melhorar significativamente a dinâmica de gestão das chaves do campus, oferecendo uma solução aplicável ao contexto institucional do IFPB.

Do ponto de vista acadêmico, este trabalho contribui para a área de computação ao demonstrar a aplicação prática de tecnologias Web atuais na solução de problemas reais de gestão organizacional. A documentação detalhada do processo de desenvolvimento, incluindo decisões arquiteturais e implementação de funcionalidades de segurança, serve como referência para projetos similares que busquem criar soluções de gerenciamento de ativos físicos. A integração das tecnologias utilizadas demonstra como construir sistemas robustos e seguros mantendo o foco no baixo custo e na simplicidade operacional, fornecendo *insights* valiosos sobre o desenvolvimento de aplicações para organizações de pequeno porte.

1.4 Organização do trabalho

Este trabalho está organizado da seguinte forma:

- o Capítulo 2 apresenta a fundamentação teórica, abordando os conceitos essenciais sobre gerenciamento de chaves físicas, códigos QR, sistemas Web atuais e segurança em aplicações Web;
- o Capítulo 3 apresenta os trabalhos relacionados, revisando a literatura sobre sistemas de controle de acesso físico, segurança de códigos QR, trilhas de auditoria e gestão de segurança da informação;
- o Capítulo 4 descreve a metodologia empregada no desenvolvimento do sistema, incluindo classificação da pesquisa, tecnologias utilizadas, arquitetura do sistema e estratégias de segurança;
- o Capítulo 5 apresenta os resultados obtidos;
- o Capítulo 6 traz as conclusões e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os conceitos necessários para o entendimento do presente trabalho. Dessa maneira, na Seção 2.1 são abordados os fundamentos de gerenciamento de chaves físicas; na Seção 2.2, é apresentada a definição de códigos QR e suas aplicações; na Seção 2.3 conceitua-se sistemas Web atuais e arquiteturas responsivas; na Seção 2.4 são apresentados os aspectos fundamentais de segurança em sistemas Web, essenciais para sistemas de controle de acesso; por fim, na Seção 2.5 é introduzida a arquitetura hexagonal, padrão arquitetural adotado no desenvolvimento do sistema.

2.1 Gerenciamento de chaves físicas

O sistema de controle de acesso físico (PACS - do inglês, *Physical Access Control System*) é definido como um sistema que controla o acesso físico de atores físicos a recursos ou espaços físicos, sendo responsável por proteger pessoas e recursos físicos contra acessos não autorizados e suas consequências desastrosas (MASOUMZADEH; LAAN; DERCKSEN, 2022). Em essência, trata-se de gerenciar quem pode acessar um recurso ou localização física e quando. Dessa forma, um sistema eficiente de gerenciamento de chaves deve proporcionar rastreabilidade, segurança e facilidade de uso para os administradores.

O gerenciamento de chaves físicas é um processo importante para organizações. Empresas menores frequentemente empregam protocolos manuais com caneta e papel, enquanto empresas maiores preferem sistemas eletrônicos de gerenciamento de chaves (REAL TIME NETWORKS, 2024). Tradicionalmente, esse processo manual envolve registro, distribuição e rastreamento propensos a erros humanos. O erro humano aumenta drasticamente o risco de perder ou extraviar chaves quando o gerenciamento é deixado inteiramente para a equipe em vez de utilizar um sistema automatizado de monitoramento (REAL TIME NETWORKS, 2024).

Existem diversas abordagens para implementação de sistemas de controle de acesso físico, variando desde soluções puramente eletrônicas (com fechaduras conectadas e controle centralizado) até sistemas híbridos que combinam fechaduras mecânicas tradicionais com rastreamento automatizado. Uma discussão mais aprofundada sobre diferentes tipos de sistemas PAC, incluindo exemplos e arquiteturas específicas, é apresentada na Seção 3.1, onde são analisadas soluções como armários eletrônicos de chaves, sistemas *Network on Card* e abordagens baseadas em identificação por código.

Nesse contexto, o *KeyCode Manager* representa uma solução digital para automatizar o processo de controle de chaves físicas, utilizando tecnologias Web atuais e códigos QR para simplificar a identificação e rastreamento de chaves em organizações.

2.2 Códigos QR e aplicações

O *QR Code* (do inglês, *Quick Response Code*) é um tipo de código de barras bidimensional que pode armazenar informações de forma compacta e ser lido rapidamente por dispositivos móveis. Desenvolvido em 1994 em resposta a demandas por códigos de barras com maior capacidade de armazenamento e leitura rápida, os códigos QR expandiram suas aplicações para diversos setores devido à sua capacidade de armazenar por volta de 7.000 caracteres numéricos (DENSO WAVE, [s.d.]b).

Dentre as principais vantagens dos códigos QR, inclui-se alta capacidade de correção de erros (DENSO WAVE, [s.d.]a) e leitura rápida de qualquer ângulo (DENSO WAVE, [s.d.]b). Essas características tornam os códigos QR ideais para sistemas de identificação, onde a precisão e velocidade são fundamentais.

Para o gerenciamento de chaves físicas, os códigos QR permitem sua identificação, facilitando processos de controle e monitoramento.

2.3 Sistemas Web atuais

Os sistemas Web atuais enfrentam expectativas de usuário mais altas e maiores demandas do que nunca. Segundo a Microsoft Learn (2022), as aplicações Web atuais devem estar disponíveis continuamente, acessíveis de qualquer lugar do mundo e utilizáveis em praticamente qualquer dispositivo ou tamanho de tela. Para atender cenários cada vez mais complexos, essas aplicações dependem de experiências de usuário ricas construídas no cliente através de JavaScript e comunicação eficiente por meio de APIs Web.

As aplicações Web contemporâneas evoluíram para incluir diferentes abordagens arquiteturais, desde *Multi-Page Applications* (MPAs) tradicionais até *Single Page Applications* (SPAs) e *Hypermedia-Driven Applications* (HDAs). Esta última sendo uma abordagem que combina a simplicidade e flexibilidade das MPAs com a melhor experiência do usuário das SPAs (GROSS, 2022).

Uma HDA utiliza sintaxe declarativa incorporada ao HTML, em vez de scripts imperativos, para alcançar melhor interatividade no *front-end*. Além disso, uma HDA interage com o

servidor em termos de hipermídia, como HTML, em vez de formatos não-hipermídia, como JSON (GROSS, 2022).

Tecnologias como HTMX exemplificam essa evolução, permitindo que desenvolvedores criem interfaces dinâmicas sem abandonar os princípios fundamentais da Web. Ao utilizar HTML como linguagem principal e manter a comunicação via HTTP, essas ferramentas oferecem uma alternativa aos *frameworks* JavaScript complexos, proporcionando experiências de usuário aprimoradas através de uma abordagem mais simples e diretamente integrada aos padrões Web estabelecidos.

2.4 Segurança em sistemas Web

Além das tecnologias utilizadas na construção de sistemas Web, é essencial considerar os aspectos de segurança envolvidos, especialmente em aplicações sensíveis como sistemas de controle de acesso.

A segurança em sistemas Web baseia-se no princípio de defesa em profundidade, que envolve múltiplas camadas de controles de segurança para proteger os ativos de uma organização. Essa estratégia fundamenta-se na ideia de que, se uma camada de segurança falhar, as outras camadas ainda serão capazes de proteger o ativo (OWASP, [s.d.]). Para sistemas de gerenciamento como o *KeyCode Manager*, a segurança é crítica devido à natureza sensível das informações de controle de acesso, sendo exemplos o CPF e número de telefone. Uma discussão sobre níveis de segurança e mecanismos de autenticação em sistemas de controle de acesso físico é apresentada na Seção 3.1.2.

A implementação de criptografia adequada é essencial para proteger dados sensíveis tanto em armazenamento quanto em trânsito. Dados sensíveis, como informações pessoais e financeiras, devem ser protegidos através de criptografia em trânsito e em repouso, utilizando funções e protocolos criptográficos como HTTPS (OWASP, [s.d.]).

Para organizações de pequeno e médio porte, que enfrentam pressão crescente para proteger seus dados e operações enquanto carecem de equipes de segurança dedicadas e recursos financeiros de empresas maiores, é essencial implementar medidas de segurança práticas e econômicas que não exijam conhecimento técnico extenso (HIETALA, 2005).

2.5 Arquitetura hexagonal

A arquitetura hexagonal, também conhecida como padrão *Ports and Adapters*, foi proposta por Alistair Cockburn (2005) como solução para desafios comuns em desenvolvimento de software

relacionados ao acoplamento entre lógica de negócio e tecnologias de infraestrutura. O padrão fundamenta-se na ideia de que aplicações devem ser isoladas de detalhes de implementação de interfaces de usuário, bancos de dados e outras tecnologias externas, permitindo que a lógica central do sistema permaneça independente e testável.

Cockburn (2005) define o objetivo da arquitetura como permitir que uma aplicação seja igualmente conduzida por usuários, programas, testes automatizados ou *scripts* em lote, e seja desenvolvida e testada em isolamento de seus dispositivos de execução e bancos de dados eventuais. A arquitetura divide o sistema em três componentes principais: o núcleo hexagonal (contendo a lógica de negócio), portas (interfaces que definem como o mundo externo pode interagir com a aplicação) e adaptadores (implementações concretas que conectam tecnologias específicas às portas).

Cockburn (2005) distingue entre adaptadores primários (ou condutores) e adaptadores secundários (ou conduzidos), classificação fundamental para compreensão do padrão. Adaptadores primários iniciam interações com a aplicação, traduzindo requisições externas (como requisições HTTP ou eventos de interface gráfica) em chamadas ao núcleo de negócio através de portas de entrada. Adaptadores secundários são acionados pela aplicação para realizar operações de infraestrutura, como persistência de dados ou envio de notificações, implementando portas de saída definidas pela lógica de negócio.

Cockburn (2005) argumenta que este padrão arquitetural oferece benefícios significativos: independência de *frameworks* e tecnologias de infraestrutura, facilitando substituição de componentes externos sem modificação da lógica de negócio; testabilidade aprimorada, permitindo testes unitários e de integração sem dependências de banco de dados ou interfaces de usuário; e maior manutenibilidade através de separação clara de responsabilidades. Esta arquitetura é particularmente adequada para sistemas que precisam integrar múltiplas interfaces ou tecnologias de infraestrutura, permitindo que a lógica de negócio permaneça estável enquanto adaptadores são modificados ou substituídos conforme necessidades operacionais.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta um levantamento bibliográfico relacionada aos principais conceitos e tecnologias que fundamentam o *KeyCode Manager*. A revisão está organizada em seções que abordam sistemas de controle de acesso físico, segurança de códigos QR, trilhas de auditoria, controle de acesso em instituições públicas, e gestão de segurança da informação.

É importante destacar que, “apesar do tamanho da indústria de PAC e seu papel essencial na proteção de nossos ambientes físicos, a pesquisa e desenvolvimento público em relação a PAC tem sido limitado” (MASOUMZADEH; LAAN; DERCKSEN, 2022). Esta escassez de pesquisa contrasta significativamente com o tamanho do mercado global de produtos de controle de acesso físico, estimado em US$ 8,8 bilhões em 2021, com projeção de US$ 12 bilhões até 2026 (MASOUMZADEH; LAAN; DERCKSEN, 2022).

3.1 Sistemas de Controle de Acesso Físico (PAC)

Os sistemas de controle de acesso físico constituem um dos pilares centrais da segurança de qualquer organização. PAC é definido como o controle do acesso físico de atores físicos a recursos ou espaços físicos, distinguindo-o do controle de acesso lógico (do inglês, *Logical Access Control* - LAC), que controla o acesso de usuários de computador a recursos computacionais (MASOUMZADEH; LAAN; DERCKSEN, 2022). Em outras palavras, PAC é responsável por proteger a superfície de ataque física, enquanto LAC protege a superfície de ataque cibernética.

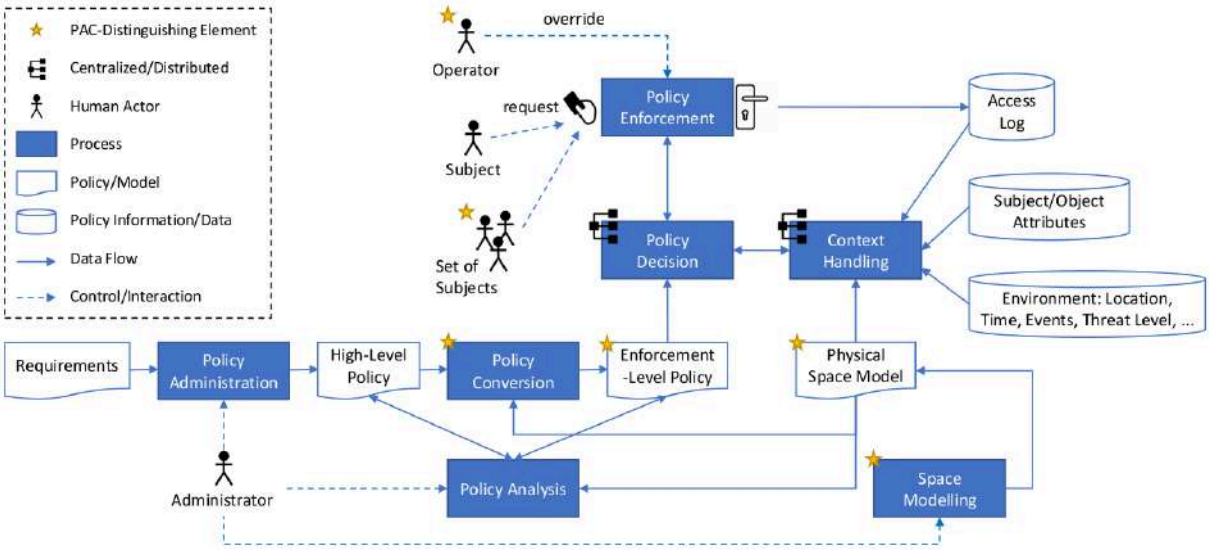
Embora modelos de política de controle de acesso originados no domínio LAC, como controle de acesso baseado em papéis (do inglês, *Role Based Access Control* - RBAC), sejam úteis também em sistemas PAC, a natureza física dos sistemas PAC introduz diversos desafios e oportunidades de pesquisa específicos.

Masoumzadeh, Laan e Dercksen (2022) destacam características que diferenciam PAC de LAC: a necessidade de modelar explicitamente espaços físicos e seus relacionamentos; o uso de barreiras físicas como pontos de aplicação de política, o envolvimento de atores humanos (tanto como sujeitos solicitando acesso físico quanto como pessoal de segurança); o tratamento de múltiplos sujeitos em uma única requisição de acesso; e a possibilidade de sobrescrita de políticas por pessoal de segurança física.

3.1.1 Arquitetura de sistemas PAC

Masoumzadeh, Laan e Dercksen (2022) propõem uma arquitetura abrangente para gerenciamento e aplicação de controle de acesso em ambientes PAC, conforme ilustrado na Figura 1. Nesta arquitetura, os seguintes componentes são essenciais:

Figura 1: Arquitetura de gerenciamento e aplicação de controle de acesso em ambientes PAC



Fonte: Masoumzadeh; Laan; Dercksen, 2022

1. **Modelo de Espaço Físico**: identifica recursos físicos, espaços e pontos de aplicação, modelando relacionamentos físicos como posicionamento, contenção e acessibilidade. Este modelo é referenciado por políticas e processos em todo o sistema.
2. **Conversão de Políticas**: traduz políticas de alto nível (legíveis por humanos, com escopo em todo o sistema) em políticas de nível de aplicação (escopo limitado por ponto de aplicação), considerando as capacidades e posicionamentos dos pontos de aplicação com base no modelo de espaço físico.
3. **Aplicação de Políticas**: pode envolver um sujeito individual ou um conjunto de sujeitos, com capacidade de sobrescrita por operadores de segurança para atender necessidades operacionais.
4. ***Log* de Acesso**: registra todas as decisões e ações de acesso, constituindo componente essencial para rastreabilidade e análise forense.

Esta arquitetura fornece um *framework* conceitual aplicável ao *KeyCode Manager*, especialmente nos aspectos de modelagem de espaços físicos (através do campo de localização nos

registros de chaves), *log* de acesso (trilha de auditoria imutável), e aplicação de políticas (entrega de chaves com validação de código QR).

3.1.2 Níveis de segurança e mecanismos de autenticação

A seleção apropriada de mecanismos de autenticação em sistemas PAC deve considerar uma abordagem baseada em risco. Ferraiolo *et al*. (2018) estabelecem um *framework* para controle de acesso físico baseado em credenciais PIV (do inglês, *Personal Identity Verification*), introduzindo o conceito de Níveis de Segurança de Instalações (do inglês, *Facility Security Levels* - FSL). Este *framework* categoriza instalações em cinco níveis (FSL-1 a FSL-5), onde FSL-1 representa instalações de baixo risco com consequências mínimas de falhas de autenticação, FSL-2 representa instalações de risco moderado, FSL-3 abrange instalações de risco moderado-alto com possibilidade de consequências severas ou catastróficas, e os níveis FSL-4 e FSL-5 representam instalações de alto risco que exigem garantia muito alta na identidade do portador de credenciais, tipicamente envolvendo áreas de exclusão com requisitos de autenticação de três fatores.

A determinação do FSL apropriado considera fatores como importância dos ativos protegidos, potencial de danos à segurança nacional, impacto operacional de incidentes de segurança e requisitos regulatórios aplicáveis (Ferraiolo *et al*., 2018). Para cada nível de segurança, o NIST (*National Institute of Standards and Technology*) especifica requisitos mínimos de autenticação, balanceando segurança e conveniência operacional. Por exemplo, em instalações FSL-1, um único fator de autenticação pode ser suficiente, enquanto instalações FSL-2 geralmente requerem autenticação de dois fatores, combinando “algo que você possui” (como uma credencial física) com “algo que você sabe” (como um PIN) ou “algo que você é” (biometria).

O NIST descreve seis mecanismos de autenticação para uso com credenciais PIV (Ferraiolo *et al*., 2018): PKI-CAK (autenticação criptográfica com infraestrutura de chave pública), SYM-CAK (autenticação criptográfica simétrica), BIO (biometria de um dedo), BIO-A (biometria de dois dedos para maior garantia), PKI-AUTH (autenticação baseada em certificado digital), e OCC-AUTH (autenticação visual do cartão). Cada mecanismo oferece diferentes níveis de garantia de autenticação, e sua seleção deve corresponder ao FSL da área sendo protegida.

No contexto de instituições educacionais brasileiras como o IFPB, o ambiente operacional tipicamente se enquadra nos níveis FSL-1 ou FSL-2. Embora áreas técnicas e administrativas que armazenam equipamentos de valor ou dados sensíveis possam justificar proteção FSL-2, a

maioria dos espaços acadêmicos se enquadra em FSL-1, onde o objetivo principal é prevenir acesso não autorizado casual e manter registros de auditoria para responsabilização. O *KeyCode Manager* foi projetado considerando este contexto: a combinação de autenticação Web (credenciais institucionais) com validação de código QR na chave física fornece dois fatores de controle (acesso ao sistema + posse da chave autorizada), apropriado para ambientes FSL-1 e FSL-2. Esta abordagem está alinhada com o princípio do NIST de que "a força da autenticação deve ser proporcional às consequências de erros de autenticação" (Ferraiolo *et al.*, 2018).

