INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS PICOS
CURSO ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

MATEUS VITAL DE SOUSA PEREIRA MARTINS

VELHA CAP HOMES: SISTEMA WEB PARA OTIMIZAR A BUSCA E LOCAÇÃO DE IMÓVEIS NA CIDADE DE OEIRAS – PI

PICOS-PI
2026

MATEUS VITAL DE SOUSA PEREIRA MARTINS

VELHA CAP HOMES: SISTEMA WEB PARA OTIMIZAR A BUSCA E LOCAÇÃO DE IMÓVEIS NA CIDADE DE OEIRAS – PI

Trabalho de Conclusão de Curso (Relatório Técnico de Software) apresentado como exigência parcial para obtenção do diploma do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos.

Orientador(a): Prof. Esp. Denis Costa Paiva

PICOS-PI

2026

MATEUS VITAL DE SOUSA PEREIRA MARTINS

VELHA CAP HOMES

Trabalho de Conclusão de Curso (Relatório Técnico de Software) apresentado como exigência parcial para obtenção do diploma do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos.

Aprovada em: _____/_____/_____.

BANCA EXAMINADORA

_____

Prof. Esp. Denis Costa Paiva (Orientador)

Instituto Federal do Piauí (IFPI)

_____

Prof. Me. Ciro Ferreira de Carvalho Junior

Instituto Federal do Piauí (IFPI)

_____

Prof. Me. Anthony Irlan Marques Luz

Instituto Federal do Piauí (IFPI)

AGRADECIMENTOS

Agradeço, primeiramente, a Deus, por me conceder saúde, força, sabedoria e perseverança ao longo desta caminhada, permitindo que eu superasse os desafios encontrados durante minha formação acadêmica.

Aos meus pais, Aldênia Maria Pereira Martins Sousa e Emanuel Vital de Sousa, dedico meu mais profundo e sincero agradecimento, por todo o amor, apoio incondicional e por serem o alicerce fundamental da minha jornada. Agradeço também à minha irmã, aos meus avós, tios, tias e a toda a minha família, pelo incentivo, carinho e confiança. Cada um de vocês teve um papel fundamental nessa trajetória, e sou eternamente grato por todo o apoio que me impulsionou a seguir em frente.

Estendo meus agradecimentos aos meus amigos e colegas, com quem compartilhei momentos de aprendizado, desafios e alegria. Misael, Leonardo, Erick, Kauê, Ezequiel, Artur, João, Matheus, Juan, Vinícius, Anthony e Caique, sem vocês, essa jornada não teria sido a mesma. Agradeço imensamente por toda ajuda, apoio e companheirismo, por tornarem essa caminhada mais leve me fazerem sentir que a jornada nunca foi difícil demais. Minha sincera gratidão a cada um de vocês.

Agradeço também ao meu companheiro de moradia, Dardimir pelo acolhimento, pela parceria e pelo apoio no momento que precisei.

Gostaria de agradecer ao meu orientador, Prof. Esp. Denis Costa Paiva, pela orientação e apoio ao longo dessa etapa. Seu acompanhamento foi valioso para o desenvolvimento deste trabalho.

Por fim, agradeço a todos que, de alguma forma, contribuíram direta ou indiretamente. Cada palavra de incentivo, gesto de apoio e demonstração de confiança teve papel fundamental nesta trajetória, tornando possível a superação dos desafios encontrados e fortalecendo minha determinação para concluir esta importante etapa da minha vida.

RESUMO

A crescente urbanização e a dinâmica social das cidades de menor porte impõem desafios informacionais à locação de imóveis, sobretudo em municípios como Oeiras – PI, onde o mercado imobiliário ainda se apoia em canais informais e dispersos em redes sociais. O presente trabalho descreve o desenvolvimento de um sistema web, denominado VelhaCap Homes, idealizado para centralizar anúncios de imóveis para locação na referida cidade, conectando locadores e locatários em uma única plataforma estruturada. A pesquisa, de natureza aplicada e abordagem qualitativa, valeu-se do modelo clássico de ciclo de vida (Cascata) e da adoção de uma arquitetura em camadas baseada no padrão *Model-View-Controller* (MVC). O sistema foi implementado com *stack* moderna, composta por *React* 19, *Vite*, *TanStack Start* e *Tailwind CSS* no *front-end*, e *Node.js*, *Express*, Prisma e *PostgreSQL* (Neon) no *back-end*, com persistência de imagens no *Cloudinary*, autenticação via JSON Web Token (JWT) e validação de dados por meio da biblioteca *Zod*. A análise de requisitos resultou em quinze requisitos funcionais e seis requisitos não funcionais, contemplando cadastro e autenticação de usuários, cadastro e gestão de imóveis, upload de imagens, busca e filtros, painel do anunciante, dashboard de estatísticas e mecanismos de contato. O sistema foi implantado em ambiente de produção, com o *back-end* hospedado no Render, o *front-end* na Vercel e o banco de dados no Neon, e submetido a testes de sistema que evidenciaram o atendimento aos requisitos não funcionais de usabilidade, desempenho, segurança, confiabilidade, disponibilidade e portabilidade. Conclui-se que a solução desenvolvida demonstrou potencial para mitigar a fragmentação informacional do mercado imobiliário de Oeiras – PI, indicando caminhos de evolução para futuras iterações.

Palavras-chaves: Moradia; Imóveis; Tecnologia; Aluguel; Oeiras–PI

ABSTRACT

The increasing urbanization and social dynamics of small cities impose informational challenges to real estate rental, especially in municipalities like Oeiras – PI, where the local market still relies on informal channels scattered across social networks. This work describes the development of a web system, named VelhaCap Homes, designed to centralize real estate rental listings in that city, connecting landlords and tenants in a single structured platform. The research, of applied nature and qualitative approach, adopted the classic Waterfall life cycle model and a layered architecture based on the Model-View-Controller (MVC) pattern. The system was implemented with a modern stack, comprising React 19, Vite, TanStack Start and Tailwind CSS on the front-end, and Node.js, Express, Prisma and PostgreSQL (Neon) on the back-end, with image persistence on Cloudinary, JSON Web Token (JWT) authentication and data validation through the Zod library. The requirements analysis yielded fifteen functional requirements and six non-functional requirements, covering user registration and authentication, property management, image upload, search and filters, landlord dashboard, statistics dashboard and contact mechanisms. The system was deployed to a production environment, with the back-end hosted on Render, the front-end on Vercel and the database on Neon, and was subjected to system tests that evidenced compliance with the non-functional requirements of usability, performance, security, reliability, availability and portability. It is concluded that the developed solution showed potential to mitigate the informational fragmentation of the real estate market in Oeiras – PI, indicating paths of evolution for future iterations.

keywords:*Housing; Real estate; Technology; Rent; Oeiras-PI*

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
API	Application Programming Interface
CSS	Cascading Style Sheets
ER	Entidade - Relacionamento
FK	Chave estrangeira (Foreign Key)
HTTP	HyperText Transfer Protocol
JWT	JSON Web Token
MVC	Model-View-Controller
ORM	Object-Relational Mapping
RF	Requisito Funcional
RNF	Requisito Não Funcional
SQL	Structured Query Language
SUS	System Usability Scale
UAT	User Acceptance Testing
UX	User Experience
UI	User Interface

LISTA DE SÍMBOLOS

%	Porcentagem

SUMÁRIO

1 INTRODUÇÃO

A crescente urbanização e a dinâmica social das cidades, mesmo em centros de menor porte como Oeiras-PI, impuseram desafios logísticos e informacionais, especialmente no que tange à locação de imóveis. Tradicionalmente, o processo de busca por moradia ou espaço comercial em cidades pequenas foi caracterizado pela informalidade, dependência de redes de contato pessoal e, mais recentemente, pela dispersão de informações em grupos e páginas de redes sociais. Essa fragmentação e a falta de um canal centralizado resultaram em ineficiência, desatualização de dados e uma experiência frustrante tanto para locadores quanto para locatários.

Neste contexto, a problemática central desta pesquisa se estabeleceu na dificuldade de locação de imóveis em cidades pequenas, como Oeiras-PI, devido à ausência de uma plataforma centralizada que conectasse de forma eficiente locadores e locatários. Isso gerava ineficiência, perda de tempo e informações desatualizadas, impactando negativamente o mercado imobiliário local.

O presente trabalho propôs o desenvolvimento de uma solução tecnológica, um sistema web, que atuou como uma plataforma centralizada para a divulgação e busca de imóveis disponíveis para locação na cidade de Oeiras-PI. A hipótese de trabalho foi de que a implementação de uma plataforma digital com funcionalidades de busca avançada, filtros e gestão de anúncios resultaria em uma redução significativa do tempo e do esforço despendidos no processo de locação, oferecendo maior transparência e acessibilidade à informação.

O trabalho foi delimitado ao desenvolvimento e avaliação da usabilidade do sistema *web*, com foco na realidade do mercado imobiliário de Oeiras-PI. A plataforma ampliou a visibilidade dos imóveis para os locadores e forneceu informações detalhadas (número de quartos, garagem, tipo de imóvel) para os locatários, auxiliando na busca pelo imóvel ideal e centralizando as informações.

2 JUSTIFICATIVA

Oeiras, primeira capital do Piauí, destacou-se por seu patrimônio histórico, arquitetura colonial preservada e forte tradição religiosa, especialmente com eventos como a Procissão do Fogaréu, que impulsionam o turismo e a economia local. Como polo do Vale do Canindé, a cidade também atraiu comércio e estudantes, intensificando o fluxo de pessoas e aumentando a demanda por moradia e hospedagem.

Esse fluxo constante de pessoas, somado ao crescimento urbano, gerou uma demanda crescente por moradia e hospedagem. No entanto, o mercado imobiliário de Oeiras ainda careceu de uma infraestrutura digital que acompanhasse essa importância histórica e econômica. Uma realidade observada e evidenciada pela constante procura por informações em canais informais, como grupos de redes sociais e páginas da internet da cidade. Essa dependência de meios não estruturados gerou problemas práticos para a população, como a perda de tempo na busca e verificação de anúncios desatualizados.

O sistema *web* proposto buscou preencher essa lacuna ao oferecer um serviço de utilidade pública que centraliza e organiza as ofertas de forma estruturada. Entre as funcionalidades implementadas, destacam-se os filtros de busca avançados (por bairro, faixa de preço, quantidade de quartos e garagem), que permitiram ao locatário encontrar exatamente o que procurava sem perda de tempo. Além disso, o sistema contou com galerias de fotos detalhadas e descrições precisas das condições do imóvel, reduzindo a necessidade de visitas presenciais a locais que não atendiam às expectativas.

Do ponto de vista econômico, a plataforma atuou como um catalisador para o mercado imobiliário local. Ao facilitar a locação de imóveis, o sistema estimulou a ocupação de espaços ociosos, gerando renda direta para os proprietários. A relevância acadêmica e tecnológica esteve vinculada ao curso de Análise e Desenvolvimento de Sistemas, pois o projeto representou a aplicação prática e integrada dos conhecimentos adquiridos ao longo das disciplinas.

3 OBJETIVOS

3.1 Objetivo geral

Desenvolver e implementar um sistema *web* para a gestão e busca de imóveis disponíveis para locação na cidade de Oeiras – PI, visando facilitar o processo para locadores e locatários.

