UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
FACULDADE DE ENGENHARIA DE COMPUTAÇÃO

WESLEY DAVID SANTOS LIMA

LAGOTUR: PROTÓTIPO DE APLICATIVO MÓVEL PARA APOIO AO TURISMO NA REGIÃO DO LAGO DE TUCURUÍ

TUCURUÍ - PA
2025

UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
FACULDADE DE ENGENHARIA DE COMPUTAÇÃO

WESLEY DAVID SANTOS LIMA

LAGOTUR: PROTÓTIPO DE APLICATIVO MÓVEL PARA APOIO AO TURISMO NA REGIÃO DO LAGO DE TUCURUÍ

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia de Computação, do Campus Universitário de Tucuruí, da Universidade Federal do Pará, como requisito parcial para obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr(a). Daniel da Conceição Pinheiro

TUCURUÍ
2025

WESLEY DAVID SANTOS LIMA

LAGOTUR: PROTÓTIPO DE APLICATIVO MÓVEL PARA APOIO AO TURISMO NA REGIÃO DO LAGO DE TUCURUÍ

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia de Computação, do Campus Universitário de Tucuruí, da Universidade Federal do Pará, como requisito parcial para obtenção do título de Bacharel em Engenharia de Computação.

Data da aprovação: ____/____/_____

Conceito: _____

BANCA EXAMINADORA

_____

Nome completo do(a) orientador(a) precedido de titulação
Instituição a que pertence

_____

Nome completo do(a) examinador(a) precedido de titulação
Instituição a que pertence

AGRADECIMENTOS

Agradeço primeiramente a Deus, pelos inúmeros favores através dos quais foi possível chegar a este ponto no percurso da minha trajetória acadêmica

À minha família, especificamente minha mãe Francinalva, meu pai Manoel e meus irmãos, Welson e Cintia que sempre me deram motivação e apoio para realização desse sonho.

Ao meu professor e orientador Dr. Daniel da Conceição Pinheiro por compartilhar conhecimento, pelo incentivo e por confiar a mim o desenvolvimento desse projeto.

Aos meus amigos da turma de Engenharia de Computação 2018, especialmente ao grupo do Zorel, pelos estudos em conjunto e apoio nas atividades ao decorrer do curso. Um agradecimento especial aos amigos Danilo, Roqueudison e Keven pela parceria e amizade durante e após o curso.

A Instância de Governança Regional LagoTur por confiar no projeto, abraçar a ideia e compartilhar conhecimento turístico na região do lago de Tucuruí.

A minha companheira Iarly pela motivação e apoio no decorrer dessa caminhada.

E a todos que chegaram a contribuir de forma positiva para conclusão dessa etapa.

EPÍGRAFE

“Pense globalmente e atue localmente.”
(John Lennon)

RESUMO

Este trabalho apresenta o desenvolvimento do protótipo de um aplicativo móvel chamado LagoTur, voltado à promoção do turismo regional na área de influência do lago de Tucuruí, visando fortalecer a visibilidade dos municípios participantes e do comércio local. A aplicação foi concebida para centralizar, em um único ambiente digital, informações sobre estabelecimentos, eventos, roteiros turísticos e pontos para visita, de modo a auxiliar turistas e visitantes na exploração organizada das cidades catalogadas, reunindo dados relevantes para o planejamento e experiência turística. O sistema foi desenvolvido utilizando o framework React Native, em conjunto com a ferramenta Expo, que possibilita testes em tempo real e integração facilitada com bibliotecas atualizadas, auxiliando no tratamento e armazenamento de dados além de autenticação de usuários. A aplicação conta também com uso da plataforma Google Firebase para armazenamento e tratamento de dados, permitindo maior flexibilidade e escalabilidade da solução. O projeto foi conduzido em estreita colaboração com a Instância de Governança Regional LagoTur, organização responsável por curadoria das informações e pela definição de requisitos funcionais alinhados à realidade do turismo na região, dessa forma o aplicativo propõe uma plataforma acessível e expansível, orientada à melhoria da experiência do visitante e ao aumento da visibilidade dos atrativos e serviços turísticos locais.

Palavras-chave: Turismo regional, Aplicativo móvel, Lago de Tucuruí, Roteiros turísticos, React Native, Firebase.

ABSTRACT

This work presents the development of a prototype mobile application called LagoTur, aimed at promoting regional tourism in the area influenced by lake Tucuruí, with the goal of strengthening the visibility of participating municipalities and local businesses. The application was designed to centralize, in a single digital environment, information about establishments, events, tourist routes, and points of interest, in order to assist tourists and visitors in the organized exploration of the cataloged cities, gathering relevant data for tourism planning and experience. The system was developed using the React Native framework, in conjunction with the Expo tool, which allows for real-time testing and easy integration with updated libraries, assisting in data processing and storage, as well as user authentication. The application also uses the Google Firebase platform for data storage and processing, allowing for greater flexibility and scalability of the solution. The project was conducted in close collaboration with the LagoTur Regional Governance Body, the organization responsible for curating the information and defining functional requirements aligned with the reality of tourism in the region. Therefore, the application proposes an accessible and expandable platform, aimed at improving the visitor experience and increasing the visibility of local tourist attractions and services.

Keywords: Regional tourism, Mobile application, Lake Tucuruí, Tourist routes, React Native, Firebase.

LISTA DE FIGURAS

LISTA DE CÓDIGOS-FONTE

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE ABREVIATURA E SIGLAS

Embratur ..... Agência Brasileira de Promoção Internacional do Turismo
FOFA ..... Forças, Oportunidades, Fraquezas, Ameaças
HTML ..... *Hypertext Markup Language*
ID ..... Identificador
IGR LagoTur ..... Instância de Governança Regional LagoTur
IGRs ..... Instâncias de Governança Regional
JSON ..... *JavaScript Object Notation*
OMT ..... Organização Mundial do Turismo
PNMT ..... Programa Nacional de Municipalização do Turismo
PRT ..... Programa de Regionalização do Turismo
SDKs ..... *Software Development Kits*
SIMTEC ..... Simpósio de Tecnologia
TICs ..... Tecnologias da Informação e Comunicação
TOPAM ..... Torneio de Pesca da Amazônia
UI ..... Interface do usuário
UX ..... Experiência do usuário

SUMÁRIO

1 INTRODUÇÃO

A atividade turística tem se consolidado como um importante vetor de desenvolvimento econômico, social e cultural, ao gerar empregos, renda e dinamizar diferentes setores produtivos, especialmente em territórios que contam com recursos naturais e atrativos paisagísticos relevantes, como é o caso do Brasil. Nesse sentido, Oliveira (2019) destaca que o turismo, quando fomentado de forma planejada, apresenta forte capacidade de impulsionar a economia local e produzir riqueza social nos destinos em que se desenvolve.

Estudos recentes que utilizam a análise FOFA (Forças, Oportunidades, Fraquezas, Ameaças), ferramenta de planejamento e análise estratégica aplicada ao município de Tucuruí apontam o lago e suas ilhas como importantes oportunidades para o desenvolvimento do turismo regional, desde que acompanhadas de planejamento adequado (SILVA *et al*., 2020). Essa análise reforça a caracterização da região como um polo turístico em consolidação, evidenciada pelo crescimento da demanda de visitantes atraídos por seus atrativos naturais, diversidade culinária e crescente aumento de eventos sediados na localidade.

Com o avanço das tecnologias digitais e dos aplicativos móveis, tornou-se possível centralizar informações turísticas em plataformas acessíveis, oferecendo ao usuário acesso ágil e em tempo real a dados sobre destinos, serviços e atividades, além de apoiar a gestão e a promoção de localidades turísticas. Tendo em vista essa narrativa e o evidente crescimento do turismo na região, a Instância de Governança Regional LagoTur (IGR LagoTur) buscou parceria com a Universidade Federal do Pará (UFPA), com objetivo de reunir em um aplicativo móvel funcionalidades que possam ajudar a fomentar o turismo local.