3.1.3 Sistemas PAC híbridos

Uma característica importante dos sistemas PAC atuais é a coexistência de fechaduras eletrônicas e mecânicas, configurando sistemas PAC híbridos. Estima-se que a proporção entre fechaduras mecânicas e eletrônicas em instalações PAC modernas seja de aproximadamente 4:1 (MASOUMZADEH; LAAN; DERCKSEN, 2022). O principal benefício das fechaduras mecânicas é seu custo significativamente menor de instalação, embora ofereçam menor nível de conveniência e flexibilidade. Por exemplo, um crachá perdido pode ser bloqueado instantaneamente e remotamente em um sistema eletrônico, enquanto a perda de uma chave mecânica requer a substituição física de todas as chaves e do cilindro da fechadura correspondente. A Tabela 1 sintetiza esta relação entre custo e rastreabilidade, posicionando o *KeyCode Manager* como alternativa que combina o baixo custo das fechaduras mecânicas com rastreamento de custódia.

Tabela 1: Comparativo entre tipos de controle de acesso físico

Tipo	**Custo**	**Controle**
Fechaduras Eletrônicas	Alto	Acesso rastreado
Fechaduras Mecânicas	Baixo	Sem rastreamento
KeyCode Manager	Baixo	Custódia rastreada

Fonte: Elaborado pelo autor (2025)

Para preencher a lacuna entre sistemas de controle de acesso *online* (eletrônicos) e *offline* (mecânicos), fabricantes desenvolveram soluções criativas, sendo duas abordagens principais descritas por Masoumzadeh, Laan e Dercksen (2022):

- **Armários de Chaves**: são armários controlados por acesso nos quais chaves físicas etiquetadas são armazenadas. Um sujeito pode abrir o armário e retirar a(s) chave(s) atribuída(s) a ela. A remoção e devolução das chaves é registrada no *log* de acesso, e o acesso ao armário é gerenciado a partir do sistema PAC. Configurações avançadas nas quais a devolução da chave é pré-requisito para sair de um local são usadas para aliviar o problema de chaves perdidas. Este tipo de configuração é tipicamente usado para engenheiros de serviço que precisam realizar trabalhos de manutenção em áreas técnicas de edifícios fora do horário comercial, ou seja, sem a presença de um recepcionista ou guarda.
- ***Network on Card***: envolve cartões de acesso regraváveis, usados para transportar políticas de acesso, listas de bloqueio e logs de acesso entre as fechaduras e o sistema de gerenciamento. Os sujeitos precisam atualizar seus cartões diariamente em um dispositivo de atualização que lê o *log* de acesso e estados de bateria das fechaduras que o sujeito visitou no dia anterior.

É importante destacar que, apesar da importância dos sistemas PAC híbridos, modelos formais de políticas de alto nível e de nível de aplicação capazes de suportar tais sistemas ainda não foram desenvolvidos e estudados na literatura (MASOUMZADEH; LAAN; DERCKSEN, 2022). Esta lacuna de pesquisa representa uma oportunidade significativa, pois impede o projeto e análise holística de políticas para os subsistemas mecânico e eletrônico.

O presente trabalho pode ser posicionado como uma solução de sistema PAC híbrido que aborda esta lacuna de pesquisa. O sistema mantém fechaduras mecânicas existentes, evitando custos elevados de substituição por fechaduras eletrônicas, enquanto adiciona rastreamento eletrônico automatizado através de códigos QR nas chaves físicas. Esta abordagem é conceitualmente similar aos armários de chaves descritos na literatura: o usuário autentica-se no sistema, recebe autorização de acesso, retira a chave e escaneia o código QR para registrar automaticamente a transação. A solução pode ser implantada em dois modos: (1) modo completo, com integração de um armário controlando eletronicamente a sua abertura; ou (2) modo assistido, onde um administrador gerencia fisicamente o armário, mas utiliza o sistema Web e códigos QR para eliminar o registro manual de dados.

3.1.4 Acesso Físico Transacional

O conceito de acesso físico transacional introduzido por Masoumzadeh, Laan e Dercksen (2022) destaca a distinção entre acesso concedido e acesso utilizado. Esta distinção é frequentemente

ignorada em sistemas LAC, mas é particularmente relevante em sistemas PAC para determinar se uma decisão de acesso concedido efetivamente resultou em um movimento físico.

Considerando um exemplo de catraca na entrada de um edifício: após autenticação, o acesso é concedido. Porém, somente se o sujeito efetivamente entrar na catraca e sair após uma rotação de 180 graus do mecanismo giratório, o acesso foi utilizado. As etapas desde autenticar, entrar na catraca, girar e sair podem ser vistas como acesso físico transacional. Se a rotação não for completada, a transação é abortada e não deve ser registrada como acesso completo (MASOUMZADEH; LAAN; DERCKSEN, 2022).

Em situações mais comuns, onde uma porta de velocidade, porta deslizante ou porta regular é usada com autenticação de fator único, só é possível assumir que a transação de acesso físico ocorreu a menos que haja outras evidências para corroboração, como filmagem de câmera de vídeo com reconhecimento facial (MASOUMZADEH; LAAN; DERCKSEN, 2022).

Este conceito de acesso transacional é aplicável ao presente trabalho: o sistema registra a transação em duas fases (emissão e devolução da chave), mas não garante que a chave foi efetivamente utilizada para destrancar uma porta. Assim como nos exemplos de portas com autenticação de fator único mencionados anteriormente, assume-se que o acesso físico ocorreu, mas não há evidências diretas. Uma transação incompleta (chave não devolvida) pode indicar um potencial incidente de segurança. A integração com sensores nas fechaduras para registrar o uso efetivo da chave está além do escopo deste trabalho, que foca no gerenciamento e rastreabilidade da posse das chaves.

3.1.5 Aceitabilidade psicológica e usabilidade

Sistemas PAC precisam ser projetados considerando um equilíbrio aceitável entre segurança e conveniência. Masoumzadeh, Laan e Dercksen (2022) observam que inconveniência e aceitação insuficiente do sistema inevitavelmente levarão a tentativas de burlá-lo. Exemplos comuns do sistema sendo contornado incluem: *tailgating* (múltiplos sujeitos seguindo um líder sem efetivamente solicitar acesso e, portanto, não sendo registrados), uso de saídas de emergência para atalhar rotas e colaboração com pessoas internas para ignorar o registro pelo sistema.

Uma direção de pesquisa interessante é desenvolver *frameworks* para consideração sistemática de fatores de usabilidade e seu impacto na aplicação de políticas, com o objetivo de reduzir o atrito entre objetivos de segurança e conveniência. A boa prática na indústria de acesso físico estabelece um tempo de resposta máximo de três segundos entre solicitar acesso e acionamento da fechadura (MASOUMZADEH; LAAN; DERCKSEN, 2022). Se este atraso for

excedido, a experiência mostra que os sujeitos ficam irritados, o que potencialmente leva a danos ao equipamento, perda de produtividade, aceitação reduzida do usuário e, subsequentemente, defeito do sistema.

O *KeyCode Manager* contribui para estes princípios de aceitabilidade psicológica. O escaneamento de códigos QR é significativamente mais rápido que a assinatura de um livro de registros físico manual. A interface Web oferece maior acessibilidade que um caderno físico localizado em um ponto específico. Embora a implementação atual envolva interação humana na entrega física das chaves, a arquitetura do sistema foi projetada com margem para extensão futura, permitindo a integração com armários de chaves controlados eletronicamente que eliminariam completamente a interação humana e atenderiam ao requisito rigoroso de três segundos aplicável a catracas eletrônicas. Nesta configuração estendida, o sistema se tornaria totalmente automatizado: após autenticação e escaneamento do código QR, o armário liberaria a chave automaticamente, registrando a transação sem necessidade de intervenção de funcionários. A conveniência da interface atual, combinada com esta possibilidade de automação completa, contribui significativamente para maior aceitação pelos usuários e viabiliza evolução incremental do sistema conforme recursos se tornem disponíveis.

3.2 Segurança de códigos QR

Embora ofereçam uma ampla gama de vantagens, incluindo alta densidade de informação e robustez, códigos QR apresentam riscos significativos de segurança, ao quais serão tratados nesta seção e que precisam ser considerados em sistemas PAC como o *KeyCode Manager*.

A literatura sobre segurança de códigos QR foca predominantemente em cenários de uso público, onde usuários finais escaneam códigos QR de fontes desconhecidas em espaços públicos (anúncios, cartazes, embalagens de produtos). Estes cenários diferem fundamentalmente de ambientes controlados como o *KeyCode Manager*, onde os códigos são gerados por administradores do sistema e fisicamente anexados a ativos institucionais em ambiente controlado. Neste trabalho, entende-se que compreender as vulnerabilidades identificadas na literatura é essencial para fundamentar as decisões de design de segurança do sistema, mesmo quando alguns ataques não se aplicam diretamente ao contexto institucional.

3.2.1 Ataques com códigos QR

Krombholz *et al*. (2014) identificam dois principais ataques que se utilizam de códigos QR: (1) a substituição completa do código QR, onde atacantes criam novos códigos QR maliciosos e os

colam sobre códigos existentes em anúncios publicitários ou outros locais; e (2) a modificação de módulos individuais do código QR, alterando módulos específicos de preto para branco e vice-versa para sobrescrever o conteúdo originalmente codificado.

Wahsheh e Luccio (2020) destacam que estudos recentes demonstram que códigos de barras podem ser maliciosamente utilizados para executar diferentes tipos de ataques, incluindo *phishing*, propagação de *malware*, *cross-site scripting* (XSS), injeção SQL e ataques às aplicações leitoras de códigos. Estes meios de ataque representam ameaças significativas em ambientes onde códigos QR são utilizados para controle de acesso físico ou identificação de ativos.

3.2.2 Exemplos de ataques reais

Krombholz *et al*. (2014) documentam diversos exemplos de ataques bem-sucedidos utilizando códigos QR. Em 2012, o especialista em segurança Ravi Borgaonkar demonstrou como códigos MMI (do inglês, *Man-Machine Interface*) podem ser utilizados para executar ataques contra dispositivos Samsung, onde um código QR contendo o código MMI `*2767*3855#` com o prefixo `tel:` foi capaz de apagar completamente os dados do dispositivo móvel ao ser escaneado.

Ataques de injeção SQL via códigos QR também foram demonstrados como viáveis. Se uma aplicação de escaneamento utiliza um banco de dados para armazenar entradas escaneadas, é possível executar uma injeção SQL escaneando valores como `1' OR 1=1 --` para contornar mecanismos de autenticação. Adicionalmente, explorações baseadas em navegador, ataques XSS e injeções de comando via códigos QR foram documentadas (Krombholz *et al*., 2014).

Um exemplo particularmente relevante é o *typosquatting*, a prática de registro intencional de erros ortográficos de endereços de sites populares. Moore e Edelman (2010), citados por Krombholz *et al*. (2014), estimam que existem pelo menos 938.000 domínios de *typosquatting* direcionados aos 3.264 principais sites “.com”, demonstrando que alterar um código QR para produzir um nome de domínio com erro ortográfico é um vetor de ataque eficaz para *phishing*.

3.2.3 Vulnerabilidades em cenários públicos

Estudos sobre comportamento do usuário em cenários públicos revelam vulnerabilidades significativas. Krombholz *et al*. (2014) citam pesquisas de Seeburger *et al*. (2012) e Vidas *et al*. (2013) que identificaram a curiosidade como principal motivação para usuários escanearem códigos QR desconhecidos, indicando que usuários ignoram ou desconhecem ameaças de segurança associadas a códigos QR de fontes não verificadas.

Duas características fundamentais contribuem para estas vulnerabilidades: (1) códigos QR não são legíveis por humanos, exigindo decodificação antes de avaliar o conteúdo; e (2) a maioria dos leitores de código QR não fornece ferramentas viáveis para detectar automaticamente ataques (Krombholz *et al.*, 2014). Esta lacuna entre comportamento do usuário e proteções disponíveis representa desafio significativo para sistemas que dependem exclusivamente de decisões do usuário final para validação de segurança.

3.2.4 Limitações de aplicações leitoras

Wahsheh e Luccio (2020) analisaram 100 aplicações leitoras de código de barras sob perspectivas de segurança e privacidade, revelando que popularidade não implica segurança. A análise identificou que muitas aplicações solicitam permissões desnecessárias, utilizam algoritmos de criptografia fracos (DES, AES-ECB), e não adotam métodos padronizados de codificação. Esta inconsistência nas proteções oferecidas por diferentes aplicações leitoras reforça que sistemas críticos não devem depender exclusivamente de proteções implementadas no lado do cliente.

3.2.5 Controles de segurança: cenários públicos versus controlados

A literatura identifica dois conjuntos distintos de controles de segurança conforme o contexto de uso de códigos QR. Para cenários públicos, onde usuários escaneam códigos de fontes desconhecidas, Wahsheh e Luccio (2020) e Krombholz *et al.* (2014) recomendam as seguintes proteções centradas no usuário final:

1. Verificação de URL e avisos de segurança na aplicação leitora
2. Exibição do conteúdo decodificado antes da execução
3. Resolução de URLs encurtadas para revelar destino final
4. Detecção visual de modificações através de análise de distribuição de módulos
5. Prevenção de execução automática de código ou comandos

Em ambientes institucionais controlados, onde administradores geram códigos QR e os anexam fisicamente a ativos, a estratégia de segurança difere fundamentalmente. Krombholz *et al.* (2014) reconhecem que assinaturas digitais permitiriam verificar o originador e detectar modificações, mas observam que “a quantidade aumentada de dados para codificar reduz a área disponível para dados reais”. Para sistemas de identificação de ativos, validação criptográfica no servidor representa abordagem mais apropriada.

O *KeyCode Manager* implementa esta estratégia através de:

1. **Validação HMAC no servidor**: todos os códigos QR contêm assinatura HMAC validada pelo servidor, protegendo contra substituição e modificação. Caso substituídos ou

modificados, falharão na validação criptográfica, sendo rejeitados automaticamente pelo sistema.

2. **Validação automática**: o sistema não requer que usuários tomem decisões de segurança sobre confiabilidade dos códigos. A validação ocorre automaticamente quando o código é escaneado, mitigando as vulnerabilidades de comportamento do usuário identificadas na literatura.
3. **Ambiente institucional**: códigos QR são gerados centralizadamente por administradores e anexados fisicamente às chaves. Embora usuários com acesso temporário às chaves possam teoricamente substituir códigos QR, tal substituição seria detectada pela validação HMAC, que rejeita códigos não gerados pelo sistema. Adicionalmente, o contexto institucional reduz motivações para ataques como *phishing* via *typosquatting*, comuns em cenários públicos.
4. **Independência de aplicação leitora**: como aplicação Web, o sistema não requer instalação de aplicação dedicada. Usuários podem utilizar qualquer leitor de QR, pois toda validação de segurança ocorre no servidor.

Esta abordagem está alinhada com o princípio identificado por Wahsheh e Luccio (2020) de que sistemas críticos não devem depender exclusivamente de proteções implementadas no lado do cliente, especialmente considerando as inconsistências nas proteções oferecidas por diferentes aplicações leitoras disponíveis no mercado.

3.3 Trilhas de auditoria e Segurança da Informação

Este seção apresenta estudos sobre trilhas de auditoria, controle de acesso em instituições públicas brasileiras e gestão de segurança da informação em organizações com recursos limitados, temas diretamente relacionados ao contexto de desenvolvimento e implantação do *KeyCode Manager*.

3.3.1 Trilhas de auditoria e proteção de logs

A auditoria de sistemas de informação constitui um mecanismo fundamental para garantir a rastreabilidade de ações e a detecção de comportamentos maliciosos. Roratto e Dias (2014) apresentam um estudo sobre trilhas de auditoria em sistemas de banco de dados, destacando sua importância para a segurança da informação em ambientes de produção e operações.

Os autores definem trilha de auditoria como um registro cronológico de atividades do sistema que permite reconstruir e examinar a sequência de eventos relacionados a uma operação

específica, sendo um mecanismo é essencial para atender requisitos legais, detectar violações de segurança e fornecer evidências em investigações. O estudo enfatiza que “o primeiro alvo de um atacante experiente será o sistema de *log* de auditoria, seja para ocultar suas ações ou para implantar informações falsas que incriminem outros usuários” (RORATTO; DIAS, 2014).

Esta observação ressalta a necessidade de proteger não apenas os dados operacionais, mas também os próprios registros de auditoria contra modificações não autorizadas. Roratto e Dias (2014) apresentam o conceito de *Forward Integrity* (FI), uma técnica criptográfica que garante que registros de auditoria anteriores permaneçam válidos mesmo que um atacante comprometa o sistema no presente.

A técnica de *Forward Integrity* utiliza *Message Authentication Codes* (MACs) encadeados, onde cada nova entrada de *log* depende criptograficamente da entrada anterior. Isso cria uma cadeia de autenticação que torna praticamente impossível modificar registros históricos sem detecção, mesmo com acesso administrativo ao sistema (RORATTO; DIAS, 2014).

O estudo analisa duas soluções comerciais que implementam trilhas de auditoria avançadas: *Oracle Database Vault* e *IBM InfoSphere Guardium*. Ambas as soluções oferecem separação de privilégios, monitoramento em tempo real e proteção contra usuários privilegiados, demonstrando que a auditoria efetiva requer controles técnicos específicos além da simples geração de logs (RORATTO; DIAS, 2014).

O *KeyCode Manager* implementa trilhas de auditoria através de registros imutáveis em nível de aplicação. Todos os campos da entidade de *log* são marcados como imutáveis no esquema do banco de dados, e a camada de aplicação não expõe operações de atualização ou exclusão de registros históricos. Esta abordagem garante que o histórico de empréstimos e devoluções não seja modificado através da interface do sistema, fornecendo rastreabilidade para responsabilização e investigação de incidentes.

Embora o sistema não implemente técnicas criptográficas de *Forward Integrity* como MACs encadeados, a imutabilidade em nível de aplicação representa um equilíbrio pragmático entre segurança e complexidade de implementação. Para ambientes que exigem garantias criptográficas mais rigorosas contra modificação de logs por administradores com acesso direto ao banco de dados, a implementação de *Forward Integrity* conforme descrita por Roratto e Dias (2014) constitui uma evolução recomendada.

3.3.2 Controle de acesso em instituições públicas

Sfreddo e Flores (2012) investigam o controle de acesso à informação arquivística em arquivos públicos estaduais brasileiros, apresentando uma pesquisa empírica que examina como sete instituições implementam políticas de segurança da informação. O estudo é fundamentado em três normas principais: ISO 15489-1 (gestão de documentos), e-ARQ Brasil (requisitos para sistemas informatizados de gestão arquivística) e ABNT NBR ISO/IEC 27002 (código de prática para gestão da segurança da informação).