3.2 Objetivos específicos

Foram estabelecidos como objetivos específicos do presente trabalho:

- Realizar o levantamento de requisitos para o sistema *web* de locação de imóveis, considerando as necessidades dos usuários de Oeiras – PI;
- Projetar o sistema focando em fácil entendimento, excelente usabilidade e uma boa curva de aprendizado do usuário;
- Implementar o sistema *web* utilizando tecnologias modernas de desenvolvimento;
- Realizar testes de sistema para o correto funcionamento do sistema e testes de aceitação do usuário para adequação das necessidades dos clientes.

4 METODOLOGIA

A presente seção descreve as metodologias de pesquisa e os procedimentos técnicos adotados no desenvolvimento do sistema VelhaCap Homes.

A classificação metodológica de uma pesquisa não se dá por um único critério, mas pela combinação de eixos complementares (natureza, objetivos, abordagem do problema e procedimentos técnicos), que juntos caracterizam com precisão o desenho da investigação (GIL, 2017). O presente trabalho é classificado, simultaneamente, segundo os quatro eixos a seguir.

Quanto à natureza, a pesquisa foi classificada como Aplicada, pois, conforme Gil (2017), foca no interesse prático e na solução de problemas específicos identificados na realidade, o que se justifica pelo objetivo central de desenvolver uma solução tecnológica concreta para mitigar dificuldades reais de locação de imóveis em Oeiras, PI.

Quanto aos objetivos, a pesquisa assumiu caráter Descritivo, por meio do método de Estudo de Caso, que, segundo Yin (2015), permite analisar um fenômeno contemporâneo em profundidade dentro de seu contexto real. O estudo de caso foi essencial para captar as particularidades socioeconômicas e logísticas de Oeiras, adaptando a plataforma às demandas específicas da cidade, e não a uma solução genérica.

Quanto à abordagem do problema, a pesquisa foi Qualitativa, manifestada na compreensão das dores e experiências subjetivas do processo de locação e na observação direta dos canais informais usados pela população de Oeiras para anunciar e buscar imóveis.

Quanto ao procedimento técnico, empregou-se a Pesquisa-Ação, que pressupõe associação estreita entre pesquisa e resolução de um problema coletivo (THIOLLENT, 2011), manifestada pela observação participante do pesquisador e por diálogos informais com moradores, comerciantes e estudantes envolvidos na busca ou oferta de imóveis, subsidiando a identificação das funcionalidades essenciais e a validação do escopo da solução.

Essas quatro dimensões (natureza aplicada, objetivo descritivo por estudo de caso, abordagem qualitativa e procedimento de pesquisa-ação) não são excludentes, mas se complementam para descrever o desenho metodológico adotado neste trabalho.

4.1 Modelo de desenvolvimento de software

A adoção de um modelo de desenvolvimento de *software* revelou-se fundamental para garantir a organização, a qualidade e o cumprimento dos prazos do projeto. Segundo Pressman (2016), o uso de um processo de *software* fornece a estrutura necessária para que o desenvolvimento ocorra de forma sistemática e disciplinada.

Para este projeto, adotou-se o Modelo Cascata (ou Ciclo de Vida Clássico). Essa escolha justificou-se por se tratar de uma metodologia tradicional e linear, adequada a projetos individuais em que os requisitos puderam ser bem definidos no início do processo. Diferente das metodologias ágeis, que demandam grandes equipes e iterações constantes, o modelo Cascata permitiu que o desenvolvedor focasse em uma etapa por vez, garantindo o rigor técnico necessário a um Trabalho de Conclusão de Curso.

As fases do ciclo de desenvolvimento, com adaptações ao contexto do projeto, seguiram a seguinte sequência:

- Levantamento e Análise de Requisitos: identificação das necessidades dos usuários e das funcionalidades do sistema;
- Design (Projeto): definição da arquitetura do *software*, da interface e da estrutura do banco de dados;
- Implementação (Codificação): tradução do projeto em código de programação;
- Implantação: disponibilização do sistema para uso em ambiente real;
- Verificação (Testes): realização de testes de sistema e de aceitação do usuário (UAT) para validar o atendimento aos requisitos;
- Validação: aplicação do questionário SUS para avaliação da usabilidade junto aos 15 participantes.

Vale ressaltar que, embora o modelo adotado tenha sido o Cascata, a evolução incremental do banco de dados ao longo do projeto, registrada por meio

de cinco *migrations Prisma* sucessivas evidenciou certa iteratividade localizada na fase de projeto, compatível com refinamentos pontuais sem prejuízo à ordenação clássica das etapas.

4.2 Arquitetura de sistemas

A arquitetura de *software* refere-se à estrutura organizacional de um sistema, definindo seus componentes, propriedades externas e as relações entre eles. Segundo Pressman (2016), uma arquitetura bem planejada é crucial para garantir que o sistema atenda aos requisitos de desempenho, segurança e manutenibilidade, servindo como base para todo o desenvolvimento técnico.

Para o desenvolvimento do VelhaCap Homes, adotou-se o padrão arquitetural Model-View-Controller (MVC), realizado por meio de uma arquitetura em camadas com separação física entre *front-end* e *back-end.* Conforme explica Sommerville (2019), o MVC organiza a aplicação em três camadas distintas com responsabilidades bem definidas:

- **Mode**l (Modelo): representado, no *back-end*, pelas definições do Prisma ORM e pelos models do banco de dados PostgreSQL, responsáveis pela persistência e pela lógica de acesso a dados;
- **View** (Visão):representada pela aplicação *front-end* em React, responsável pela interface do usuário, pela apresentação dos dados e pela captura das interações;
- **Controlle**r (Controlador): representado, no *back-end*, pelos arquivos de rotas e *controllers* em *Express*, que recebem as entradas das requisições *HTTP*, processam-nas acionando camadas de serviço/modelo e retornam a resposta adequada; no *front-end*, pelas funções de serviço (*api.ts*) e pela camada de roteamento do *TanStack Router.*

A arquitetura em camadas, baseada em MVC, mostrou-se adequada ao desenvolvimento deste *software* porque permitiu a separação clara entre a interface e a lógica de dados, facilitando a manutenção e futuras expansões do sistema de locação, garantindo que mudanças no design não afetassem o funcionamento do

banco de dados e vice-versa. Além disso, essa organização modular tornou o desenvolvimento individual mais eficiente e estruturado, assegurando a integridade e a segurança das informações de imóveis e usuários.

4.3 Usabilidade e experiência do usuário (UX/UI)

A eficácia do sistema esteve diretamente ligada à sua usabilidade, especialmente em plataformas de busca e filtragem de informações. Segundo Shneiderman et al. (2016), o design de interface (UI) deve focar na redução da carga cognitiva do usuário, permitindo que ele encontre o imóvel desejado de forma intuitiva e rápida. Em sistemas de busca imobiliária, isso se traduziu em interfaces limpas, nas quais os filtros de pesquisa e os resultados foram apresentados de maneira clara e organizada.

A Experiência do Usuário (UX) foi priorizada por meio da implementação de um *Design* Responsivo. Conforme preconizado por Krug (2014), é essencial que a aplicação *web* ofereça uma boa experiência de navegação tanto em computadores de mesa quanto em dispositivos móveis, como *smartphones* e *tablets*. Dado que grande parte das buscas por imóveis em Oeiras ocorre por dispositivos móveis em redes sociais, o sistema foi construído para se adaptar automaticamente a diferentes tamanhos de tela, garantindo que a legibilidade e a facilidade de interação fossem mantidas em qualquer contexto de uso. Essa abordagem visou maximizar a acessibilidade e a satisfação do usuário final, tornando o processo de locação mais eficiente e menos custoso.

5 LEVANTAMENTO DE REQUISITOS

O levantamento de requisitos do sistema foi realizado a partir da análise do problema a ser resolvido: a dificuldade de locadores e locatários de Oeiras – PI para encontrar e divulgar imóveis disponíveis para locação de forma centralizada e eficiente. Essa etapa é crucial, pois consiste na compreensão das necessidades dos usuários e do negócio, estabelecendo as bases de *design* e construção que garantem que o produto final realmente solucione o problema identificado (PRESSMAN, 2016). Diferentemente de projetos tradicionais que iniciam com entrevistas e pesquisas com usuários, neste trabalho os requisitos foram definidos com base em:

- Observação participante da realidade local, em diálogo informal com moradores, comerciantes e estudantes em situação de busca ou oferta de imóveis.
- Necessidade prática identificada pelo autor, a partir da vivência em Oeiras – PI e da constatação da fragmentação dos anúncios imobiliários em grupos de redes sociais;
- Diretrizes do projeto de pesquisa, que já definia funcionalidades essenciais como filtros de busca, galeria de imagens e painel do anunciante;
- Análise de plataformas similares e da literatura da área (TURBAN et al., 2015; SHNEIDERMAN et al., 2016; GARRETT, 2011);

Os requisitos foram organizados em Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF), garantindo uma visão clara tanto das funcionalidades quanto das qualidades esperadas do sistema.

6 ANÁLISE DE REQUISITOS

A análise de requisitos pode ser definida como a atividade de transformar as necessidades levantadas em especificações estruturadas, que orientam o projeto e a implementação do *software*. Para Sommerville (2019), representa o elo entre a fase de levantamento e a concretização das decisões de projeto, traduzindo dores reais em exigências técnicas capazes de guiar o desenvolvimento. A análise aumenta sutilmente a clareza dos requisitos, eliminando ambiguidades e destacando prioridades, missão indispensável para que o *software* atenda de fato ao problema a que se propõe.

No presente projeto, a análise dos requisitos derivados conforme a Seção 5 culminou em dois conjuntos de especificações: os requisitos funcionais (que descrevem o que o sistema deve fazer) e os requisitos não funcionais (que descrevem como o sistema deve comportar-se). Por fim, a interação entre os atores e o sistema foi representada por meio de um diagrama de caso de uso na notação UML.

6.1 Requisitos funcionais

Os requisitos funcionais descrevem as funcionalidades específicas que o sistema deve oferecer aos seus usuários, expressando serviços essenciais, comportamentos esperados e respostas a entradas. Conforme Sommerville (2019), esses requisitos são fundamentais para demarcar o escopo do *software* e direcionar sua posterior validação por meio de testes funcionais.

Quadro 1 - Quadro de requisitos funcionais

ID	Requisito	Descrição
RF01	Cadastro de usuário	O sistema deve permitir que um novo usuário cadastre-se informando nome, e-mail e senha.
RF02	Autenticação (login)	O sistema deve permitir que o usuário autentique-se por meio de e-mail e senha, recebendo um token JWT válido por sete dias.
	Encerramento de sessão (logout)	O sistema deve permitir que o usuário autenticado encerre a sessão, removendo o token armazenado no

RF03		navegador.
RF04	Listagem pública de imóveis	O sistema deve exibir, em uma página pública, os imóveis marcados como disponíveis, ordenados por ordem de criação decrescente.
RF05	Detalhamento de imóvel	O sistema deve exibir uma página de detalhes, contendo título, descrição, preço, localização, características, galeria de imagens e dados de contato do anunciante.
RF06	Busca e filtros	O sistema deve permitir filtrar os imóveis por palavra-chave, por tipo de imóvel, por número mínimo de quartos e por preço máximo.
RF07	Cadastro de imóvel (autenticado)	O sistema deve permitir que o usuário autenticado cadastre um imóvel, com até seis imagens, informando título, descrição, preço, cidade, bairro, tipo, número de quartos, banheiros, presença de garagem, disponibilidade e dados do proprietário.
RF08	Edição de imóvel (autenticado)	O sistema deve permitir que o usuário autenticado edite os imóveis que cadastrou, com verificação de posse.
RF09	Exclusão de imóvel (autenticado)	O sistema deve permitir que o usuário autenticado exclua os imóveis que cadastrou, removendo também as respectivas imagens armazenadas no Cloudinary.
RF10	Alternar disponibilidade	O sistema deve permitir que o usuário autenticado alterne o status de disponibilidade de um imóvel por meio de interruptor dedicado.
RF11	Upload de imagens	O sistema deve permitir o upload de imagens nos formatos JPG, JPEG, PNG e WEBP, com redimensionamento automático e hospedagem em serviço de nuvem.
RF12	Painel do anunciante	O sistema deve exibir um painel privado, restrito ao usuário autenticado, com a listagem exclusiva dos seus imóveis.
RF13	Dashboard de estatísticas	O sistema deve exibir, no painel, os totais de imóveis cadastrados, de visualizações e de contatos recebidos pelo usuário autenticado.
RF14	Contato com o anunciante	O sistema deve permitir que o visitante entre em contato com o anunciante por meio de link para WhatsApp e para e-mail.
RF15	Métricas de visualização e contato	O sistema deve manter um contador de visualizações por imóvel e um registro de cada contato efetuado via própria plataforma.

