INSTITUTO FEDERAL
Rondônia

MINISTÉRIO DA EDUCAÇÃO
Secretaria de Educação Profissional e Tecnológica
Instituto Federal de Educação, Ciência e Tecnologia de Rondônia

Campus Vilhena

Coordenação do Curso Superior em Tecnologia em Análise e Desenvolvimento de Sistemas

HENRIQUE ZORZI

Venda Fácil PDV: Desenvolvimento de um Sistema de Ponto de Venda Móvel para Pequenos Comerciantes

VILHENA - RO

2026

HENRIQUE ZORZI

Venda Fácil PDV: Desenvolvimento de um Sistema de Ponto de Venda Móvel para Pequenos Comerciantes

Artigo entregue como Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO), Campus Vilhena, como requisito parcial para obtenção do grau de Tecnólogo, junto ao Curso Superior em Tecnologia em Análise e Desenvolvimento de Sistemas, sob a orientação do professor Mestre Gilberto Pereira da Silva.

VILHENA - RO

2026

Ficha catalográfica elaborada pelo Sistema Gerador de Ficha Catalográfica do IFRO.

Zorzi, Henrique.
Venda Fácil PDV: desenvolvimento de um sistema de ponto de venda móvel para pequenos comerciantes / Henrique Zorzi. - Vilhena, 2026.
30 f. : il.

Orientador(a): Prof. Me. Gilberto Pereira da Silva.

Trabalho de Conclusão de Curso (Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO, Vilhena, 2026.

1. Aplicativo móvel. 2. Kotlin. 3. Pequenos comerciantes. 4. Gestão de vendas. 5. Android. I. Silva, Gilberto Pereira da (orient.). II. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO. III. Título.

Bibliotecário(a) Responsável: Rosilene Maria do Couto Marques, CRB-11/321

HENRIQUE ZORZI

Venda Fácil PDV: Desenvolvimento de um Sistema de Ponto de Venda Móvel para Pequenos Comerciantes

Artigo entregue como Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO), Campus Vilhena, como requisito parcial para obtenção do grau de Tecnólogo, junto ao Curso Superior em Tecnologia em Análise e Desenvolvimento de Sistemas, sob a orientação do professor Mestre Gilberto Pereira da Silva.

Aprovado em 17/11/2026 pela banca examinadora.

Prof. Mestre José Lucas Brandão Montes

Examinador interno

Prof. Mestre Wesley Jhonnes Ramos Rolim

Examinador interno

Prof. Mestre Gilberto Pereira da Silva

Orientador

Resumo

Muitos microempreendedores brasileiros ainda registram vendas, estoque e caixa à mão, o que gera erros e limita a tomada de decisão. Este trabalho apresenta o Venda Fácil PDV, um sistema de ponto de venda composto por aplicativo Android e API REST, concebido para reunir em uma única operação os registros que o pequeno vendedor costuma manter separados. A pesquisa, aplicada e descritiva, levantou doze requisitos funcionais e quatro não funcionais a partir de soluções existentes e da prática do varejo, construiu o sistema com Kotlin, Jetpack Compose, TypeScript, Express, JWT e MongoDB Atlas e o verificou por testes automatizados e verificação funcional. Os 1.184 testes da API foram aprovados, com 99,00% de cobertura de instruções, assim como os 248 testes do aplicativo; os doze requisitos funcionais e os quatro não funcionais foram atendidos. O trabalho oferece uma alternativa de código aberto e hospedagem própria, verificada em ambiente controlado; o efeito sobre a rotina do comerciante não foi medido e fica, com o funcionamento offline, como trabalho futuro.

Palavras-chave: Aplicativo móvel; Kotlin; Pequenos comerciantes; Gestão de vendas; Android.

Abstract

Many Brazilian micro-retailers still record sales, stock and cash by hand, which causes errors and limits decision-making. This paper presents Venda Fácil PDV, a point-of-sale system composed of an Android application and a REST API, designed to combine, in a single operation, the records that small retailers usually keep apart. The applied and descriptive research elicited twelve functional and four non-functional requirements from existing solutions and retail practice, built the system with Kotlin, Jetpack Compose, TypeScript, Express, JWT and MongoDB Atlas, and verified it through automated tests and functional checks. All 1,184 API tests passed with 99.00% statement coverage, as did the 248 application tests; all twelve functional requirements and the four non-functional ones were met. The work offers an open-source, self-hostable alternative verified in a controlled environment; its effect on retailers' routines was not measured and remains, with offline operation, as future work.

Keywords: Mobile application; Kotlin; Small retailers; Sales management; Android.

1. Introdução

No cenário contemporâneo, as organizações buscam constantemente ampliar a lucratividade e reduzir os custos operacionais. Para que esses objetivos sejam alcançados, torna-se indispensável o planejamento estratégico aliado à análise de dados, permitindo que as empresas acompanhem a evolução tecnológica e evitem a obsolescência em um mercado cada vez mais digital (FREITAS; PASCHOAL, 2020). No Brasil, os pequenos negócios representam cerca de 95% das empresas e respondem por 30% da formação do Produto Interno Bruto (SEBRAE, 2024).

Além disso, a velocidade das mudanças no mercado e a constante variação das preferências dos consumidores exigem que o empreendedor tenha acesso a informações precisas e confiáveis. Dessa forma, tomadas de decisão mais rápidas e seguras dependem da utilização de sistemas capazes de reduzir falhas humanas e proporcionar maior controle sobre a gestão financeira e o fluxo de mercadorias (URBANO; ZOTTESSO; RUIZ, 2024).

Sem o auxílio dessas ferramentas, o administrador torna-se mais vulnerável à perda de informações e à lentidão operacional, fatores que podem comprometer a sustentabilidade do negócio. Atualmente, observa-se que muitas empresas adotam estratégias digitais para obter vantagens competitivas. Entretanto, diversos pequenos negócios ainda dependem de métodos manuais de controle, dificultando a identificação da rotatividade de produtos e do comportamento de compra dos clientes. Nesse sentido, o gerenciamento de estoque possui impacto direto no desempenho financeiro da organização, uma vez que os itens armazenados representam a capacidade da empresa de atender às demandas do mercado (SILVA, K. B. A.; MADEIRA, 2004).

A ausência de uma visão sistêmica dificulta que o gestor alcance maior eficiência nos processos administrativos. Embora existam desafios relacionados à obtenção de infraestrutura tecnológica, o planejamento continua sendo um fator essencial para a sobrevivência e o crescimento sustentável do empreendimento. Diante disso, sistemas de Ponto de Venda (PDV) tornam-se importantes por integrarem operações de venda, controle de estoque e gestão financeira, contribuindo para a otimização dos processos internos (LEAL, 2020; URBANO; ZOTTESSO; RUIZ, 2024).

Diante desse contexto, o problema de pesquisa que orienta este trabalho é: como um aplicativo móvel integrado a uma Interface de Programação de Aplicações (API) pode ser projetado e verificado para reunir o controle de vendas, estoque, caixa e clientes de pequenos comerciantes em uma única solução, de modo que cada operação atualize os registros que hoje são mantidos separadamente? O objetivo geral consiste em desenvolver o Venda Fácil PDV, uma solução móvel para apoiar a gestão operacional de pequenos vendedores. Como objetivos específicos, definem-se: levantar requisitos funcionais relacionados às rotinas de venda e controle comercial; projetar uma API para autenticação, autorização, regras de negócio e persistência de dados; desenvolver um aplicativo Android para interação do usuário com o sistema; implementar funcionalidades de venda, estoque, caixa, cadastros e relatórios; e validar o funcionamento da solução por meio de testes automatizados e verificação dos principais fluxos de uso.

A justificativa do trabalho está relacionada à necessidade de oferecer aos pequenos comerciantes uma alternativa acessível, móvel e centralizada para tarefas que impactam diretamente a continuidade do negócio. Ao reunir vendas, estoque, caixa e

relatórios em uma mesma solução, o sistema busca reduzir falhas de registro, facilitar o acompanhamento das operações e apoiar decisões baseadas em dados. Além disso, o uso de tecnologias atuais, como Kotlin, Jetpack Compose, Node.js, TypeScript e MongoDB Atlas, permite construir uma aplicação alinhada às práticas contemporâneas de desenvolvimento de software.

Em relação às soluções de mercado analisadas na Seção 2.8, a contribuição está em oferecer, com código aberto, uma API própria que o comerciante ou a instituição podem hospedar, com controle de permissões por rota e verbo HTTP e sem dependência de conta em serviço de terceiro. O efeito do sistema sobre a rotina do comerciante não foi medido nesta etapa; o trabalho verifica, por testes automatizados e verificação funcional, que o sistema faz o que foi especificado.

Este artigo está organizado da seguinte forma: a Seção 2 apresenta a fundamentação teórica; a Seção 3 descreve a metodologia; a Seção 4 detalha o desenvolvimento e a implementação do Venda Fácil PDV; a Seção 5 apresenta e analisa os resultados; e a Seção 6 traz as considerações finais.

2. Fundamentação Teórica

2.1. O Cenário das Micro e Pequenas Empresas (MPEs) no Brasil

As micro e pequenas empresas desempenham um papel fundamental na economia nacional, e em 2023 geraram mais de 80% dos empregos formais do país (SEBRAE, 2024).

Apesar dessa relevância, a taxa de mortalidade desses empreendimentos é preocupante. O Sebrae calculou em 24,4% a taxa de mortalidade das empresas com até dois anos de atividade abertas em 2007 (SEBRAE, 2013, p. 19) e apontou a falta de planejamento antes da abertura e a falta de experiência na gestão do negócio como fatores que podem estar associados ao fechamento (SEBRAE, 2013, p. 23). O levantamento refere-se a empresas abertas em 2007 e é usado aqui pela ordem de grandeza do fenômeno. Segundo Pereira et al. (2009), essa vulnerabilidade está ligada a falhas gerenciais graves, como a falta de capital de giro (42% das respostas) e problemas financeiros diversos (21%), muitas vezes decorrentes de níveis de controle e avaliação precários.

Nesse contexto, visto que a análise financeira e o planejamento são pilares para a sobrevivência organizacional, a automação desses processos por meio de um sistema de PDV móvel torna-se estratégica. Tal ferramenta permite que o microempreendedor obtenha controle em tempo real sobre o fluxo de caixa e estoque, combatendo diretamente as causas de mortalidade identificadas e profissionalizando a tomada de decisão.