Os autores estabelecem que o controle de acesso é um componente crítico da segurança da informação arquivística, destacando que "as organizações têm de poder controlar quem está autorizado a aceder aos documentos e à informação e, por outro lado, os utilizadores autorizados necessitam de garantias de que os registros a que acedem são autênticos e fiáveis" (SFREDDO; FLORES, 2012). A pesquisa revelou que, embora 71% das instituições possuam documentos normativos sobre controle de acesso, apenas 29% têm políticas formalmente estabelecidas. Esta lacuna entre a existência de documentação e a implementação efetiva de políticas de segurança é preocupante no contexto de instituições públicas que custodiam informações sensíveis.

Sfreddo e Flores (2012) identificam que os mecanismos de controle de acesso mais comuns incluem autenticação por usuário e senha (100%), definição de perfis de acesso (86%), e controles baseados em classificação de sigilo (71%). No entanto, o estudo constatou deficiências na aplicação consistente destes controles e na ausência de revisões periódicas de permissões.

Os autores também enfatizam a importância da ABNT NBR ISO/IEC 27002, que define controles de segurança organizados em domínios como política de segurança, gestão de ativos, controle de acesso, e gestão de incidentes. A norma estabelece que o controle de acesso deve ser baseado em requisitos de negócio e segurança da informação, incluindo segregação de funções e princípio do menor privilégio, onde usuários recebem apenas as permissões necessárias para suas funções.

Para o *KeyCode Manager*, este estudo oferece diretrizes importantes sobre implementação de controles de acesso em contexto institucional público. O sistema implementa controle de acesso baseado em três papéis de usuário: estudante, professor e terceirizado. Todos os usuários autenticados possuem capacidade de retirar e devolver chaves, mantendo a simplicidade operacional adequada ao contexto educacional. Esta abordagem de classificação por papéis estabelece fundação para futuras implementações de segregação de funções mais granular, conforme recomendado pela ABNT NBR ISO/IEC 27002.

No contexto brasileiro, sistemas desenvolvidos para instituições públicas devem também considerar o arcabouço regulatório nacional. A Lei Geral de Proteção de Dados (BRASIL, 2018) estabelece princípios e requisitos para tratamento de dados pessoais, incluindo o princípio de minimização (coleta apenas de dados necessários) e a implementação de medidas de segurança técnicas e administrativas para proteção dos dados. Para instituições federais, a Instrução Normativa GSI/PR nº 1 (BRASIL, 2008) disciplina a gestão de segurança da informação e comunicações, estabelecendo requisitos para controle de acesso físico e lógico, proteção contra acesso não autorizado, e implementação de trilhas de auditoria. As medidas de segurança implementadas no *KeyCode Manager*, descritas na Seção 4.4.2, abordam diversos destes requisitos, embora a conformidade regulatória completa esteja fora do escopo deste trabalho acadêmico.

3.3.3 Gestão da Segurança da Informação em PMEs

Netto e Silveira (2007) investigam os fatores que influenciam a adoção de práticas de gestão da segurança da informação em pequenas e médias empresas (PMEs), através de um estudo empírico com 43 empresas do setor metalúrgico da região do ABC paulista. O estudo propõe um modelo de três camadas de segurança: física, lógica e humana.

A camada de segurança física engloba controles sobre instalações, equipamentos e suportes de armazenamento. Netto e Silveira (2007) identificam que a camada física apresentou o melhor nível de implementação entre as três camadas nas empresas pesquisadas, incluindo controles como proteção de equipamentos contra falhas de energia elétrica, salas de servidores em locais restritos, e uso de firewall para proteção da rede.

A camada lógica refere-se a controles implementados através de software e hardware, incluindo autenticação, criptografia, firewalls e sistemas de detecção de intrusão. O estudo revelou que 100% das PMEs pesquisadas possuem antivírus instalado, 97,6% implementam políticas de backup sistemáticas, 82,9% utilizam firewall, e aproximadamente 44% adotam criptografia em banco de dados ou para troca de informações (NETTO; SILVEIRA, 2007).

A camada humana aborda treinamento, conscientização e políticas de uso aceitável. Netto e Silveira (2007) constataram que esta é a camada mais negligenciada pelas PMEs pesquisadas, apresentando “a maior carência de cuidados por parte das empresas, seguida pela camada lógica”. O estudo revelou baixa adoção de controles como conscientização e treinamento em segurança, políticas de segurança da informação formalizadas, e termos de responsabilidade assinados por funcionários.

O estudo identifica que o principal fator motivador para adoção de gestão da segurança da informação em PMEs é “evitar possíveis perdas financeiras”, sendo o único fator que apresentou diferença estatisticamente significativa em relação aos demais (recomendação de especialista externo, ocorrência de incidente anterior, e consciência do gestor). Em relação aos fatores inibidores, o estudo constatou que todos apresentaram graus de importância estatisticamente equivalentes: valor do investimento, dificuldade em mensurar custo/benefício, falta de conhecimento, e cultura organizacional (NETTO; SILVEIRA, 2007).

Netto e Silveira (2007) argumentam que a gestão efetiva de segurança da informação em PMEs requer abordagem balanceada entre as três camadas, com políticas formais, investimento proporcional aos riscos, e liderança organizacional comprometida.

Para o *KeyCode Manager*, este modelo de três camadas oferece uma estrutura conceitual relevante. A camada física é diretamente abordada pelo sistema através do controle de acesso a chaves. A camada lógica se manifesta na autenticação de usuários, autorização baseada em perfis, e auditoria de operações. A camada humana deve ser considerada através de treinamento de servidores responsáveis pelo sistema e políticas institucionais claras sobre uso e responsabilização. O estudo também reforça que, no contexto de instituições públicas de ensino (que compartilham características com PMEs em termos de recursos limitados), a segurança física permanece como fundamento essencial sobre o qual controles adicionais são construídos.

3.4 Síntese da revisão

A revisão da literatura apresentada neste capítulo demonstra que o *KeyCode Manager* se posiciona na interseção de diversos domínios de pesquisa em segurança da informação. A análise de sistemas de controle de acesso físico revelou a lacuna existente entre soluções puramente eletrônicas (custosas e complexas) e sistemas mecânicos tradicionais (econômicos mas com rastreabilidade limitada). O conceito de sistemas PAC híbridos, particularmente armários de chaves controlados, oferece fundamentação teórica para a abordagem adotada neste trabalho.

A investigação sobre segurança de códigos QR evidenciou vulnerabilidades significativas relacionadas a substituição, modificação e engenharia social, justificando a implementação de validação criptográfica no lado do servidor através de HMAC. Os estudos sobre comportamento do usuário confirmam que a proteção não pode depender exclusivamente de decisões de segurança por parte dos usuários finais, reforçando a escolha arquitetural de validação automática centralizada.

A análise de trilhas de auditoria e técnicas de *Forward Integrity* apresenta o estado da arte em proteção criptográfica de logs, identificando tanto soluções avançadas (como MACs encadeados) quanto abordagens pragmáticas (como imutabilidade em nível de aplicação). Esta revisão demonstra que diferentes contextos operacionais justificam diferentes níveis de proteção de logs, equilibrando segurança e complexidade de implementação. Os estudos sobre controle de acesso em instituições públicas brasileiras revelam que a existência de documentação normativa não garante implementação efetiva de políticas de segurança, destacando a necessidade de sistemas que simplifiquem e automatizem a aplicação de controles.

O modelo de três camadas de segurança (física, lógica e humana) oferece *framework* conceitual para análise holística do *KeyCode Manager*, evidenciando que soluções efetivas de segurança da informação em ambientes com recursos limitados devem equilibrar proteção técnica, usabilidade operacional e aceitabilidade organizacional. A literatura revisada confirma que sistemas de controle de acesso físico híbridos, que combinam fechaduras mecânicas existentes com rastreamento eletrônico automatizado, representam solução pragmática e economicamente viável para instituições de ensino, área ainda pouco explorada na pesquisa acadêmica sobre PAC.

Por fim, conforme mencionado na introdução, o *KeyCode Manager* representa uma reimplementação de um protótipo desenvolvido anteriormente por estudantes do IFPB campus Campina Grande. O sistema mantém os requisitos funcionais originais (cadastro de usuários e chaves, entrega e devolução e histórico de movimentações), concentrando as melhorias em aspectos arquiteturais e de segurança. A Tabela 2 apresenta uma comparação entre o protótipo original e a nova implementação.

Tabela 2: Comparativo entre o protótipo original e o atual

Aspecto	**Protótipo Original**	**Protótipo Atual**
Linguagem	Python	Go
Arquitetura	Monolítica sem separação	Hexagonal
Armazenamento de senhas	Texto plano	*Hashing* com Argon2id
Validação de QR Code	Sem validação	HMAC-SHA256
Trilha de auditoria	Mutável pela aplicação	Imutável na aplicação
Testes automatizados	Ausentes	424 testes (>80% cobertura)
Geração de relatórios	Ausente	Exportação em PDF
Controle de permissões	Por domínio de e-mail	Perfis de usuário

Fonte: Elaborado pelo autor (2025)

4 METODOLOGIA

Este capítulo descreve a metodologia empregada no desenvolvimento do *KeyCode Manager*, abrangendo desde a classificação da pesquisa até os aspectos técnicos de implementação e estratégias de validação do sistema. Inicialmente, apresenta-se a classificação metodológica da pesquisa, caracterizando sua natureza aplicada e abordagem exploratória incremental. Em seguida, são detalhadas as tecnologias utilizadas no desenvolvimento, incluindo linguagens de programação, *frameworks* e ferramentas de apoio ao desenvolvimento. A arquitetura do sistema é apresentada através de sua organização hexagonal e modelo de dados, fornecendo visão estrutural da solução. Os aspectos de implementação são discutidos com foco em três áreas críticas: a utilização de códigos QR para identificação de chaves, os mecanismos de segurança implementados para proteção de dados e controle de acesso e as funcionalidades principais que compõem o ciclo de vida do gerenciamento de chaves. Por fim, são apresentadas a estratégia de testes adotada para garantir a qualidade do software e as considerações sobre *deployment* do sistema em ambientes reais.

4.1 Classificação da pesquisa

Este trabalho caracteriza-se como uma pesquisa aplicada com desenvolvimento experimental. Segundo Prodanov e Freitas (2013), a pesquisa aplicada objetiva gerar conhecimentos para aplicação prática, dirigidos à solução de problemas específicos, envolvendo verdades e interesses locais. Neste contexto, o desenvolvimento do *KeyCode Manager* foi motivado pela observação direta do problema de controle manual de chaves no Instituto Federal de Educação da Paraíba — campus Campina Grande — buscando gerar conhecimento aplicável à solução prática deste problema institucional.

Neste trabalho, seguiu-se uma abordagem exploratória e incremental. As tecnologias, padrões arquiteturais e boas práticas foram estudados e aplicados durante o próprio processo de desenvolvimento, resultando em ciclos de implementação, aprendizado e refatoração. Esta abordagem permitiu a evolução gradual tanto do sistema quanto da compreensão técnica necessária para sua construção, com a revisão da literatura sendo conduzida de forma paralela ao desenvolvimento para fundamentar as decisões técnicas e contexto acadêmico do trabalho.

O trabalho foi estruturado em três grandes etapas. A primeira etapa consistiu na concepção e no planejamento inicial do sistema, onde a proposta do *KeyCode Manager* foi apresentada e discutida com os orientadores, definindo-se o escopo geral da solução e as

principais funcionalidades a serem implementadas. A segunda etapa, a mais extensa, envolveu o desenvolvimento e o aprendizado contínuo, caracterizada pela implementação incremental do sistema simultaneamente ao estudo de tecnologias Web, padrões arquiteturais e boas práticas. Nesta fase, o código foi constantemente refatorado à medida que novos conhecimentos eram adquiridos, buscando-se sempre uma base de código limpa e bem estruturada. A terceira etapa compreendeu a consolidação e a validação, onde, após estabelecer uma fundação sólida do sistema, foram implementadas as funcionalidades finais e desenvolvida a suíte de testes unitários e de integração para validar o comportamento do sistema. A revisão da literatura foi conduzida de forma paralela ao desenvolvimento, fundamentando as decisões técnicas e fornecendo o contexto acadêmico necessário para o trabalho.

4.2 Desenvolvimento e tecnologias

O desenvolvimento seguiu uma abordagem incremental com refatoração contínua, utilizando exclusivamente tecnologias *open-source* e gratuitas. O *backend* foi implementado em Go[3] com o *framework* Echo[4] para gerenciamento de rotas e *middleware*. A persistência utiliza SQLite[5] com o ORM Ent[6]. A camada de apresentação combina Templ[7] para *templates* HTML[8], HTMX[9] para interações dinâmicas e Alpine.js[10] para reatividade no cliente, com estilização via Bulma[11]. Configurações são gerenciadas pela biblioteca Viper[12], e testes utilizam a biblioteca padrão do Go complementada por Testify[13].

[3]Disponível em: https://go.dev. Acesso em: 15 dez. 2025.

[4]Disponível em: https://echo.labstack.com. Acesso em: 15 dez. 2025.

[5]Disponível em: https://www.sqlite.org. Acesso em: 15 dez. 2025.

[6]Disponível em: https://entgo.io. Acesso em: 15 dez. 2025.

[7]Disponível em: https://templ.guide. Acesso em: 15 dez. 2025.

[8]Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTML. Acesso em: 15 dez. 2025.

[9]Disponível em: https://htmx.org. Acesso em: 15 dez. 2025.

[10]Disponível em: https://alpinejs.dev. Acesso em: 15 dez. 2025.

[11]Disponível em: https://bulma.io. Acesso em: 15 dez. 2025.

[12]Disponível em: https://github.com/spf13/viper. Acesso em: 15 dez. 2025.

[13]Disponível em: https://github.com/stretchr/testify. Acesso em: 15 dez. 2025.

O ambiente de desenvolvimento utiliza ferramentas *open-source* como Air[14] para recarregamento automático durante o desenvolvimento, golangci-lint[15] e Biome[16] para análise estática e Lefthook[17] para automação de verificações de qualidade via *hooks* de Git[18].

4.3 Arquitetura e modelagem

4.3.1 Arquitetura hexagonal

O sistema implementa arquitetura hexagonal com separação clara de responsabilidades, conforme ilustrado na Figura 2. A camada de *handler* funciona como adaptador primário, responsável pelo processamento de requisições HTTP, realizando validação de formato de entrada e conversão entre DTOs e modelos de domínio, desacoplando assim a representação HTTP da lógica de negócio.

O núcleo hexagonal é composto pela camada de domínio e pela camada de serviço. A camada de domínio contém entidades e objetos de valor que encapsulam regras de validação e comportamentos fundamentais. A camada de serviço implementa a lógica de negócio e define portas de saída através de interfaces de repositório, orquestrando operações entre diferentes repositórios e implementando regras de negócio de mais alto nível como fluxos de autenticação e autorização.

A camada de repositório implementa as portas definidas pela camada de serviço, atuando como adaptador secundário que abstrai o acesso a dados e isola a lógica de persistência através de conversões entre modelos de domínio e entidades ORM. O fluxo de uma requisição atravessa todas essas camadas sequencialmente, com transformações de dados apropriadas em cada etapa.

[14]Disponível em: https://github.com/air-verse/air. Acesso em: 15 dez. 2025.

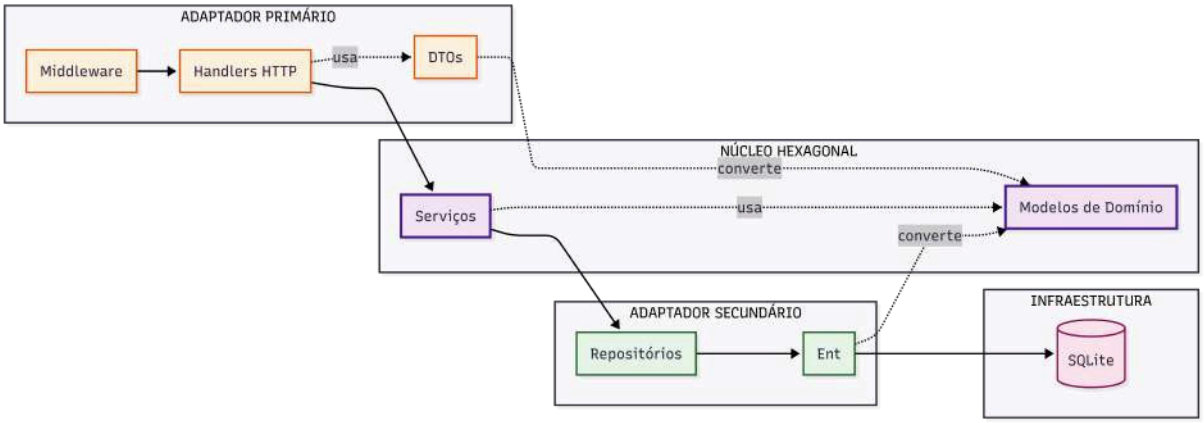
[15]Disponível em: https://golangci-lint.run. Acesso em: 15 dez. 2025.

[16]Disponível em: https://biomejs.dev. Acesso em: 15 dez. 2025.

[17]Disponível em: https://github.com/evilmartians/lefthook. Acesso em: 15 dez. 2025.

[18]Disponível em: https://git-scm.com. Acesso em: 15 dez. 2025.

Figura 2: Arquitetura hexagonal do KeyCode Manager

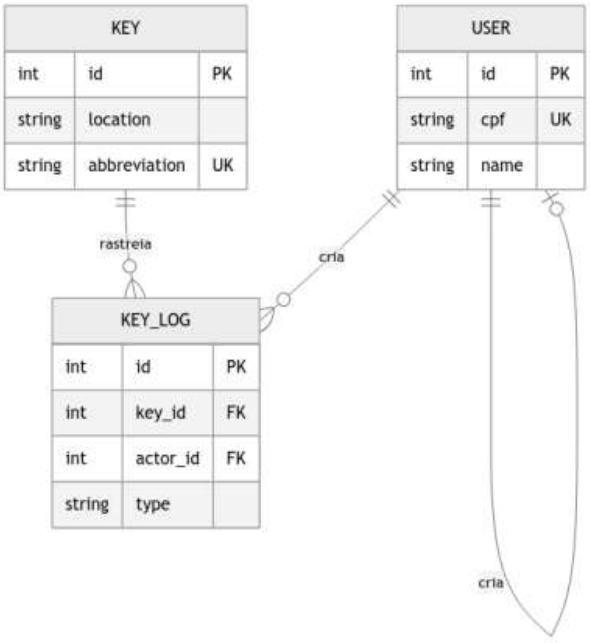


Fonte: Elaborado pelo autor (2025)

4.3.2 Modelo de dados

O banco de dados possui três entidades principais, conforme ilustrado na Figura 3: `User` armazena nome, CPF, telefone e PIN com *hash* seguro, classificados em três papéis (professor, estudante ou terceirizado) com flag `admin` para privilégios administrativos; `Key` representa chaves físicas com localização, abreviação, código QR e campos de controle de disponibilidade; `KeyLog` implementa *log* de auditoria imutável capturando operação, usuário, chave e *timestamp*.