Diante desse cenário, o presente trabalho apresenta o desenvolvimento do protótipo do aplicativo móvel LagoTur, concebido para centralizar informações sobre estabelecimentos, eventos e roteiros turísticos da região, apoiando a experiência do turista e fortalecendo o comércio local. A solução foi desenvolvida em React Native, com uso do Expo para testes em tempo real, integração a recursos nativos de dispositivo e utilização do Google Firebase para armazenamento de dados e autenticação de usuários. O projeto é conduzido em parceria com a IGR LagoTur, que contribui para alinhar as funcionalidades do sistema às necessidades reais do turismo regional, configurando o sistema desenvolvido como uma iniciativa tecnológica acessível e orientada ao desenvolvimento regional.

1.1 Justificativa

A justificativa para o desenvolvimento do aplicativo LagoTur fundamenta-se na relevância do turismo como atividade capaz de gerar emprego, renda e visibilidade para cidades com potencial turístico, como as que compõem a área de influência do Lago de Tucuruí. Na região, as informações sobre estabelecimentos, eventos, roteiros e serviços em sua maioria encontram-se dispersas em diferentes canais, o que dificulta o planejamento e tomada de decisão por parte dos visitantes.

Diante desse cenário, torna-se pertinente a criação de uma solução tecnológica que centralize, em um único ambiente digital, dados confiáveis sobre o destino, organizados de forma simples. Sabendo disso, o projeto LagoTur tem como objetivo implementar uma ferramenta de apoio ao turista e ao morador, permitindo acesso rápido a informações sobre hospedagem, alimentação, lazer, serviços e programações locais, ao mesmo tempo em que contribui para fortalecer a imagem da região, ampliar a visibilidade dos empreendimentos cadastrados e apoiar as ações turísticas da localidade.

Sob a perspectiva acadêmica e tecnológica, o projeto busca justificativas ao possibilitar uma aplicação integrada de conhecimentos adquiridos no curso de Engenharia da Computação para desenvolvimento de um protótipo funcional em React Native com uso do Expo e utilização do Firebase para armazenamento e autenticação. Além disso, o desenvolvimento de um protótipo funcional, avaliado por usuários reais, contribui para a formação prática do discente, ao mesmo tempo em que gera uma solução tecnológica, passível de evolução futura com novos recursos.

1.2 Objetivos

1.2.1 Objetivos gerais

Desenvolver um protótipo de aplicativo móvel multiplataforma, voltado à centralização e organização de informações turísticas da região do Lago de Tucuruí, com o intuito de aprimorar a experiência do visitante e apoiar a divulgação e o fortalecimento do turismo e do comércio local.

1.2.2 Objetivos Específicos

- Analisar e planejar os requisitos funcionais e não funcionais do aplicativo em conjunto com a IGR LagoTur, identificando necessidades de turistas, visitantes e do comércio local da região do Lago de Tucuruí.
- Projetar a arquitetura da aplicação, definindo módulos funcionais, estrutura de navegação, organização das informações e exibição dos dados para o usuário.
- Implementar um protótipo funcional do aplicativo móvel LagoTur utilizando React Native com Expo para desenvolvimento e o Google Firebase para armazenamento de dados e autenticação de usuários.
- Cadastrar e estruturar informações reais de estabelecimentos, eventos e roteiros turísticos de forma organizada e coerente.
- Desenvolver modos de interação do usuário com o sistema, como registros de avaliações e comentários sobre os itens exibidos na aplicação.
- Realizar testes práticos de usabilidade com usuários reais, a fim de coletar percepções sobre utilidade, facilidade de uso, organização das informações e desempenho do aplicativo.
- Analisar os dados obtidos nos testes com usuários, avaliando em que medida o protótipo atende aos objetivos propostos, identificando pontos fortes, limitações e possibilidades de aprimoramento para versões futuras do LagoTur.

1.3 Metodologia de pesquisa

A metodologia adotada neste trabalho caracteriza-se como uma pesquisa aplicada, de natureza descritiva e exploratória. É aplicada pois visa o desenvolvimento de um artefato tecnológico voltado a disponibilização de informações turísticas na região do Lago de Tucuruí. Assume caráter descritivo e exploratório por buscar, simultaneamente, compreender o contexto do turismo local e descrever as funcionalidades propostas para apoiar turistas, visitantes e o comércio regional.

Inicialmente, realizou-se um levantamento bibliográfico e documental sobre turismo regional, uso de tecnologias móveis em apoio ao turismo, usabilidade de aplicativos e as principais tecnologias envolvidas no desenvolvimento do projeto. Em seguida, desenvolveu-se uma pesquisa junto à Instância Governamental Regional LagoTur para alinhamento das ideias à realidade da região, os dados obtidos foram utilizados para a definição do funcionamento geral da aplicação.

Paralelamente, foram conduzidos estudos sobre a aplicação dessas diretrizes no framework React Native, bem como a melhor forma de estar armazenando os

dados do sistema a ser desenvolvido, visando à integração e à implementação das funcionalidades da forma mais adequada à proposta sugerida para desenvolvimento.

1.4 Estrutura do trabalho

O presente trabalho está organizado em sete capítulos principais, além da seção de referências bibliográficas, de modo a apresentar de forma clara e sistemática todo o desenvolvimento da pesquisa. Cada capítulo possui objetivos específicos e contribui para a compreensão global do estudo, conforme descrito a seguir:

- Capítulo 1 – Introdução: Apresenta o tema da pesquisa, a problematização, os objetivos gerais e específicos, a justificativa, e a relevância do estudo. Este capítulo estabelece o contexto e fundamenta a necessidade da pesquisa.
- Capítulo 2 – Fundamentação Teórica: Contém a revisão da literatura relacionada ao tema, conceitos, teorias e estudos que sustentam o trabalho. Aqui são abordados os principais conhecimentos prévios que embasam o desenvolvimento do projeto.
- Capítulo 3 – Trabalhos Relacionados: Apresenta pesquisas e projetos semelhantes já realizados, permitindo identificar lacunas, limitações e oportunidades que justificam a condução do estudo.
- Capítulo 4 – Metodologia: Descreve os procedimentos, métodos, técnicas e ferramentas utilizadas na pesquisa ou no desenvolvimento do projeto. Este capítulo detalha como os dados foram coletados, analisados e interpretados.
- Capítulo 5 – Desenvolvimento: Apresenta detalhadamente o processo de implementação do projeto ou da solução proposta, incluindo etapas, recursos utilizados e decisões de projeto.
- Capítulo 6 – Resultados: Exibe os resultados obtidos, analisando e discutindo os dados ou produtos gerados durante a execução do trabalho, demonstrando a eficácia da metodologia aplicada.
- Capítulo 7 – Conclusão: Resume as principais contribuições do estudo, apresenta considerações finais, limitações e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 O turismo e sua importância para o desenvolvimento regional

Reconhecido como um setor de grande importância econômica e cultural, o turismo constitui um vetor estratégico para o desenvolvimento regional. A atividade contribui para a preservação da identidade e das tradições locais, bem como para a conservação do patrimônio histórico (BARBOSA, 2005). Para Netto e Trigo (2009), configura-se como um convite à interação entre pessoas, etnias e culturas distintas, promovendo o contato com diferentes modos de vida. Trata-se de uma oportunidade de conhecer a diversidade natural do planeta, assim como a riqueza que cada destino carrega.

No âmbito econômico, o turismo atua como um meio de redução das desigualdades regionais, geração de receita e criação de empregos, por ser uma atividade predominantemente do setor de serviços, intensiva em mão de obra e com elevado potencial multiplicador sobre a economia local e nacional (RABAHY, 2020).

No Brasil, a consolidação do turismo enquanto setor estruturado ocorreu a partir de políticas públicas específicas, que tiveram início com a criação da Agência Brasileira de Promoção Internacional do Turismo (Embratur), a partir do Decreto-Lei nº 55, de 1966 (BRASIL, 1966). Desde então, o país tem avançado na formulação de instrumentos voltados ao fortalecimento regional e à gestão estratégica da atividade. Atualmente, o setor é orientado pela Política Nacional de Turismo, instituída pela Lei nº 11.771/2008, cuja execução ocorre por meio dos Planos Nacionais de Turismo, os quais estabelecem diretrizes, estratégias e metas voltadas à promoção do turismo sustentável em todo o território nacional (BRASIL, 2008).