2.2. Automação Comercial no Varejo Brasileiro

A evolução da automação comercial no Brasil foi marcada pela substituição das antigas caixas registradoras mecânicas pelos modernos sistemas de PDV, responsáveis por proporcionar maior agilidade, organização e controle às operações comerciais. A necessidade de profissionalização do pequeno varejo foi um dos principais fatores que impulsionaram a expansão dessas tecnologias no país, levando empresas a buscarem soluções capazes de otimizar atividades operacionais e melhorar o gerenciamento das vendas e do estoque (SILVA, J. F., 2022).

Nesse contexto, a automação comercial passou a desempenhar um papel importante na modernização dos estabelecimentos varejistas, especialmente entre MPEs. Entretanto, a transformação digital nesse segmento ainda ocorre de forma gradual. Um estudo com pequenos comércios de Sertãozinho-PB aponta que grande parte desses empreendimentos permanece em níveis básicos de digitalização, enquanto apenas uma pequena parcela apresenta maior maturidade tecnológica (SILVA, Y. B., 2026). Apesar disso, observa-se um crescimento na busca por softwares de gestão e sistemas de PDV capazes de centralizar informações de vendas, estoque e controle financeiro, oferecendo uma base mais estruturada para a administração do negócio (SILVA, Y. B., 2026).

2.3. Conceitos e Gestão Estratégica de Estoques

O maior desafio da gestão de estoques é saber quando e quanto repor cada produto e quanto manter em estoque de segurança (KUNIGAMI; OSÓRIO, 2009 apud CARVALHO; MAZZOTTI, 2024, p. 5). Esse controle exige equilíbrio, pois o excesso de estoque representa um custo desnecessário, enquanto a falta de produtos pode causar perda de lucro (AMARAL; DOURADO, 2011 apud CARVALHO; MAZZOTTI, 2024, p. 5). Por isso, a gestão de estoques precisa estar em sintonia com as outras áreas da empresa e ocorrer de forma equilibrada (CARVALHO; MAZZOTTI, 2024, p. 5).

Nas micro e pequenas empresas, o gerenciamento adequado dos estoques permite reduzir custos, otimizar o fluxo de caixa, atender à demanda dos clientes e melhorar a eficiência operacional (KOS et al., 2014 apud CARVALHO; MAZZOTTI, 2024, p. 7).

Apesar disso, muitas empresas ainda controlam o estoque de forma manual, o que aumenta o risco de erros e dificulta a obtenção de informações precisas e em tempo real (TACHIZAWA, 2002 apud CARVALHO; MAZZOTTI, 2024, p. 7). Para as microempresas, sistemas informatizados de baixo custo podem reduzir erros, agilizar o registro e dar mais precisão às informações (BARBOSA, 2019 apud CARVALHO; MAZZOTTI, 2024, p. 7).

2.4. Controle de Fluxo de Caixa em MPEs

O fluxo de caixa é definido como o processo de registrar e controlar todas as movimentações financeiras de uma empresa, tendo evoluído de uma simples ferramenta de registro para um instrumento estratégico de planejamento financeiro de curto e longo prazo (RAMOS, 2024). Diferentemente do regime de competência, que registra a venda no momento da contratação, o regime de caixa reflete o movimento real dos recursos financeiros, apresentando uma visão precisa da liquidez disponível, aspecto crítico para empresas que operam com margens reduzidas, como as MPEs (RAMOS, 2024).

Diante das causas de fechamento vistas na Seção 2.1, o controle do fluxo de caixa atua como mecanismo preventivo, permitindo ao gestor visualizar o saldo disponível e antecipar necessidades de capital (RAMOS, 2024).

Complementarmente, as projeções de fluxo de caixa permitem identificar sazonalidades e antecipar períodos de baixa ou alta demanda, facilitando o planejamento de estoques, abordado na Seção 2.3, e de investimentos de curto e longo prazo. Dessa forma, o controle financeiro e o controle de estoque tornam-se processos interdependentes, cuja integração é fundamental para a sustentabilidade operacional do negócio (RAMOS, 2024; CARVALHO; MAZZOTTI, 2024, p. 2).

Por fim, vale destacar que muitos gestores de MPEs ainda enxergam o fluxo de caixa apenas como um registro de entradas e saídas, sem compreender seu potencial estratégico. Para que o sistema de PDV, apresentado na Seção 2.2, contribua efetivamente para a sustentabilidade financeira, o controle deve ser sistemático e orientado ao planejamento, com o uso de tecnologias que permitam o acompanhamento das movimentações em tempo real e possibilitam reações mais rápidas às mudanças do mercado (RAMOS, 2024).

2.5. Desenvolvimento Android Moderno com Kotlin

O desenvolvimento para a plataforma Android tem evoluído significativamente, migrando de uma abordagem tradicional baseada em Java para um ecossistema mais moderno, centrado na linguagem Kotlin. Desde 2019, o Google passou a adotar oficialmente o modelo *Kotlin-first*, recomendando a linguagem como padrão para o desenvolvimento de aplicações Android (GOOGLE, [s. d.]).

Nesse contexto, a adoção do Kotlin no desenvolvimento do sistema de PDV móvel proposto justifica-se por características que impactam diretamente a qualidade e a confiabilidade do software. Entre elas, destaca-se a maior concisão da linguagem, que permite a implementação de funcionalidades com menor quantidade de código, favorecendo a produtividade e a manutenção (SHARMA; PRASAD, 2024 apud CARVALHO, 2025). Recursos como *data classes*, funções de extensão e expressões lambda contribuem para uma modelagem mais clara e eficiente das regras de negócio (JETBRAINS, 2024 apud CARVALHO, 2025).

Outro aspecto relevante é a segurança contra referências nulas, conhecida como null safety, incorporada ao sistema de tipos da linguagem, reduzindo significativamente a ocorrência de falhas em tempo de execução, como o erro *NullPointerException*, comum em aplicativos escritos em Java (JETBRAINS, 2025 apud CARVALHO, 2025). Essa característica é especialmente importante em sistemas de vendas, nos quais a confiabilidade das transações é um fator crítico.

2.6. Persistência de Dados e Banco de Dados em Nuvem

A persistência deste projeto usa o MongoDB Atlas, banco de dados em nuvem orientado a documentos.

Diferentemente dos Sistemas Gerenciadores de Bancos de Dados (SGBDs) relacionais, que utilizam estruturas rígidas de tabelas e colunas, o MongoDB armazena as informações em documentos no formato BSON (Binary JSON), uma versão binária do JSON (JavaScript Object Notation). Segundo Bortoloto (2022), essa abordagem rompe parcialmente com os paradigmas tradicionais ao adotar uma estrutura de esquema flexível, permitindo que documentos de uma mesma coleção apresentem atributos distintos.

Essa flexibilidade pode ser vantajosa para sistemas de PDV, nos quais diferentes categorias de produtos podem vir a possuir características específicas, como tamanhos, pesos e classificações variadas, sem a necessidade de reestruturar o banco de dados. No Venda Fácil PDV, os documentos seguem esquemas definidos com o Mongoose, o que garante consistência aos dados sem impedir a inclusão de novos atributos no futuro. Além disso, destacam-se como vantagens técnicas do MongoDB a alta performance, a escalabilidade e a disponibilidade dos dados (BORTOLOTO, 2022).

Neste projeto, porém, o que pesou na escolha não foram a escalabilidade horizontal nem a replicação, dispensáveis para um pequeno comércio, mas três fatores: o uso de JSON de ponta a ponta entre aplicativo, API e banco, que reduz a complexidade do desenvolvimento (BORTOLOTO, 2022); a disponibilidade de um nível gratuito no Atlas já configurado como conjunto de réplicas, o que habilita transações multidocumento; e a familiaridade do autor com a plataforma. O modelo resultante, descrito na Seção 4.2, é referencial, com coleções ligadas por identificadores e operações em transação, de modo que um banco relacional também o atenderia; a flexibilidade de esquema fica como possibilidade futura, não como motivo da escolha.

2.7. APIs REST e Arquitetura Cliente-Servidor

Na arquitetura cliente-servidor, o sistema é dividido em duas partes que se comunicam por uma rede: o cliente, que oferece as telas e permite a interação do usuário, e o servidor, que armazena as informações e as fornece ao cliente quando necessário (SILVA, S. de O.; COSTA, 2024, p. 58). Quando essa divisão é feita em três camadas, a de apresentação corresponde ao cliente, a de aplicação processa os dados e aplica as regras de negócio, e a de banco de dados armazena as informações com segurança (SILVA, S. de O.; COSTA, 2024, p. 58-59).

A comunicação entre cliente e servidor pode ser feita pelo protocolo HTTP (Hypertext Transfer Protocol), no qual as requisições são feitas por URLs (Uniform Resource Locators) e cada método indica a ação desejada, como GET para solicitar dados, POST para enviar, PUT e PATCH para atualizar e DELETE para remover (SILVA, S. de O.; COSTA, 2024, p. 62). Sobre esse protocolo se apoia o REST (Representational State Transfer), um estilo arquitetural para serviços web que segue o modelo cliente-servidor, realiza as operações pelos métodos HTTP e exige que cada requisição traga todas as informações necessárias para ser processada, sem depender de requisições anteriores (FIELDING, 2000). Os dados costumam ser trocados no formato JSON, organizado em pares de chave e valor (SILVA, S. de O.; COSTA, 2024, p. 63).

No Venda Fácil PDV, essa organização se aplica diretamente: o aplicativo Android atua como cliente, e a API, junto com o banco de dados, forma o servidor, que concentra as regras de negócio e a persistência dos dados, como detalhado na Seção 4.

2.8. Verificação de Software e Soluções de Ponto de Venda Existentes

A verificação de software busca evidências de que o produto faz o que foi especificado, e os testes são seu principal instrumento (PRESSMAN; MAXIM, 2021). A pirâmide de testes recomenda base ampla de testes de unidade, camada menor de integração e poucos testes de interface (VOCKE, 2018); este trabalho segue essa organização na API e no aplicativo e usa a cobertura como medida do código exercitado, não como prova de ausência de defeitos.

Entre os trabalhos acadêmicos próximos, Silva e Viegas (2022) fizeram um sistema de estoque, vendas e caixa em Java para computador de mesa, com perfis de usuário e administrador e sem relato de verificação; Oliveira (2018) construiu o SVEN, aplicação web sobre Firebase para pedidos de vendedores externos, sem caixa, avaliada com uma usuária; Grassetti (2018) prototipou um aplicativo Android para fichas de consumo em bares, testado em dois aparelhos. Nenhum combina aplicativo móvel, API

própria, caixa por turno e permissões por rota, nem relata testes automatizados (Quadro 2).