Figura 3: Diagrama entidade-relacionamento simplificado do banco de dados



Fonte: Elaborado pelo autor (2025)

4.4 Implementação

4.4.1 Códigos QR

Códigos QR identificam chaves físicas através do uso da biblioteca *open-source* `skip2/go-qrcode`[19] com nível M de correção de erros. Cada código contém string estruturada no formato `KEYCODE_<uuid>_<hmac>`, onde o HMAC autentica o código no servidor, prevenindo falsificações. Como discutido na Seção 3.2, a validação HMAC mitiga riscos de substituição identificados na literatura, ocorrendo automaticamente no servidor sem depender de decisões do usuário.

4.4.2 Segurança

A segurança do *KeyCode Manager* foi projetada seguindo o princípio de defesa em profundidade, conforme discutido na Seção 2.4, implementando múltiplas camadas de proteção que se complementam para mitigar diferentes vetores de ataque. O objetivo central é proteger a confidencialidade dos dados dos usuários, garantir a integridade das operações de gerenciamento de chaves, e manter rastreabilidade completa das ações realizadas no sistema.

A camada de autenticação e gerenciamento de sessões foi implementada utilizando a biblioteca *open-source* `gorilla/sessions`[20], que gerencia sessões com validade de 7 dias, balanceando conveniência operacional e segurança. As credenciais dos usuários (PINs) são protegidas através de *hash* Argon2id, variante recomendada pelo RFC 9106[21] por sua resistência superior contra ataques de força bruta baseados em GPU e ataques de *side-channel*. Um *pepper* configurável adiciona camada extra de proteção, garantindo que mesmo com acesso ao banco de dados, atacantes não consigam reverter os *hashes* facilmente. A regeneração automática de identificadores de sessão após autenticação previne ataques de fixação de sessão, onde um atacante poderia forçar um usuário a utilizar um identificador de sessão conhecido.

A proteção das sessões ativas é reforçada através de configurações seguras nos *cookies* de sessão: a *flag* `HttpOnly` impede acesso via JavaScript, mitigando ataques XSS (do inglês, *Cross-Site Scripting*); `SameSite=Strict` bloqueia envio de *cookies* em requisições originadas de outros domínios, protegendo contra CSRF (do inglês, *Cross-Site Request Forgery*); e `Secure` garante transmissão exclusivamente via HTTPS em ambiente de produção, prevenindo

[19] Disponível em https://github.com/skip2/go-qrcode. Acesso em: 15 dez. 2025.

[20] Disponível em: https://github.com/gorilla/sessions. Acesso em: 15 dez. 2025.

[21] Disponível em: https://datatracker.ietf.org/doc/html/rfc9106. Acesso em: 15 dez, 2025.

interceptação em trânsito. Adicionalmente, todas as operações de mutação de dados exigem validação de token CSRF via cabeçalho `X-CSRF-Token`, criando uma segunda camada de defesa contra ataques de falsificação de requisição entre sites.

O modelo de autorização implementa controle de acesso baseado em papéis, onde usuários são classificados em três categorias (professor, estudante, terceirizado) para fins de registro e rastreabilidade, complementado por uma *flag* administrativa que controla acesso ao painel de gestão do sistema. Usuários não administrativos têm acesso ao quiosque, enquanto administradores acessam funcionalidades de gerenciamento de usuários, chaves e visualização de auditoria. Esta abordagem atende ao princípio do menor privilégio, separando operações rotineiras de gestão administrativa.

A validação de dados ocorre em múltiplas camadas arquiteturais para garantir consistência e prevenir vulnerabilidades. Objetos de valor como `CPF`, `PIN` e `PublicID` encapsulam regras de validação, garantindo que dados inválidos nunca sejam representados no sistema. Na camada de *handlers* HTTP, validações de formato verificam requisições antes do processamento. A camada de serviço aplica regras de negócio complexas, e a camada de repositório utiliza consultas parametrizadas via ORM Ent, prevenindo ataques de injeção SQL através da separação automática entre código SQL e dados fornecidos por usuários.

Para atender requisitos de auditoria e detecção de incidentes, o sistema implementa *logging* estruturado utilizando a biblioteca padrão do Go `slog`. Diferentemente de *logs* tradicionais baseados em texto livre, o *logging* estruturado representa cada evento como conjunto de pares chave-valor, facilitando análise automatizada, filtragem e correlação de eventos. O sistema utiliza formato JSON para serialização dos *logs*, permitindo integração direta com ferramentas de terceiros para agregação, análise e visualização de *logs*, como Grafana[22], Loki[23], Elasticsearch[24] ou serviços de monitoramento em nuvem. O sistema registra todas as operações sensíveis com contexto rico, incluindo identificadores de requisição, endereços IP, tipos de operação e *timestamps* precisos. Esta abordagem permite identificação rápida de padrões anômalos, rastreamento de ações específicas de usuários, investigação forense de incidentes de segurança e geração de métricas operacionais através de *profiling* e análise de desempenho.

A camada de infraestrutura implementa proteções adicionais contra vetores de ataque em nível de transporte e aplicação. Em ambiente de produção, o sistema força comunicação

[22]Disponível em: https://grafana.com. Acessom em: 15 dez. 2025.

[23]Disponível em: https://grafana.com/oss/loki. Acesso em: 15 dez, 2025.

[24]Disponível em: https://www.elastic.co/elasticsearch. Acesso em: 15 dez, 2025.

exclusivamente via HTTPS através de redirecionamento automático, garantindo que todas as transmissões de dados entre cliente e servidor sejam criptografadas. Adicionalmente, cabeçalhos HTTP de segurança instruem navegadores a aplicar políticas de proteção: `X-Frame-Options` previne ataques de *clickjacking* ao impedir que a aplicação seja incorporada em *iframes* maliciosos, `X-Content-Type-Options` bloqueia *MIME type sniffing* que poderia permitir execução de conteúdo malicioso, `X-XSS-Protection` ativa filtros de XSS nos navegadores, e `Strict-Transport-Security` força uso de HTTPS por período prolongado mesmo em acessos futuros.

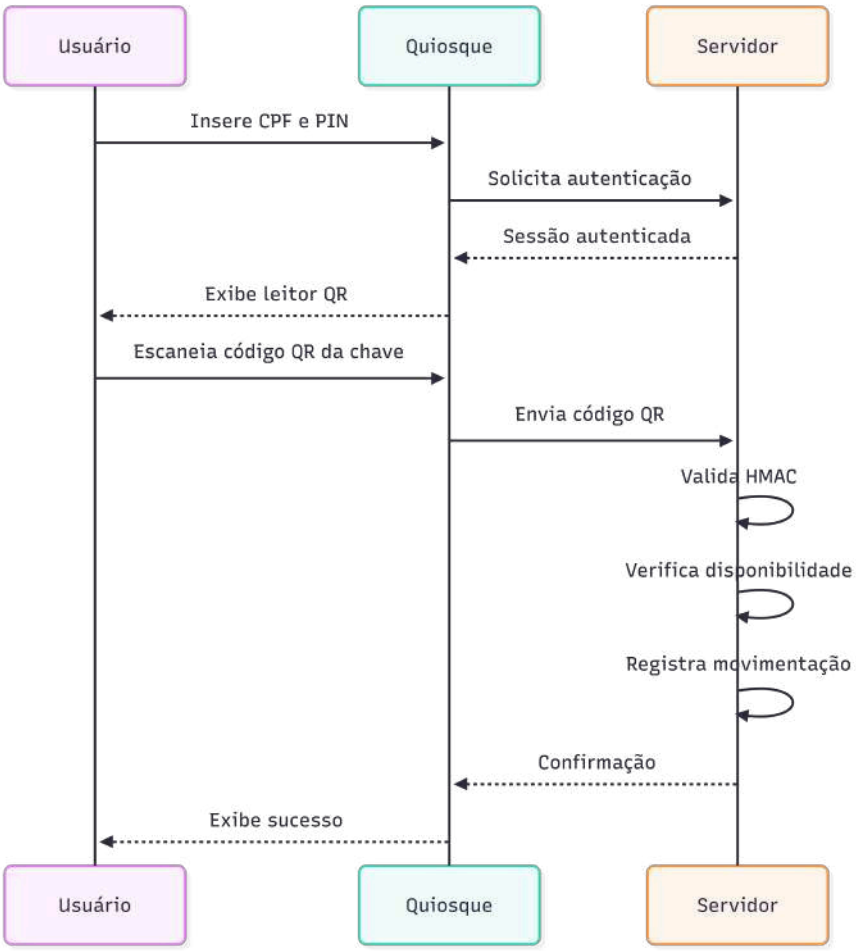
A proteção contra ataques de negação de serviço opera em múltiplas camadas. A limitação de taxa implementa restrição de 100 requisições por segundo por endereço IP, prevenindo que atacantes sobrecarreguem o servidor com volume excessivo de requisições. Os *endpoints* de autenticação recebem limites adicionais mais restritivos para prevenir ataques de força bruta distribuídos que tentam adivinhar credenciais através de múltiplas tentativas. O *timeout* de 30 segundos para processamento de requisições previne ataques *slowloris*, onde atacantes mantêm conexões abertas indefinidamente enviando dados de forma extremamente lenta, esgotando recursos do servidor. A limitação do tamanho do corpo das requisições a 1KB previne ataques baseados em *payloads* excessivamente grandes que poderiam causar consumo excessivo de memória ou processamento.

A configuração de CORS (do inglês, *Cross-Origin Resource Sharing*) restringe quais origens podem realizar requisições à API e quais métodos HTTP são permitidos, reduzindo a superfície de ataque ao bloquear métodos não utilizados pela aplicação e prevenir requisições não autorizadas de domínios externos.

4.4.3 Funcionalidades principais

O sistema gerencia o ciclo de vida completo das chaves através de duas interfaces: quiosque (autoatendimento) e painel administrativo. Administradores cadastram usuários (professores, estudantes, terceirizados) e chaves (localização, abreviação, QR code automático). No quiosque, usuários autenticam-se com CPF e PIN, escaneiam o código QR da chave via webcam e o sistema registra a transação automaticamente. A Figura 4 ilustra o fluxo de retirada de chave, desde a autenticação do usuário até a confirmação da operação.

Figura 4: Diagrama de sequência do fluxo de retirada de chave



Fonte: Elaborado pelo autor (2025)

O rastreamento de disponibilidade mantém o estado atualizado (disponível, em uso, perdida), evitando tentativas de retirada de itens indisponíveis. A trilha de auditoria imutável registra todas as operações de retirada e devolução, permitindo identificar o responsável atual por cada chave e reconstruir o histórico completo de utilização.

O sistema oferece funcionalidade de geração de relatórios em PDF utilizando a biblioteca *open-source* Maroto[25], permitindo que administradores exportem dados de auditoria com filtros personalizados (período, tipo de operação, usuário, presença de auxiliar). Os relatórios incluem estatísticas como total de movimentações, média diária, usuário mais ativo, dia mais movimentado e ranking das chaves mais utilizadas, facilitando análise de padrões de uso e prestação de contas institucional.

[25]Disponível em: https://github.com/johnfercher/maroto. Acesso em: 15 dez. 2025.

O sistema aborda problemas identificados no IFPB (Seção 1.2): a interface digital elimina livros de assinatura, o rastreamento automático identifica responsáveis atuais, QR codes aceleram registro, e o controle de acesso baseado em papéis implementa gestão de permissões ausente no processo manual.

4.5 Estratégia de testes

A estratégia de testes adotada segue abordagem pragmática focada em maximizar a cobertura das camadas que contêm lógica de negócio crítica, concentrando esforços onde erros teriam maior impacto na segurança e integridade do sistema. Os testes foram organizados em três categorias correspondentes às camadas arquiteturais: domínio, serviço e repositório.

A cobertura de código foi medida utilizando a ferramenta nativa `go test` com a *flag* `-cover`, que implementa cobertura por instruções (do inglês, *statement coverage*). Esta métrica contabiliza a porcentagem de instruções do código-fonte que foram executadas durante os testes, inserindo contadores de instrumentação no código compilado. Embora existam métricas mais rigorosas como cobertura por ramificações (do inglês, *branch coverage*) ou cobertura por condições (do inglês, *condition coverage*), a cobertura por instruções oferece equilíbrio adequado entre precisão e praticidade para o escopo deste projeto.

Os testes unitários na camada de domínio validam regras de negócio fundamentais encapsuladas em objetos de valor e entidades. Esta categoria verifica formatação e validação de dados, algoritmos de *hash* criptográfico e transições de estado de chaves e usuários. Os testes garantem que apenas dados válidos sejam representados no sistema e que mudanças de estado inválidas sejam rejeitadas, prevenindo inconsistências no modelo de domínio.

Já na camada de serviço, os testes unitários utilizam *mocks* de dependências para isolar a lógica de negócio. Esta categoria verifica fluxos completos de casos de uso críticos: autenticação e autorização de usuários, criação de registros de auditoria durante transferências de chaves, aplicação de regras de autorização baseadas em papéis, validação de disponibilidade antes de operações e garantia de imutabilidade da trilha de auditoria.

Os testes de integração na camada de repositório executam validações contra banco de dados SQLite real em ambiente de teste isolado. Esta abordagem garante que operações de persistência funcionem corretamente em condições reais, verificando operações CRUD (do inglês, *Create*, *Read*, *Update*, *Delete*), ordenação e paginação de resultados, aplicação de filtros de busca e gerenciamento de transações com reversão automática em caso de erros.

Aspectos de segurança foram verificados de forma transversal nas três camadas: validação criptográfica de senhas com Argon2id, prevenção de operações não autorizadas através de controle de acesso baseado em papéis, garantia de imutabilidade dos registros de auditoria, validação de códigos QR com HMAC para prevenir falsificações e rejeição de operações sobre recursos em estados inválidos.

Camadas de apresentação (*handlers* HTTP e *templates*) e infraestrutura (*middleware*, configuração) foram deliberadamente excluídas da suíte automatizada, sendo validadas manualmente através de testes exploratórios. Testes automatizados dessas camadas demandariam volume extenso de código de teste e ferramentas adicionais de simulação de navegador que extrapolavam o escopo do projeto. A execução paralela dos testes, habilitada nativamente pelo Go, garante *feedback* rápido durante o desenvolvimento.

4.6 *Deployment*

O sistema foi projetado com abstração de hardware através da interface `Controller`, permitindo diferentes estratégias de *deployment* conforme o ambiente. Esta interface define contratos para operações de controle físico (abertura e fechamento de fechaduras), desacoplando a lógica de negócio da implementação específica de hardware. Durante o desenvolvimento, utiliza-se a implementação `NoOpController`, um *stub* que simula operações sem hardware real, permitindo testes da lógica de aplicação de forma independente.

Para validação do conceito de integração com hardware, foi desenvolvida a implementação `EmbeddedController` que permite integração com protótipo físico baseado em Raspberry Pi. Este protótipo, desenvolvido pelos estudantes mencionados na introdução, integra três componentes principais: tela para interface do usuário (quiosque de auto-atendimento), webcam para leitura de códigos QR das chaves e servo motor controlado via GPIO que atua como mecanismo de travamento. A implementação `EmbeddedController` comunica-se com o GPIO através de *scripts* Python, enviando sinais elétricos para controle do servo motor.

O sistema completo, integrando a aplicação Web desenvolvida neste trabalho com o protótipo físico, foi apresentado na modalidade Amostra de Projetos do 6º SIMPIF (Simpósio de Pesquisa, Inovação e Pós-Graduação do IFPB), realizado de 23 a 25 de julho de 2025 em João Pessoa, onde conquistou segundo lugar.

5 RESULTADOS

Este capítulo apresenta os resultados obtidos durante o desenvolvimento e validação do *KeyCode Manager*. São demonstradas as interfaces do sistema implementado, o protótipo físico construído e apresentado no 6º SIMPIF e os resultados da validação da estratégia de testes. A apresentação dos resultados segue a organização proposta na metodologia, mostrando tanto os aspectos de *software* quanto a integração com hardware.

5.1 Interface do sistema

As interfaces desenvolvidas materializam as funcionalidades descritas na Seção 4.4.3, oferecendo aos usuários e administradores os meios de interação com o sistema. A aplicação Web foi construída priorizando simplicidade operacional e responsividade, adaptando-se a diferentes tamanhos de tela. Ambas as interfaces oferecem suporte a temas claro e escuro, com tema inicial determinado pelas preferências do sistema operacional do usuário e opção de alternância manual, sendo as capturas de tela apresentadas em tema claro para melhor legibilidade em formato impresso. Esta seção apresenta as principais telas implementadas, organizadas em dois grupos: interface do quiosque para autoatendimento e painel administrativo para gestão do sistema.

5.1.1 Interface do quiosque

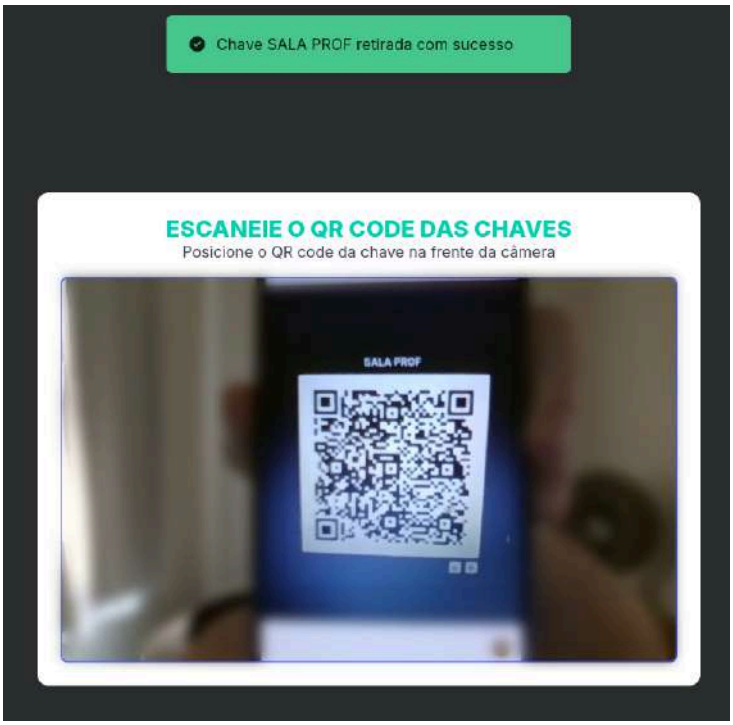
A interface do quiosque representa o ponto de interação principal entre usuários finais e o sistema, projetada para operação em modo de autoatendimento. A Figura 5 apresenta a tela inicial do quiosque, onde usuários inserem CPF e PIN para autenticação. Esta interface foi otimizada para ser usada apenas com o teclado numérico, com os campos de login sendo focados automaticamente ao serem preenchidos e a requisição de login sendo feita assim que tudo for preenchido. Em caso de erro de autenticação, o sistema exibe mensagem de erro informativa, permitindo nova tentativa imediata sem necessidade de interação adicional.