De acordo com dados da Organização Mundial do Turismo (OMT), o Brasil tem registrado recordes de entrada de turistas internacionais. Apenas no primeiro trimestre de 2025, o país apresentou crescimento de 48% em relação a 2024, ocupando a segunda posição global em desempenho, atrás apenas do Paraguai (BRASIL, 2025). Segundo a Embratur, até junho de 2025 os visitantes estrangeiros injetaram mais de R$ 23 bilhões na economia nacional, consolidando o turismo como um motor relevante da economia nacional (EMBRATUR, 2025).

Um dos marcos na política turística brasileira foi a criação do Programa de Regionalização do Turismo (PRT), no ano de 2004, em substituição ao Programa Nacional de Municipalização do Turismo (PNMT). O PRT foi implantado com base nos

princípios de descentralização, integração, participação social e sustentabilidade, em prol de fortalecer a articulação entre os municípios e a gestão regional do setor turístico no país (BRASIL, 2013).

Estudos apontam que o programa tem produzido resultados positivos, mas também enfrenta desafios. Oliveira e Pereira (2020) afirmam que o PRT fortaleceu a cadeia produtiva do turismo local, contribuindo para o avanço do turismo sustentável. Entretanto, ao analisar sua implementação prática, Lima (2025) observou baixa adesão e envolvimento tanto por parte dos gestores quanto da comunidade, além de deficiências na infraestrutura e carência de pessoal qualificado

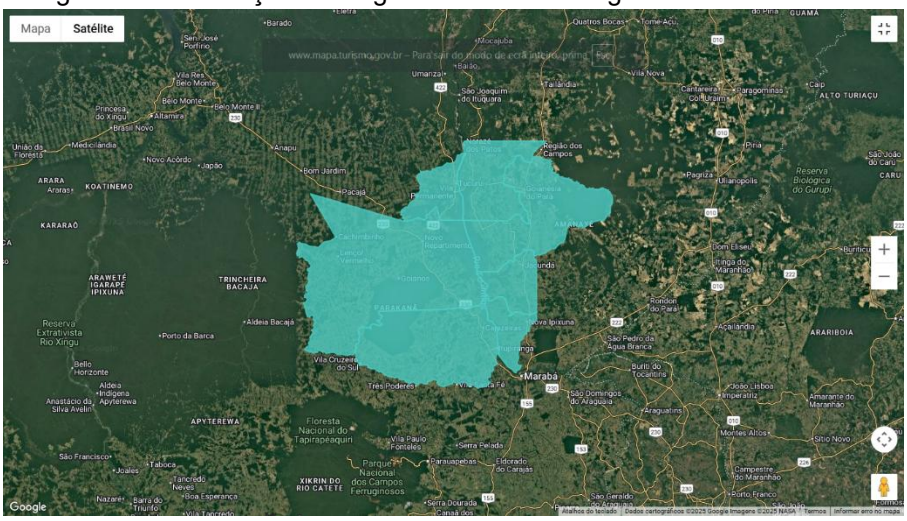
2.2 Instância de Governança Regional - IGR

No contexto da regionalização do turismo, surgem as denominadas regiões turísticas que, segundo o Ministério do Turismo, são formadas por vários municípios que compartilham características históricas, culturais, econômicas ou geográficas e se articulam por meio de Instâncias de Governança Regional (IGRs) para desenvolver o turismo de forma integrada (BRASIL, 2017).

De acordo com o *Programa de Regionalização do Turismo – Roteiros do Brasil*, “a Instância de Governança Regional é uma organização com participação do poder público e dos atores privados dos municípios componentes das regiões turísticas, com o papel de coordenar o Programa em âmbito regional” (BRASIL, 2007, Mód. 3, p. 16). Dessa forma, a função central das IGRs é articular os diversos atores locais e regionais, de modo que haja integração e cooperação entre os municípios de uma mesma região turística, e o consequente planejamento e execução do fortalecimento e da valorização desses territórios.

Ao todo, são 357 regiões turísticas cadastradas no Mapa do Turismo Brasileiro, outro instrumento do PRT que organiza, classifica e delimita as áreas a serem priorizadas pelo Ministério do Turismo para o planejamento, a gestão e a implementação de políticas públicas voltadas ao turismo (BRASIL, 2025). A Região Turística do Lago de Tucuruí está entre as 14 do estado do Pará, abrangendo sete municípios: Tucuruí, Breu Branco, Goianésia do Pará, Jacundá, Nova Ipixuna, Itupiranga e Novo Repartimento (Figura 1).

Figura 1 - Delimitação da Região Turística do Lago de Tucuruí.



Fonte: Mapa do Turismo Brasileiro, 2025.

2.3 Região turística do Lago de Tucuruí

Inaugurada em 1984, a Usina Hidrelétrica de Tucuruí, localizada na bacia do rio Tocantins, constitui um dos maiores empreendimentos hidrelétricos do Brasil. Seu reservatório possui área inundada de aproximadamente 2.850 km², abrangendo cerca de 1.800 ilhas em seu perímetro (COMISSÃO MUNDIAL DE BARRAGENS, 1999)

O represamento do rio Tocantins pela usina deu origem ao Lago de Tucuruí, uma formação artificial que constitui uma Área de Proteção Ambiental. O reservatório, composto por fragmentos de Floresta Ombrófila Densa, configura uma paisagem singular com potencial para fomentar diferentes modalidades de turismo, como o convencional, o ecológico, o esportivo e o comunitário, além de abrigar distintas atividades econômicas, com destaque para a pesca, a aquicultura e a pecuária. Entre alguns atrativos da região estão o bosque, a caixa d'água, a antiga estrada de ferro, o Pedral do Lourenço, o acesso à eclusa da usina e a antiga rodovia Transamazônica (IDEFLOR-Bio, 2025). Ao longo das margens do reservatório observam-se também formações de áreas de praia, como a Praia da Matinha e a Praia do Mangal. O cenário conta ainda com a Orla de Tucuruí, espaço urbano de lazer e convivência que reforça o papel do município na oferta de equipamentos turísticos e culturais (TUCURUÍ, 2025; PEÑA, 2024).

A criação do Lago de Tucuruí trouxe para a região uma relevância especial no que se refere à prática da pesca esportiva. Segundo Souza e Cañete (2015), o novo ecossistema, favorável à proliferação do tucunaré (um peixe agressivo e de difícil captura), fez com que essa espécie se tornasse um dos principais atrativos do

ecoturismo regional, tanto pela aventura da pesca quanto pela valorização do turismo sustentável.

A pesca esportiva, considerada segmento do turismo ecológico pela Lei nº 11.959/2009, consolidou-se como a principal atividade econômica nas margens do Lago de Tucuruí, sobretudo nas proximidades da barragem, que concentra grande parte do turismo local (SOUZA; CAÑETE, 2012). Essa prática é fortalecida por torneios como o TOPAM (Torneio de Pesca da Amazônia), que atraem pescadores de diferentes regiões do Brasil e do exterior, dinamizando o setor turístico e gerando emprego, renda e novas oportunidades (SANTOS *et al*., 2025). Nesse contexto, Souza e Cañete (2012) destacam a presença dos pescadores artesanais locais que, como detentores de profundo conhecimento da geografia da região, se beneficiam ao prestarem serviços de condução e orientação aos turistas, criando arranjos socioeconômicos ao integrar saberes tradicionais e favorecer a inclusão.

Nesse sentido, o surgimento do Lago de Tucuruí não apenas transformou a paisagem regional e possibilitou a criação de novas práticas turísticas, mas também exemplifica como determinados contextos históricos podem dar origem a destinos relevantes no cenário local. Contudo, a consolidação e a expansão do turismo em regiões como essa não dependem apenas de suas características naturais e históricas, mas também da adoção de recursos capazes de ampliar sua visibilidade e facilitar a experiência dos visitantes. É nesse ponto que se destacam as inovações tecnológicas que vêm desempenhando um papel central na forma como os indivíduos planejam, vivenciam e avaliam suas viagens na atualidade.