Entre as soluções de mercado analisadas no levantamento de requisitos (Seção 3.5), Loyverse e Square são gratuitas no uso básico, com dados na nuvem do fornecedor e recursos avançados cobrados, e a Square não opera no Brasil; Chromis é de código aberto, mas só para computador de mesa; Simple POS não tem documentação pública suficiente e ficou fora da comparação. O Quadro 1 resume. O Venda Fácil PDV se distingue por combinar aplicativo Android, API própria de código aberto hospedável pelo comerciante ou pela instituição, caixa por turno e permissões por rota e verbo HTTP.

Quadro 1. Comparação com soluções de ponto de venda existentes

Solução	Plataforma	Venda, estoque e caixa por turno	Permissões por usuário	Custo e propriedade dos dados
Loyverse POS (LOYVERSE, [s. d.])	Android e iOS	Sim	Sim, em módulo pago	Gratuito no básico; dados na nuvem do fornecedor; código fechado
Square POS (SQUARE, [s. d.])	Android e iOS	Sim	Sim	Gratuito, com tarifa por transação; não opera no Brasil; código fechado
Chromis POS (CHROMIS, [s. d.])	Computador de mesa (Java)	Venda e estoque; caixa por sessão	Sim	Gratuito e de código aberto; dados no próprio computador
Venda Fácil PDV (este trabalho)	Android, com API própria	Sim	Sim, por rota e verbo HTTP	Gratuito e de código aberto; dados em banco controlado por quem hospeda a API

Fonte: elaborada pelo autor (2026), a partir das páginas dos fornecedores.

Quadro 2. Comparação com trabalhos acadêmicos correlatos

Trabalho	Plataforma	Venda, estoque e caixa	Permissões	Verificação relatada
Silva e Viegas (2022)	Computador de mesa, Java e MySQL	Sim	Usuário e administrador	Não informada
Oliveira (2018)	Web, Firebase	Venda e estoque; sem caixa	Administrador e vendedor	Usabilidade com uma usuária e questionário
Grassetti (2018)	Android, Firebase	Fichas de consumo; sem estoque nem caixa	Login e perfil administrador	Jornada completa em dois aparelhos
Venda Fácil PDV (este trabalho)	Android, com API REST própria	Sim, com caixa por turno	Por grupo, rota e verbo HTTP	1.184 testes na API, 248 no aplicativo e roteiro funcional de 24 casos

Fonte: elaborada pelo autor (2026).

3. Metodologia

Esta seção apresenta os procedimentos adotados para o desenvolvimento e a validação do Venda Fácil PDV. A ordem foi modelagem e prototipação, depois API e seus testes e, por fim, o aplicativo integrado a ela.

3.1. Caracterização da Pesquisa

Esta pesquisa classifica-se como aplicada, pois busca gerar conhecimentos para a solução de um problema específico (PRODANOV; FREITAS, 2013, p. 51), no caso o controle manual de vendas, estoque e caixa em pequenos comércios. Quanto aos objetivos, é

descritiva, por registrar e descrever o sistema desenvolvido e seu comportamento (PRODANOV; FREITAS, 2013, p. 51-52).

Quanto à abordagem, é predominantemente qualitativa, na análise das soluções existentes e na definição dos requisitos; as contagens de testes, a cobertura de código e o número de requisitos atendidos são medidas descritivas de apoio, e não um tratamento estatístico (PRODANOV; FREITAS, 2013, p. 71). Quanto aos procedimentos, utiliza a pesquisa bibliográfica, elaborada a partir de material já publicado (PRODANOV; FREITAS, 2013, p. 54), e o desenvolvimento e a verificação de um artefato de software, procedimento próprio da pesquisa tecnológica, em que o sistema é construído e seu comportamento é verificado sob condições controladas por testes automatizados e verificação funcional (PRESSMAN; MAXIM, 2021). Não se trata de pesquisa experimental no sentido de Prodanov e Freitas (2013, p. 57), pois não há variável manipulada nem grupo de comparação.

3.2. Unidade de Análise e Observação

A unidade de análise desta pesquisa é a rotina de controle de vendas, estoque e caixa dos pequenos comerciantes, que formam o universo ao qual a solução se destina (PRODANOV; FREITAS, 2013, p. 98). A unidade de observação é o próprio sistema Venda Fácil PDV, tratado como artefato cujo comportamento foi verificado sob condições controladas por meio dos testes automatizados e da verificação funcional dos requisitos. Essa escolha se justifica porque não houve coleta de dados junto a usuários, de modo que a avaliação se concentra no funcionamento do sistema diante das rotinas que ele pretende atender.

3.3. Técnica de Coleta de Dados

Neste trabalho, não houve coleta junto a usuários, e os dados foram obtidos por três técnicas. A primeira foi o uso de dados secundários, isto é, informações já disponíveis e não produzidas especificamente para a pesquisa (PRODANOV; FREITAS, 2013, p. 102), obtidos na análise de soluções de ponto de venda existentes no mercado e somados ao conhecimento prévio das rotinas do varejo, ambos utilizados na definição dos requisitos descrita na Seção 3.5. A segunda foi a execução da suíte de testes automatizados da API e do aplicativo, que constituem a principal forma de obter evidências de que o software faz o que foi especificado (PRESSMAN; MAXIM, 2021). A terceira foi a verificação funcional dos requisitos da Quadro 3, feita por observação sistemática, que é planejada, ocorre em condições controladas e exige um plano prévio de registro das informações (PRODANOV; FREITAS, 2013, p. 104). Para isso, foi elaborado um roteiro com vinte e quatro casos de teste executados no aplicativo, cada um com pré-condição, passos realizados, resultado esperado e resultado obtido. O roteiro foi executado pelo autor no aparelho; os requisitos não funcionais da Seção 3.5 foram verificados pela configuração do projeto e pelos resultados dos testes.

A combinação dessas técnicas se justifica pela triangulação, que compara dados de diferentes fontes e utiliza diferentes métodos sobre um mesmo objeto para tornar as informações mais precisas (PRODANOV; FREITAS, 2013, p. 129). Assim, os testes automatizados verificam as regras de negócio de forma isolada e repetível, enquanto a verificação funcional confirma o comportamento do sistema durante o uso do aplicativo.

3.4. Técnica de Análise dos Dados

Como cada técnica de coleta deve ter um tratamento adequado à sua natureza (PRODANOV; FREITAS, 2013, p. 113), os dados foram analisados de três formas. Os dados sobre as soluções de mercado e as rotinas do varejo foram analisados de forma qualitativa, para definir os requisitos do sistema. Os resultados dos testes automatizados foram analisados pela quantidade de testes aprovados e pela cobertura de código, que mostra quanto do código foi executado pelos testes. Já a verificação funcional foi analisada pela contagem de requisitos atendidos, parcialmente atendidos e não atendidos. Esses resultados são apresentados e discutidos na Seção 5.

3.5. Levantamento e Definição dos Requisitos

A primeira etapa do desenvolvimento consistiu no levantamento das funcionalidades necessárias ao sistema, realizado a partir de duas fontes principais.

A primeira foi a análise de soluções já existentes no mercado, considerando tanto sistemas proprietários, como Square POS e Loyverse POS, quanto sistemas de código aberto, como Simple POS e Chromis POS. Essa análise permitiu identificar o conjunto de funcionalidades considerado essencial em um sistema de ponto de venda, bem como recursos cuja complexidade os tornava inviáveis dentro do prazo estabelecido para o projeto.

A segunda fonte foi o conhecimento prévio das rotinas operacionais do varejo, obtido em atividade profissional de suporte técnico a sistemas de gestão empresarial e de frente de caixa. Esse conhecimento contribuiu para a identificação das falhas mais recorrentes no controle manual e das operações consideradas indispensáveis no cotidiano do pequeno comerciante.

Os requisitos elicitados foram posteriormente avaliados em conjunto com o orientador do trabalho, etapa que definiu o escopo final da aplicação e delimitou o que seria implementado dentro do cronograma estabelecido.

Foram definidos doze requisitos funcionais, organizados nos grupos apresentados a seguir, e quatro requisitos não funcionais, verificáveis pela configuração do projeto e pelos testes, listados ao final da Quadro 3.

Quadro 3. Requisitos funcionais e não funcionais do sistema

Grupo	Requisitos funcionais
Cadastros	Produtos com grupos e controle de estoque (RF01), clientes (RF02), fornecedores (RF03) e usuários vinculados a grupos de permissão configuráveis (RF04)
Caixa	Abertura e fechamento de turno com movimentações de entrada e saída (RF05)
Vendas	Registro de vendas com múltiplos itens, desconto e seleção da forma de pagamento (RF06) e cancelamento com estorno automático de estoque e caixa (RF07)
Estoque	Entrada manual vinculada opcionalmente a fornecedor (RF08) e saída manual, distinta das saídas geradas por venda (RF09)
Autenticação e acesso	Controle de acesso por rota e verbo HTTP (RF10) e autenticação por credenciais com token de acesso e token de renovação (RF12)
Relatórios	Vendas por período, produtos mais vendidos, fluxo de caixa e posição de estoque (RF11)

Grupo	Requisitos funcionais
Não funcionais	Autenticação obrigatória em todas as rotas de negócio (RNF01); comunicação em HTTPS no ambiente publicado (RNF02); operações que alteram mais de uma coleção executadas em transação (RNF03); cobertura mínima de 90% nos testes automatizados da API (RNF04)

Fonte: elaborada pelo autor (2026).

O controle de acesso foi projetado a partir de um conjunto fechado de grupos de usuários, composto por Administrador e Operador, cuja criação e exclusão não são expostas pela API. As permissões atribuídas a cada grupo, no entanto, permanecem configuráveis por rota e por verbo HTTP. Essa decisão restringe a superfície de configuração do sistema e evita que a criação livre de grupos resulte em combinações de permissões não previstas, preservando a flexibilidade necessária para adequar o sistema à realidade de cada estabelecimento.

A partir desses requisitos foi definida a estrutura de entidades do sistema. O modelo adota o cliente e o fornecedor como cadastros independentes, o produto vinculado a um grupo de produtos, e o usuário vinculado a um grupo de permissões. As operações comerciais organizam-se em torno do caixa, que representa um turno de trabalho e ao qual as vendas são vinculadas. Cada venda, por sua vez, possui itens registrados em coleção própria e gera a respectiva baixa de estoque.

O fluxo principal definido foi:

Usuário → Caixa → Venda → Itens da venda → Movimentação de estoque

3.6. Organização do Desenvolvimento

Para a organização das atividades foi adotado um fluxo de trabalho baseado em Kanban, operacionalizado por meio dos quadros de issues da plataforma GitLab. O projeto foi estruturado em dois repositórios distintos, pdv_api e pdv_mobile, cada um com seu próprio quadro de acompanhamento, no qual as tarefas passaram pelos estados Aberto, Fazendo e Concluído.