Figura 5: Interface de autenticação do quiosque para retirada de chave

Fonte: Elaborado pelo autor (2025)

Após autenticação bem-sucedida, o sistema apresenta a interface de leitura de códigos QR, conforme ilustrado na Figura 6. Esta tela mostra a webcam do dispositivo, permitindo que o usuário posicione o código QR da chave física em frente à câmera. O sistema processa o código em tempo real, valida sua autenticidade através do HMAC conforme descrito na Seção 4.4.1 e registra a operação de retirada ou devolução. A figura também apresenta a notificação de sucesso exibida ao usuário após a conclusão da operação, informando qual chave foi retirada.

Figura 6: Interface de leitura de código QR com webcam ativa e confirmação de retirada



Fonte: Elaborado pelo autor (2025)

A Figura 7 demonstra o fluxo de devolução de chave, que segue um processo diferente: o usuário não autenticado posiciona o código QR em frente à câmera, que automaticamente checa o último usuário que retirou a chave e assume ser o mesmo que está devolvendo. A notificação de sucesso confirma ao usuário que a devolução foi registrada corretamente. Esta decisão de projeto prioriza agilidade operacional, permitindo devoluções sem etapa de autenticação, mas introduz possibilidade de devoluções falsas onde um terceiro poderia registrar devolução em nome do responsável real. O risco foi considerado aceitável para o contexto institucional, onde o ambiente controlado e a presença de funcionários reduzem a probabilidade de uso malicioso. Caso a instituição identifique necessidade de maior rigor, a arquitetura do sistema permite adicionar autenticação obrigatória nas devoluções sem alterações estruturais significativas.

Figura 7: Confirmação de devolução de chave no quiosque

Fonte: Elaborado pelo autor (2025)

5.1.2 Painel administrativo

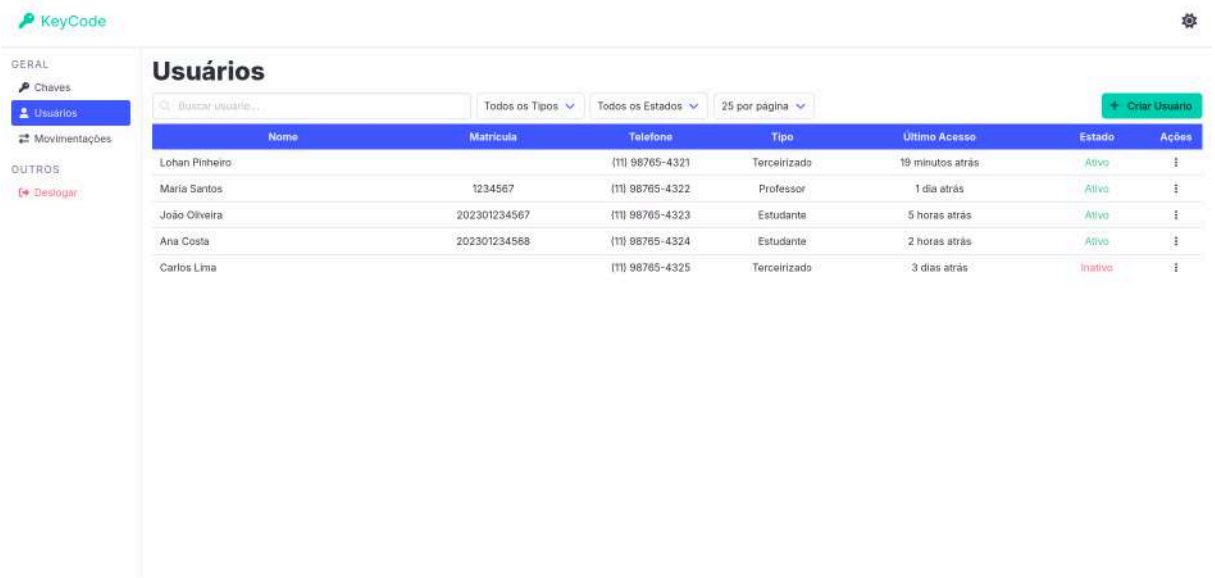
O painel administrativo oferece aos gestores do sistema controle completo sobre usuários, chaves e registros de auditoria. A Figura 8 apresenta a tela de autenticação administrativa, que utiliza os mesmos mecanismos de segurança descritos na Seção 4.4.2, porém destinada a usuários com privilégios administrativos.

Figura 8: Tela de autenticação do painel administrativo

Fonte: Elaborado pelo autor (2025)

A gestão de usuários é realizada através da interface apresentada na Figura 9, que lista todos os usuários cadastrados com informações como matrícula, telefone, tipo (professor, estudante ou terceirizado), último acesso e estado atual (ativo ou inativo). A interface oferece recursos de busca, filtragem por tipo e estado, e paginação para gerenciar grandes volumes de registros.

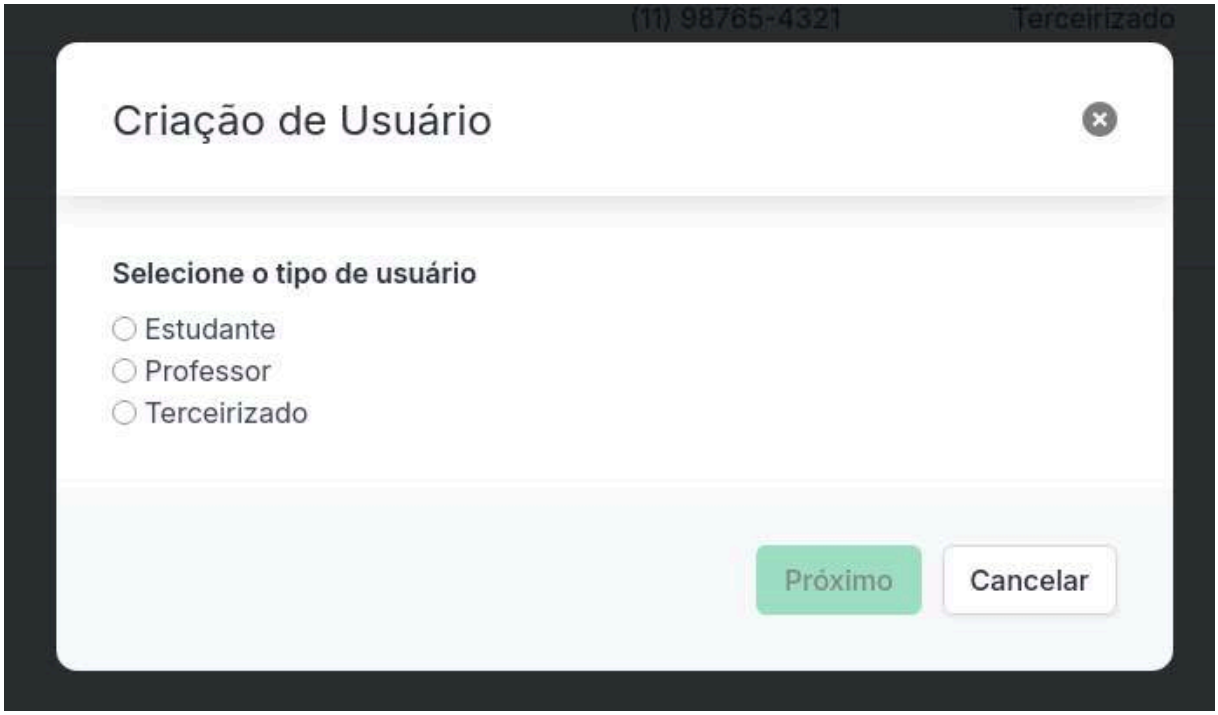
Figura 9: Painel de gerenciamento de usuários do sistema



Fonte: Elaborado pelo autor (2025)

O cadastro de novos usuários é conduzido através de um processo em duas etapas. A Figura 10 apresenta a primeira etapa, onde o administrador seleciona o tipo de usuário a ser criado. Esta escolha determina os campos e as validações aplicados na etapa seguinte.

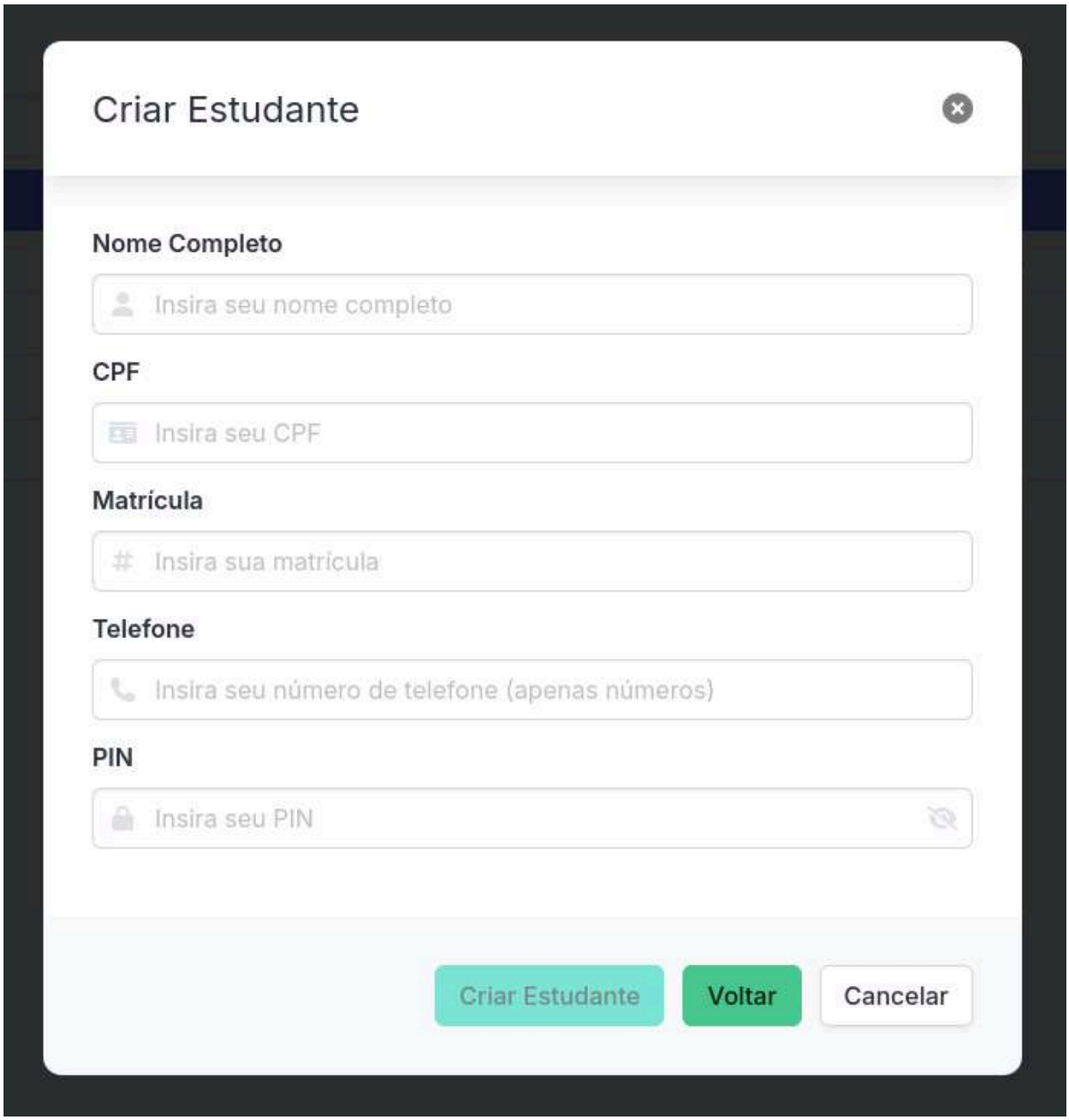
Figura 10: Primeira etapa do cadastro de usuário: seleção de tipo



Fonte: Elaborado pelo autor (2025)

A Figura 11 ilustra a segunda etapa na criação de um estudante, onde são inseridos os dados do usuário, incluindo nome completo, CPF, telefone, matrícula e PIN. O sistema aplica validações em todos os campos para garantir consistência dos dados, conforme descrito na Seção 4.4.2. O cadastro de usuários é realizado manualmente pelos administradores, não havendo integração com o SUAP (Sistema Unificado de Administração Pública) nesta versão do sistema. A integração automática com o SUAP para importação de dados de usuários representa oportunidade de evolução futura, que seria especialmente útil para reduzir o trabalho administrativo de cadastro inicial e manter sincronização com a base institucional.

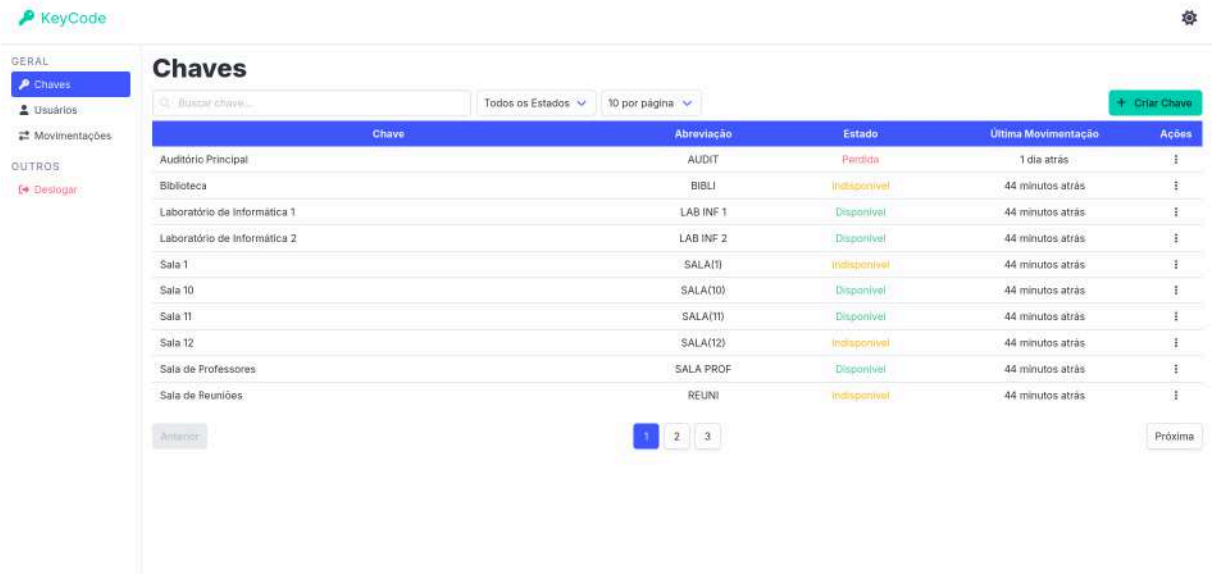
Figura 11: Segunda etapa do cadastro de usuário: inserção de dados



Fonte: Elaborado pelo autor (2025)

O gerenciamento de chaves segue estrutura similar ao de usuários. A Figura 12 apresenta a listagem de chaves cadastradas, exibindo localização, abreviação, estado atual (disponível, indisponível ou perdida) e última movimentação registrada. A interface oferece recursos de busca, filtragem por estado e paginação.

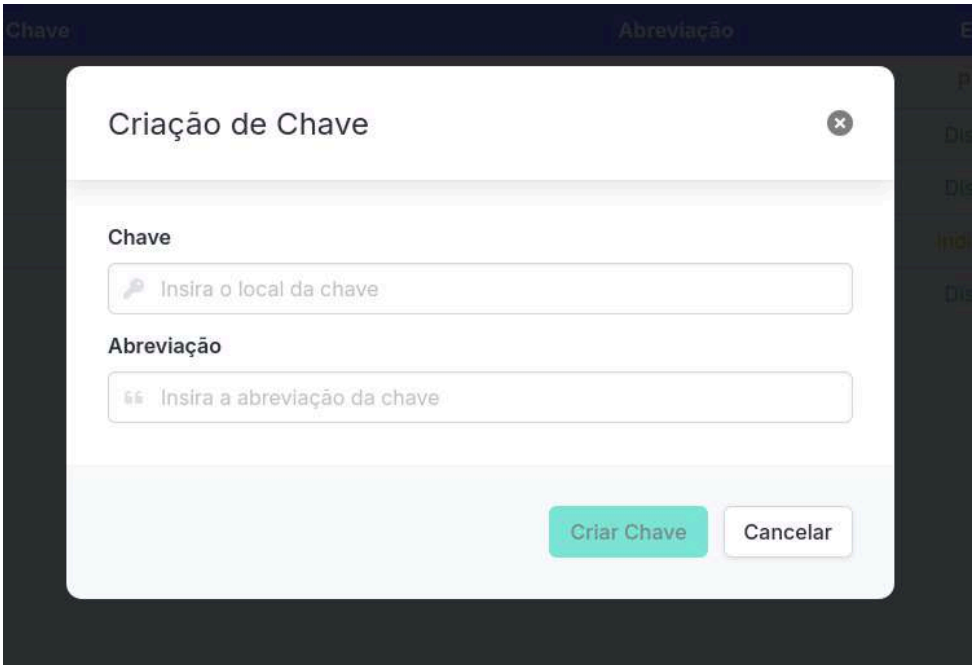
Figura 12: Painel de gerenciamento de chaves do sistema



Fonte: Elaborado pelo autor (2025)

A criação de novas chaves é realizada através do formulário apresentado na Figura 13, onde o administrador insere o nome da localização e a abreviação da chave. O sistema gera automaticamente um código QR único contendo identificador seguro e HMAC para autenticação, conforme descrito na Seção 4.4.1.

Figura 13: Formulário de cadastro de nova chave



Fonte: Elaborado pelo autor (2025)

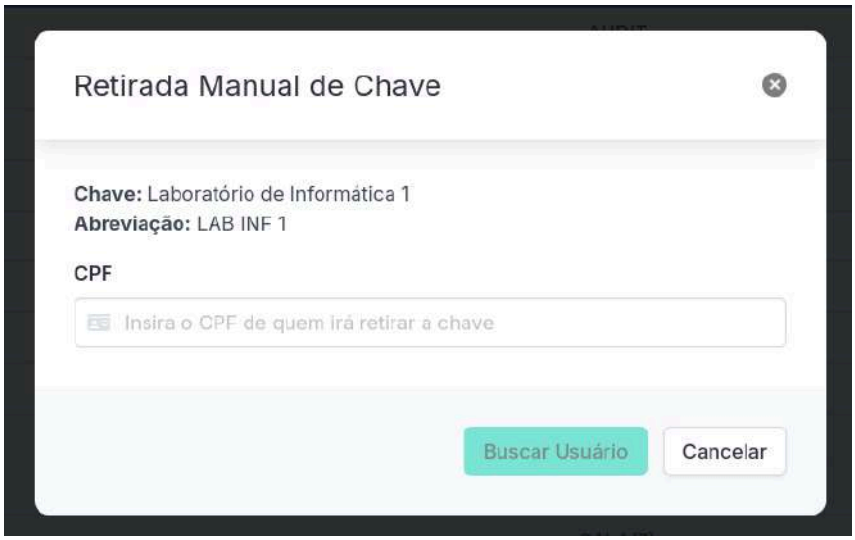
Após criação da chave, o administrador pode visualizar, baixar e imprimir o código QR gerado, conforme ilustrado na Figura 14. Este código deve ser afixado na chave física correspondente, permitindo sua identificação durante operações de retirada e devolução no quiosque.