Fonte: Elaborado pelo autor

6.2 Requisitos não-funcionais

Os requisitos não funcionais definem atributos de qualidade do sistema, discriminando restrições de comportamento e propriedades operacionais que devem

ser respeitadas pela solução. Conforme Sommerville (2019), tais requisitos tutelam aspectos como usabilidade, desempenho, segurança, confiabilidade e disponibilidade, e diante da crescente variedade de dispositivos de acesso, foram complementados por outros como portabilidade.

Quadro 2 - Quadro de requisitos não-funcionais

ID	Requisito	Descrição
RNF01	Desempenho	O sistema deve apresentar tempos de carregamento adequados para as telas públicas, sem prejuízo à fluidez da navegação. As rotas do *back-end* devem responder em até 1.500 ms nas operações de listagem e detalhamento.
RNF02	Usabilidade	O sistema deve apresentar interface limpa, organizada e responsiva, com componentes de interface reutilizáveis baseados na biblioteca *shadcn/ui* e no *framework Tailwind CSS*, oferecendo experiência intuitiva em desktop e em mobile.
RNF03	Segurança	O sistema deve armazenar senhas por meio de hash criptográfico (*bcrypt js*), autenticar as requisições protegidas por *JWT* no cabeçalho *HTTP*, validar os dados de entrada com a biblioteca *Zod* e verificar a posse do recurso antes de editar ou excluir um imóvel.
RNF04	Confiabilidade	O sistema deve manter os dados em banco de dados relacional PostgreSQL sob *backup* gerenciado pelo serviço *Neon*, com o esquema versionado por meio de *migrations* Prisma.
RFN05	Disponibilidade	O sistema deve permanecer disponível em ambiente de produção por meio de hospedagem em nuvem (*back-end* no *Render* e *front-end* na *Vercel*), com *SLA* compatível com a infraestrutura gratuita dos referidos provedores.
RFN06	Portabilidade	O sistema deve funcionar nos principais navegadores modernos (*Google Chrome, Mozilla Firefox, Microsoft Edge e Safari*) e adaptar-se a diferentes dimensões de tela (*desktop, tablet e smartphone*), dada a natureza responsiva das telas.

Fonte: Elaborado pelo autor

6.3 Diagrama de Caso de Uso UML

O diagrama de caso de uso da *Unified Modeling Language* (UML) descreve, de forma gráfica, as interações entre os atores do sistema e os casos de uso por ele oferecidos, funcionando como instrumento para visualizar o escopo funcional sob a perspectiva do usuário. Pressman (2016) ressalta que o diagrama contribui para o entendimento compartilhado dos requisitos entre *stakeholders* e para a comunicação entre cliente, desenvolvedor e orientadores.

Foram identificados, no VelhaCap Homes, dois atores distintos:

- **Visitante**: usuário não autenticado que interage apenas com páginas públicas (acessa a Home, lista e detalha imóveis, busca/filtra e aciona contatos) e pode realizar cadastro e login para acessar as funcionalidades restritas;
- **Locador**: usuário autenticado que, após realizar cadastro e login, pode cadastrar novo imóvel, editar, excluir, alternar disponibilidade, inserir contatos e acessar o painel do anunciante e o *dashboard* de estatísticas.

Figura 1 – Diagrama de Caso de Uso do VelhaCap Homes

Fonte: Elaborado pelo autor

7. IMPLEMENTAÇÃO DO SISTEMA

7.1 Tecnologias envolvidas

Para a implementação do VelhaCap Homes, adotou-se uma *stack* tecnológica moderna, baseada em *TypeScript* em ambas as extremidades da aplicação, com bibliotecas e *frameworks* consolidados no mercado, em sua maioria de código aberto. A escolha das tecnologias buscou conciliar maturidade, suporte da comunidade e adequação às necessidades de um projeto individual de Trabalho de Conclusão de Curso.

Quadro 3 – Tecnologias envolvidas no projeto

Tecnologia	**Versão**	**Observação**
TypeScript	6.0	Linguagem principal de tipagem estática, adotada nas duas extremidades para garantir segurança de tipos.
Node.js	22.20.0	Ambiente de execução do *back-end.*
Express	5.2	*Framework* web do *back-end*, responsável pela exposição das rotas e pelo fluxo de *middleware*.
Prisma ORM	5.22	ORM de mapeamento objeto-relacional, responsável pela interação com o banco de dados e pela definição do esquema.
PostgreSQL	17	Sistema gerenciador de banco de dados

		relacional, hospedado no serviço Neon.
bcryptjs	3.0	Biblioteca de *hash* usada para criptografar as senhas dos usuários.
JSON Web Token (jsonwebtoken)	9.0	Padrão de *token* usado para autenticar as requisições protegidas.
Zod	4.4./3.24	Biblioteca de validação de esquemas, usada para validar as entradas nos controladores.
Cloudinary	1.41	Serviço de hospedagem e processamento de imagens em nuvem.
multer-storage-cloudinary	4.0	*Adapter* do multer para envio direto das imagens ao *Cloudinary*.
cors	2.8	*Middleware* de configuração do Compartilhamento de Recursos de Origem Cruzada.
React	19.2	Biblioteca de construção de interfaces.
Vite	7.3	Ferramenta de *build* e servidor de desenvolvimento do *front-end*.
TanStack Start / Router / Query	1.16	*Framework* de SSR +

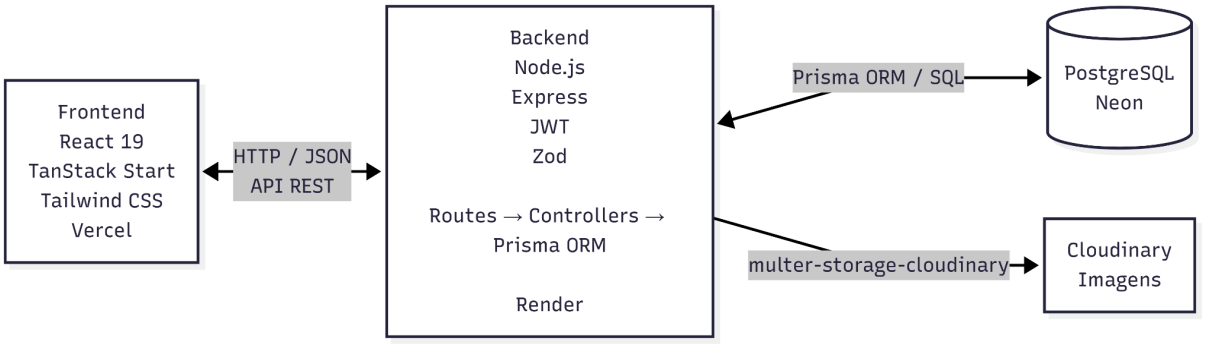
		roteamento *file-based* e gerência de estado assíncrono no *front-end*.
Tailwind CSS	4.2	*Framework* de estilos utilitários
shadcn/ui (Radix UI)	-	Conjunto de componentes acessíveis e reutilizáveis usados na interface.
react-hook-form	7.71	Biblioteca de gerência de formulários
Git	2.45	Sistema de controle de versão distribuído.
GitHub	-	Plataforma de hospedagem do repositório de código.
VSCode	1.128	Editor de código.
Render	-	Plataforma de hospedagem do *back-end* em ambiente de produção.
Vercel	-	Plataforma de hospedagem do *front-end* em ambiente de produção.
Neon	-	Plataforma de hospedagem do banco de dados PostgreSQL em ambiente de produção.

Fonte: Elaborado pelo autor

7.2 Arquitetura Do Sistema

O VelhaCap Homes opera em dois níveis arquiteturais distintos e complementares. No nível macro, o sistema segue o padrão cliente-servidor desacoplado, no qual componentes fisicamente independentes — *front-end*, *back-end*, banco de dados e armazenamento de imagens — comunicam-se por meio de requisições HTTP e da interface programática (API REST). No nível micro, o *back-end* organiza-se internamente segundo o padrão Model-View-Controller (MVC), que separa as responsabilidades em camadas de controle (rotas e controllers), de modelo (Prisma ORM e banco de dados) e de visão (representada pelo *front-end* React, que consome a API remotamente e apresenta os dados ao usuário).