O trabalho ocorreu em fluxo contínuo, sem divisão em ciclos de duração fixa. Cada *issue* correspondia a uma funcionalidade ou a um ajuste específico e recebia rótulos indicando sua natureza, como codificação ou testes, o que permitiu acompanhar a distribuição do esforço ao longo do projeto.

O controle de versão do código-fonte foi realizado por meio do Git. Cada *issue* originava uma ramificação própria, posteriormente integrada à ramificação principal por meio de uma requisição de incorporação, chamada merge request. Esse fluxo garantiu rastreabilidade entre a funcionalidade planejada, o código produzido e sua integração ao sistema. Para as mensagens de commit adotou-se, na maior parte do histórico, o padrão Conventional Commits, com prefixos como feat, fix, build e chore para identificar a natureza de cada alteração.

O cronograma do projeto compreendeu o período de março a novembro de 2026, iniciando pela elaboração do canvas do trabalho, seguido da elicitação de requisitos, da codificação, dos testes automatizados e, por fim, da escrita e validação do artigo.

3.7. Tecnologias e Ferramentas Utilizadas

As tecnologias foram escolhidas conforme as necessidades de cada parte do sistema. Os motivos da escolha do Kotlin e do MongoDB foram apresentados nas Seções 2.5 e 2.6.

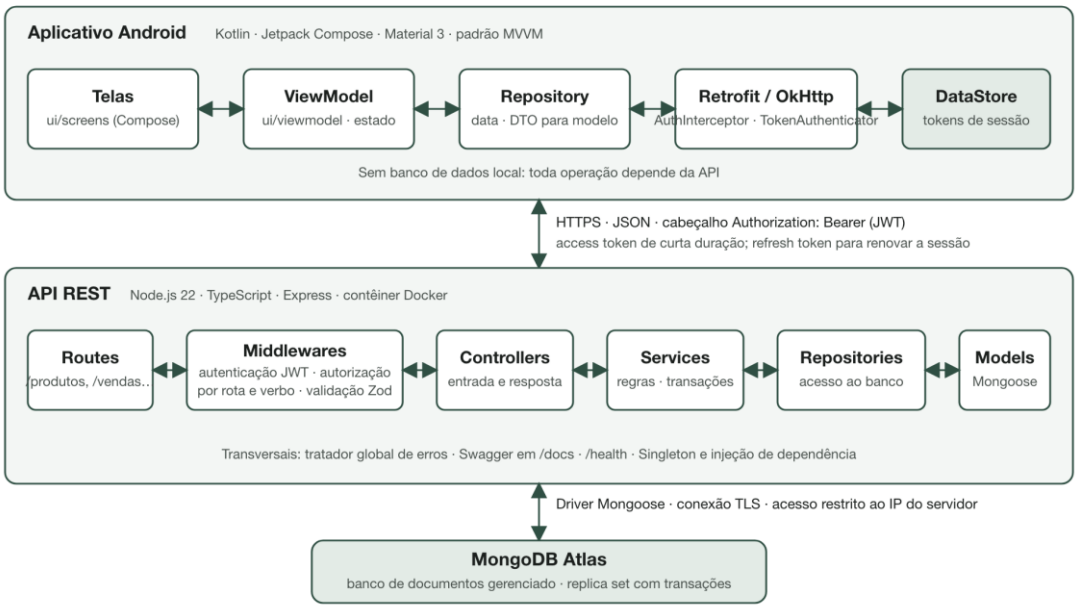
A API foi desenvolvida em TypeScript, executada no Node.js e construída com o Express, que organiza as rotas HTTP. Os dados ficam armazenados no MongoDB Atlas, e o Mongoose define a estrutura dos documentos salvos. A autenticação utiliza tokens JWT (JSON Web Token), e o Docker foi usado para padronizar o ambiente de execução da API.

O aplicativo Android foi desenvolvido em Kotlin, com as telas construídas em Jetpack Compose, e se comunica com a API por meio da biblioteca Retrofit. Os testes automatizados foram feitos com Jest na API e com JUnit no aplicativo. Por fim, as telas foram prototipadas no Figma, e o código foi versionado no GitLab, conforme descrito na Seção 3.6.

4. Desenvolvimento e Implementação

O sistema adota arquitetura cliente-servidor organizada em três camadas: o aplicativo Android, executado no dispositivo do operador; a API REST, que concentra as regras de negócio e o controle de acesso; e o banco de dados MongoDB Atlas. A comunicação entre o aplicativo e a API ocorre por HTTP, com mensagens em JSON e autenticação por token JWT transmitido no cabeçalho Authorization. A Figura 1 apresenta essa organização, e as seções seguintes detalham cada camada.

Figura 1. Arquitetura geral do sistema



Fonte: elaborada pelo autor (2026).

4.1. Arquitetura da API

A API foi implementada em TypeScript com Node.js e Express. A estrutura segue uma divisão em camadas com os mesmos nomes utilizados no código. A camada routes define os endpoints HTTP; middlewares trata autenticação, autorização, confirmação de senha

e fuso horário; controllers recebe as requisições e aciona os casos de uso; services concentra as regras de negócio; repositories isola o acesso ao banco de dados; e models representa as coleções Mongoose.

A documentação Swagger é servida nas rotas /docs e /docs.json, e a rota /health informa se o processo está respondendo e o estado da conexão com o MongoDB.

As principais rotas de negócio foram montadas com prefixos próprios, como /produtos, /vendas, /caixas, /entrada-estoque, /saida-estoque e /relatorios. Essa organização também contribui para o controle de autorização, pois as permissões dos grupos de usuários são verificadas com base na rota acessada e no verbo HTTP utilizado.

Na implementação, controllers e repositories são instâncias únicas (Singleton) que recebem suas dependências pelo construtor, com valor padrão, o que permite substituí-las por versões simuladas nos testes. Os dados de entrada são validados com a biblioteca Zod antes das regras de negócio, com erros convertidos em respostas 400; as buscas usam um FilterBuilder por entidade; a paginação fica em um repositório base; e os erros de todas as rotas vão a um único tratador global. Registros com histórico são preservados: um cliente que já possui vendas é apenas inativado.

4.2. Modelagem

A persistência foi implementada com MongoDB Atlas e Mongoose. A modelagem foi estruturada em coleções relacionadas aos cadastros, ao controle de acesso e às operações comerciais. Entre as principais coleções estão usuarios, usuarios_grupo, tokens, clientes, fornecedores, produtos, produtos_grupo, entrada_estoque, saida_estoque, caixa, vendas e vendas_itens. Há ainda a coleção counters, de uso interno, responsável pela geração do número sequencial das vendas. O contador é incrementado de forma atômica dentro da mesma transação que finaliza a venda, de modo que uma transação abortada também desfaz o incremento, evitando lacunas na numeração das vendas. O número é atribuído somente no momento da finalização, e vendas canceladas conservam o número recebido.

O produto é uma entidade central do sistema, pois participa tanto das vendas quanto das movimentações de estoque. Cada produto possui grupo, preço, estoque, estoque mínimo para alerta de reposição, status de ativação e indicação sobre permitir ou não estoque negativo. As entradas de estoque registram reposições vinculadas a um usuário e, opcionalmente, a um fornecedor. As saídas de estoque podem ser manuais ou geradas automaticamente por uma venda finalizada.

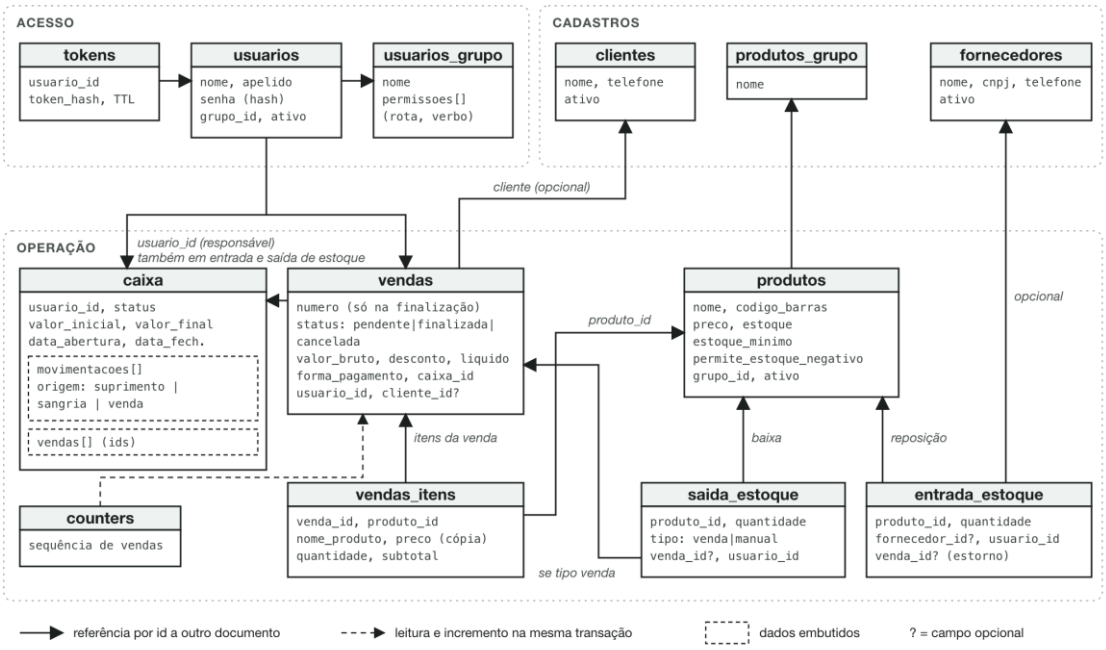
A venda foi modelada em duas coleções: vendas, que armazena o cabeçalho da operação, e vendas_itens, que armazena os itens individualmente. Essa separação permite consultar a venda como transação principal e, quando necessário, recuperar seus produtos associados. Os itens registram uma cópia do nome e do preço do produto no momento da venda, evitando que alterações futuras no cadastro modifiquem o histórico comercial.

O caixa representa o turno de operação do usuário. Ele registra valor inicial, status, data de abertura, data de fechamento, movimentações, vendas vinculadas, totais de entrada, totais de saída e valor final. As movimentações podem ter origem em suprimento, sangria ou venda, permitindo diferenciar entradas manuais, retiradas de dinheiro e valores provenientes de vendas finalizadas.