Figura 14: Visualização do código QR gerado para a chave

Fonte: Elaborado pelo autor (2025)

Além das operações automatizadas no quiosque, o sistema oferece funcionalidade de retirada manual de chaves por administradores. A Figura 15 apresenta a interface inicial de retirada, onde o administrador seleciona a chave e insere o CPF do usuário que a receberá. Após buscar o usuário, o sistema exibe os dados para confirmação, conforme ilustrado na Figura 16, permitindo que o administrador registre a transferência manual da chave. Esta funcionalidade é útil em situações onde o quiosque não está disponível ou quando há necessidade de intervenção administrativa direta.

Figura 15: Interface de retirada manual de chave: seleção de usuário



Fonte: Elaborado pelo autor (2025)

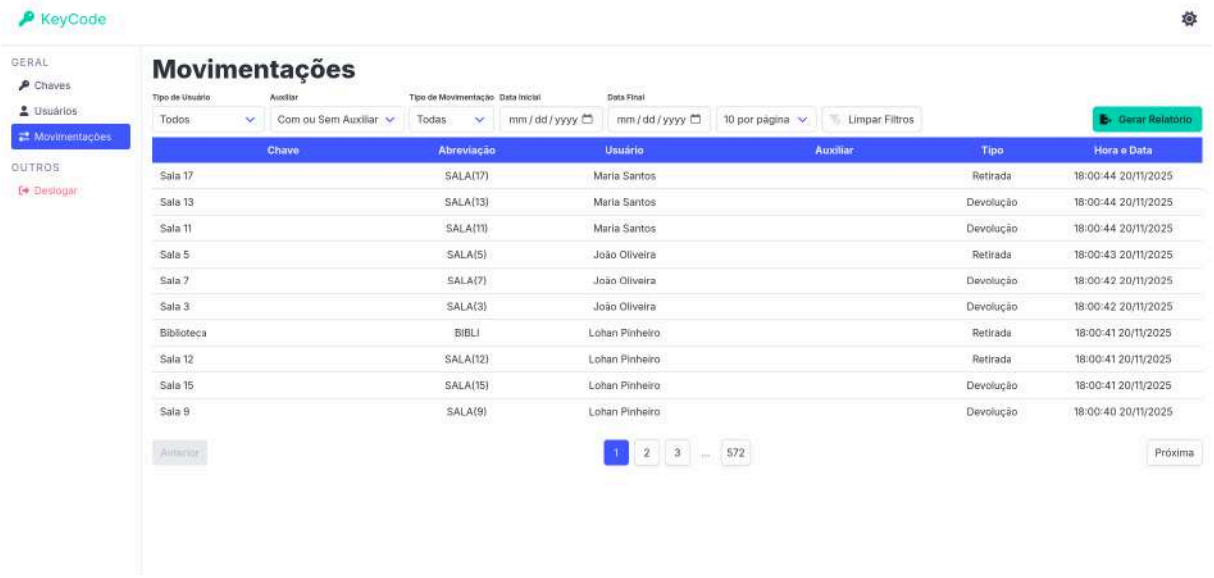
Figura 16: Interface de retirada manual de chave: confirmação de transferência

Fonte: Elaborado pelo autor (2025)

A trilha de auditoria completa do sistema é acessível através da interface de movimentações, apresentada na Figura 17. Esta tela lista todos os registros de retirada e devolução de chaves dos últimos 2 anos por padrão, identificando a chave movimentada, o usuário responsável, o auxiliar (quando aplicável), o tipo de operação e a data e hora exatas. A interface oferece recursos avançados de filtragem por tipo de usuário, presença de auxiliar, tipo de movimentação e intervalo de datas, permitindo análises detalhadas do histórico de utilização do sistema e identificação rápida

do responsável atual por cada chave. Adicionalmente, a interface disponibiliza funcionalidade de geração de relatórios em PDF, permitindo exportar os dados filtrados para análise externa. Um exemplo da primeira página de um relatório gerado pode ser visualizado no Apêndice A.

Figura 17: Painel de visualização da trilha de auditoria de movimentações



Fonte: Elaborado pelo autor (2025)

As interfaces apresentadas implementam todas as funcionalidades especificadas nos objetivos do trabalho, oferecendo controle completo sobre o ciclo de vida das chaves físicas através de aplicação Web responsiva.

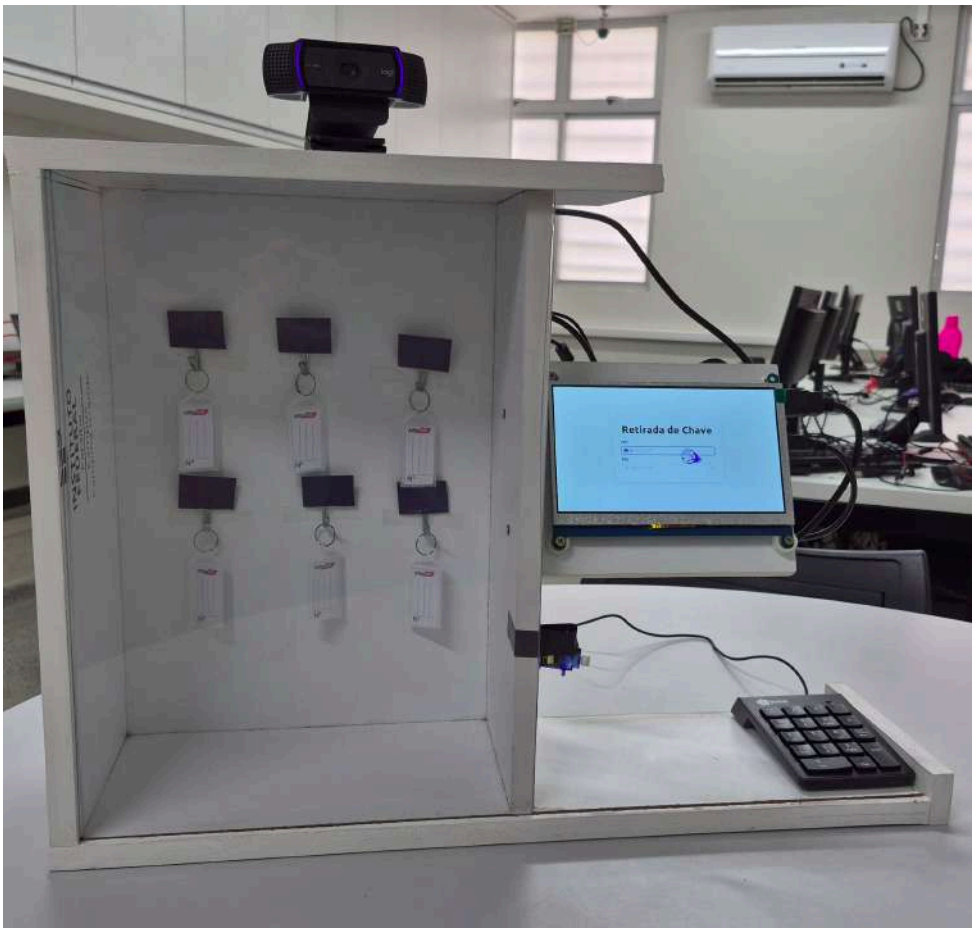
5.2 Protótipo físico

Conforme descrito na Seção 4.6, o *KeyCode Manager* foi desenvolvido como parte de um projeto maior envolvendo múltiplos estudantes. Este trabalho concentrou-se exclusivamente no desenvolvimento da aplicação Web, enquanto outros estudantes foram responsáveis pela construção do protótipo físico de hardware. O sistema completo, integrando software e hardware, foi apresentado publicamente durante o 6º SIMPIF (Simpósio de Pesquisa, Inovação e Pós-Graduação do IFPB), realizado entre 23 e 25 de julho de 2025 em João Pessoa, onde conquistou segundo lugar na modalidade Amostra de Projetos.

A Figura 18 apresenta a visão frontal do protótipo, mostrando o armário de chaves físicas com seis posições de armazenamento, cada uma identificada por código QR único. À direita, posiciona-se a tela do terminal do quiosque executando a interface de retirada de chave, e acima

do armário localiza-se a webcam responsável pela leitura dos códigos QR. Na base do sistema, o servo motor azul atua como mecanismo de travamento controlado via GPIO do Raspberry Pi.

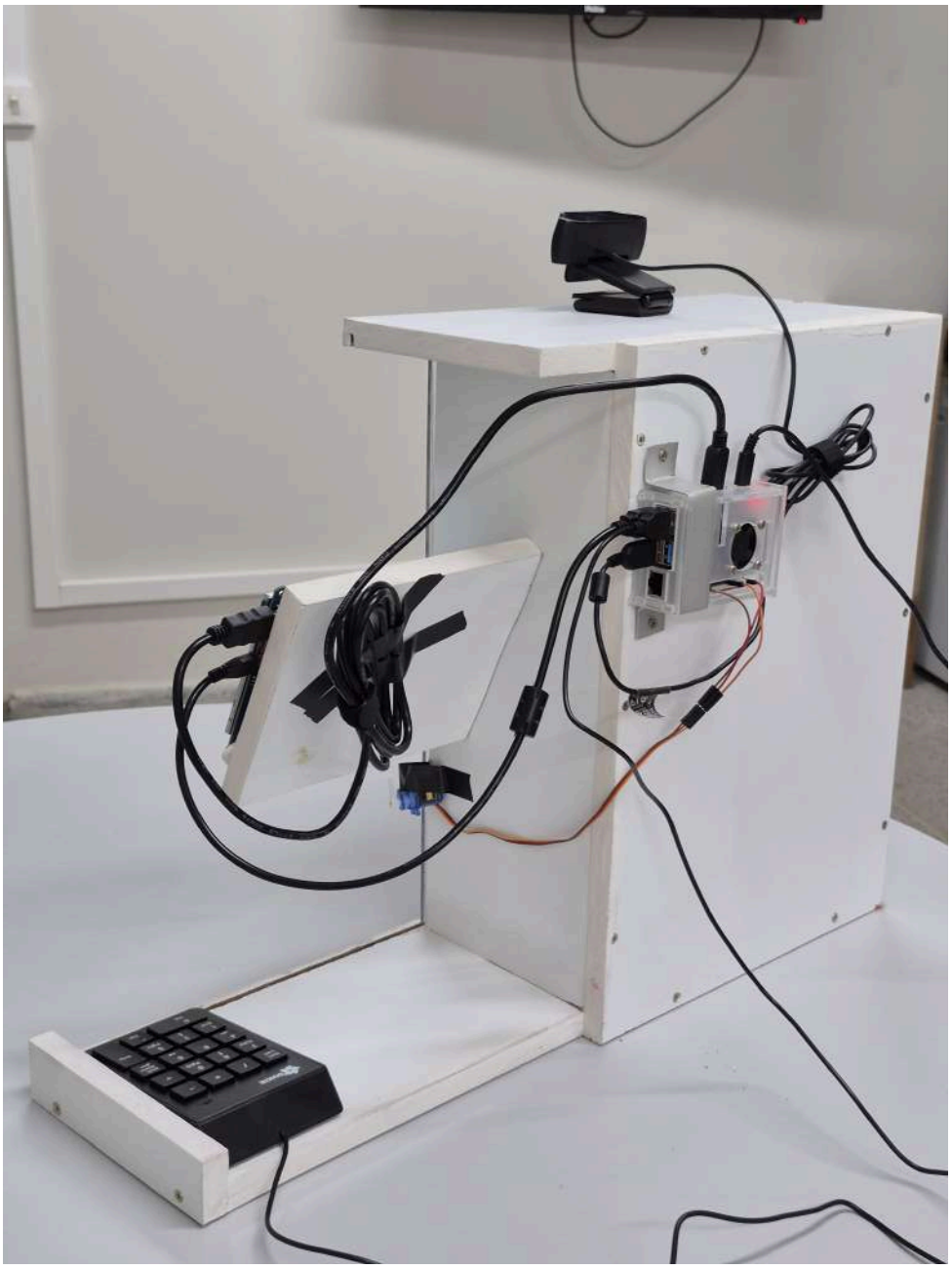
Figura 18: Visão frontal do protótipo mostrando armário de chaves, terminal e webcam



Fonte: Elaborado pelo autor (2025)

A Figura 19 revela a organização interna dos componentes eletrônicos do protótipo. No centro superior, o Raspberry Pi processa as requisições da aplicação Web e controla o servo motor através de GPIO. A fiação conecta fonte de alimentação, servo motor e periféricos, demonstrando a implementação física do controlador embarcado. Esta visão evidencia a separação entre os componentes de processamento, interface e atuação mecânica.

Figura 19: Visão posterior do protótipo mostrando Raspberry Pi e fiação



Fonte: Elaborado pelo autor (2025)

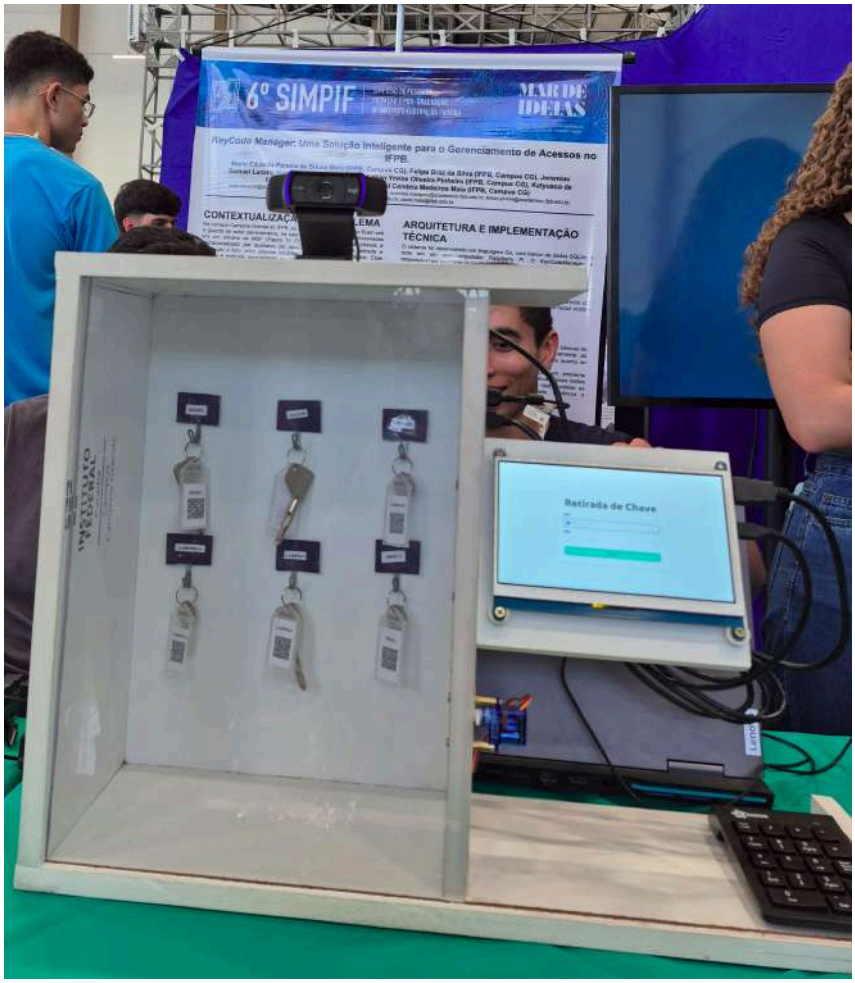
Durante os testes realizados na preparação para o SIMPIF e durante o evento, o sistema integrado operou conforme especificado: usuários autenticavam-se no terminal através da interface desenvolvida, escaneavam códigos QR das chaves, e o hardware respondia adequadamente liberando ou travando o acesso ao armário. A integração entre software e hardware demonstrou a viabilidade prática da solução completa para uso em ambientes reais.

O *feedback* recebido de visitantes, incluindo professores, estudantes e profissionais da área, foi positivo quanto à usabilidade da aplicação Web e aplicabilidade da solução integrada.

A conquista do segundo lugar na categoria pelo projeto completo demonstra o reconhecimento da qualidade técnica do sistema e sua relevância para o contexto institucional.

A Figura 20 apresenta o sistema completo durante a apresentação no 6º SIMPIF, mostrando o protótipo em operação no contexto do evento acadêmico, com o pôster explicativo ao fundo detalhando a arquitetura e funcionalidades do sistema.

Figura 20: Apresentação do sistema completo no 6º SIMPIF



Fonte: Elaborado pelo autor (2025)

5.3 Cobertura de testes

Conforme a estratégia de testes descrita na Seção 4.5, a suíte de testes automatizados foi executada utilizando a ferramenta `go test` com instrumentação de cobertura por instruções (*statement coverage*). A Tabela 3 apresenta os resultados obtidos para cada camada arquitetural.

Tabela 3: Resultados de cobertura de testes por camada arquitetural

Camada	**Casos de Teste**	**Cobertura**
Domínio	223	78,4%
Serviço	73	79,9%
Repositório	128	83,1%
Total	**424**	**80,5%**

Fonte: Elaborado pelo autor (2025)

Os resultados demonstram cobertura consistente nas três camadas centrais do sistema, com média de 80,5%. A camada de repositório apresentou a maior cobertura (83,1%), refletindo a natureza determinística das operações de persistência. As camadas de domínio e serviço alcançaram coberturas próximas (78,4% e 79,9% respectivamente), superando o objetivo típico de 70% recomendado para sistemas críticos.

A distribuição dos casos de teste reflete a complexidade de cada camada: o domínio concentra o maior número de testes (223) devido à quantidade de objetos de valor e regras de validação, enquanto o repositório apresenta volume expressivo (128) pela necessidade de testar múltiplas operações de banco de dados. A camada de serviço, embora com menor quantidade de testes (73), cobre fluxos de negócio complexos que orquestram múltiplas operações.

5.4 Validação de estabilidade

Para validar a estabilidade do sistema sob operação contínua, foram realizados cinco testes automatizados utilizando um *script* desenvolvido especificamente para simular o comportamento de múltiplos usuários (apresentado no Apêndice B). O *script* de teste, implementado em Bash, automatiza requisições HTTP diretas à API do sistema, enviando as mesmas requisições que a interface Web geraria durante operações reais de retirada e devolução de chaves. Os identificadores de chaves são obtidos através de consultas diretas ao banco de dados, simulando a seleção que ocorreria após a leitura de códigos QR na interface real. Esta abordagem permite validar as camadas de lógica de negócio, repositório e segurança da aplicação sem dependência de hardware ou interface gráfica.