2.4 Aplicativos de Turismo

O avanço das Tecnologias da Informação e Comunicação (TICs) transformou profundamente a forma como a sociedade se comunica, consome informações e interage com o mundo. Se antes o acesso a dados e serviços dependia de computadores de mesa, hoje a popularização dos smartphones possibilitam que milhões de pessoas tenham o mundo na palma da mão. De acordo com Coutinho (2014), o número de usuários de smartphones no Brasil cresce de maneira significativa, deixando de se restringir a uma pequena parcela da população e integrando um processo de universalização e democratização.

Segundo Theoharidou, Mylonas e Gritzaldis (2013), o smartphone pode ser entendido como um dispositivo móvel avançado, dotado de hardware sofisticado e de

sistemas operacionais capazes de suportar uma ampla variedade de aplicativos, que vão desde funções utilitárias até opções voltadas ao entretenimento. Esses aparelhos ultrapassaram a função originalmente destinada às chamadas telefônicas e passaram a concentrar uma ampla gama de recursos.

As tecnologias móveis têm afetado significativamente a vida cotidiana das pessoas, modificando os modos de interação social, a forma de consumir bens e serviços e a maneira como o tempo e o espaço são vivenciados. Segundo Coutinho (2014), *"estes dispositivos funcionam como catalisadores de mudanças sociais"*. O autor acrescenta que tais transformações se tornaram hábitos, com a incorporação crescente de diferentes aplicativos e funcionalidades nas rotinas pessoais, repercutindo em áreas como medicina, educação, segurança e transporte. No turismo, observa-se a mesma tendência, caracterizada pela redefinição das formas de consumo das experiências de viagem, uma vez que as TICS passaram a estar fortemente presentes, desempenhando papel relevante em todas as etapas do processo: antes, durante e após a atividade turística (GRETZEL; FESENMAIER; O'LEARY, 2006).

Atualmente, existe uma ampla variedade de aplicativos móveis voltados ao suporte de turistas, como TripAdvisor, Booking e Airbnb, que oferecem desde opções de hospedagem até recomendações de restaurantes, roteiros de passeios e avaliações de outros usuários. De acordo com o estudo de Dias e Afonso (2021), os principais benefícios advindos do uso desses aplicativos são a ubiquidade, disponibilidade imediata, acesso à informação, conveniência, organização e apoio ao planejamento.

Wang, Park e Fesenmaier (2012) fazem uma reflexão acerca da imprevisibilidade inerente às viagens, um fato que dificulta com que os turistas antecipem todas as situações possíveis e elaborem planos totalmente eficazes. Diante disso, a ocorrência de eventos inesperados que podem comprometer a experiência, ou mesmo a mudança de roteiro por vontade própria dos turistas, exigem acesso imediato a informações que auxiliem na tomada de decisão. Nesse cenário, o suporte informacional em tempo real revela-se fundamental para atender às necessidades emergentes, melhorando a eficiência do deslocamento e ampliando o valor da experiência turística.

Os viajantes contemporâneos encontram-se fortalecidos pelo uso da Internet, que lhes garante acesso rápido e econômico a diversas fontes de informação e a

extensas comunidades virtuais (GRETZEL; FESENMAIER; O'LEARY, 2006). Esse processo reduz a dependência de intermediários tradicionais e amplia a autonomia do turista na elaboração de seus próprios roteiros de viagem, o que pode resultar em experiências mais personalizadas.

2.5 Desenvolvimento multiplataforma

O desenvolvimento multiplataforma (*cross-platform*) consiste em uma abordagem que permite criar aplicativos capazes de funcionar em diferentes sistemas operacionais a partir de um único código-fonte. Em contraste, o desenvolvimento nativo cria softwares voltados para uma plataforma específica, utilizando suas próprias linguagens, ferramentas e recursos (HEITKÖTTER *et al*., 2012).

Uma das principais vantagens da abordagem multiplataforma é a redução do tempo e do esforço necessários para o desenvolvimento, uma vez que os desenvolvedores escrevem o código do aplicativo apenas uma vez, sem necessidade de se criar versões distintas para cada sistema operacional. Esse fator também facilita a manutenção do código, pois todas as alterações podem ser aplicadas de forma centralizada em um único projeto (HEITKÖTTER *et al*., 2012; EL-KASSAS *et al*., 2015). No contexto do aplicativo de turismo da região do Lago de Tucuruí, essa estratégia é especialmente vantajosa, pois possibilita atingir um público mais amplo de usuários de diferentes dispositivos.

Apesar desses benefícios, a utilização do desenvolvimento multiplataforma apresenta certas limitações. El-Kassas *et al*. (2015) apontam restrições de desempenho em comparação a aplicativos nativos, dificuldades no acesso a recursos específicos de cada sistema operacional e a ausência de uma solução definitiva que atenda a todas as demandas do processo de criação móvel. Em complemento, Júnior (2015) ressalta os desafios relacionados à padronização da interface do usuário, dado que cada plataforma possui elementos próprios de design, além de dificuldades na implementação de novas funcionalidades, já que as ferramentas de terceiros nem sempre oferecem suporte imediato a tais atualizações.

2.6 React Native

O React Native é um framework de código aberto desenvolvido e mantido pela Meta, criado com o propósito de desenvolver aplicativos móveis nativos para os sistemas Android e iOS a partir de uma única base de código, usufruindo dos recursos da biblioteca React e a usando a linguagem JavaScript (META, 2025).

O JavaScript é uma linguagem de programação interpretada, amplamente utilizada no desenvolvimento de aplicações web e móveis, cuja principal característica é a capacidade de adicionar interatividade, dinamismo e controle lógico às interfaces digitais. De acordo com Flanagan (2020), o JavaScript permite manipular elementos da interface, responder a eventos do usuário e comunicar-se de forma assíncrona com servidores, sendo executado diretamente no ambiente do navegador ou em plataformas externas, como aplicações móveis. Dessa forma, a linguagem tornou-se essencial no desenvolvimento moderno de software, especialmente por sua flexibilidade, ampla adoção e integração com diferentes tecnologias e frameworks, como o React Native.

Conforme destacam Liu *et al*. (2023), o React Native possibilita a comunicação entre o código JavaScript e o código nativo por meio de um mecanismo de ponte (*bridge*), permitindo integrar funcionalidades nativas ao aplicativo sem comprometer sua performance. Dessa forma, o código escrito em JavaScript é traduzido em componentes nativos da plataforma, permitindo que a interface do usuário funcione como se tivesse sido construída com as ferramentas próprias do sistema.

A interface de um aplicativo no React Native é organizada em componentes, que funcionam como blocos reutilizáveis de código, responsáveis por exibir elementos visuais e controlar sua lógica. Essa estrutura possibilita segmentar a interface do usuário em partes menores e mais organizadas, permitindo que cada elemento seja desenvolvido e compreendido de forma isolada antes de ser integrado à aplicação (KOMPERLA *et al*., 2022).

Os componentes do framework se dividem em funcionais e de classe. Os primeiros são funções JavaScript simples que recebem propriedades (*props*) e retornam elementos de interface (JavaScript XML), sendo mais utilizados por sua leveza e praticidade. Já os componentes de classe são definidos como classes que estendem *React.Component*, possibilitando o uso de métodos do ciclo de vida e o controle de estado interno (*state*), o que os torna adequados para aplicações mais complexas e dinâmicas (KOMPERLA *et al*., 2022).