Figura 2 - Diagrama de Arquitetura do Sistema



Fonte: Elaborado pelo autor

7.2.1 Modelo de dados

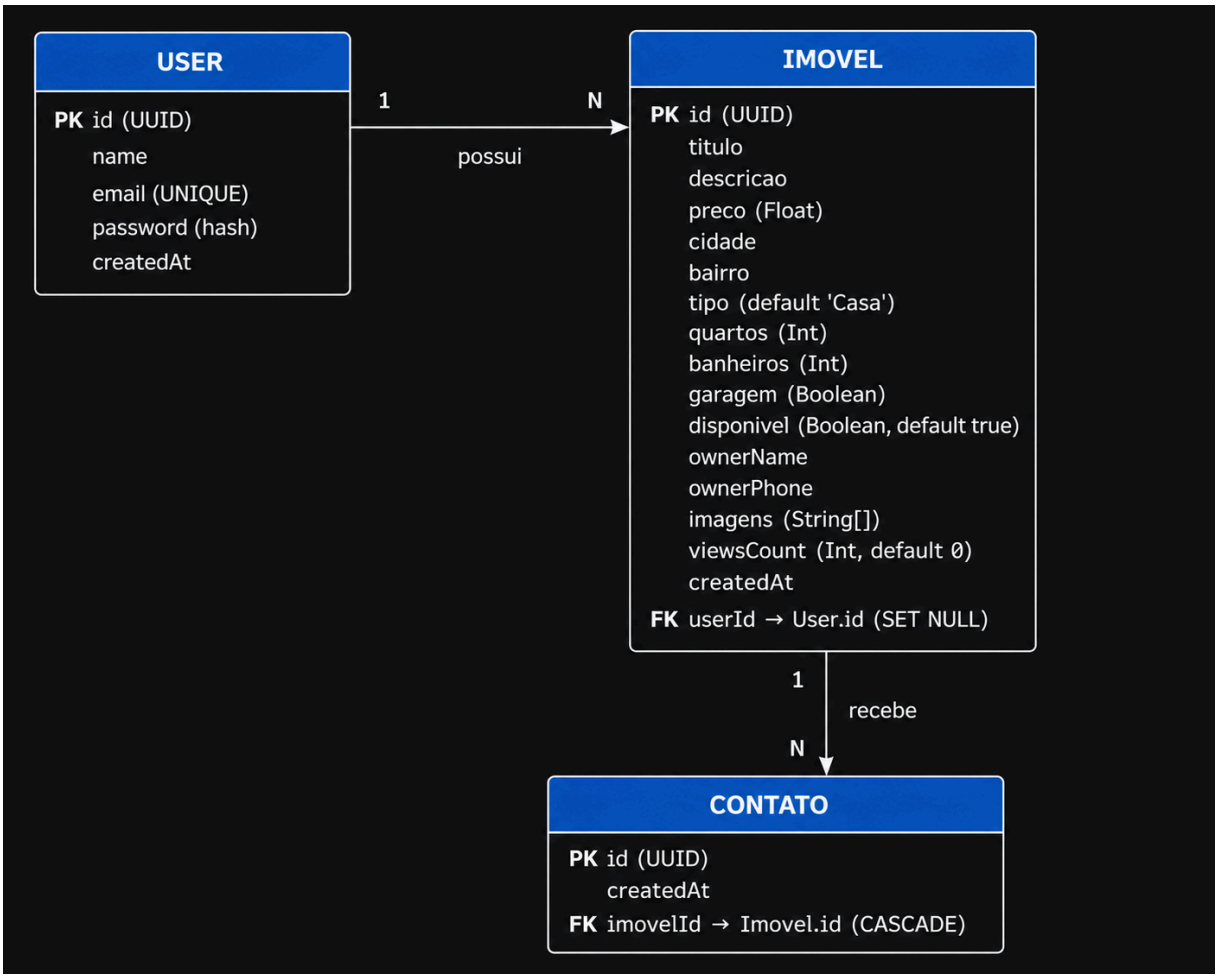
O esquema do banco de dados é declarado por meio da linguagem de definição de dados do Prisma e evoluiu progressivamente ao longo do projeto, registrando cinco *migrations* sucessivas. O esquema final é composto por três entidades:

- **User**:representa o usuário (locador), com campos *id* (UUID), *name*, email (único), password (em *hash*), telefone (opcional) e *createdAt*. Relaciona-se em um-para-muitos com Imovel. Caso o usuário seja removido, a exclusão de seus imóveis aciona *ON DELETE SET NULL*, preservando o histórico dos imóveis sem localizar proprietário;

Imovel:representa cada imóvel anunciado, com os campos id, titulo, descricao, preço (Float), cidade, bairro, tipo (*string*, validada por *Zod* como enum: Casa, Apartamento, Kitnet, Sobrado ou Sala Comercial), quartos, banheiros, garagem, disponivel, ownerName, *ownerPhone*, imagens (array de *strings*, URLs do *Cloudinary*), *viewsCount* (inteiro), *userId* (chave estrangeira para User, opcional), contatos (relacionamento um-para-muitos com **Contato**, com *ON DELETE CASCADE*) e *createdAt*;

- **Contato**: registra cada contato efetuado via o sistema para um imóvel, com *id*, *imovelId* (FK) e *createdAt*. Função analítica para o *dashboard* do anunciante.

Figura 3 – Diagrama Entidade-Relacionamento do banco de dados



Fonte: Elaborado pelo autor (2026).

7.2.2 Autenticação

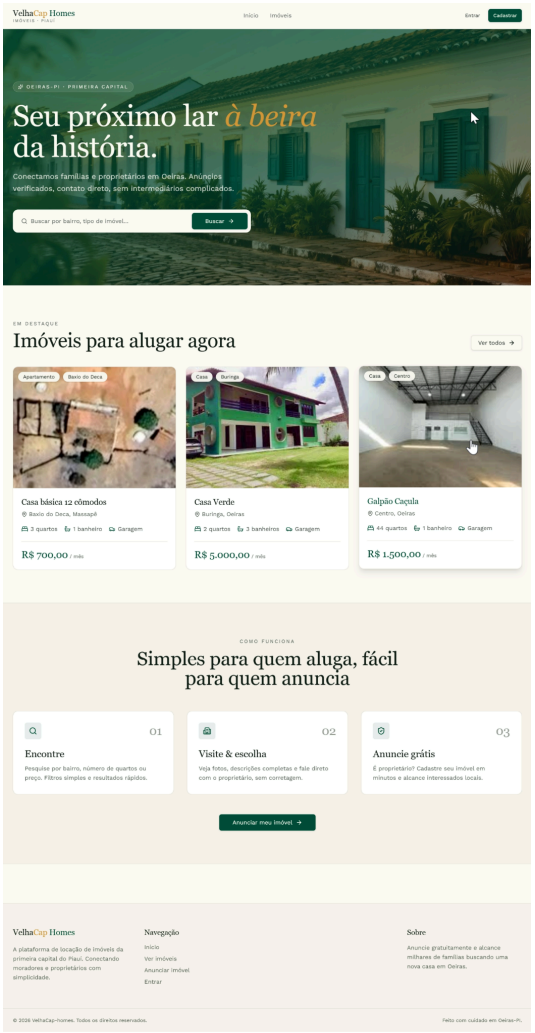
A autenticação do sistema segue o padrão stateless baseado em *JSON Web Token (JWT)*, com expiração de sete dias. As senhas são armazenadas em hash por meio da biblioteca *bcryptjs*. Requisições a endpoints protegidos são validadas por um middleware (auth.middleware.ts) que decodifica o *token* com *jwt.verify*, anexa o identificador do usuário à requisição (*req.userId*) e verifica a posse do recurso antes de permitir alterações em imóveis.

7.3 Interface

A interface do usuário foi construída com *React* em conjunto com o *framework Tailwind CSS* e os componentes acessíveis do *shadcn/ui*, que reúne primitivos do *Radix UI* estilizados de forma consistente. A escolha dessa combinação visou, conforme recomenda Shneiderman et al. (2016), reduzir a carga cognitiva do usuário por meio de elementos padronizados, estado visual previsível e comportamento acessível. O conceito visual, marcado pela paleta de tons quentes, faz alusão à identidade da marca VelhaCap Homes, evocando a ambientação histórica de Oeiras. A aplicação é responsiva, com breakpoints definidos via *Tailwind* e adaptando o cabeçalho (*Navbar*) para um menu hambúrguer em telas estreitas. A navegação é baseada em arquivos (*TanStack Router*), permitindo acesso direto via URL recarregável a qualquer rota. As principais telas implementadas são descritas a seguir:

7.3.1 Página inicial (Home)

Figura 4 - Tela inicial (Home)

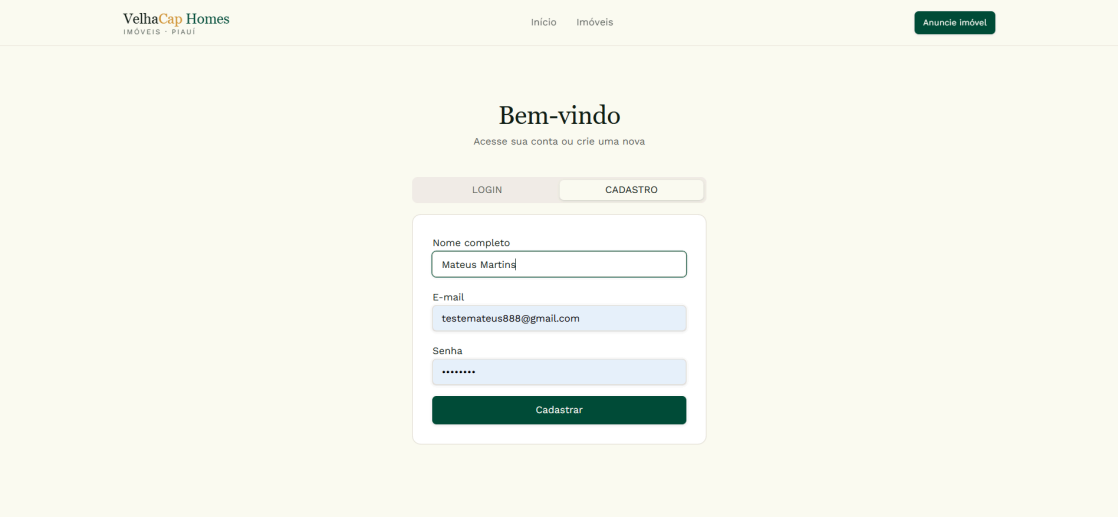


Fonte: Autor (2026).

A página inicial é a primeira experiência do usuário com a plataforma. No topo, uma sessão de apresentação (*hero*) exibe uma imagem de fundo da cidade de Oeiras acompanhada de texto institucional que apresenta o VelhaCap Homes como a plataforma de locação de imóveis da cidade. Ao rolar a página, o visitante encontra uma seção explicativa "Como funciona", seguida por uma grade com seis imóveis em destaque. Cada card de imóvel apresenta imagem principal, tipo do imóvel (Casa, Apartamento, Kitnet, Sobrado ou Sala Comercial), bairro, quantidade de quartos, banheiros e vagas de garagem, além do preço mensal formatado em Real brasileiro. O visitante pode clicar em qualquer card para acessar a página de detalhes do imóvel

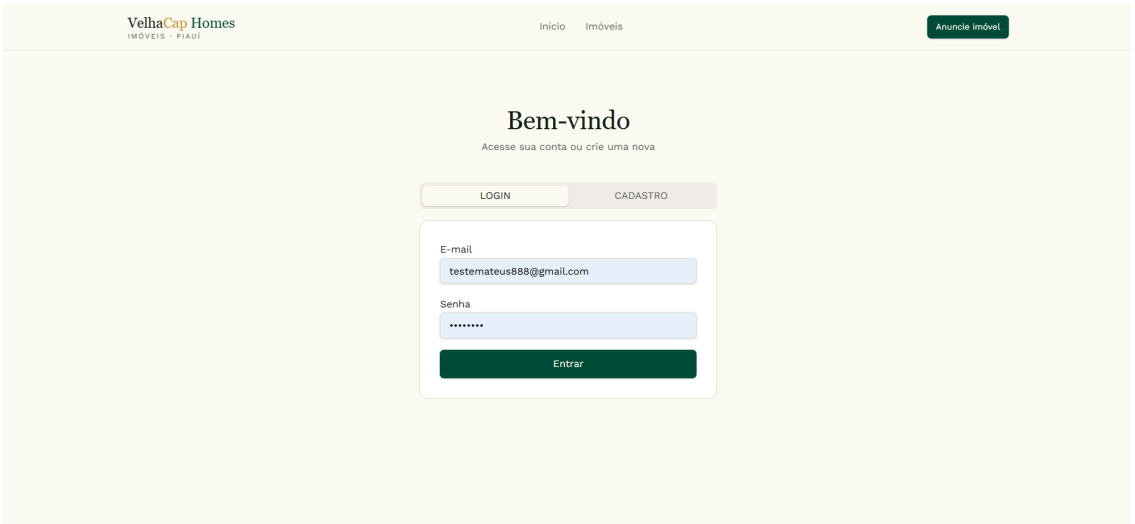
7.3.2 Telas de Cadastro e Login

Figura 5 - Tela de autenticação (login/cadastro)



Fonte: Autor (2026).

A tela de autenticação reúne login e cadastro em uma única página, organizada por abas. Na aba "*LOGIN*", o usuário informa e-mail e senha e clica em "Entrar".