Cada execução consistiu em 200 ciclos sequenciais, onde cada ciclo representa uma sessão completa de usuário selecionado aleatoriamente dentre quatro usuários cadastrados. O

comportamento simulado busca refletir cenários realistas de uso institucional: quando um usuário autentica, o sistema verifica através da trilha de auditoria se ele possui chaves não devolvidas de sessões anteriores. Caso positivo, essas chaves são devolvidas antes de novas operações, e há 50% de probabilidade de o usuário retirar novas chaves na mesma sessão. Caso o usuário não possua chaves pendentes, ele retira de uma a três chaves selecionadas aleatoriamente do conjunto de 26 chaves cadastradas. As chaves retiradas permanecem com o usuário após *logout*, sendo devolvidas apenas quando o mesmo usuário autentica novamente em ciclo posterior. Este padrão permite avaliar o gerenciamento correto de estado entre sessões, a consistência da trilha de auditoria e a capacidade do sistema de rastrear responsáveis por chaves ao longo do tempo.

A Tabela 4 apresenta os resultados consolidados das cinco execuções. Todos os testes foram realizados em ambiente de produção com o servidor compilado em modo *release*, executando em máquina equipada com processador Intel Core i5-12450H de 12ª geração (8 núcleos, 12 *threads*), 32 GB de memória RAM e sistema operacional Linux Pop!_OS com kernel na versão 6.16.3. É importante destacar que o objetivo dos testes foi validar a corretude do sistema e sua estabilidade sob operação contínua, e não avaliar desempenho absoluto, uma vez que o ambiente de execução difere do hardware onde o sistema seria implantado em produção.

Tabela 4: Resultados dos testes de estabilidade sob operação contínua

Execução	Duração	Operações	Retiradas	Devoluções	Taxa de Sucesso
1	4m 47s	529	266	263	100%
2	4m 50s	542	270	272	100%
3	4m 42s	520	259	261	100%
4	4m 44s	525	265	260	100%
5	4m 42s	524	261	263	100%
Média	**4m 45s**	**528**	**264,2**	**263,8**	**100%**

Fonte: Elaborado pelo autor (2025)

Os resultados demonstram estabilidade consistente do sistema durante operação contínua. As cinco execuções completaram 200 ciclos cada sem falhas, totalizando 1000 sessões de usuário e 2640 operações (1321 retiradas e 1319 devoluções) com taxa de sucesso de 100% em todas as requisições. A duração média de 4 minutos e 45 segundos por execução de 200 ciclos resulta

em *throughput* médio de 1,85 operações por segundo. O tempo médio de resposta observado foi de 13,41 milissegundos por operação, demonstrando latência consistentemente baixa. Vale ressaltar que estes números refletem a execução em ambiente com recursos computacionais adequados para servidor dedicado, representando o cenário de implantação em produção onde a aplicação Web seria hospedada separadamente do Raspberry Pi utilizado no protótipo físico.

A ausência completa de falhas durante os testes indica estabilidade das camadas de validação, gerenciamento de estado de sessão e integridade das operações de banco de dados sob carga sustentada. A variação mínima entre execuções (duração entre 4m 42s e 4m 50s, operações entre 520 e 542) demonstra comportamento previsível do sistema. O número ligeiramente diferente de operações entre execuções decorre da natureza aleatória da simulação: cada ciclo retira quantidade aleatória de chaves (1 a 3), e a decisão de retirar novas chaves após devolver chaves pendentes ocorre com 50% de probabilidade, criando padrões de uso variados mas realistas.

Estes resultados validam a adequação da arquitetura implementada para operação em ambiente institucional real, onde o sistema enfrentaria carga significativamente menor que a testada. Em cenário típico de uso no IFPB campus Campina Grande, o sistema processaria dezenas de operações diárias distribuídas ao longo do horário de funcionamento, ordem de grandeza inferior à carga contínua e concentrada aplicada durante os testes.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho abordou o problema do gerenciamento manual de chaves físicas no Instituto Federal de Educação da Paraíba — campus Campina Grande — caracterizado por processos baseados em registros em papel, ausência de controle automatizado de acesso e dificuldades na rastreabilidade de responsáveis. Conforme apresentado no Capítulo 1, esses processos manuais apresentam taxas de erro significativas e geram vulnerabilidades operacionais que comprometem a eficiência e segurança institucional.

A solução desenvolvida consistiu no *KeyCode Manager*, uma aplicação Web que utiliza códigos QR para identificação de chaves físicas, oferecendo interfaces para autoatendimento e painel administrativo para gestão completa do sistema. A arquitetura hexagonal implementada, descrita no Capítulo 4, permite flexibilidade na integração com diferentes configurações de hardware. O sistema implementa múltiplas camadas de segurança, incluindo autenticação robusta com Argon2id, sessões seguras, validação HMAC de códigos QR, proteções contra ataques comuns em nível de infraestrutura e trilha de auditoria imutável.

Dentre os resultados alcançados destacam-se: a implementação completa da aplicação Web; a validação através de suíte de 424 testes automatizados com cobertura média superior a 80% nas camadas centrais da arquitetura, demonstrando qualidade e confiabilidade da lógica de negócio; a validação de estabilidade através de testes automatizados que simularam 1000 sessões de usuário com taxa de sucesso de 100%; a validação manual extensiva das interfaces e dos fluxos de usuário; a construção da solução fundamentada em discussões e entrevistas com funcionários do setor de chaves do IFPB, garantindo que o sistema atende necessidades reais do processo operacional; e a apresentação no 6º SIMPIF, onde o sistema integrado conquistou segundo lugar na modalidade Amostra de Projetos, demonstrando viabilidade técnica da integração entre aplicação Web e controle físico de acesso.

6.1 Contribuições

Este trabalho apresenta contribuições nos âmbitos acadêmico, técnico e institucional, demonstrando a aplicação prática de tecnologias atuais na solução de problemas reais de gestão organizacional.

A contribuição acadêmica reside na pesquisa aplicada que demonstra solução prática para controle de acesso físico em instituições educacionais. A integração de tecnologia de códigos QR com sistemas Web para gerenciamento institucional de chaves físicas constitui caso

de estudo aplicável a instituições similares que enfrentam desafios de gestão de ativos físicos. A documentação detalhada do processo de desenvolvimento, incluindo decisões arquiteturais, implementação de funcionalidades de segurança e estratégias de teste, serve como referência para projetos similares, fornecendo *insights* valiosos sobre construção de sistemas robustos mantendo foco em baixo custo e simplicidade operacional.

A contribuição técnica manifesta-se através da aplicação Web de código aberto desenvolvida com arquitetura hexagonal utilizando Go, HTMX e Alpine.js. O padrão de abstração de hardware implementado permite flexibilidade nas estratégias de *deployment*, desde execução direta em dispositivos embarcados até arquiteturas distribuídas com múltiplos controladores dedicados. A prova de conceito demonstra viabilidade técnica da integração entre sistemas Web e controle físico de acesso através de códigos QR. A implementação abrangente de segurança seguindo boas práticas, incluindo defesa em profundidade com proteções em múltiplas camadas, demonstra como construir sistemas seguros para ambientes institucionais. A suíte de testes automatizados com 424 casos de teste e cobertura superior a 80% nas camadas de lógica de negócio estabelece fundação sólida para a manutenção e a evolução contínua do sistema.

A contribuição institucional oferece solução viável para problema real identificado no IFPB campus Campina Grande. O sistema elimina dependência de registros em papel e processos manuais suscetíveis a erros, substituindo-os por controle digital automatizado. A trilha de auditoria imutável melhora significativamente a prestação de contas através de rastreabilidade completa de todas as operações, permitindo identificação precisa de responsáveis e reconstrução de histórico de utilização. A validação bem-sucedida no 6º SIMPIF e o reconhecimento através do segundo lugar na categoria demonstram maturidade técnica da solução e estabelecem fundação para futuro *deployment* em produção no ambiente institucional.

6.2 Limitações

Apesar dos resultados positivos alcançados, o trabalho apresenta limitações que devem ser reconhecidas e que indicam oportunidades para trabalhos futuros.

O protótipo físico de hardware desenvolvido pelos colaboradores do projeto, embora funcional para validação do conceito, não está preparado para ambiente de produção. Conforme discutido na Seção 4.6, a execução da aplicação Web completa diretamente no Raspberry Pi apresenta restrições de desempenho, especialmente sob múltiplos acessos simultâneos. A arquitetura de integração testada serve como prova de conceito, mas ambientes de produção

se beneficiariam de arquitetura distribuída com servidor Web centralizado e múltiplos controladores de hardware dedicados.

O sistema encontra-se em estágio de prova de conceito e não foi ainda implantado em produção no IFPB. Embora a validação no SIMPIF tenha demonstrado funcionamento adequado em ambiente controlado, o *deployment* em produção requer considerações adicionais como manutenção contínua, treinamento de usuários, procedimentos operacionais e integração com sistemas institucionais existentes.

A estratégia de testes adotada, conforme justificado na Seção 4.5, concentrou esforços nas camadas de lógica de negócio, resultando em cobertura limitada para as camadas de apresentação e *handlers* HTTP. Esta decisão pragmática priorizou qualidade onde erros teriam maior impacto na segurança e integridade do sistema, sendo as demais camadas validadas através de testes exploratórios manuais.

A integração com hardware foi testada exclusivamente com o protótipo específico baseado em Raspberry Pi desenvolvido pelos colaboradores. Embora a abstração implementada permita diferentes configurações de hardware, validações adicionais seriam necessárias para outros cenários de *deployment*, especialmente considerando variações em controladores de servo motor, leitores de QR code e configurações de GPIO.

6.3 Trabalhos futuros

As limitações identificadas e a experiência adquirida durante o desenvolvimento apontam diversas direções promissoras para evolução do *KeyCode Manager*.

No âmbito de *deployment* e produção, antes da implantação definitiva, faz-se necessária uma fase de validação mais abrangente no ambiente real de operação do IFPB campus Campina Grande. Esta validação deve incluir testes com usuários reais do setor de chaves, simulação de cenários operacionais diversos, identificação de requisitos não previstos durante o desenvolvimento e coleta de *feedback* dos funcionários que operarão o sistema diariamente. Após esta validação e eventuais ajustes necessários, a prioridade consiste na implantação do sistema em ambiente de produção, substituindo definitivamente os processos manuais atuais. Esta implantação deve considerar otimizações de desempenho para cargas de trabalho reais e adoção de arquitetura distribuída conforme discutido na Seção 4.6, separando servidor Web centralizado de controladores de hardware dedicados. A validação bem-sucedida neste campus estabeleceria fundação para expansão futura a outras unidades do IFPB, ou instituições que lidam com artefatos físicos de maneira geral, adaptando o sistema às particularidades de cada localidade.

A automação completa do processo de gerenciamento de chaves representa evolução natural do sistema. A integração com armários eletrônicos de chaves, conforme discutido na Seção 3.1.5, eliminaria necessidade de intervenção humana na retirada física das chaves. Neste cenário, após autenticação e escaneamento do código QR no quiosque, o armário eletrônico liberaria automaticamente a chave correspondente, com sistema automatizado de ordenação e organização mantendo as chaves organizadas internamente.

As funcionalidades adicionais necessárias para transição do protótipo validado no SIMPIF para sistema em produção podem ser organizadas em três categorias de prioridade. As funcionalidades críticas para implantação em produção incluem: verificar possibilidade de integração com o SUAP, que em caso de sucesso eliminaria o cadastro manual de usuários e sincronizando automaticamente dados de matrícula, vinculação institucional e situação acadêmica, reduzindo carga administrativa inicial e garantindo que usuários desligados da instituição tenham acesso revogado automaticamente, atualização em tempo real do painel administrativo utilizando *Server-Sent Events* (SSE), permitindo que administradores visualizem mudanças de status de chaves instantaneamente conforme operações ocorrem nos quiosques, notificações automáticas alertando administradores sobre devoluções atrasadas, identificando chaves retidas além do prazo esperado e permitindo ação proativa para recuperação, e transferência direta entre usuários, removendo a necessidade do quiosque quando alguém deseja entregar uma chave para outra pessoa cadastrada no sistema.

Os aprimoramentos operacionais que aumentariam a conveniência sem serem estritamente necessários incluem: transferência de chaves entre usuários via interface Web, permitindo que usuários escaneiem códigos QR através da câmera do navegador em dispositivos móveis, facilitando cenários onde chaves precisam ser repassadas sem retorno ao quiosque, e capacidades avançadas de filtragem e geração de relatórios personalizados, permitindo análise de padrões de utilização e identificação de anomalias operacionais.

A evolução de longo prazo do sistema, após estabelecimento sólido em produção, poderia incluir: painel de análise avançada com visualizações de métricas de utilização, identificando chaves subutilizadas ou gargalos operacionais através de análise temporal, e expansão do sistema para outras unidades do IFPB, aproveitando a experiência operacional do campus Campina Grande e adaptando o sistema às particularidades de cada localidade.

A qualidade e confiabilidade do sistema se beneficiariam de expansão da estratégia de testes. Testes *end-to-end* automatizados para interfaces *frontend* garantiriam funcionamento correto de fluxos completos de usuário. Testes unitários para camada de *handlers* HTTP

complementariam a cobertura existente nas camadas de domínio, serviço e repositório. Testes de carga com simulação de tráfego em nível de produção validariam capacidade do sistema sob demanda real.

A sustentabilidade a longo prazo requer estabelecimento de plano de manutenção contínua, incluindo procedimentos para atualizações de segurança, correção de *bugs* e adição de funcionalidades. Documentação abrangente para administradores de sistema facilitaria operação e resolução de problemas. Materiais de treinamento para funcionários do IFPB garantiriam utilização efetiva do sistema e reduziriam curva de aprendizado durante adoção.

Estas direções futuras não apenas abordam as limitações identificadas, mas também expandem significativamente o valor e aplicabilidade do sistema, estabelecendo caminho claro para evolução contínua da solução.

7 REFERÊNCIAS

Are Manual Workflows Costing Your Enterprise?. **Unqork**, 8 jul. 2021. Disponível em: https://unqork.com/resource-center/blogs/are-manual-workflows-costing-your-enterprise/. Acesso em: 29 maio 2025.

BRASIL. Gabinete de Segurança Institucional da Presidência da República. Instrução Normativa nº 1, de 13 de junho de 2008. Disciplina a Gestão de Segurança da Informação e Comunicações na Administração Pública Federal. **Diário Oficial da União**, Brasília, DF, 16 jun. 2008. Disponível em: https://antigo.mctic.gov.br/mctic/opencms/legislacao/outros_atos/instrucoes_normativas/Instrucao_Normativa_GSIPR_n_1_de_13062008.html. Acesso em: 29 out. 2025.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. Dispõe sobre a proteção de dados pessoais e altera a Lei nº 12.965, de 23 de abril de 2014 (Marco Civil da Internet). **Diário Oficial da União**, Brasília, DF, 15 ago. 2018. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato 2015-2018/2018/lei/l13709.htm. Acesso em: 29 out. 2025.

Characteristics of modern web applications. **Microsoft Learn**, 21 set. 2022. Disponível em: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/modern-web--applications-characteristics. Acesso em: 29 maio 2025.

COCKBURN, Alistair. **Hexagonal Architecture**. HaT Technical Report 2005.02, 04 abr 2005. Disponível em: https://alistair.cockburn.us/hexagonal-architecture. Acesso em: 21 nov. 2025.

Error correction feature. **DENSO WAVE**, [s.d.]a. Disponível em: https://www.qrcode.com/en/about/error_correction.html. Acesso em: 29 maio 2025.

FERRAIOLO, Hildegard; GHADIALI, Nabil; MOHLER, Jason; JOHNSON, Vincent; BRADY, Steven. **Guidelines for the Use of PIV Credentials in Facility Access**. NIST Special Publication 800-116 Revision 1. Gaithersburg, MD: National Institute of Standards and Technology, June 2018. DOI: 10.6028/NIST.SP.800-116r1.

GROSS, Carson. Hypermedia-Driven Applications. **HTMX**, 6 fev. 2022. Disponível em: https://htmx.org/essays/hypermedia-driven-applications/. Acesso em: 28 setembro 2025.

HIETALA, Jim. Network Security - A Guide for Small and Mid-sized Businesses. **SANS**, 26 jan. 2005. Disponível em: https://www.sans.org/reading-room/whitepapers/basics/network--security-guide-small-mid-sized-businesses-1539. Acesso em: 29 maio 2025.

History of QR Code. **DENSO WAVE**, [s.d.]b. Disponível em: https://www.qrcode.com/en/history/. Acesso em: 29 maio 2025.

KROMBHOLZ, Katharina; FRÜHWIRT, Peter; KIESEBERG, Peter; KAPSALIS, Ioannis; HUBER, Markus; WEIPPL, Edgar. QR Code Security: A Survey of Attacks and Challenges

for Usable Security. In: TRYFONAS, T.; ASKOXYLAKIS, I. (Eds.). **Human Aspects of Information Security, Privacy, and Trust**. Springer International Publishing, 2014. p. 79-90. DOI: 10.1007/978-3-319-07620-1_8.

MASOUMZADEH, A.; LAAN, H. van der; DERCKSEN, A. BlueSky: Physical Access Control: Characteristics, Challenges, and Research Opportunities. *In*: **Proceedings of the 27th ACM Symposium on Access Control Models and Technologies (SACMAT)**. New York, NY, USA: ACM, 2022. p. 1-10. DOI: 10.1145/3532105.3535019.

NETTO, Abner da Silva; SILVEIRA, Marco Antonio Pinheiro da. Gestão da segurança da informação: fatores que influenciam sua adoção em pequenas e médias empresas. **Revista de Gestão da Tecnologia e Sistemas de Informação / Journal of Information Systems and Technology Management**, v. 4, n. 3, p. 375-397, 2007. ISSN online: 1807-1775. Disponível em: https://www.scielo.br/j/jistm/a/Vx8Ypv6mDjxdYkKKrfYVgqz/?lang=pt. Acesso em: 27 out. 2025.

Physical Key Management System: Why Every Organization Needs One. **Genesis Resource**, [s.d.]. Disponível em: https://genesisresource.com/physical-key-management-system-why-every-organization-needs-one/. Acesso em: 28 setembro 2025.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar de. **Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico**. 2. ed. Novo Hamburgo: Editora Feevale, 2013. E-book (276 p.). ISBN 978-85-7717-158-3. Disponível em: https://www.feevale.br/Comum/midias/0163c988-1f5d-496f-b118-a6e009a7a2f9/E-book%20Metodologia%20do%20Trabalho%20Cientifico.pdf. Acesso em: 18 nov. 2025.

RORATTO, Rodrigo; DIAS, Evandro Dotto. Segurança da informação de produção e operações: um estudo sobre trilhas de auditoria em sistemas de banco de dados. **JISTEM - Journal of Information Systems and Technology Management**, v. 11, n. 3, p. 717-734, set./dez. 2014. DOI: 10.4301/S1807-17752014000300010.

Secure Product Design Cheat Sheet. **OWASP**, [s.d.]. Disponível em: https://cheatsheetseries.owasp.org/cheatsheets/Secure_Product_Design_Cheat_Sheet.html. Acesso em: 29 maio 2025.

SFREDDO, Josiane Ayres; FLORES, Daniel. Segurança da informação arquivística: o controle de acesso em arquivos públicos estaduais. **Perspectivas em Ciência da Informação**, v. 17, n. 2, p. 158-178, abr./jun. 2012. Disponível em: https://www.scielo.br/j/pci/a/mZKkFG8BBGHxztZvZNzRbMR/?lang=pt. Acesso em: 27 out. 2025.