O React Native se destaca entre as tecnologias multiplataforma por oferecer desempenho muito próximo ao de aplicativos nativos, aliando eficiência, agilidade e baixo custo. Entre suas vantagens ressalta-se a reutilização de código, que possibilita o uso de uma mesma base para diferentes plataformas, reduzindo o tempo de desenvolvimento e facilitando manutenções. Além disso, sua documentação ampla e

bem estruturada, aliada a uma comunidade ativa de desenvolvedores, contribui para o aprimoramento contínuo da tecnologia e a resolução de problemas (BRITO *et al*., 2018). Komperla *et al*. (2022) acrescenta que a possibilidade de reciclar componentes é um dos principais benefícios desse framework, pois torna o processo de atualização das aplicações mais ágil e simplificado, contribuindo diretamente para o aumento da produtividade.

2.7 Expo

O Expo foi criado com o objetivo de facilitar a criação de aplicativos nativos para Android, iOS e Web, utilizando como base o React Native. Ele reúne os principais recursos do ambiente móvel e web em uma única plataforma unificada, oferecendo ferramentas, bibliotecas e serviços que dão suporte a todas as etapas do ciclo de criação de aplicativos, desde a inicialização do projeto até sua execução, testes e distribuição (EXPO, 2025).

Dentre suas funcionalidades, destacam-se módulos pré-configurados que permitem o acesso a componentes nativos, como câmera, localização e notificações, além de serviços que automatizam etapas de compilação e atualização. Há ainda interfaces que possibilitam testar e visualizar em tempo real o funcionamento dos *apps*, como ocorre com o Expo Go, aplicativo integrante do ecossistema Expo (EXPO, 2025).

O Expo mantém uma relação direta com o React Native, atuando como um framework complementar, otimizando o processo de criação de aplicativos. Embora o React Native possa ser utilizado de forma independente, sua própria documentação reconhece que a maioria dos desenvolvedores obtém melhores resultados ao utilizá-lo em conjunto com o Expo (REACT NATIVE, 2025). Essa diferença também é observada em estudos comparativos entre as duas abordagens. Conforme relata Saarikoski (2025) “Com um aplicativo React Native puro, uma quantidade significativa de tempo foi gasta na construção do ambiente de desenvolvimento e na correção de erros que ocorriam em diferentes etapas, enquanto com o framework Expo, a etapa de programação foi concluída em menos de meia hora” (tradução nossa).

Dessa forma, o ambiente fornecido pelo Expo simplifica a implementação de funcionalidades complexas, dando maior agilidade e praticidade ao processo, motivo pelo qual ele foi adotado neste trabalho.

2.8 Firebase

Um banco de dados é uma coleção organizada de registros ou informações mantidas em formato digital, projetada para que programas possam consultá-la por meio de consultas específicas, retornando dados úteis para a tomada de decisões (BERRINGTON, 2014).

Para o desenvolvimento de aplicativos, os bancos de dados são fundamentais. Nesse contexto, o Firebase destaca-se como uma plataforma de backend em nuvem voltada à criação de aplicações móveis e web. Adquirido pelo Google em 2014, o serviço foi aprimorado para oferecer infraestrutura escalável e sincronização de dados em tempo real, tornando-se uma solução que simplifica o gerenciamento do backend e acelera o processo de desenvolvimento (BERNARDINO; BARROS, 2025)

A plataforma disponibiliza SDKs (Software Development Kits), que atuam como intermediários entre o aplicativo e os serviços do Firebase, permitindo uma integração mais eficiente. Por meio desses kits, o desenvolvedor pode acessar as principais ferramentas da plataforma, como bancos de dados online, autenticação de usuários, armazenamento de arquivos e análise de desempenho (GOOGLE, 2025). Segundo Nguyen (2025), essa infraestrutura dispensa a necessidade de configurações complexas no lado do servidor.

Em outras palavras, o Firebase encurta o caminho, ampliando as possibilidades de criação de aplicações mais complexas e dinâmicas. A plataforma reduz a carga de configuração para o desenvolvedor, já que permite o aproveitamento direto de seus recursos sem a necessidade de criar um servidor intermediário. Essa abordagem otimiza o processo de desenvolvimento e possibilita que os profissionais concentrem seus esforços na melhoria da experiência do usuário e na entrega de soluções mais eficientes e de alto desempenho (NGUYEN,2025).

3 TRABALHOS RELACIONADOS

Nesta seção são apresentados trabalhos relacionados que compartilham princípios de funcionamento semelhantes ao projeto proposto, em especial aplicações móveis voltadas ao turismo, à divulgação de destinos e à organização de roteiros em ambientes urbanos e regionais. A análise dessas soluções contribui para evidenciar a relevância das abordagens adotadas neste trabalho e amplia a compreensão das potencialidades dos aplicativos móveis aplicados ao turismo.

3.1 Aplicativo Turistando Beribé

Um dos trabalhos que se aproxima diretamente da proposta deste projeto é o desenvolvido por Falcão (2022), intitulado *"Desenvolvimento do aplicativo Turistando Beberibe utilizando React Native"*. Nesse trabalho, o autor descreve o processo de concepção e implementação de um aplicativo móvel voltado à divulgação e contratação de experiências turísticas no município de Beberibe, no Ceará, tendo como foco principal a oferta de passeios e serviços turísticos para viajantes, utilizando React Native e a plataforma Expo em uma abordagem *cross-platform* para Android e iOS.

O Turistando Beberibe é estruturado como um catálogo de experiências, permitindo que viajantes visualizem serviços, filtrem opções, realizem reservas e interajam com prestadores, enquanto estes podem cadastrar, gerenciar e divulgar seus serviços dentro do próprio aplicativo. A solução utiliza Firebase para autenticação, armazenamento de dados e gerenciamento de conteúdo multimídia, além de explorar princípios de Design Thinking e metodologias ágeis no processo de desenvolvimento. O trabalho também apresenta uma avaliação de usabilidade baseada no método System Usability Scale (SUS), permitindo analisar a aceitação do aplicativo pelos usuários finais.

O trabalho de Falcão (2022) foi utilizado como referência tanto tecnológica quanto metodológica. Do ponto de vista tecnológico, a experiência relatada com React Native, Expo e Firebase contribuiu para reforçar a escolha dessas ferramentas no desenvolvimento do LagoTur, bem como para compreender vantagens e limitações da abordagem *cross-platform* na construção de aplicativos turísticos. A forma como o autor descreve o processo de concepção do aplicativo, organização em módulos, apresentação de casos de uso e realização de testes de usabilidade serviu como base para estruturar partes deste trabalho. Apesar dessas aproximações, o LagoTur

diferencia-se ao focar a região do Lago de Tucuruí, trabalhar com múltiplos municípios e priorizar a centralização de informações sobre estabelecimentos, eventos, roteiros turísticos e serviços regionais, em vez de se concentrar apenas na contratação de experiências em uma única cidade.

3.2 Desbrava: aplicativo de incentivo ao turismo na paraíba

Um dos trabalhos que se relacionam com a proposta deste projeto é o desenvolvido por Maciel (2019), intitulado *"Desbrava: interface de aplicativo de incentivo ao turismo de experiência na Paraíba"*. Nesse estudo, o autor desenvolve a interface de um aplicativo voltado ao turismo de experiência no estado da Paraíba, utilizando fundamentos de UX/UI Design (experiência do usuário e interface do usuário), arquitetura da informação e design de interação para conceber uma plataforma na qual o usuário pode cadastrar e utilizar diferentes segmentos de experiências turísticas, explorando atrativos menos conhecidos e valorizando aspectos culturais e vivenciais do território.

Ao longo desse processo, Maciel (2019) enfatiza a importância da construção de interfaces consistentes, centradas no usuário e guiadas por princípios de usabilidade, bem como a necessidade de articular identidade visual, navegação e conteúdo para compor uma experiência coerente e envolvente.

O Desbrava foi utilizado principalmente como referência conceitual e metodológica para o planejamento da experiência do usuário no LagoTur. As abordagens de pesquisa com usuários, a organização do fluxo de navegação e a ênfase em UX/UI serviram de base para estruturar os fluxos e os padrões de interface do aplicativo LagoTur. Apesar dessa aproximação, o presente projeto diferencia-se por ter foco na região do Lago de Tucuruí, trabalhar com múltiplos municípios e enfatizar não apenas o turismo de experiência, mas também a centralização de informações sobre estabelecimentos, eventos, roteiros turísticos referentes a várias cidades da região.