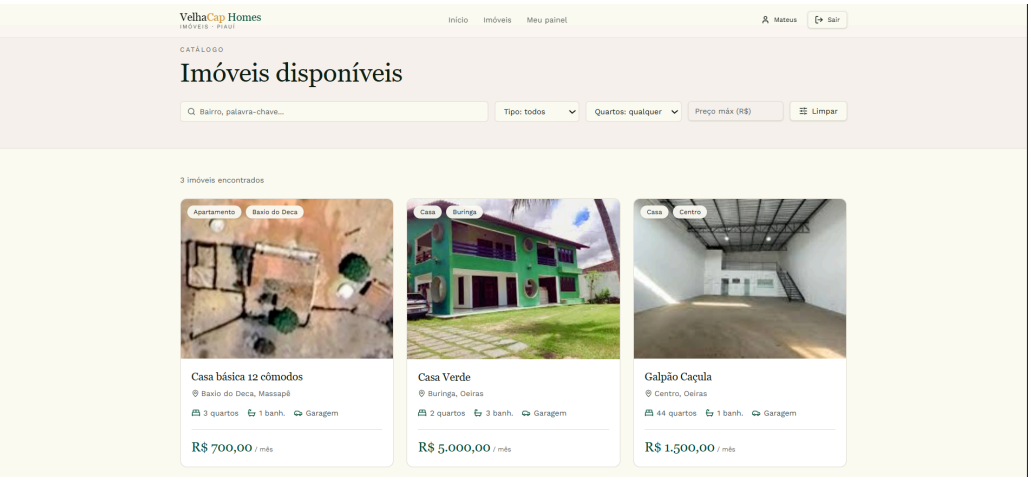


Fonte: Autor (2026).

Na aba "CADASTRO", preenche nome, e-mail e senha e clica em "Cadastrar". Ao se cadastrar, o usuário é conectado automaticamente e levado ao painel do anunciante. Se o e-mail já estiver em uso, uma mensagem de erro aparece. Caso o visitante não esteja logado, o botão "Anuncie imóvel" na barra de navegação o direciona para esta tela

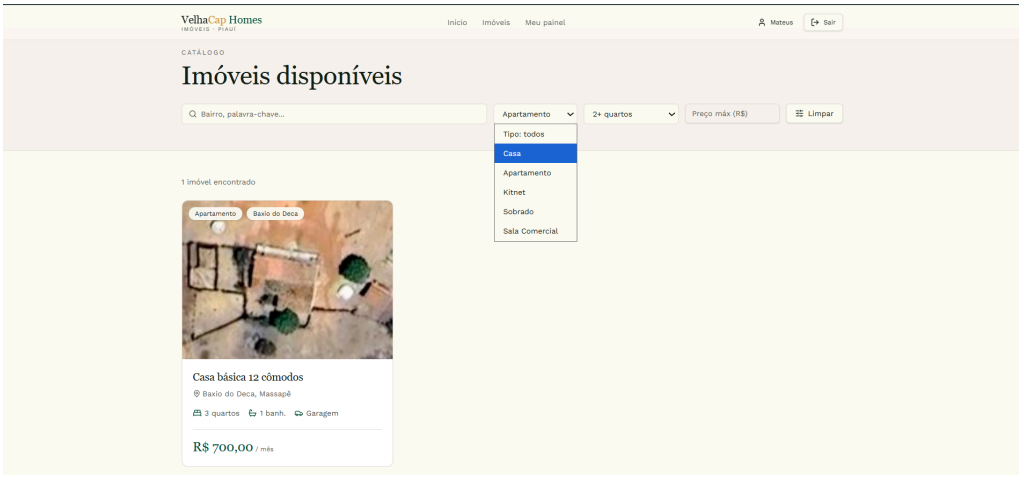
7.3.3 Listagem pública de imóveis

Figura 6 - Tela de listagem pública de imóveis



Fonte: Autor (2026).

A tela de listagem apresenta todos os imóveis disponíveis para locação, organizados em grade responsiva de cards.

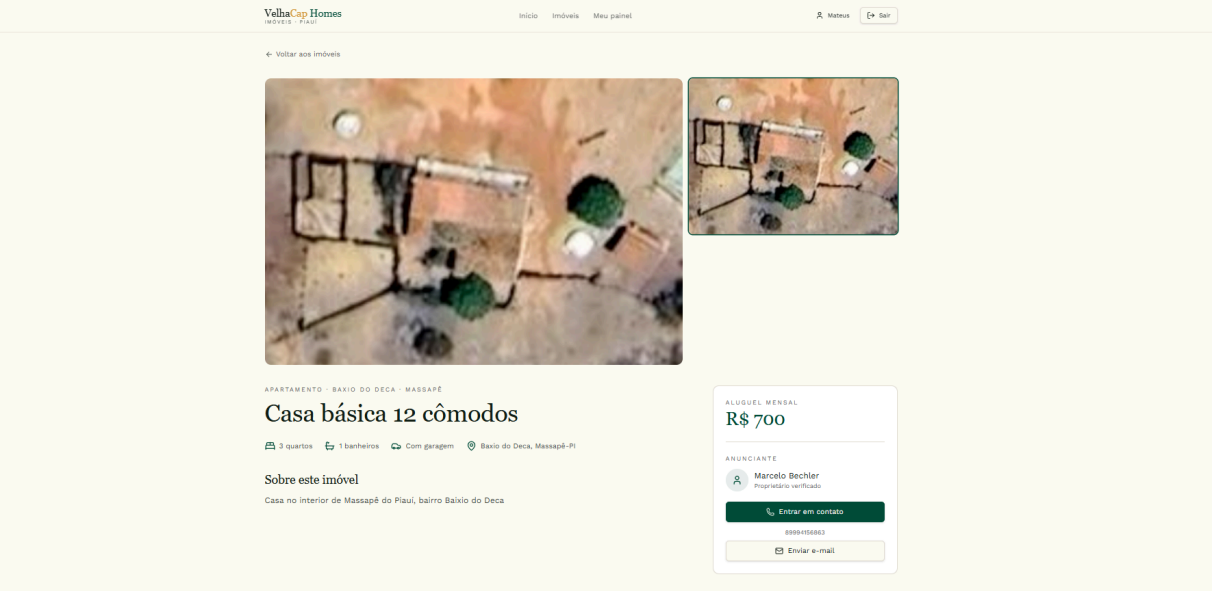


Fonte: Autor (2026).

No topo da página, uma barra de busca permite filtrar imóveis por palavra-chave (título, descrição ou bairro), enquanto filtros laterais ou acima da grade permitem refinar por tipo de imóvel, número mínimo de quartos e preço máximo. Os filtros são aplicados em tempo real, reorganizando a grade sem necessidade de recarregar a página. Cada card exibe imagem, tipo, bairro, características físicas e preço. O visitante pode clicar em qualquer card para acessar os detalhes do imóvel.

7.3.4 Detalhamento de imóvel

Figura 7 - Tela de detalhamento de imóvel

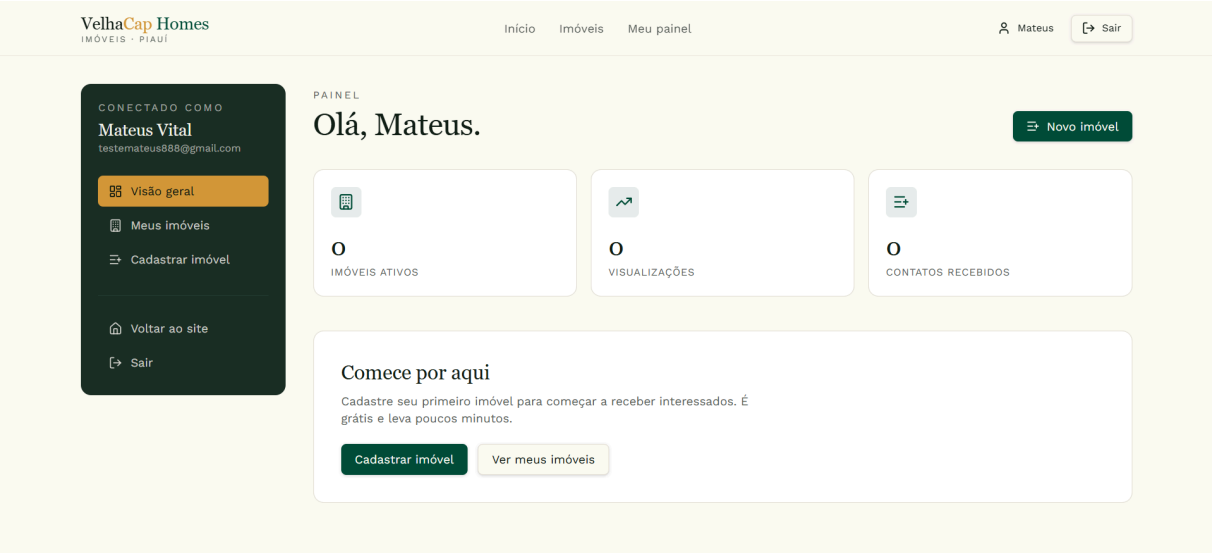


Fonte: Autor (2026).

A tela de detalhes de imóvel exibe, na parte superior, uma galeria de imagens do imóvel, seguida por informações detalhadas: título, descrição completa, preço mensal em Real, localização (cidade e bairro), tipo do imóvel e quantidade de quartos, banheiros e vagas de garagem. Na lateral ou abaixo das informações, são exibidos o nome e telefone do proprietário. Dois botões de contato permitem ao visitante iniciar uma conversa diretamente pelo *WhatsApp* (com mensagem pré-preenchida contendo o título do imóvel) ou enviar um e-mail para o anunciante. O contador de visualizações do imóvel é atualizado automaticamente a cada acesso à página

7.3.5 Painel do anunciante

Figura 8 - Painel do anunciante – dashboard de estatísticas



Fonte: Autor (2026).

A tela principal do painel do anunciante (acessível apenas por usuários autenticados) apresenta um *dashboard* com três cartões de estatísticas em tempo real: o total de imóveis cadastrados pelo usuário, o total de visualizações acumuladas entre todos os seus imóveis e o total de contatos recebidos via plataforma. Esses dados são atualizados a cada acesso à página e fornecem ao locador uma visão geral do desempenho dos seus anúncios.