Quanto a dados pessoais, o sistema trata duas categorias de titulares: usuários do aplicativo (nome, apelido e senha, esta armazenada apenas como hash) e clientes do

comércio (nome, CPF, e-mail e telefone, informados pelo comerciante, com identificação opcional na venda); fornecedores são pessoas jurídicas. A finalidade é a operação do caixa e a identificação do responsável por cada movimentação, e o acesso é restrito por autenticação e por grupo de permissão (Seção 4.3). Esta etapa não contemplou consentimento do cliente, prazo de retenção, anonimização na exclusão nem registro das operações de tratamento previstos na Lei Geral de Proteção de Dados (BRASIL, 2018), o que fica declarado como limitação na Seção 5.4.

Figura 2. Modelo de documentos das coleções do sistema



Fonte: elaborada pelo autor (2026).

4.3. Autenticação e Autorização

O controle de acesso foi implementado com JWT (JONES; BRADLEY; SAKIMURA, 2015). Ao realizar login, a API emite um token de acesso e um token de renovação. O token de acesso é utilizado nas rotas protegidas, enquanto o token de renovação permite manter a sessão ativa por meio da rota de refresh. Para reduzir riscos de exposição, o token de renovação não é armazenado diretamente no banco; a API salva apenas seu hash na coleção tokens, que possui expiração automática dos registros. Ao encerrar a sessão, o token de renovação é revogado e não pode mais ser utilizado.

As rotas protegidas utilizam dois middlewares principais. O primeiro valida o token de acesso e verifica se ainda existe sessão ativa para o usuário. O segundo realiza a autorização com base no grupo do usuário. O grupo possui permissões por rota e por ação, relacionando verbos HTTP a operações como buscar, enviar, modificar e excluir. Usuários administradores possuem acesso amplo, enquanto operadores ficam restritos às permissões configuradas. Na configuração inicial, o grupo Operador possui a permissão de excluir apenas na rota /vendas, necessária para remover itens do carrinho e descartar a própria venda pendente.

O Quadro 4 mostra o trecho em que o verbo HTTP da requisição é convertido na ação correspondente à permissão. Para usuários que não são administradores, o verbo

PUT é recusado antes da consulta às permissões, pois as atualizações são feitas apenas pelo verbo PATCH; a recusa responde com o status 403 e o código interno METHOD_NOT_ALLOWED. O administrador, que tem acesso amplo, não passa por essa verificação.

Quadro 4. Trecho do middleware de autorização

```
const verbosHttp: Record<string, string> = {
  GET: 'buscar',
  POST: 'enviar',
  PATCH: 'modificar',
  DELETE: 'excluir',
  PUT: 'substituir',
};
const verboHttp = req.method.toUpperCase();
const campoPermissao = verbosHttp[verboHttp];
if (verboHttp === 'PUT') {
  return next(httpError(
    ERROR_CODES.METHOD_NOT_ALLOWED,
    'Verbo HTTP PUT não é permitido. Use PATCH para modificações.',
    HTTP_STATUS.FORBIDDEN
  ));
}
```

Fonte: elaborada pelo autor (2026).

No aplicativo Android, os tokens são armazenados localmente por meio de uma implementação de SessaoLocal baseada em DataStore. O cliente HTTP utiliza um AuthInterceptor para adicionar o cabeçalho Authorization às requisições protegidas.

Quando a API retorna erro de autenticação por expiração do token, o TokenAuthenticator tenta renovar a sessão automaticamente e repete a requisição original com o novo token. Caso a API recuse a renovação, a sessão é limpa e o usuário retorna ao fluxo de login; uma falha de rede durante a renovação apenas interrompe a requisição, sem encerrar a sessão.

4.4. Fluxo de Venda e Caixa

O fluxo de venda foi implementado como um carrinho persistido no servidor. A venda nasce com status pendente quando o operador adiciona o primeiro produto no PDV. Enquanto estiver pendente, o aplicativo pode adicionar itens, remover itens, alterar quantidades, aplicar desconto e definir a forma de pagamento. Essa abordagem evita que o carrinho exista apenas na memória do aplicativo e permite recuperar uma venda pendente caso o usuário retorne à tela posteriormente.

Para criar uma venda, a API exige que exista um caixa aberto para o operador autenticado. A venda fica vinculada ao caixa do turno em que foi criada. No aplicativo, a tela de PDV verifica a situação do caixa antes de liberar a operação. Se não houver caixa aberto, o usuário é orientado a abrir um turno antes de vender.

A finalização da venda ocorre em transação no MongoDB. Nesse processo, a API relê a venda e seus itens, calcula o valor bruto, aplica o desconto, define o valor líquido, gera um número sequencial de venda, altera o status para finalizada, baixa o estoque dos produtos, cria registros de saída de estoque do tipo venda e registra uma entrada no caixa com origem venda. Essa transação evita que a venda seja finalizada sem baixa de estoque ou sem movimentação correspondente no caixa.

O cancelamento possui tratamentos diferentes conforme o status da venda. Uma venda pendente pode ser descartada sem impacto em estoque ou caixa, e nesse caso a venda e seus itens são apagados, pois ainda não representam uma operação concluída. Já uma venda finalizada exige permissão administrativa, pois gera estorno. Nesse caso, a API altera o status para cancelada, devolve as quantidades ao estoque por meio de entradas de estoque e registra uma movimentação compensatória no caixa. Se o produto de algum item ou o caixa da venda não existirem mais, a transação é abortada com erro 404 e nada é alterado, de modo que não se registra entrada de estoque sem reposição nem cancelamento sem compensação no caixa.

O módulo de caixa permite abrir turno, consultar caixa aberto, listar turnos, visualizar extrato, registrar suprimento, registrar sangria e fechar caixa. A API impede que um usuário tenha mais de um caixa aberto, e ao fechar o caixa as vendas que ainda estavam pendentes no turno são descartadas na mesma operação. No aplicativo, a tela de caixa mostra o turno atual, suas movimentações, os totais e a consulta de turnos anteriores.

4.5. Implementação do Aplicativo Móvel

O aplicativo foi desenvolvido em Kotlin com Jetpack Compose, Material 3, Navigation Compose, Retrofit, OkHttp, Gson, Coroutines e DataStore. A tela principal é controlada pela MainActivity, que alterna entre o fluxo de login e o AppNavHost conforme o estado de autenticação. Após o login, o usuário acessa uma navegação com abas principais para início, produtos, PDV, vendas e mais opções, além de rotas internas para caixa, estoque, clientes, fornecedores e usuários. O aplicativo adota o padrão arquitetural Model-View-ViewModel (MVVM), no qual as telas reagem ao estado exposto pelos modelos de visão. Nas telas, o aplicativo se identifica como PDV Mobile, nome de trabalho adotado durante o desenvolvimento; Venda Fácil PDV é o nome do sistema completo, aplicativo e API.

A camada de rede é centralizada no RetrofitClient, que configura a URL base da API, timeouts, interceptadores, autenticação automática e criação das interfaces Retrofit. Em modo de desenvolvimento, o aplicativo utiliza http://localhost:3000/, com suporte a adb reverse para redirecionar o tráfego do dispositivo Android para a API local. Em build de release, a URL base é a da API publicada (Seção 4.8). Em qualquer build, o usuário pode informar outro endereço HTTPS na tela de login, recurso pensado para apontar o aplicativo a uma instalação própria da API.

Cada tela que carrega dados possui uma classe em ui/viewmodel responsável por carregar dados, manter estado de interface e chamar o repositório correspondente em data. Os repositórios convertem DTOs (Data Transfer Objects) vindos de network/dto para modelos de domínio definidos em model. Essa separação foi aplicada em módulos como venda, caixa, produtos, estoque, clientes, fornecedores, usuários e início; o histórico de vendas reutiliza o repositório de vendas.

A tela inicial resume as vendas do dia, as dos últimos sete dias e a situação do caixa. O PDV busca produtos por nome ou código de barras, filtra por grupo, monta o carrinho, valida o estoque conhecido ao adicionar itens e alterar quantidades, aplica desconto, escolhe a forma de pagamento e finaliza a venda. O histórico lista vendas por período e status, com cancelamento pelo administrador; a tela de estoque reúne entradas e saídas, com filtros e correções manuais; os cadastros cobrem produtos, clientes, fornecedores e usuários; a tela de relatórios reúne os quatro relatórios por período; e a de

permissões define os verbos liberados por rota para cada grupo. As permissões recebidas no login ocultam as ações não autorizadas, mas a autorização definitiva é feita pela API.

4.6. Testes Automatizados

A validação da API foi realizada por meio de uma suíte de testes automatizados executada com Jest, complementada pelo Supertest nas verificações feitas sobre a aplicação Express. A suíte, cujos números constam da Seção 5.2, segue a organização da pirâmide de testes (Seção 2.8), com testes de unidade e de integração.

Os testes de unidade verificam isoladamente o comportamento de controladores, serviços, middlewares, repositórios, rotas e utilitários, com as dependências externas substituídas por implementações controladas. Os testes de integração concentram-se nos fluxos que envolvem mais de uma coleção e exigem consistência transacional, como a montagem do carrinho, a finalização e o cancelamento da venda, as movimentações de estoque, as operações de caixa e a geração dos relatórios.

Essa separação permitiu validar tanto as regras de negócio isoladas quanto os efeitos colaterais das transações. No caso da finalização da venda, por exemplo, os testes verificam se a operação efetivamente reduz o estoque dos produtos, gera os registros de saída correspondentes e credita o valor no caixa aberto, garantindo que nenhuma dessas etapas ocorre isoladamente.

O aplicativo também possui testes automatizados, executados com JUnit, que verificam os repositórios responsáveis pela comunicação com a API, usando um servidor simulado no lugar da API real, e parte dos modelos de visão, como os das telas de venda, caixa, histórico, permissões e relatórios.

4.7. Uso de Inteligência Artificial

Foram usadas ferramentas de inteligência artificial da Anthropic na escrita e na correção do código da API e do aplicativo, na criação dos testes e na revisão deste texto: o Claude Code, no editor VS Code, com os modelos Claude Opus 4.8, Claude Opus 5.5 e Claude Sonnet 5.5, e o Claude, com acesso só de leitura ao documento. Os repositórios não têm arquivos de configuração da ferramenta.

O uso seguiu um processo fixo: pedido de análise somente leitura do código existente, instruções detalhadas do que alterar e do que manter, conferência e testes pelo autor antes de cada commit, feito por ele. Na escrita, o Claude sugeriu e redigiu trechos que o autor revisou, e as citações foram conferidas nas fontes originais, com a página de cada trecho. Requisitos, decisões de arquitetura, regras de negócio, conferência dos resultados, testes no aparelho e a versão final do sistema e do texto ficaram sob responsabilidade do autor.