3.3 Jampa Rotas: compartilhamento de roteiros turísticos em João Pessoa

Um trabalho que também se aproxima da proposta deste projeto é o desenvolvido por Anjos (2022), intitulado *"Jampa Rotas: um protótipo de um aplicativo de compartilhamento de roteiros turísticos da cidade de João Pessoa"*. Nesse estudo, a autora propõe um aplicativo cujo foco é permitir que usuários planejem, adaptem e compartilhem roteiros turísticos de João Pessoa, com o objetivo de ampliar o turismo

local e divulgar tanto percursos convencionais quanto não convencionais. Esse protótipo é estruturado com base em conceitos de design de serviços, turismo e experiência do usuário, enfatizando a importância da tecnologia móvel como mediadora da relação entre visitantes e destino turístico.

Do ponto de vista metodológico, o trabalho de Anjos (2022) adota o Design de Serviço e o Design Centrado no Usuário, seguindo etapas de pesquisa/exploração, ideação, prototipação e avaliação. Entre as técnicas empregadas destacam-se a análise de aplicativos similares, construção de cenários de uso, desenvolvimento de *wireframes*, criação de um *design system* e produção de um protótipo de alta fidelidade, posteriormente testado com usuários para verificar aspectos de usabilidade, navegação e clareza das informações.

O Jampa Rotas foi utilizado principalmente como referência conceitual e de processo de projeto levando em conta as estratégias de pesquisa com usuários em fase de prototipação. Entretanto, o presente trabalho diferencia-se por ter como recorte a região do Lago de Tucuruí e a capacidade de contemplar não só roteiros turísticos, mas também outras informações relevantes a respeito da região estudada.

4 METODOLOGIA

4.1 Concepção e planejamento do aplicativo LagoTur

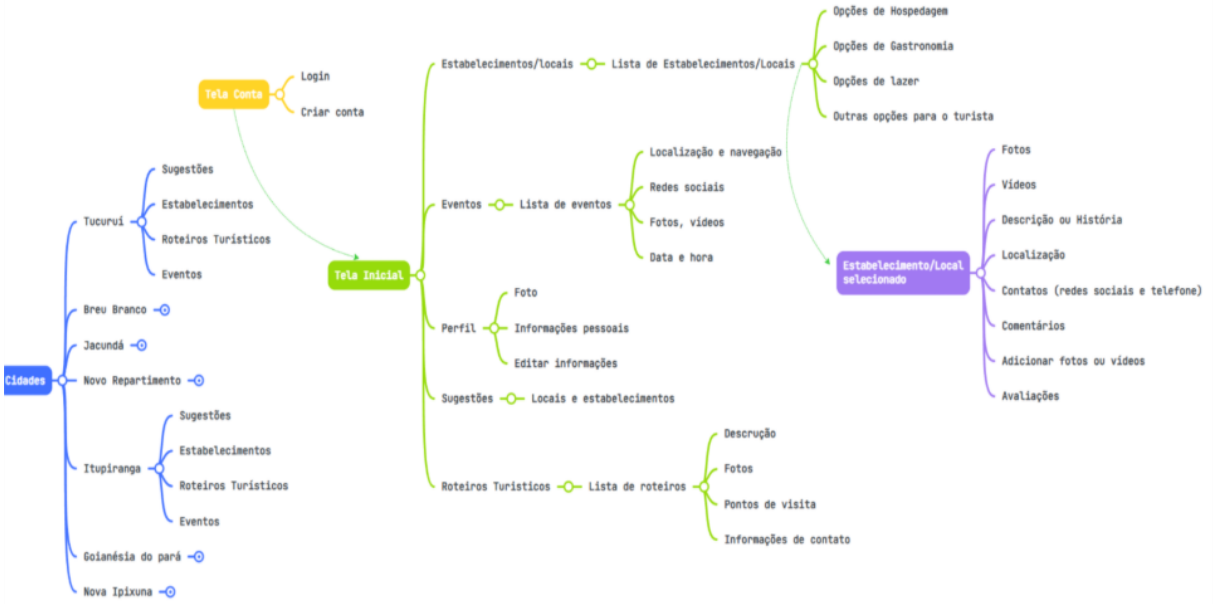
A concepção do aplicativo surgiu a partir de uma demanda apresentada pela IGR LagoTur, que buscava uma solução tecnológica capaz de apoiar o turismo na região do Lago de Tucuruí, fortalecendo o comércio local e facilitando o acesso a informações relevantes para visitantes e moradores. Ao longo da fase inicial do projeto foram realizadas diversas reuniões com a diretoria dessa organização, nas quais foram discutidas as principais dificuldades enfrentadas pelos turistas que visitam a localidade, a partir disso tornou-se possível identificar necessidades recorrentes e estruturar as primeiras ideias para a aplicação definindo quais funcionalidades mais iriam agregar o sistema.

Com base nesse diagnóstico conjunto, foram planejadas as funcionalidades do sistema, bem como o modo como a aplicação deveria se portar do ponto de vista de usabilidade e organização das informações. Dessa forma definiu-se que a aplicação abordaria a região de forma estruturada, separando os conteúdos por cidades e somente a partir da seleção de uma cidade, mais informações detalhadas seriam apresentadas.

Para estruturar as ideias iniciais do aplicativo LagoTur, definir o comportamento da interface, registrar anotações, planejar as funcionalidades e interações do usuário com a aplicação utilizou-se a ferramenta online de mapas mentais MindMeister (Figura 2). Por meio dela foi possível transformar uma concepção ainda abstrata em um modelo visual organizado onde os módulos, as informações e os recursos desejados eram representados de forma hierárquica e interligada.

Esse mapeamento permitiu detalhar quais ações o usuário poderia realizar em cada parte do sistema, quais dados deveriam ser exibidos em cada tela e como se dariam os fluxos de navegação entre as diferentes partes do sistema. O uso dessa ferramenta facilitou o registro contínuo de novas ideias, o ajuste de requisitos ao longo das discussões com a IGR LagoTur e a comunicação da estrutura proposta para outras pessoas envolvidas no projeto, servindo como base conceitual para as etapas posteriores de modelagem da arquitetura e implementação do aplicativo.

Figura 2 - Exemplo de fluxo de planejamento da aplicação no MindMeister.



Fonte: Elaborado pelo autor (2025).

Diante desse cenário, optou-se por desenvolver o aplicativo utilizando o framework React Native, uma vez que se trata de uma tecnologia multiplataforma, capaz de atender simultaneamente usuários de dispositivos Android e iOS, o que se mostrou um requisito essencial para a proposta do projeto, visto que os turistas podem ter celulares que possuem diferentes sistemas operacionais. Essa ferramenta apresenta alta flexibilidade de desenvolvimento, ampla documentação oficial e um ecossistema de bibliotecas modernas e constantemente atualizadas, o que facilita a integração de recursos adicionais e a manutenção da aplicação ao longo do tempo.

Dessa maneira, a etapa inicial de concepção e planejamento do LagoTur consolidou-se como um momento estratégico para alinhar as necessidades reais do turismo na região do Lago de Tucuruí com as soluções tecnológicas propostas. As reuniões com a IGR LagoTur, a definição de funcionalidades, a organização das informações por cidades e o uso sistemático de mapas mentais no MindMeister permitiram transformar demandas abstratas em requisitos claros, fluxos de navegação bem definidos e uma estrutura de conteúdo coerente, esse conjunto de decisões forneceu uma base sólida para as fases seguintes de modelagem da arquitetura, implementação e validação do aplicativo, orientando o desenvolvimento de forma incremental e com objetivo de garantir que o sistema permanecesse alinhado ao objetivo central de apoiar o turista e fortalecer o comércio local.