7.3.6 Painel do anunciante – meus imóveis

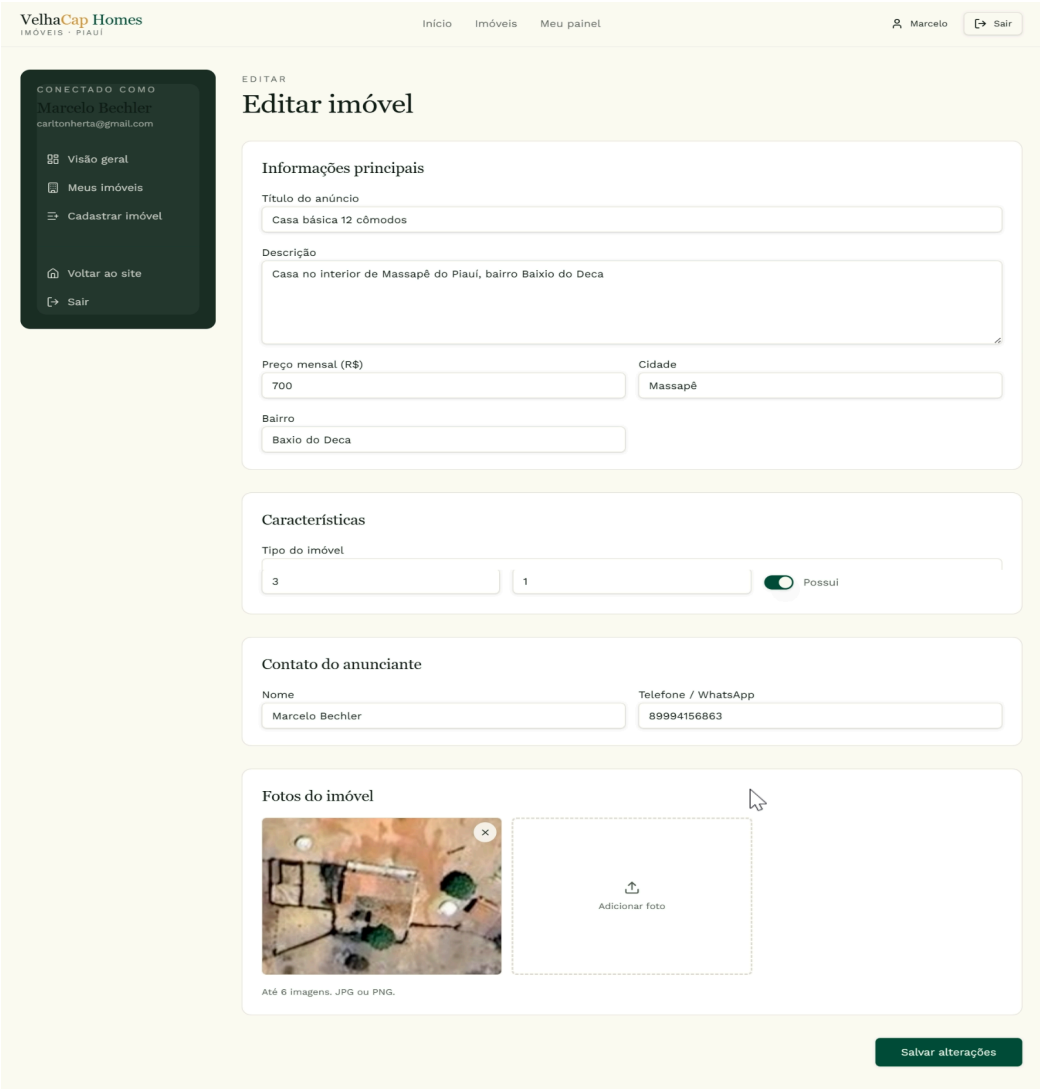
Figura 9 - Painel do anunciante – meus imóveis

Fonte: Autor (2026).

A tela "Meus imóveis" exibe uma tabela com todos os imóveis cadastrados pelo usuário autenticado, ordenados por data de criação. Cada linha da tabela apresenta uma miniatura da imagem principal, o título do imóvel, o preço, o status de disponibilidade (ativo ou inativo) e botões de ação. O usuário pode alternar a disponibilidade do imóvel por meio de um interruptor (*switch*), editar as informações do imóvel em um formulário dedicado ou excluí-lo permanentemente, sendo que esta última ação remove também as imagens armazenadas em nuvem e solicita confirmação antes de prosseguir.

7.3.7 Cadastro / edição de imóvel

Figura 10 - Tela de cadastro/edição de imóvel com upload de imagem



Fonte: Autor (2026).

A tela de cadastro e edição de imóvel apresenta um formulário dividido em duas seções: na parte superior, uma área dedicada ao upload de imagens, onde o usuário pode selecionar até seis fotografias do imóvel nos formatos *JPG*, *JPEG*, *PNG* ou *WEBP*, com barra de progresso visível durante o envio para a nuvem; na parte inferior, os campos de informações do imóvel, incluindo título, descrição, preço, cidade (predefinida como Oeiras), bairro, tipo, número de quartos, banheiros, presença de garagem, nome e telefone do proprietário. Ao preencher o formulário e acionar o botão "Salvar", o imóvel é registrado ou atualizado no sistema. Na versão de edição, as imagens já cadastradas são exibidas com opção de remoção individual.

8 IMPLANTAÇÃO

O *back-end* da aplicação foi implantado na plataforma Render, que hospeda a *API REST* desenvolvida em *Node.js* com *Express*. Ao receber o código atualizado do repositório no GitHub, o Render executa automaticamente o processo de compilação do *TypeScript*, aplica as *migrations* do Prisma e reinicia o servidor, mantendo a *API* disponível para acesso público sem necessidade de intervenção manual. O banco de dados PostgreSQL, responsável por armazenar os usuários, imóveis e contatos registrados no sistema, foi hospedado no Neon, serviço de nuvem que oferece persistência gerenciada, garantindo a integridade e a disponibilidade das informações sem exigir configuração manual de servidor.

O *front-end*, desenvolvido em React com Vite e *TanStack Start*, foi implantado na Vercel, que publica a aplicação como uma página web otimizada e acessível diretamente pelo navegador. A Vercel distribui o conteúdo globalmente, garantindo tempos de carregamento reduzidos para o usuário final. As imagens dos imóveis enviadas pelos anunciantes são armazenadas e processadas no *Cloudinary*, que cuida do redimensionamento automático, da conversão de formato e da disponibilização das fotografias por meio de URLs públicas. Assim como o Render, a Vercel também atualiza o site automaticamente a cada envio de código ao GitHub, garantindo que a versão publicada corresponda sempre à versão mais recente do repositório.

A plataforma VelhaCap está disponível para acesso e avaliação em ambiente on-line.[1] O repositório da aplicação também está disponível para demonstração.[2]

[1] Disponível em: https://velhacap-homes.vercel.app/ Acesso em: 29 jul. 2026.
[2] Disponível em: https://github.com/MateusVitall/meu-tcc-imoveis.git/ Acesso em: 29 jul. 2026

9 TESTES

9.1 Teste de sistema (System Testing)

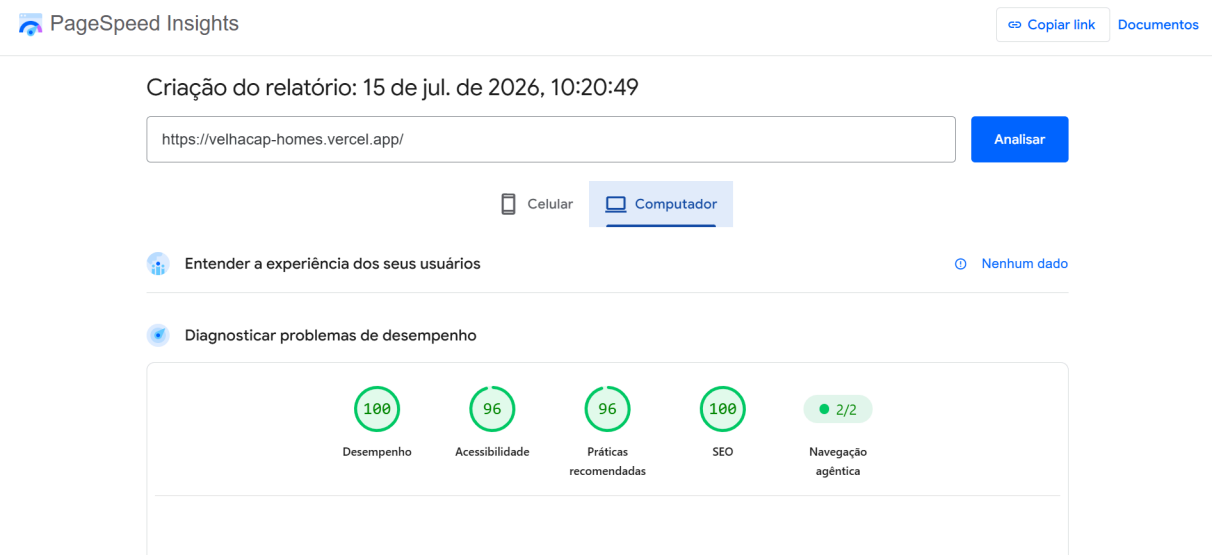
O teste de sistema, conforme define Pressman (2016), consiste na verificação do software como um todo, em ambiente que simula o uso real, com o objetivo de validar os requisitos não funcionais previamente especificados. Diferente dos testes unitários, que enfocam componentes isolados, o teste de sistema examina a aplicação de ponta a ponta, observando o comportamento das funcionalidades encadeadas e dos atributos de qualidade previstos. Para o VelhaCap Homes, o teste de sistema foi conduzido na versão publicada em ambiente de produção. As validações procuraram comprovar o quanto o sistema efetivamente cumpre os seis requisitos não funcionais descritos na Tabela 2, por meio de observação direta e instrumentos disponíveis no navegador (*DevTools*). Os resultados são descritos a seguir, organizados por atributo.

9.1.1 Teste de Desempenho

O tempo de carregamento das páginas públicas foi observado de forma manual no navegador e, de forma complementar, por meio de ferramenta automatizada. A observação manual mostrou que, em conexão de banda larga, as páginas carregam de forma fluida após a inicialização do *back-end*, sendo que o plano gratuito do Render impõe uma pequena espera inicial quando o servidor permanece inativo por algum tempo, comportamento conhecido como *cold start*. Como instrumento complementar, utilizou-se o *PageSpeed Insights*, ferramenta gratuita do Google que analisa uma página web e atribui notas de 0 a 100 em quatro aspectos: velocidade de carregamento, facilidade de uso por pessoas com deficiência (acessibilidade), adoção de boas práticas de desenvolvimento e adequação a motores de busca (SEO). Em termos práticos, a ferramenta simula o acesso de um usuário real à página e aponta tanto a pontuação obtida quanto oportunidades de melhoria. A análise foi realizada na versão *desktop* da página inicial do sistema e o resultado é apresentado na Figura 11. As notas obtidas foram todas próximas ou iguais ao valor máximo, o que confirma, de forma objetiva, que a página inicial carrega de maneira rápida, segue boas práticas de desenvolvimento e está bem ajustada para mecanismos de busca, corroborando as observações

manuais descritas anteriormente.

Figura 11 - Resultado do PageSpeed Insights (Google) – versão desktop



Fonte: Elaborado pelo autor (2026).

Em conjunto, os resultados evidenciam conformidade parcial com o RNF01: nos cenários comuns, os tempos ficaram dentro da faixa esperada e a auditoria do PageSpeed Insights confirmou índices máximos ou próximos do máximo no *desktop*; todavia, o *cold start* do plano gratuito do Render impõe uma degradação inicial periódica, limitação coerente com a escolha de hospedagem gratuita e ponto explicitado nas considerações finais como oportunidade de melhoria futura.

9.1.2 Teste de Usabilidade

A interface foi percorrida em três dimensões de tela (desktop 1920x1080, tablet 768x1024 e *smartphone* 375x812) por meio das ferramentas de responsividade do navegador. Verificou-se que:

- A *Navbar* adapta-se corretamente, recolhendo o menu em *hambúrguer* em telas inferiores a 768 px;
- A grade de cartões de imóveis na listagem pública reorganiza o número de colunas de três (desktop) para duas (tablet) e para uma (smartphone), sem sobreposição ou *scroll* horizontal indesejado;
- Os formulários (cadastro, login, novo imóvel, editar imóvel) preservaram legibilidade e usabilidade em *smartphone*, com campos empilhados

verticalmente;

- O *dashboard* do painel apresentou os cartões de estatísticas em *grid* responsivo adequado, com quebra de linha em telas pequenas;
- Os componentes do *shadcn/ui* (Botão, *Dialog*, *Switch*, *Select*) preservaram comportamento e estilo em todas as dimensões testadas. Conclui-se que o RNF02 foi atendido, observando-se padrão consistente de usabilidade em distintos dispositivos.