4.8. Publicação da API

A API é executada em um contêiner Docker, montado em duas etapas a partir da imagem node:22-slim. Na primeira etapa, o código TypeScript é compilado para JavaScript. Na segunda, são instaladas apenas as dependências de produção e copiado somente o código compilado, o que deixa a imagem final menor. O processo é executado por um usuário sem privilégios de administrador e atende na porta 3000. O arquivo docker-compose define uma verificação periódica da rota /health, que confirma se a API está respondendo;

o estado da conexão com o banco é informado no corpo da resposta, mas não altera o resultado da verificação.

A publicação foi feita em uma máquina virtual com Ubuntu Server 24.04, criada na plataforma Light Cloud do FSLab, laboratório do IFRO Campus Vilhena, com um processador virtual e 2 GB de memória. O código foi baixado do GitLab com uma chave de acesso somente para leitura, e a API foi iniciada com o Docker Compose. As configurações sensíveis, como o endereço do banco de dados e as chaves de assinatura dos tokens, não ficam no código e são informadas por variáveis de ambiente. O banco de dados permanece no MongoDB Atlas, que aceita conexões apenas a partir do endereço da máquina virtual. A API ficou disponível pelo endereço https://pdv-api.app.fslab.dev, com acesso por HTTPS, e a versão de release do aplicativo foi configurada para acessar esse endereço. A terminação TLS e o certificado do domínio são providos pela infraestrutura do FSLab, por proxy reverso externo ao contêiner da API, que atende apenas em HTTP na rede interna da máquina virtual.

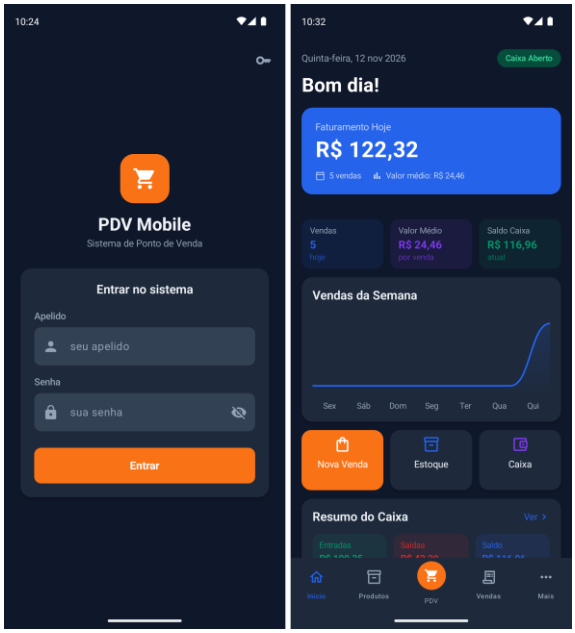
5. Apresentação e Análise dos Resultados

5.1. O Sistema Desenvolvido

Esta seção apresenta as principais telas do aplicativo e os requisitos da Quadro 3 que cada uma atende.

A Figura 3 mostra a tela de login e a tela inicial. O acesso exige apelido e senha, conforme o requisito RF12. Após o login, a tela inicial resume o faturamento do dia, a quantidade de vendas, o valor médio por venda, o saldo do caixa e um gráfico com as vendas da semana; os relatórios completos ficam na tela própria, mostrada na Figura 8.

Figura 3. Tela de login e tela inicial

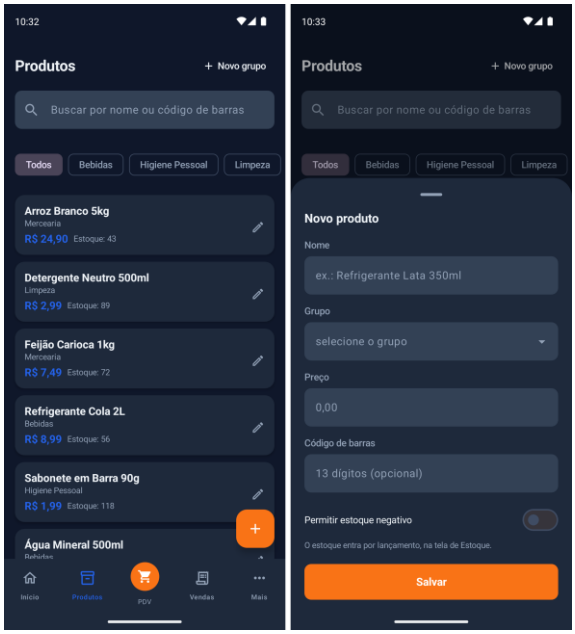


Fonte: elaborada pelo autor (2026), com dados de demonstração.

A Figura 4 mostra a lista de produtos e o formulário de cadastro, que atendem ao requisito RF01. Cada produto pertence a um grupo, como Mercearia ou Bebidas, e tem preço, código de barras e a opção de permitir ou não estoque negativo. A quantidade em

estoque não é digitada no cadastro, pois só muda por lançamentos na tela de Estoque ou por vendas.

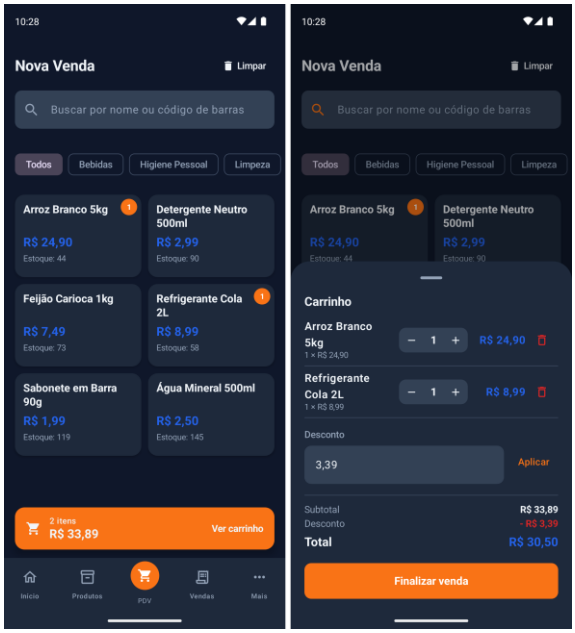
Figura 4. Lista e cadastro de produtos



Fonte: elaborada pelo autor (2026), com dados de demonstração.

As Figuras 5 e 6 mostram o fluxo de venda, que atende ao requisito RF06. Na tela de PDV, o operador busca os produtos pelo nome ou pelo código de barras e toca neles para adicioná-los ao carrinho, onde pode alterar as quantidades, remover itens e aplicar desconto.

Figura 5. Tela de PDV e carrinho

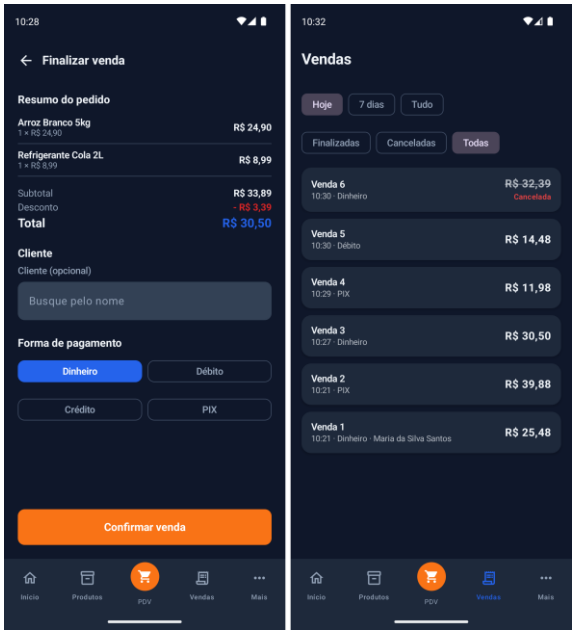


Fonte: elaborada pelo autor (2026), com dados de demonstração.

Na finalização, o operador escolhe a forma de pagamento e, se quiser, identifica o cliente. Ao confirmar, a venda atualiza o estoque e o caixa na mesma operação, como

descrito na Seção 4.4. As vendas ficam registradas no histórico, que pode ser filtrado por período e por situação.

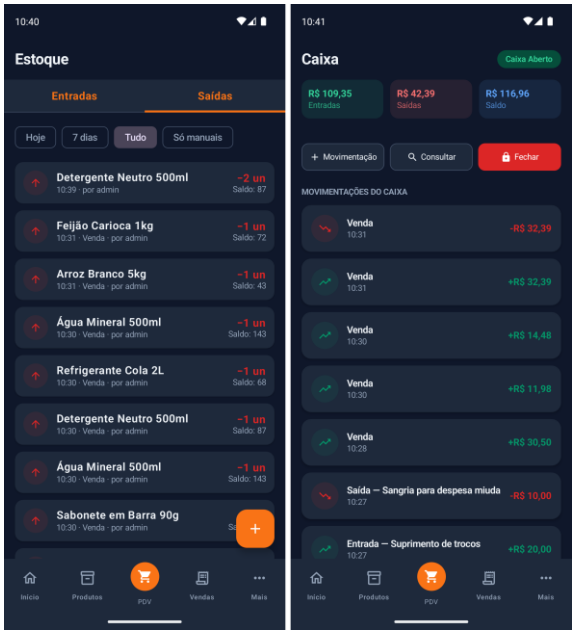
Figura 6. Finalização da venda e histórico de vendas



Fonte: elaborada pelo autor (2026), com dados de demonstração.

A Figura 7 mostra as saídas de estoque e a tela de caixa. Cada saída mostra o produto, a quantidade, o usuário responsável e o saldo atual do produto, e as baixas geradas por vendas aparecem identificadas como Venda, atendendo ao requisito RF09. A tela de caixa exibe o total de entradas, de saídas e o saldo do turno aberto, além das movimentações, atendendo ao requisito RF05.

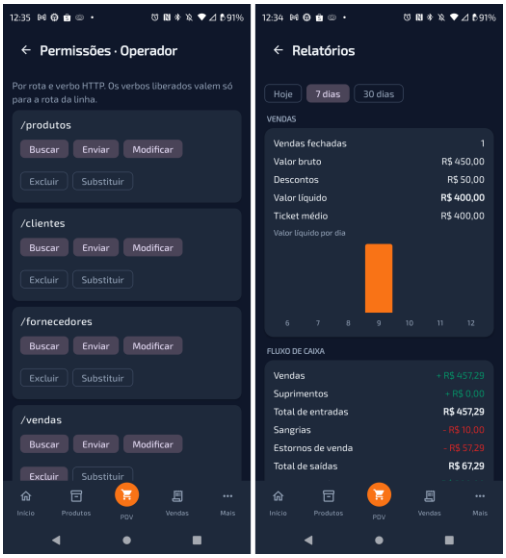
Figura 7. Saídas de estoque e tela de caixa



Fonte: elaborada pelo autor (2026), com dados de demonstração.