4.2 Estrutura funcional e apresentação das informações no LagoTur

Com base no levantamento de requisitos, definiu-se que o aplicativo passaria a apresentar as informações de forma contextualizada à cidade selecionada pelo usuário. Para organizar esses dados bem como permitir a interação de maneira clara e coerente, foram estabelecidos quarto funcionalidades principais, responsáveis por estruturar e disponibilizar as informações turísticas, sendo eles:

- Estabelecimentos: Reúne uma lista estruturada de locais de visita, lazer, gastronomia e demais pontos de interesse da cidade selecionada, apresentando ao usuário informações iniciais desses pontos bem como a opção de poder estar selecionando um desses itens para acessar mais informações.
- Eventos: Concentra uma lista com as programações pontuais da localidade, incluindo festividades, festivais, feiras e outras atividades culturais ou sazonais, permitindo ao turista visualizar informações detalhadas ao selecionar um desses itens.
- Roteiros Turísticos: Disponibiliza uma lista de roteiros que o usuário pode realizar na região, onde cada rota apresenta uma breve descrição do percurso, uma imagem representativa e um resumo introdutório da proposta daquele passeio, o mesmo poderá ser selecionado para ceder informações detalhadas.
- Sugestões: Agrega uma lista que deverá aparecer para o usuário de forma inicial ao acessar os dados da cidade selecionada, voltadas à indicação de locais, estabelecimentos, pontos de visita e opções de lazer ao turista, apresentando listas de conteúdos recomendados de modo a ampliar o repertório de possibilidades de exploração da região.

Cada um dos itens mencionados acima trata-se de um agrupamento lógico de funcionalidades e dados que tratam de um mesmo assunto no sistema, dessa forma tratando-se de módulos funcionais compostos por múltiplos componentes de interface e lógica, mas que do ponto de vista do usuário, se apresentam como áreas coerentes de navegação e uso.

Além dos quatro módulos principais voltados à apresentação de conteúdo, foi definido um módulo específico para o gerenciamento de dados do turista e de suas interações com o sistema, sendo esse o módulo de perfil de usuário, responsável por

armazenar informações pessoais básicas, preferências e demais dados associados ao uso da aplicação.

Após a fase de modelagem conceitual, definiu-se que a estrutura inicial de funcionamento do aplicativo LagoTur seria baseada na apresentação de listas de informações organizadas de acordo com Estabelecimentos, Eventos, Roteiros Turísticos e Sugestões, tendo como objetivo facilitar a navegação e a descoberta de conteúdos pelo usuário, visando dessa forma evitar a exibição de dados de maneira desorganizada.

Estabeleceu-se que cada item dessas listas seria apresentados por meio de *cards* interativos (Figura 3), de acordo com a documentação oficial do Android Developers o componente *Card* tem conteúdo e ações sobre um único assunto (GOOGLE, 2025), tratando-se de uma superfície visual que agrupa informações e ações relacionadas, favorecendo uma leitura rápida e uma hierarquia clara do conteúdo. Esse conceito foi adotado para encapsular informações, a aplicação possuirá uma lista de cards com informações da localidade, visando contribuir para uma apresentação objetiva, hierarquizada e de fácil compreensão.

No desenvolvimento da aplicação, os cards foram concebidos como componentes reutilizáveis, estruturados a partir de contêineres visuais (*View*), elementos de mídia (*Image*) e textos (*Text*), todos envolvidos por componentes de interação como o *TouchableOpacity*, responsável por atribuir comportamento ao item quando selecionado pelo usuário, fazendo com que cada um dos itens das listas da aplicação funcione como uma unidade de informação e elemento interativo simultaneamente, que ao ser acionado direciona o usuário para uma tela específica contendo os detalhes correspondente a seleção. Essa abordagem favorece a padronização visual e a manutenção do código, uma vez que o mesmo modelo de card pode ser reutilizado em diferentes listas e módulos do sistema, reduzindo redundâncias e facilitando a evolução incremental da aplicação.

Figura 3 - Exemplo de card interativo utilizado para apresentação de informações.

Fonte: Elaborado pelo autor (2025).

No LagoTur, os cards foram definidos para exibir de forma compacta, informações preliminares de cada elemento, tais como: nome do estabelecimento ou evento, imagem principal representativa, descrição breve, categoria a que pertence e a distância aproximada em relação ao usuário.

Essa padronização da estrutura em listas de cards visou contribuir diretamente para a coerência e a consistência da experiência de uso, uma vez que o mesmo formato de visualização e interação foi implementado em diferentes partes do aplicativo, padronizando o funcionamento da interface.

4.3 Telas de detalhes e organização das informações

A estrutura de apresentação das informação dos itens cadastrados na aplicação foi planejada para que os dados sejam apresentados ao usuário de forma organizada de acordo com a exibição das listas de Estabelecimentos, Sugestões, Roteiros Turísticos e Eventos, bem como o conteúdo disponibilizado quando o utilizador acessa cada um desses módulos. A definição desses campos e de sua disposição na interface foi realizada em conjunto com a IGR LagoTur, de modo a garantir que as informações apresentadas fossem relevantes, consistentes e alinhadas às necessidades dos turistas e aos objetivos do projeto.

- Estabelecimentos: Ao acessar os detalhes de um item dessa lista, o usuário é direcionado para uma tela que apresenta um conjunto ampliado de informações, incluindo nome do estabelecimento, imagens ilustrativas, vídeos, localização geográfica, dados de contato, breve descrição, além de informações deixadas por outros usuários. Esses atributos foram definidos com

o objetivo de fornecer ao turista subsídios suficientes para apoiar a tomada de decisão sobre onde ir, quais serviços utilizar e como chegar ao local, ao mesmo tempo em que tem por objetivo contribuir para aumentar a visibilidade dos empreendimentos cadastrados e fortalecer o comércio e os serviços da região.

- Eventos: Ao selecionar um evento específico, o usuário acessa uma tela detalhada que apresenta descrição completa, fotos, vídeos, localização no mapa e informações de contato para obtenção de mais detalhes. Esse conjunto de dados foi definido com o objetivo de facilitar a descoberta de atividades em andamento ou futuras, permitindo que turistas e moradores planejem melhor sua participação em eventos da região e se mantenham informados sobre a agenda local.
- Roteiros Turísticos: Ao selecionar um roteiro, o usuário é encaminhado para uma tela detalhada que descreve, de forma mais completa a experiência oferecida, incluindo uma introdução descritiva, informações sobre o ponto de partida, os principais pontos de visita que compõem a rota turística e dados de contato com a IGR LagoTur para esclarecimento de dúvidas adicionais. Esses roteiros foram planejados com o intuito de consolidar informações turísticas relevantes em percursos coerentes, proporcionando uma experiência guiada e contextualizada, que valoriza os atrativos locais e facilita o planejamento do tempo de visita.
- Sugestões: São apresentados ao usuário itens recomendados com base em sua relevância turística seguindo o mesmo padrão utilizado para o módulo de Estabelecimentos, porém com foco em destacar conteúdos considerados prioritários, como pontos de grande fluxo de visitantes, atrativos emblemáticos ou opções indicadas para pessoas que estão conhecendo a região pela primeira vez. Ao selecionar um item da lista de Sugestões, o usuário é direcionado para uma tela detalhada que reutiliza o mesmo layout de Estabelecimentos, exibindo o conjunto completo de informações planejadas para aquele local.

De forma geral as telas de detalhes disponibilizam informações mais completas e contextualizadas, essa organização tem como objetivo contribuir para uma navegação mais intuitiva, reduzindo a sobrecarga cognitiva do usuário e reforçando o propósito central do LagoTur de apoiar a experiência turística de maneira clara, acessível e alinhada à realidade da região do Lago de Tucuruí.

4.4 Perfil do Usuário e Interação com o Sistema

O módulo de Perfil do Usuário foi concebido como o principal ponto de gerenciamento de dados do turista e de suas interações com o sistema, sua função central é estabelecer uma camada de personalização e identificação individual, permitindo que cada usuário possua um espaço próprio dentro da aplicação, armazenando elementos pessoais básicos associados ao uso da aplicação, constituindo a base para recursos de interação e para o acompanhamento contínuo do comportamento de uso da plataforma.