9.1.3 Teste de Segurança

Foram simuladas, via Postman (ferramenta para teste de APIs) e via inspeção manual, as seguintes verificações:

- Armazenamento de senha: ao consultar diretamente o banco de dados, constatou-se que o campo *password* da tabela *User* encontra-se armazenado em formato de *hash* (valor criptográfico unidirecional, gerado pelo algoritmo *bcrypt*), sem exposição da senha em texto claro;
- Autenticação *JWT*: requisições a *endpoints* (pontos de acesso da API) protegidos (*POST* /imoveis, *GET* /imoveis/meus-imoveis, *GET* /imoveis/stats, *PUT* /imoveis/:id, *DELETE* /imoveis/:id, *POST* /*upload*) sem *token* retornaram código **401** (não autorizado) com a mensagem "Token não informado"; requisições com *token* adulterado retornaram código *401* (não autorizado) com a mensagem "Token inválido"; requisições com *token* válido prosseguiram normalmente;
- Verificação de posse: ao tentar editar (*PUT*) e excluir (*DELETE*) imóvel pertencente a outro usuário, o sistema retornou código ***403*** (acesso proibido) com a mensagem "Acesso negado";
- Validação Zod: ao submeter cadastro de usuário com e-mail inválido e senha abaixo de 6 caracteres, o sistema retornou código **400** (requisição inválida) com mensagem indicando os campos inválidos;
- Upload: o *endpoint POST* /*upload* permaneceu protegido por *middleware* (componente intermediário que processa requisições) de autenticação, recusando requisições sem *token*. Conclui-se que o RNF03 foi atendido em seus pontos essenciais.

9.1.4 Confiabilidade

A persistência foi avaliada em regime de operação contínua, ao longo de uma semana de observação, verificando-se que:

- A instância PostgreSQL no Neon manteve-se disponível e sem indisponibilidades registradas;
- As cinco *migrations* (scripts de versionamento do banco de dados) do Prisma aplicaram-se sem erros, e o esquema refletiu corretamente as entidades **User**, **Imovel** e **Contato**;
- A integridade referencial foi preservada, com a regra *ON DELETE CASCADE* (exclusão em cascata, que remove registros dependentes automaticamente) em **Contato** acionado corretamente em testes de exclusão de imóvel;
- A aplicação tolerou a reinicialização do *dyno* (unidade de processamento do Render), mantendo o estado no banco de dados. Conclui-se pela conformidade com o RNF04, observando-se o *backup* e a persistência confiável oferecidos pelo Neon como serviço gerenciado.

9.1.5 Disponibilidade

A disponibilidade do sistema em ambiente de produção foi observada durante a fase de testes, com acesso periódico via navegador e via requisições *HTTP* automatizadas. Foram observados:

- Disponibilidade contínua da aplicação *front-end* na Vercel, sem janelas de indisponibilidade;
- Disponibilidade do *back-end* no *Render* condicionada ao plano gratuito, que impõe *sleep* após 15 minutos sem requisições, com reativação sob demora de aproximadamente 30 a 60 segundos. Tal comportamento é próprio do plano gratuito e foi plenamente identificado durante os testes;
- Disponibilidade do banco de dados no Neon sem flutuações;
- Disponibilidade das imagens no Cloudinary sem flutuações.

Conclui-se pela conformidade parcial com o RNF05: o sistema está acessível, mas sujeito à degradação do *cold start* no *back-end*, constituindo uma limitação a ser mitigada mediante a adesão futura a um plano pago.

9.1.6 Portabilidade

Foram realizadas validações cruzadas em quatro navegadores (*Google*

Chrome 130, *Mozilla Firefox* 132, *Microsoft Edge* 130 e *Safari* 17.4), em *desktop* e em dispositivo móvel (*iPhone* com iOS 17 e *Android* 14), observando-se que:

- Em todos os navegadores, as páginas renderizaram corretamente, sem quebras de *layout*;
- As interações de *hover*, foco e formulários comportaram-se de forma padronizada, com suporte dos componentes *Radix UI* em todos os navegadores modernos;
- O upload de imagens, autenticação e navegação ocorreram normalmente em todos os navegadores testados;
- Em *smartphone*, o *layout* adaptou-se automaticamente, sem necessidade de *zoom* ou *scroll* horizontal.

Conclui-se pela conformidade total com o RNF06.

9.1.7 Síntese dos resultados do teste de sistema

O Quadro a seguir sintetiza os resultados alcançados em cada atributo.

Quadro 4 – Resultados do Teste de Sistema

Requisito	**Status alcançado**	**Observações**
Desempenho	Atendido parcialmente	O *cold start* do Render gratuito impõe degradação inicial periódica; em condições comuns, os tempos estão dentro da faixa.
Usabilidade	Atendido	Interface responsiva consistente em distintos tamanhos de tela.
Segurança	Atendido	Senha em *hash*, autenticação *JWT* ativa, verificação de posse e validação Zod em funcionamento.
Confiabilidade	Atendido	Persistência PostgreSQL

		no Neon com *migrations* versionadas e integridade referencial preservada.
Disponibilidade	Atendido	Limitação do plano gratuito do Render *sleep* por inatividade).
Portabilidade	Atendido	Funcionamento conforme em *Chrome, Firefox, Edge e Safari, em desktop e mobile*.

Fonte: Elaborado pelo autor (2026).

9.2 Teste de aceitação do usuário (User Acceptance Testing)

O teste de aceitação do usuário (UAT – *User Acceptance Testing*) consiste na validação do software por parte dos futuros usuários, com o objetivo de verificar se o sistema atende às suas expectativas e necessidades reais de uso. Diferente do teste de sistema, que avalia aspectos técnicos e funcionais de forma isolada, o UAT envolve a interação direta do participante com a aplicação, permitindo mensurar percepções subjetivas como usabilidade, facilidade de aprendizado e satisfação geral (Pressman, 2016).

Para a realização do UAT do VelhaCap Homes, utilizou-se a Escala de Usabilidade de Sistema (SUS – System Usability Scale), instrumento padronizado proposto por Brooke (1996) e validado para o português brasileiro por Lourenço, Carmona e Lopes (2023). O SUS é composto por 10 afirmações avaliadas em escala Likert de 1 a 5 (de "Discordo totalmente" a "Concordo totalmente"), sendo 5 de natureza positiva (ímpares) e 5 de natureza negativa (pares). A pontuação final é calculada por meio de uma fórmula que resulta em um valor entre 0 e 100, sendo 68 a média de referência e valores acima de 72,6 classificados como "Bom" (Sauro, 2011). O tamanho amostral de 15 participantes está alinhado às recomendações de Sauro (2011) para avaliações com a Escala de Usabilidade de Sistema, que indicam amostras dessa ordem como suficientes para identificar as principais tendências de percepção de usabilidade, ainda que não configurem representatividade estatística

da população geral de usuários. O instrumento foi aplicado por meio de formulário eletrônico no Google Forms, com duração estimada de 2 minutos por participante. O período de aplicação compreendeu os dias 15 a 18 de julho de 2026.

O instrumento foi aplicado por meio de formulário eletrônico no Google Forms, com duração estimada de 2 minutos por participante. O período de aplicação compreendeu os dias 15 a 18 de julho de 2026. Participaram da avaliação 15 pessoas, sendo a maioria na faixa etária de 19 a 25 anos (8 participantes), seguida por 26 a 35 anos (5 participantes), acima de 35 anos (1 participante) e até 18 anos (1 participante). Todos os participantes acessaram o sistema em produção antes de responder ao formulário, sendo que 12 deles navegaram pelas páginas do sistema e 2 acessaram apenas a página inicial. A coleta de dados foi realizada de forma anônima, com consentimento implícito por meio da submissão do formulário, conforme os princípios éticos que orientam a pesquisa acadêmica. O roteiro completo do instrumento de coleta encontra-se disponível no Anexo A deste trabalho.

O Quadro 5 apresenta as respostas individuais dos 15 participantes às 10 afirmações do SUS, com a pontuação calculada para cada um.

Quadro 5 – Respostas dos participantes à Escala de Usabilidade de Sistema (SUS)

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Pontuação SUS
P01	4	4	4	3	3	4	5	4	2	3	57,5
P02	2	2	4	3	3	3	3	4	2	2	47,5
P03	5	2	4	1	4	4	2	5	1	1	70,0
P04	3	1	5	1	5	1	5	1	5	1	90,0
P05	5	1	5	1	5	1	5	1	5	1	90,0
P06	4	1	5	2	4	2	4	1	3	1	77,5
P07	3	1	4	2	5	2	5	2	4	1	80,0
P08	3	1	4	2	4	2	4	2	3	2	67,5
P09	3	2	4	2	4	2	5	2	3	2	67,5
P10	5	1	5	1	5	1	5	2	4	1	87,5
P11	4	1	5	1	5	2	5	2	4	1	82,5
P12	5	1	5	1	5	2	5	2	4	1	82,5
P13	4	1	5	1	5	2	5	2	4	1	82,5
P14	5	1	5	1	5	2	5	2	4	1	82,5
P15	3	1	5	1	5	2	5	2	4	1	80,0

Fonte: Elaborado pelo autor (2026).

A pontuação média dos 15 participantes foi de 76,8 pontos, classificando o sistema na faixa "Bom" de acordo com à escala de interpretação de Sauro (2011). A nota mínima registrada foi de 47,5 (participante P02, que nunca havia acessado o

sistema anteriormente) e a nota máxima foi de 90,0 (participantes P04 e P05). Dos 15 participantes, 12 (80%) obtiveram pontuação acima de 68 (média de referência), enquanto apenas 2 ficaram abaixo de 60.

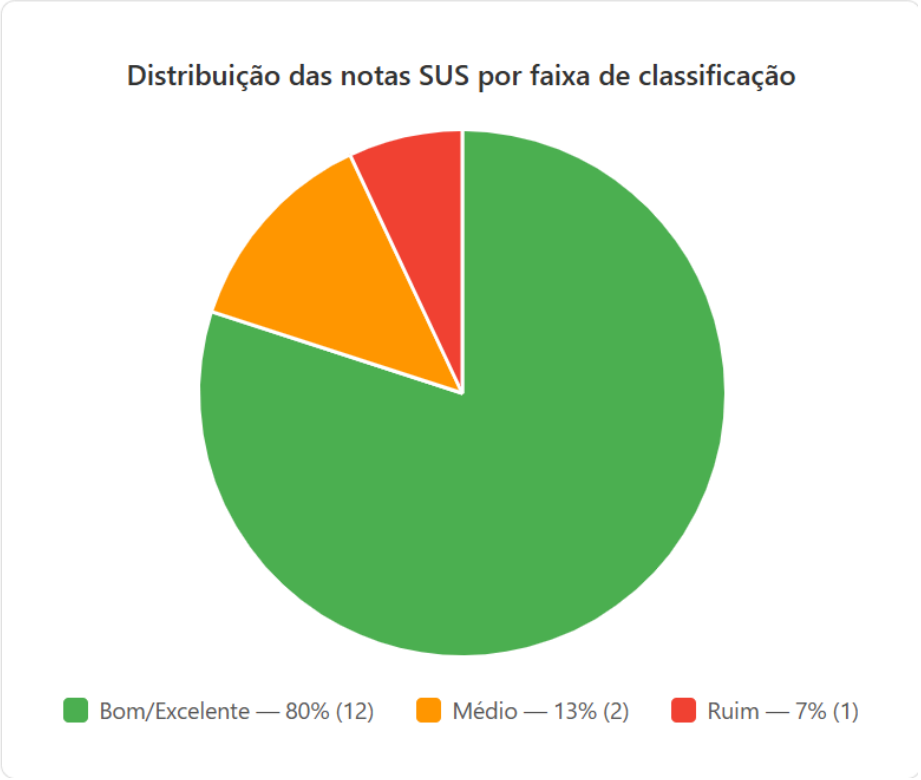
10 VALIDAÇÃO

A validação do sistema, além dos testes técnicos descritos na Seção 9.1, foi complementada pela análise das respostas obtidas no teste de aceitação do usuário (Seção 9.2), permitindo avaliar a percepção dos participantes sobre atributos como usabilidade, curva de aprendizado, satisfação e aceitação.