A Figura 8 mostra as telas de permissões e de relatórios. Na primeira, o administrador define, por grupo, os verbos HTTP liberados em cada rota da API (RF04). A segunda reúne, para hoje, sete ou trinta dias, vendas por dia, fluxo de caixa, produtos mais vendidos e posição do estoque (RF11).

Figura 8. Permissões do grupo Operador e tela de relatórios



Fonte: elaborada pelo autor (2026), com dados de demonstração.

5.2. Avaliação do Sistema

Os testes automatizados da API, escritos com Jest e Supertest, foram executados novamente em 27 de outubro de 2026. Os 106 arquivos de teste reuniram 1.184 casos, e todos foram aprovados. Desses casos, 642 são testes de unidade sem banco de dados, 103 são testes de unidade com banco de dados em memória, 233 são testes de integração dos serviços e 206 são testes de integração das rotas HTTP.

A cobertura de código ficou acima do limite mínimo de 90% configurado no projeto em todas as medidas, com 99,00% das instruções, 91,68% das ramificações, 98,24% das funções e 99,06% das linhas executadas pelos testes. No aplicativo, os 248 testes escritos com JUnit, distribuídos em 19 classes, também foram aprovados (um 20º arquivo, gerado pelo modelo de projeto do Android Studio, contém um caso trivial e não é contado), mas a cobertura não foi medida, como apontado na Seção 5.4.

A verificação funcional seguiu o roteiro de vinte e quatro casos de teste descrito na Seção 3.3, executado no aplicativo instalado em um celular Android e conectado à API publicada. O Quadro 5 apresenta, para cada requisito, os dois casos do roteiro que o verificam, o resultado obtido e a situação final, e inclui os requisitos não funcionais, verificados pela configuração do projeto e pelos resultados dos testes.

Quadro 5. Resultado da verificação funcional dos requisitos, com os casos do roteiro

Requisito	Casos verificados	Resultado obtido	Situação
RF01 Produtos	C01 cadastrar produto com grupo, preço e código; C02 procurar campo de quantidade no cadastro	Produto cadastrado; quantidade ausente do formulário, estoque alterado só por vendas e lançamentos	Atendido

Requisito	Casos verificados	Resultado obtido	Situação
RF02 Clientes	C03 cadastrar e editar; C04 inativar e consultar inativos	Operações realizadas; inativos consultados pelo filtro	Atendido
RF03 Fornecedores	C05 cadastrar e editar; C06 telefone com mais de onze dígitos	Operações realizadas; telefone restrito a onze dígitos	Atendido
RF04 Usuários e grupos	C07 criar usuário no grupo Operador; C08 alterar um verbo do grupo Operador na tela de permissões e salvar	Usuário criado e vinculado; permissão gravada na API e refletida na próxima sessão do operador	Atendido
RF05 Caixa	C09 abrir caixa, suprimento, sangria e sangria acima do saldo; C10 fechar e consultar turnos	Abertura, suprimento e sangria corretos, com bloqueio de sangria sem saldo; fechamento correto	Atendido
RF06 Vendas	C11 itens, quantidade, desconto, pagamento e finalização; C12 iniciar venda sem caixa aberto	Venda finalizada com baixa de estoque e entrada no caixa; operador orientado a abrir o turno	Atendido
RF07 Cancelamento	C13 cancelar venda finalizada como administrador; C14 tentar como operador	Estorno realizado pelo administrador; recusado para o operador	Atendido
RF08 Entrada de estoque	C15 entrada com fornecedor; C16 entrada sem fornecedor	Entradas registradas nos dois casos	Atendido
RF09 Saída de estoque	C17 saída manual dentro do estoque; C18 saída acima do estoque sem estoque negativo	Saída registrada como manual; bloqueio por estoque insuficiente	Atendido
RF10 Controle de acesso	C19 operador procura ações restritas no aplicativo; C20 operador chama rota não permitida na API	Ações ocultas no aplicativo; recusadas pela API	Atendido
RF11 Relatórios	C21 tela inicial com vendas no dia; C22 abrir a tela de relatórios e trocar o período para sete dias	Resumo do dia exibido; os quatro relatórios exibidos com os valores do período	Atendido
RF12 Autenticação	C23 senha errada e depois correta; C24 uso após expirar o token e saída	Recusa e depois acesso; renovação automática e saída corretos	Atendido
RNF01 Autenticação obrigatória	Rotas de negócio exigem token válido	Middlewares em todas as rotas, exceto login e refresh; testes de rota confirmam 401 sem token	Atendido
RNF02 HTTPS	API publicada acessível apenas por HTTPS	Endereço publicado em HTTPS; aplicativo em release sem tráfego em texto claro	Atendido
RNF03 Transações	Finalização, cancelamento, abertura e fechamento de caixa em transação	Operações em sessão transacional, com testes de integração dos efeitos conjuntos	Atendido
RNF04 Cobertura mínima	Cobertura de 90% nas quatro medidas	99,00%, 91,68%, 98,24% e 99,06%, com limite configurado no projeto	Atendido

Fonte: elaborada pelo autor (2026). Pré-condição comum: usuário autenticado como administrador, salvo quando o caso indica operador; C09 a C18 exigem caixa aberto.

Assim, os doze requisitos funcionais e os quatro requisitos não funcionais foram atendidos de forma completa.

5.3. Implicações Teóricas e Gerenciais

Os resultados se relacionam com os pontos levantados na fundamentação teórica. Pereira et al. (2009) associam o fechamento de pequenas empresas a falhas de controle financeiro, e o Sebrae relaciona esse fechamento à falta de experiência na gestão do negócio (SEBRAE, 2013, p. 23). O Venda Fácil PDV foi projetado para atender às rotinas que a literatura associa a essas causas: registra cada venda, cada movimentação de caixa e cada baixa de estoque em uma única operação, de modo a dispensar anotações separadas. Essa

concepção é compatível com a ideia de que o controle manual aumenta o risco de erros e dificulta a obtenção de informações em tempo real (TACHIZAWA, 2002 apud CARVALHO; MAZZOTTI, 2024, p. 7), e com a defesa de um controle de caixa sistemático, acompanhado em tempo real (RAMOS, 2024).

Do ponto de vista gerencial, o comerciante passa a acompanhar pelo celular o faturamento do dia, o valor médio por venda e o saldo do caixa, como mostra a Figura 3. O caixa controlado por turno mostra quanto entrou e saiu em cada período, e a baixa automática do estoque evita que a quantidade registrada fique diferente da real. Espera-se que esses recursos ajudem o comerciante a decidir compras e reposições com base em dados, e não em anotações separadas; esse efeito, porém, não foi medido neste trabalho, que se limitou a verificar o funcionamento do sistema.

Em relação ao que foi planejado, os resultados atendem aos objetivos específicos definidos na Introdução: os requisitos foram levantados, a API e o aplicativo foram desenvolvidos e integrados, a API foi publicada e a solução foi validada por testes automatizados e pela verificação dos principais fluxos de uso. A diferença entre o planejado e o entregue está em dois requisitos atendidos em parte, cujas funções existem na API, mas ainda não possuem tela no aplicativo.

5.4. Limitações e Sugestões para Trabalhos Futuros

O sistema apresenta limitações que devem ser consideradas. A principal é a falta de funcionamento offline, pois o aplicativo não possui banco de dados local e toda operação depende da conexão com a API. O sistema também não emite documento fiscal, como a Nota Fiscal de Consumidor Eletrônica (NFC-e), e não se integra a meios de pagamento, como maquininhas de cartão, de modo que a forma de pagamento é apenas registrada na venda.

Quanto à avaliação, o sistema não foi testado por comerciantes em uma rotina real, os requisitos não foram validados com comerciantes e a cobertura de código dos testes do aplicativo não foi medida. Na parte de segurança, a API não limita o número de tentativas de login, aceita senhas de quatro caracteres no cadastro de usuários, e os tokens são gravados no aparelho sem criptografia, e o aplicativo tem o backup em nuvem do Android desativado para que eles não saiam do aparelho. O encerramento de sessão revoga apenas o token de renovação enviado, de modo que um token de acesso de outra sessão do mesmo usuário continua válido até expirar, e o aplicativo permite trocar o endereço da API na tela de login em qualquer versão. Quanto à proteção de dados, não foram implementados consentimento, prazo de retenção nem anonimização previstos na LGPD (BRASIL, 2018).

Como sugestões para trabalhos futuros, propõe-se implementar o funcionamento offline, com sincronização dos dados quando a conexão voltar, a emissão de NFC-e e a integração com meios de pagamento. Também se sugere limitar as tentativas de login, exigir senhas mais longas e criptografar os tokens no aparelho, incorporar os mecanismos da LGPD listados acima, configurar a execução automática dos testes a cada alteração no código e realizar uma avaliação com pequenos comerciantes em uso real, que é o que permitirá medir o efeito sobre a rotina do comerciante que este trabalho apenas projeta.

6. Considerações Finais

Este trabalho teve como objetivo desenvolver o Venda Fácil PDV, uma solução móvel para apoiar a gestão operacional de pequenos comerciantes. Retomando a questão da pesquisa, que perguntava como um aplicativo móvel integrado a uma API pode ser projetado e verificado para reunir o controle de vendas, estoque, caixa e clientes em uma única solução, os resultados mostram que isso é possível com um carrinho persistido no servidor e uma finalização em transação: cada venda finalizada atualiza o estoque e o caixa na mesma operação, o que elimina, por construção, a divergência entre o que foi vendido, o que saiu do estoque e o que entrou no caixa. Se isso reduz a dependência de registros manuais na rotina do comerciante é a hipótese que motivou o trabalho, e só uma avaliação em uso real poderá confirmá-la.

Quanto aos objetivos específicos, os requisitos foram levantados a partir da análise de soluções de mercado, conforme descrito na Seção 3.5. A API foi projetada com autenticação, autorização por grupo de usuários e regras de negócio que garantem a consistência das operações, como apresentado nas Seções 4.1 a 4.4. O aplicativo Android foi desenvolvido e integrado à API, oferecendo telas de venda, caixa, estoque, cadastros e um resumo das vendas, conforme a Seção 4.5. Por fim, a validação apresentada na Seção 5.2 mostrou que os 1.184 testes automatizados da API foram aprovados, com cobertura de 99,00% das instruções do código, que os 248 testes do aplicativo também foram aprovados e que os doze requisitos funcionais e os quatro não funcionais definidos foram atendidos de forma completa.