WAHSHEH, Heider Ahmad; LUCCIO, Flaminia L.. Security and Privacy of QR Code Applications: A Comprehensive Study, General Guidelines and Solutions. **Information**, v. 11, n. 4, p. 217, 2020. DOI: 10.3390/info11040217.

What is a Physical Key Management System? Everything You Need to Know. **Real Time Networks**, 26 jun. 2024. Disponível em: https://www.realtimenetworks.com/blog/what-is-a-physical-key-management-system. Acesso em: 29 maio 2025.

ZAMAN, Mostafa; PURYEAR, Nathan; ABDELWAHED, Sherif; ZOHRABI, Nasibeh. A Review of IoT-Based Smart City Development and Management. **Smart Cities**, v. 7, n. 3, p. 1462-1501, 2024. DOI: 10.3390/smartcities7030061.

APÊNDICE A – EXEMPLO DA PRIMEIRA PÁGINA DE UM RELATÓRIO GERADO

KeyCode Manager - Relatório de Movimentações

Informações do Relatório

Gerado por: Lohan Yrvine Oliveira Pinheiro
Data de geração: 18:20:05 20/11/2025
Total de registros: 5717

Filtros Ativos

Tipo de usuário: todos
Auxiliar: com ou sem auxiliar
Tipo de movimentação: todas
Período: 20/11/2023 - 20/11/2025

Resumo Estatístico

Total de retiradas: 2876
Total de devoluções: 2841
Média por dia: 2513.2 movimentações
Usuário mais ativo: Maria Silva Santos (1487 movimentações)
Dia mais movimentado: 20/11/2025 (5716 movimentações)
Chaves mais movimentadas:

1. Laboratório de Informática 1 - LAB INF 1 (325 movimentações)
2. Sala de Professores - SALA PROF (286 movimentações)
3. Sala 1 - SALA(1) (271 movimentações)

Movimentações

Chave	Abreviação	Usuário	Auxiliar	Tipo	Hora e Data
Sala 17	SALA(17)	Maria Santos		Retirada	18:00:44 20/11/2025
Sala 13	SALA(13)	Maria Santos		Devolução	18:00:44 20/11/2025
Sala 11	SALA(11)	Maria Santos		Devolução	18:00:44 20/11/2025
Sala 5	SALA(5)	João Oliveira		Retirada	18:00:43 20/11/2025
Sala 7	SALA(7)	João Oliveira		Devolução	18:00:42 20/11/2025
Sala 3	SALA(3)	João Oliveira		Devolução	18:00:42 20/11/2025
Biblioteca	BIBLI	Lohan Pinheiro		Retirada	18:00:41 20/11/2025
Sala 12	SALA(12)	Lohan Pinheiro		Retirada	18:00:41 20/11/2025
Sala 15	SALA(15)	Lohan Pinheiro		Devolução	18:00:41 20/11/2025
Sala 9	SALA(9)	Lohan Pinheiro		Devolução	18:00:40 20/11/2025
Sala de Professores	SALA PROF	Lohan Pinheiro		Devolução	18:00:40 20/11/2025
Sala 11	SALA(11)	Maria Santos		Retirada	18:00:39 20/11/2025
Sala 13	SALA(13)	Maria Santos		Retirada	18:00:39 20/11/2025
Sala 8	SALA(8)	Maria Santos		Devolução	18:00:38 20/11/2025
Sala 5	SALA(5)	Maria Santos		Devolução	18:00:38 20/11/2025

APÊNDICE B – SCRIPT DE TESTE DE ESTABILIDADE

```
#!/usr/bin/env bash

set -uo pipefail

BASE_URL="${BASE_URL:-http://localhost:8080}"
CYCLES="${CYCLES:-10}"
SESSION_FILE="/tmp/kiosk_session_$$"
COOKIE_JAR="/tmp/kiosk_cookies_$$"

declare -A USERS=(
    ["92636865055"]="081632:Lohan Yrvine Oliveira Pinheiro"
    ["50830029060"]="111110:Maria Silva Santos"
    ["65904061897"]="222220:João Pedro Oliveira"
    ["20410232289"]="333330:Ana Carolina Costa"
)

RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
BLUE='\033[0;34m'
CYAN='\033[0;36m'
NC='\033[0m'

total_withdrawals=0
successful_withdrawals=0
failed_withdrawals=0
total_returns=0
successful_returns=0
failed_returns=0
total_sessions=0

start_time=0
end_time=0
total_response_time=0

cleanup() {
    rm -f "$SESSION_FILE" "$COOKIE_JAR"
    if [ -n "${session_active:-}" ]; then
        echo -e "\n${YELLOW}Logging out...${NC}"
        logout
    fi
}
trap cleanup EXIT

log_info() {
    echo -e "${BLUE}[INFO]${NC} $1"
}

log_success() {
    echo -e "${GREEN}[SUCCESS]${NC} $1"
}

log_error() {
    echo -e "${RED}[ERROR]${NC} $1"
}

log_warning() {
    echo -e "${YELLOW}[WARNING]${NC} $1"
}

log_user() {
    echo -e "${CYAN}[USER]${NC} $1"
}

get_csrf_token() {
    curl -s -c "$COOKIE_JAR" "$BASE_URL/kiosk" >/dev/null
    csrf_token=$(grep "_csrf" "$COOKIE_JAR" | awk '{print $7}')
    if [ -z "$csrf_token" ]; then
```

```
        log_error "Failed to get CSRF token" >&2
        return 1
    fi
    echo "$csrf_token"
    return 0
}

refresh_csrf() {
    csrf_token=$(grep "_csrf" "$COOKIE_JAR" 2>/dev/null | awk '{print $7}' || echo "")
    if [ -n "$csrf_token" ]; then
        CSRF_TOKEN="$csrf_token"
        return 0
    fi
    CSRF_TOKEN=$(get_csrf_token)
    return $?
}

login() {
    local cpf=$1
    local pin=$2
    local name=$3

    log_user "Logging in: $name (CPF: $cpf)"

    CSRF_TOKEN=$(get_csrf_token)
    if [ $? -ne 0 ]; then
        return 1
    fi

    response=$(curl -s -w "\n%{http_code}" \
        -X POST "$BASE_URL/auth/kiosk/login" \
        -H "Content-Type: application/json" \
        -H "x-csrf-token: $CSRF_TOKEN" \
        -b "$COOKIE_JAR" \
        -c "$COOKIE_JAR" \
        -d "{\"cpf\":\"$cpf\",\"pin\":\"$pin\"}")

    http_code=$(echo "$response" | tail -n1)
    body=$(echo "$response" | head -n-1)

    if [ "$http_code" -eq 200 ]; then
        log_success "Logged in successfully"
        session_active=true
        ((total_sessions++))
        return 0
    else
        log_error "Login failed (HTTP $http_code): $body"
        return 1
    fi
}

logout() {
    if [ -z "${session_active:-}" ]; then
        return 0
    fi

    refresh_csrf

    response=$(curl -s -w "\n%{http_code}" \
        -X POST "$BASE_URL/auth/kiosk/logout" \
        -H "x-csrf-token: $CSRF_TOKEN" \
        -b "$COOKIE_JAR" \
        -c "$COOKIE_JAR")

    http_code=$(echo "$response" | tail -n1)

    if [ "$http_code" -eq 204 ] || [ "$http_code" -eq 200 ]; then
        log_success "Logged out successfully"
    elif [ "$http_code" -eq 403 ]; then
        log_warning "Session already invalidated (HTTP 403)"
    else
        log_warning "Logout returned HTTP $http_code (session may already be cleared)"
    fi
```

```
    session_active=""
    return 0
}

get_available_keys_from_db() {
    sqlite3 keycode.db "SELECT public_id FROM keys WHERE available = 1 AND lost = 0 AND delete_time IS NULL" 2>/dev/
null || echo ""
}

get_unavailable_keys_from_db() {
    sqlite3 keycode.db "SELECT public_id FROM keys WHERE available = 0 AND lost = 0 AND delete_time IS NULL" 2>/dev/
null || echo ""
}

withdraw_key() {
    local key_id=$1

    ((total_withdrawals++))

    refresh_csrf

    local op_start=$(date +%s%3N)
    response=$(curl -s -w "\n%{http_code}" \
        -X POST "$BASE_URL/api/kiosk/withdrawal" \
        -H "Content-Type: application/json" \
        -H "x-csrf-token: $CSRF_TOKEN" \
        -b "$COOKIE_JAR" \
        -c "$COOKIE_JAR" \
        -d "{\"key_id\":\"$key_id\"}")
    local op_end=$(date +%s%3N)
    local op_duration=$((op_end - op_start))
    total_response_time=$((total_response_time + op_duration))

    http_code=$(echo "$response" | tail -n1)
    body=$(echo "$response" | head -n-1)

    if [ "$http_code" -eq 200 ]; then
        ((successful_withdrawals++))
        log_success "Withdrawal #$total_withdrawals successful (key: $key_id) [${op_duration}ms]"
        return 0
    else
        ((failed_withdrawals++))
        error_msg=$(echo "$body" | grep -o '"message":"[^"]*"' | cut -d'"' -f4 || echo "$body")
        log_error "Withdrawal #$total_withdrawals failed (HTTP $http_code)"
        echo "  Key ID: $key_id"
        echo "  Error: $error_msg"
        return 1
    fi
}

return_key() {
    local key_id=$1

    ((total_returns++))

    refresh_csrf

    local op_start=$(date +%s%3N)
    response=$(curl -s -w "\n%{http_code}" \
        -X POST "$BASE_URL/api/kiosk/return" \
        -H "Content-Type: application/json" \
        -H "x-csrf-token: $CSRF_TOKEN" \
        -b "$COOKIE_JAR" \
        -c "$COOKIE_JAR" \
        -d "{\"key_id\":\"$key_id\"}")
    local op_end=$(date +%s%3N)
    local op_duration=$((op_end - op_start))
    total_response_time=$((total_response_time + op_duration))

    http_code=$(echo "$response" | tail -n1)
    body=$(echo "$response" | head -n-1)
```

```
    if [ "$http_code" -eq 204 ] || [ "$http_code" -eq 200 ]; then
        ((successful_returns++))
        log_success "Return #$total_returns successful (key: $key_id) [${op_duration}ms]"
        return 0
    else
        ((failed_returns++))
        error_msg=$(echo "$body" | grep -o '"message":"[^"]*"' | cut -d'"' -f4 || echo "$body")
        log_error "Return #$total_returns failed (HTTP $http_code)"
        echo "  Key ID: $key_id"
        echo "  Error: $error_msg"
        return 1
    fi
}

print_banner() {
    echo "========================================"
    echo "  Kiosk Multi-User Stress Test"
    echo "========================================"
    echo "Base URL: $BASE_URL"
    echo "Users: ${#USERS[@]}"
    echo "Cycles: $CYCLES"
    echo "========================================"
    echo ""
}

print_results() {
    echo ""
    echo "========================================"
    echo "  Test Results"
    echo "========================================"

    local duration=$((end_time - start_time))
    local duration_formatted
    if [ $duration -ge 60 ]; then
        local minutes=$((duration / 60))
        local seconds=$((duration % 60))
        duration_formatted="${minutes}m ${seconds}s"
    else
        duration_formatted="${duration}s"
    fi

    echo "Execution Time:"
    echo "  Duration: $duration_formatted ($duration seconds)"
    echo ""

    echo "Sessions:"
    echo "  Total Sessions: $total_sessions"
    echo ""
    echo "Withdrawals:"
    echo "  Total:      $total_withdrawals"
    echo "  Successful: $successful_withdrawals"
    echo "  Failed:     $failed_withdrawals"

    if [ $total_withdrawals -gt 0 ]; then
        withdrawal_rate=$(awk "BEGIN {printf \"%.2f\", ($successful_withdrawals/$total_withdrawals)*100}")
        echo "  Success Rate: ${withdrawal_rate}%"
    fi

    echo ""
    echo "Returns:"
    echo "  Total:      $total_returns"
    echo "  Successful: $successful_returns"
    echo "  Failed:     $failed_returns"

    if [ $total_returns -gt 0 ]; then
        return_rate=$(awk "BEGIN {printf \"%.2f\", ($successful_returns/$total_returns)*100}")
        echo "  Success Rate: ${return_rate}%"
    fi

    echo ""
    echo "Overall:"
    total_operations=$((total_withdrawals + total_returns))
    successful_operations=$((successful_withdrawals + successful_returns))
```

```
    failed_operations=$((failed_withdrawals + failed_returns))

    echo "  Total Operations:      $total_operations"
    echo "  Successful Operations: $successful_operations"
    echo "  Failed Operations:     $failed_operations"

    if [ $total_operations -gt 0 ]; then
        overall_rate=$(awk "BEGIN {printf \"%.2f\", ($successful_operations/$total_operations)*100}")
        echo "  Overall Success Rate:  ${overall_rate}%"
    fi

    echo ""
    echo "Performance:"
    if [ $duration -gt 0 ] && [ $total_operations -gt 0 ]; then
        ops_per_sec=$(awk "BEGIN {printf \"%.2f\", $total_operations/$duration}")
        echo "  Operations per Second: $ops_per_sec"

        avg_response_time=$(awk "BEGIN {printf \"%.2f\", $total_response_time/$total_operations}")
        echo "  Average Response Time: ${avg_response_time}ms"
    else
        echo "  Operations per Second: N/A"
        echo "  Average Response Time: N/A"
    fi

    echo "========================================"
}

main() {
    start_time=$(date +%s)

    print_banner

    log_info "Starting multi-user stress test..."
    echo ""

    cpfs=(${!USERS[@]})

    for ((cycle = 1; cycle <= CYCLES; cycle++)); do
        echo ""
        log_info "========================================"
        log_info "  Cycle $cycle/$CYCLES"
        log_info "========================================"
        echo ""

        random_index=$((RANDOM % ${#cpfs[@]}))
        selected_cpf="${cpfs[$random_index]}"

        user_data="${USERS[$selected_cpf]}"
        pin="${user_data%%:*}"
        name="${user_data#*:}"

        if ! login "$selected_cpf" "$pin" "$name"; then
            log_error "Cannot proceed without login for this cycle"
            continue
        fi

        echo ""

        unavailable_keys=$(get_unavailable_keys_from_db)
        user_has_keys=false

        if [ -n "$unavailable_keys" ]; then
            user_keys=$(sqlite3 keycode.db "SELECT k.public_id FROM keys k
                JOIN key_logs kl ON k.id = kl.key_id
                JOIN users u ON kl.actor_id = u.id
                WHERE u.cpf = '$selected_cpf'
                AND k.available = 0
                AND k.lost = 0
                AND k.delete_time IS NULL
                AND kl.type = 'withdrawal'
                AND kl.id = (SELECT MAX(kl2.id) FROM key_logs kl2 WHERE kl2.key_id = k.id)" 2>/dev/null || echo "")

            if [ -n "$user_keys" ]; then
```

```
            user_has_keys=true
        fi
    fi

    if [ "$user_has_keys" = true ]; then
        log_info "User has keys from previous session, returning them"
        echo ""

        for key_id in $user_keys; do
            if [ -n "$key_id" ]; then
                log_info "Returning key: $key_id"
                return_key "$key_id"
                sleep 0.3
            fi
        done

        should_withdraw_again=$((RANDOM % 10))
        if [ $should_withdraw_again -lt 5 ]; then
            echo ""
            log_info "User will now withdraw new keys"
        else
            echo ""
            log_info "User done for this session"
            echo ""
            logout
            sleep 0.5
            continue
        fi
    fi

    num_keys=$((RANDOM % 3 + 1))
    log_info "User will attempt to withdraw $num_keys key(s)"
    echo ""

    withdrawn_keys=()

    for ((i = 1; i <= num_keys; i++)); do
        available_keys=$(get_available_keys_from_db)

        if [ -z "$available_keys" ]; then
            log_warning "No available keys found in database"
            break
        fi

        key_array=($available_keys)
        random_key_index=$((RANDOM % ${#key_array[@]}))
        selected_key="${key_array[$random_key_index]}"

        log_info "Attempting to withdraw key $i/$num_keys: $selected_key"
        if withdraw_key "$selected_key"; then
            withdrawn_keys+=("$selected_key")
            sleep 0.3
        fi
    done

    if [ ${#withdrawn_keys[@]} -gt 0 ]; then
        echo ""
        log_info "User keeping ${#withdrawn_keys[@]} key(s) (will be returned in future session)"
    fi

    echo ""
    logout
    sleep 0.5
done

available_keys=$(get_available_keys_from_db)
if [ -z "$available_keys" ]; then
    echo ""
    log_info "Some keys still withdrawn, returning them..."

    cpf="${cpfs[0]}"
    user_data="${USERS[$cpf]}"
    pin="${user_data%%:*}"
```

```
        name="${user_data#*:}"

        echo ""
        if login "$cpf" "$pin" "$name"; then
            echo ""
            unavailable_keys=$(sqlite3 keycode.db "SELECT public_id FROM keys WHERE available = 0 AND lost = 0 AND
delete_time IS NULL" 2>/dev/null || echo "")

            if [ -n "$unavailable_keys" ]; then
                for key_id in $unavailable_keys; do
                    if [ -n "$key_id" ]; then
                        log_info "Returning key: $key_id"
                        return_key "$key_id"
                        sleep 0.3
                    fi
                done
            fi

            echo ""
            logout
        fi
    fi

    end_time=$(date +%s)

    print_results
}

if [ "$#" -gt 0 ] && [ "$1" = "--help" ]; then
    echo "Usage: $0 [CYCLES]"
    echo ""
    echo "Multi-user stress test for kiosk system."
    echo "Each cycle randomly selects a user who withdraws 1-3 keys."
    echo ""
    echo "Environment variables:"
    echo "  BASE_URL - Base URL of the application (default: http://localhost:8080)"
    echo "  CYCLES   - Number of user sessions (default: 10)"
    echo ""
    echo "Examples:"
    echo "  $0              # Run 10 sessions"
    echo "  $0 20           # Run 20 sessions"
    echo "  BASE_URL=http://192.168.1.100:8080 CYCLES=50 $0"
    exit 0
fi

if [ "$#" -eq 1 ]; then
    CYCLES=$1
fi

main
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Campina Grande - Código INEP: 25137409
R. Tranqüilino Coelho Lemos, 671, Dinamérica, CEP 58432-300, Campina Grande (PB)
CNPJ: 10.783.898/0003-37 - Telefone: (83) 2102.6200

Documento Digitalizado Ostensivo (Público)

TCC

Assunto:	TCC
Assinado por:	Lohan Yrvine
Tipo do Documento:	Tese
Situação:	Finalizado
Nível de Acesso:	Ostensivo (Público)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Lohan Yrvine Oliveira Pinheiro**, **ALUNO (202011250010) DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO - CAMPINA GRANDE**, em 30/01/2026 10:46:02.

Este documento foi armazenado no SUAP em 30/01/2026. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1748583
Código de Autenticação: f76d8afd6a