10.1 Distribuição das notas SUS

A Figura 12 apresenta a distribuição das pontuações obtidas pelos participantes, agrupadas em três faixas de classificação.

Figura 12 – Distribuição das notas SUS de acordo à faixa de classificação



Fonte: Elaborado pelo autor (2026).

A análise revela que a esmagadora maioria dos participantes (80%) classificou o sistema como "Bom" ou "Excelente", com pontuações entre 70 e 90. Apenas dois participantes obtiveram notas na faixa "Médio" (entre 50 e 68), e um participante na faixa "Ruim" (abaixo de 50).

10.2 Análise por atributo de usabilidade

A análise das respostas individuais permite identificar os atributos de usabilidade que obtiveram melhor e pior desempenho:

- Facilidade de uso (Q3): 13 dos 15 participantes (87%) concordaram ou concordaram totalmente que o sistema é fácil de usar. Apenas 2 participantes deram nota neutra.
- Integração das funções (Q5): 12 participantes (80%) avaliaram positivamente a integração entre as funcionalidades do sistema.
- Curva de aprendizado (Q7): 11 participantes (73%) concordaram que a maioria das pessoas aprende a usar o sistema rapidamente.
- Necessidade de ajuda técnica (Q4): 10 participantes (67%) discordaram de que precisariam de ajuda técnica para usar o sistema, indicando boa autonomia do usuário.
- Inconsistência (Q6): 9 participantes (60%) discordaram de que o sistema apresenta inconsistências, embora 4 tenham dado nota neutra, sinalizando possíveis ajustes na uniformidade da interface.

10.3 Análise das respostas abertas

As respostas abertas do formulário complementam a análise quantitativa:

- Funcionalidade mais utilizada: Ver detalhes de imóvel (6 participantes), cadastrar imóvel (4), buscar imóveis com filtros (3), painel do anunciante (1), contatar anunciante (1).O
- O que foi mais fácil: Navegação simples e intuitiva, cadastro rápido, busca por filtros, contato direto via WhatsApp.
- O que poderia melhorar: Um participante sugeriu a inclusão de um botão de pesquisa nos filtros (atualmente a busca é apenas por teclado). Outros participantes não identificaram dificuldades relevantes, descrevendo o sistema como "completo", "prático" e "fácil de usar".

10.4 Hipótese para a nota baixa do participante P02

O participante P02 obteve a menor pontuação (47,5) e é o único que declarou nunca ter acessado o sistema anteriormente. Como hipótese explicativa, não confirmada por entrevista de acompanhamento, propõe-se que a falta de familiaridade prévia com a plataforma tenha influenciado a percepção de complexidade e a sensação de necessidade de ajuda técnica relatada nas respostas. Essa hipótese, embora não verificável a partir dos dados coletados, reforça a importância de guias iniciais ou tutoriais para novos usuários, sugestão já apontada nas considerações finais.

10.5 Síntese da validação

A validação com usuários, por meio da Escala de Usabilidade de Sistema, confirmou que o VelhaCap Homes apresenta nível de usabilidade classificado como "Bom" (média SUS = 76,8), com altos índices de satisfação nas dimensões de facilidade de uso, integração de funções e curva de aprendizado. As poucas respostas negativas concentraram-se em participantes com menor contato prévio com o sistema, indicando que a experiência de uso tende a melhorar com a navegação. Esses resultados complementam os testes técnicos da Seção 9.1 e reforçam a conclusão de que o sistema atende adequadamente aos requisitos não funcionais de usabilidade (RNF02).

11 CONSIDERAÇÕES FINAIS

O presente trabalho descreveu o desenvolvimento do VelhaCap Homes, sistema web para anúncios de imóveis para locação em Oeiras – PI. A solução cumpriu seus objetivos ao centralizar ofertas em uma plataforma estruturada, permitir busca e filtros por tipo, bairro, preço e quartos, viabilizar o cadastro e a gestão de imóveis por locadores autenticados e estabelecer contato direto entre locatário e anunciante. Para os locadores, o sistema ampliou a visibilidade dos anúncios e forneceu métricas analíticas no painel do anunciante; para os locatários, proporcionou maior transparência e reduziu o tempo de busca. Em termos acadêmicos, o projeto contemplou todas as fases do ciclo de desenvolvimento de software.

Os testes de sistema evidenciaram o atendimento dos requisitos não funcionais de usabilidade, segurança, confiabilidade e portabilidade. O desempenho, embora apresente limitação pontual com o *cold start* do Render gratuito, obteve notas máximas no *PageSpeed Insights* (ver Figura 11). A disponibilidade encontra-se condicionada à mesma limitação do plano gratuito. A validação com 15 usuários por meio da Escala de Usabilidade de Sistema (SUS) resultou em pontuação média de 76,8 (classificação "Bom"), confirmando a percepção positiva de usabilidade e facilidade de aprendizado.

Embora o sistema tenha atendido aos requisitos essenciais, há pontos que podem ser aprimorados para oferecer uma melhor experiência ao usuário. A ausência de recuperação de senha obriga o usuário a entrar em contato com o desenvolvedor caso esqueça suas credenciais, o que é inconveniente para uma plataforma em uso real a implementação de recuperação por link temporário e confirmação de e-mail eliminaria essa barreira. Outra melhoria seria um mecanismo de favoritos que permitiria que o locatário salvasse imóveis de interesse para comparação posterior, tornando a busca mais produtiva. No aspecto técnico, a migração do armazenamento do *token JWT* para *cookies httpOnly* aumentaria a segurança da sessão, e a migração do *back-end* para um plano pago no Render eliminaria o *cold start*, garantindo carregamento instantâneo a qualquer momento.

REFERÊNCIAS

BROOKE, John. *SUS: a 'quick and dirty' usability scale.* In: JORDAN, P. W. et al. (Eds.). Usability Evaluation in Industry. London: Taylor & Francis, 1996. p. 189-194.

CLOUDINARY. *Cloudinary Documentation.* Disponível em: https://cloudinary.com/documentation. Acesso em: 15 jul. 2026.

EXPRESS. *Express documentation.* Disponível em: https://expressjs.com/. Acesso em: 15 jul. 2026.

GARRETT, Jesse James. *The Elements of User Experience*: User-Centered Design for the Web and Be yond. 2. ed. Berkeley: New Riders, 2011.

GIL, Antonio Carlos. *Como elaborar projetos de pesquisa.* 6. ed. São Paulo: Atlas, 2017.

GOOGLE. *PageSpeed Insights.* Disponível em: https://pagespeed.web.dev/. Acesso em: 15 jul. 2026.

KRUG, Steve. *Não Me Faça Pensar: Atualizado*: Uma Abordagem de Senso Comum à Usabilidade na Web e Mobile. 3. ed. Rio de Janeiro: Alta Books, 2014.

LOURENÇO, Cintia de Carvalho et al. *Adaptação transcultural do System Usability Scale para o português do Brasil.* Revista da Escola de Enfermagem da USP, v. 57, e20230059, 2023. DOI: 10.1590/1980-220X-REEUSP-2023-0059.

NEON. *Neon Documentation.* Disponível em: https://neon.tech/docs/introduction. Acesso em: 15 jul. 2026.

NODE.JS FOUNDATION. *Node.js Documentation.* Disponível em: https://nodejs.org/docs/latest/api/. Acesso em: 15 jul. 2026.

PRISMA. *Prisma Documentation.* Disponível em: https://www.prisma.io/docs. Acesso em: 15 jul. 2026.

PRESSMAN, Roger S. *Engenharia de Software: Uma Abordagem Profissional.* 8. ed. Porto Alegre: AMGH, 2016.

REACT. *React Documentation.* Disponível em: https://react.dev/. Acesso em: 15 jul. 2026.

RENDER. *Render Documentation.* Disponível em: https://render.com/docs. Acesso em: 15 jul. 2026.

SAURO, Jeff. *A Practical Guide to the System Usability Scale*: A Hands-On Guide for UX Researchers. Denver: MeasuringU Press, 2011.

SHNEIDERMAN, Ben et al. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. 6. ed. Boston: Pearson, 2016.

SOMMERVILLE, Ian. *Engenharia de Software.* 10. ed. São Paulo: Pearson Brasil, 2019.

TANSTACK. *TanStack Documentation.* Disponível em: https://tanstack.com/. Acesso em: 15 jul. 2026.

THIOLLENT, Michel. *Metodologia da pesquisa-ação.* 18. ed. São Paulo: Cortez, 2011.

TURBAN, Efraim et al. *Electronic Commerce: A Managerial and Social Networks Perspective.* 8. ed. Cham: Springer, 2015.

VERCEL. *Vercel Documentation.* Disponível em: https://vercel.com/docs. Acesso em: 15 jul. 2026.

YIN, Robert K. *Estudo de Caso: Planejamento e Métodos.* 5. ed. Porto Alegre: Bookman, 2015.

ZOD. *Zod Documentation.* Disponível em: https://zod.dev. Acesso em: 15 jul. 2026.

ANEXOS

ANEXO A — ROTEIRO DO QUESTIONÁRIO DE USABILIDADE (SUS)

Instruções: Responda a cada uma das 10 afirmações abaixo indicando seu grau de concordância, variando de 1 (Discordo Totalmente) a 5 (Concordo Totalmente). Responda com base em sua experiência de uso do sistema VelhaCap Homes.

Nº	Afirmação	1	2	3	4	5
1	Acho que usaria o sistema frequentemente	☐	☐	☐	☐	☐
2	Acho o sistema desnecessariamente complexo	☐	☐	☐	☐	☐
3	Acho o sistema fácil de usar	☐	☐	☐	☐	☐
4	Acho que precisaria de ajuda técnica para usar o sistema	☐	☐	☐	☐	☐
5	Acho que as diversas funções do sistema estão bem integradas	☐	☐	☐	☐	☐
6	Acho que o sistema apresenta muitas inconsistências	☐	☐	☐	☐	☐
7	Imagino que a maioria das pessoas aprenderia a usar o sistema rapidamente	☐	☐	☐	☐	☐
8	Acho o sistema trabalhoso de usar	☐	☐	☐	☐	☐
9	Sinto-me confiante ao usar o sistema	☐	☐	☐	☐	☐
10	Preciso aprender muitas coisas antes de conseguir usar o sistema	☐	☐	☐	☐	☐

Perguntas complementares:

1. Qual funcionalidade do sistema você mais utilizou?
2. O que você considerou mais fácil de usar no sistema?
3. O que você sugere que seja melhorado no sistema?