Do ponto de vista gerencial, o sistema oferece ao pequeno comerciante uma visão atualizada das vendas e do caixa a partir do próprio celular. O controle do caixa por turno, a baixa automática do estoque a cada venda e o resumo das vendas do dia e da semana são os recursos de maior peso nos resultados, pois foram projetados para as falhas de controle financeiro e de estoque que a fundamentação teórica aponta como causas frequentes do fechamento de pequenos negócios.

Entre as limitações, destaca-se a dependência de conexão com a internet, e como trabalhos futuros propõem-se o funcionamento offline, a emissão de NFC-e e a avaliação com pequenos comerciantes em uso real, conforme apresentado na Seção 5.4.

Referências

BORTOLOTO, V. Mongo DB, uma alternativa aos SGBDs relacionais. 2022. 48 f. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) – Faculdade de Tecnologia de Americana "Ministro Ralph Biasi", Americana, 2022. Disponível em: https://ric.cps.sp.gov.br/handle/123456789/12895. Acesso em: 1 out. 2026.

BRASIL. **Lei n. 13.709, de 14 de agosto de 2018**. Lei Geral de Proteção de Dados Pessoais (LGPD). Brasília, DF: Presidência da República, 2018. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm. Acesso em: 12 nov. 2026.

CARVALHO, D. I. A. Desenvolvimento Android moderno com Kotlin e Jetpack Compose: um guia prático baseado no app PlainTextKotlin. 2025. 116 f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia da Computação) – Faculdade de

Tecnologia, Universidade Federal do Amazonas, Manaus, 2025. Disponível em: http://riu.ufam.edu.br/handle/prefix/8925. Acesso em: 1 out. 2026.

CARVALHO, E. C.; MAZZOTTI, M. Aplicação das boas práticas de gestão de estoques em uma microempresa de Santa Catarina. **Revista Produção Online**, Florianópolis, v. 24, n. 2, e5220, p. 1-34, 2024. DOI: 10.14488/1676-1901.v24i2.5220. Disponível em: https://www.producaoonline.org.br/rpo/article/view/5220. Acesso em: 19 out. 2026.

CHROMIS. **Chromis POS**: open source point of sale. [S. l.], [s. d.]. Disponível em: https://chromis.co.uk/. Acesso em: 12 nov. 2026.

COSTA, J. M. da; ROSA, S. de O. Gestão empresarial aplicada a empresas de tecnologia. **Humanidades & Inovação**, Palmas, v. 7, n. 17, p. 36-43, 2020. Disponível em: https://revista.unitins.br/index.php/humanidadeseinovacao/article/view/4182. Acesso em: 1 out. 2026.

FIELDING, R. T. **Architectural styles and the design of network-based software architectures**. 2000. 162 f. Tese (Doutorado em Ciência da Computação) – University of California, Irvine, 2000. Disponível em: https://ics.uci.edu/ fielding/pubs/dissertation/top.htm. Acesso em: 12 nov. 2026.

FREITAS, M. G. T. C. de; PASCHOAL, L. R. P. Três importantes instrumentos que auxiliam a gestão financeira: uma breve revisão. **Ciência & Tecnologia**, Jaboticabal, v. 12, n. 1, p. 27-35, 2020. DOI: 10.52138/citec.v12i1.14. Disponível em: https://publicacoes.fatecjaboticabal.edu.br/citec/article/view/14. Acesso em: 1 out. 2026.

GOOGLE. **Android's Kotlin-first approach**. Android Developers, [s. d.]. Disponível em: https://developer.android.com/kotlin/first. Acesso em: 12 nov. 2026.

GRASSETTI, P. C. **Desenvolvimento do protótipo de um app Android para automação comercial de fichas de consumo**. 2018. 66 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Controle e Automação) – Instituto de Ciência e Tecnologia, Universidade Estadual Paulista, Sorocaba, 2018. Disponível em: https://www.sorocaba.unesp.br/Home/Graduacao/EngenhariadeControleeAutomacao/galdenoro1906/tg_pedro_final_finak.pdf. Acesso em: 12 nov. 2026.

JONES, M.; BRADLEY, J.; SAKIMURA, N. **JSON Web Token (JWT)**: RFC 7519. [S. l.]: Internet Engineering Task Force, 2015. DOI: 10.17487/RFC7519. Disponível em: https://www.rfc-editor.org/rfc/rfc7519. Acesso em: 12 nov. 2026.

LEAL, M. Modelo de implantação de sistemas PDV. 2020. 47 f. Trabalho de Conclusão de Curso (Tecnologia em Segurança da Informação) – Faculdade de Tecnologia de Americana "Ministro Ralph Biasi", Americana, 2020. Disponível em: http://ric.cps.sp.gov.br/handle/123456789/8306. Acesso em: 1 out. 2026.

LOYVERSE. **Loyverse**: sistema de ponto de venda gratuito. [S. l.], [s. d.]. Disponível em: https://loyverse.com/br. Acesso em: 12 nov. 2026.

OLIVEIRA, H. C. A. **SVEN**: um sistema para auxílio de gestão de estoque e vendas. 2018. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Universidade Federal Fluminense, Niterói, 2018. Disponível em: https://app.uff.br/riuff/handle/1/8527. Acesso em: 12 nov. 2026.

PEREIRA, M. F. et al. Fatores de inovação para a sobrevivência das micro e pequenas empresas no Brasil. **RAI – Revista de Administração e Inovação**, São Paulo, v. 6, n. 1, p. 50-65, 2009. Disponível em: https://revistas.usp.br/rai/article/view/79129. Acesso em: 1 out. 2026.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software**: uma abordagem profissional. 9. ed. Porto Alegre: AMGH, 2021.

PRODANOV, C. C.; FREITAS, E. C. de. **Metodologia do trabalho científico**: métodos e técnicas da pesquisa e do trabalho acadêmico. 2. ed. Novo Hamburgo: Feevale, 2013. 276 p. Disponível em: https://www.feevale.br/Comum/midias/0163c988-1f5d-496f-b118-a6e009a7a2f9/E-book%20Metodologia%20do%20Trabalho%20Cientifico.pdf. Acesso em: 9 out. 2026.

RAMOS, P. H. C. da S. A importância do fluxo de caixa e sua aplicação em micro e pequenas empresas. 2024. 29 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências Contábeis) – Universidade Estadual de Goiás, Aparecida de Goiânia, 2024. Disponível em: https://repositorio.ueg.br/jspui/handle/riueg/5848. Acesso em: 1 out. 2026.

SERVIÇO BRASILEIRO DE APOIO ÀS MICRO E PEQUENAS EMPRESAS (SEBRAE). **Sobrevivência das empresas no Brasil**. Brasília: Sebrae, 2013. 67 p. Disponível em: https://bibliotecas.sebrae.com.br/chronus/ARQUIVOS_CHRONUS/bds/bds.nsf/06d11457d15140966a4e33ee19a31991/$File/5607.pdf. Acesso em: 14 out. 2026.

SERVIÇO BRASILEIRO DE APOIO ÀS MICRO E PEQUENAS EMPRESAS (SEBRAE). Resultado positivo do PIB de 2023 confirma a relevância dos pequenos negócios. **Agência Sebrae de Notícias**, 1 mar. 2024. Disponível em: https://agenciasebrae.com.br/economia-e-politica/resultado-positivo-do-pib-de-2023-confirma-a-relevancia-dos-pequenos. Acesso em: 14 out. 2026.

SILVA, J. F. da. Automação: um estudo de caso em uma indústria alimentícia no município de Guarabira-PB. 2022. 22 f. Trabalho de Conclusão de Curso (Tecnologia em Gestão Comercial) – Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Guarabira, 2022. Disponível em: https://repositorio.ifpb.edu.br/handle/177683/2008. Acesso em: 1 out. 2026.

SILVA, K. B. A. da; MADEIRA, G. J. Gestão de estoques e lucro da empresa. In: CONGRESSO BRASILEIRO DE CUSTOS, 11., 2004, Porto Seguro. **Anais** [...]. São Leopoldo: ABC, 2004. Disponível em: https://anaiscbc.emnuvens.com.br/anais/article/view/2402. Acesso em: 1 out. 2026.

SILVA, L. F. C. da; VIEGAS, S. C. Sistema de gerenciamento de estoque e vendas para pequenos e médios negócios. **REFAQI – Revista de Gestão, Educação e Tecnologia**, Gravataí, v. 8, n. 1, 2022. Disponível em: https://refaqi.faqi.edu.br/index.php/refaqi/pt_BR/article/view/18. Acesso em: 12 nov. 2026.

SILVA, S. de O.; COSTA, B. F. SI: um exemplo de sistema distribuído com padrões de arquitetura atuais e populares. **Cadernos do IME – Série Informática**, Rio de Janeiro, v. 50, p. 58-77, dez. 2024. DOI: 10.12957/cadinf.2024.84436. Disponível em:

https://www.e-publicacoes.uerj.br/cadinf/article/view/84436. Acesso em: 14 out. 2026.

SILVA, Y. B. da. A importância do uso da tecnologia da informação em pequenos comércios na cidade de Sertãozinho-PB. 2026. 35 f. Trabalho de Conclusão de Curso (Tecnologia em Gestão Comercial) – Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Guarabira, 2026. Disponível em: https://repositorio.ifpb.edu.br/handle/177683/5221. Acesso em: 1 out. 2026.

SQUARE. **Square Point of Sale**. [S. l.], [s. d.]. Disponível em: https://squareup.com/us/en/point-of-sale. Acesso em: 12 nov. 2026.

URBANO, M. M.; ZOTTESSO, R. H. D.; RUIZ, E. A. C. Gestão financeira e gerenciamento de estoque resulta no conceito do PDV – Ponto de Venda. In: SEMANA DE TECNOLOGIA DA INFORMAÇÃO, 11., 2024, Paranavaí. **Anais** [...]. Paranavaí: IFPR, 2024. Disponível em: https://tecnoif.com.br/setif-anais/2024/resumos/32.pdf. Acesso em: 1 out. 2026.

VOCKE, H. **The practical test pyramid**. [S. l.]: martinfowler.com, 2018. Disponível em: https://martinfowler.com/articles/practical-test-pyramid.html. Acesso em: 12 nov. 2026.