Definiu-se que por meio do perfil de usuário o turista poderá registrar avaliações sobre estabelecimentos e inserir comentários relacionados à sua experiência. Essas interações serão armazenadas de forma estruturada, permitindo a consolidação de um histórico individual de uso que poderá ser utilizado tanto para o aprimoramento contínuo da experiência oferecida ao longo do tempo quanto para servir de referência a outros utilizadores, tornando o ambiente mais participativo e interativo.

Do ponto de vista da experiência do usuário, ter um perfil individual contribui diretamente para o aumento do engajamento e para a sensação de pertencimento em relação ao aplicativo. A existência de um espaço próprio para gerenciamento de dados favorece interações mais estruturadas com os conteúdos apresentados, além de abrir espaço para futuras funcionalidades de personalização de acordo com o conteúdo consumido.

Dessa forma, o módulo de Perfil do Usuário não se limita a uma funcionalidade acessória, mas configura-se como uma fundação para a expansão de recursos sociais e de personalização no LagoTur, viabilizando a construção de uma relação mais próxima entre o turista e a aplicação, ao mesmo tempo em que prepara o sistema para futuras extensões e outras iniciativas que possam enriquecer a experiência turística na região do Lago de Tucuruí.

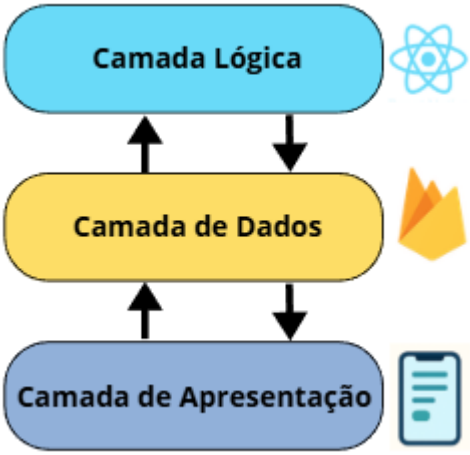
4.5 Princípios de Engenharia de Software e Arquitetura em Camadas

A ideia principal do sistema foi guiada por princípios de engenharia de software modular, priorizando a escalabilidade, facilidade na manutenção e a reutilização de código, para isso foi adotado o framework React Native com Expo.

O modelo de organização segue o princípio de arquitetura em camadas, no qual a aplicação é dividida em níveis de responsabilidade distintos, a fim de reduzir a

dependência entre diferentes módulos do aplicativo e melhorar a manutenção do sistema. Conforme Fowler (2002), a separação por camadas permite que cada parte do sistema execute seu papel de forma independente, garantindo clareza estrutural e facilitando a evolução do código ao longo do tempo. Dessa forma, a lógica do aplicativo mantém-se organizada em três camadas principais demonstado na Figura 4, sendo elas a Camada de Lógica (*Logic Layer*), a Camada de Dados (*Data Layer*) e a Camada de Apresentação (*UI Layer*).

Figura 4 - Arquitetura em camadas da aplicação.



Fonte: Elaborado pelo autor (2025).

4.6 Camada de lógica

A camada de lógica é o local onde são aplicadas as regras de negócio, transformações, validações e cálculos sobre os dados. É nesta camada que se encontram as classes-modelo da aplicação, responsáveis por representar entidades e encapsular comportamentos específicos. De acordo com Larman (2000) a camada de lógica de negócio concentra as regras e políticas que governam o domínio do sistema, garantindo que o comportamento da aplicação não esteja misturado com a interface ou com o armazenamento de dados. Essa separação melhora a clareza, a reutilização e a manutenção do software.

4.6.1 Princípios de modularidade, coesão e acoplamento na aplicação

A aplicação é organizada em unidades independentes denominadas componentes, que combinam estrutura visual e lógica de funcionamento em blocos coesos e reutilizáveis. Segundo Sommerville (2016), dividir um software em unidades menores com responsabilidades claramente definidas contribui para a clareza

estrutural, facilita a manutenção e favorece a evolução do sistema ao longo do tempo. Em consonância com esse princípio, cada componente do aplicativo foi projetado para representar uma função específica.

Desde as etapas iniciais do projeto buscou-se adotar boas práticas de engenharia de software, para isso, a estrutura do sistema foi planejada de forma a assegurar uma separação clara entre as camadas de lógica, interface e dados, objetivando promover organização e facilidade de manutenção.

Segundo Banks e Porcello (2020), a filosofia do React baseia-se na ideia de que interfaces complexas podem ser construídas a partir de componentes pequenos, reutilizáveis e isolados, que podem ser combinados para formar estruturas maiores, mantendo a previsibilidade do software. Sabendo disso, arquitetura do aplicativo adota o princípio de componentização, no qual o sistema é estruturado a partir de pequenas unidades independentes de código, chamadas componentes, sendo cada um desenvolvido com responsabilidades definidas, seguindo uma abordagem modular dos princípios de baixo acoplamento e alta coesão.

De acordo com Pressman (2011), a modularidade eficaz em um software, isto é, a existência de módulos independentes, facilita o desenvolvimento, manutenção e teste do sistema, uma vez que as funções podem ser compartimentalizadas e as interfaces simplificadas. Essa independência entre módulos do sistema reduz a propagação de erros e possibilita a criação de componentes reutilizáveis, visando manter a qualidade do software.

Pressman (2011) também ressalta que a independência entre módulos pode ser analisada por meio de dois critérios qualitativos fundamentais: coesão e acoplamento. A coesão refere-se ao grau de integração e clareza funcional existente dentro de um módulo, enquanto o acoplamento está relacionado ao nível de dependência entre os diferentes módulos do sistema. De acordo com Zimmer (2020), ao se projetar uma arquitetura com alta coesão e baixo acoplamento, obtém-se um design mais modular, compreensível e de fácil manutenção, características que favorecem a simplicidade, elegância e eficiência no desenvolvimento de software.

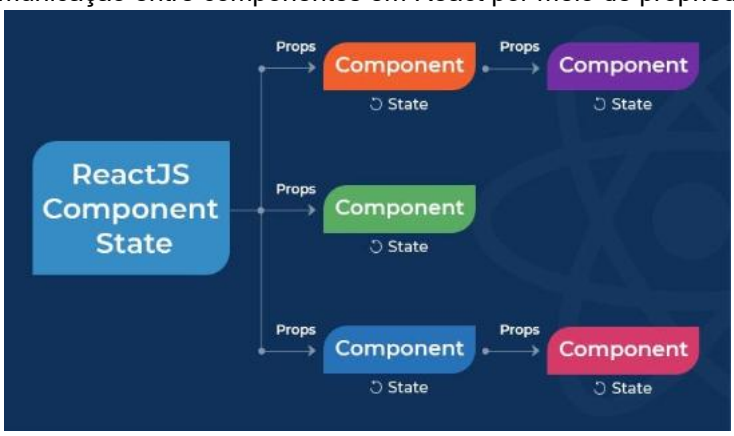
Essa abordagem modular foi implementada priorizando a independência entre os componentes, visando garantir que cada módulo atue de forma autônoma, sem dependências diretas desnecessárias. Essa estratégia tem como objetivo contribuir para uma estrutura mais clara e compreensível, reduzindo o

acoplamento entre as partes do sistema e favorecendo tanto a clareza estrutural quanto a facilidade de manutenção, expansão e atualização futura do código.

A comunicação entre componentes deve ser realizada de forma unidirecional, por meio de *props*, garantindo acoplamento fraco, alta coesão e previsibilidade no fluxo de dados, como aponta Fowler (2023). A abordagem do React Native reforça esse modelo ao destacar que a estrutura baseada em 'component-tree' promove modularização e manutenibilidade (META, 2024).

Stefanov (2016), descreve as *props* como valores transmitidos de um componente de nível superior (componente-pai) para componentes de nível inferior (componentes-filho), funciona como parâmetros que determinam o comportamento ou o conteúdo exibido por esses componentes. A documentação oficial do React destaca que o uso de *props* contribui para o princípio de baixa acoplagem entre componentes, pois cada unidade do sistema recebe apenas os dados essenciais para executar sua tarefa (META, 2023). Essa modularidade favorece a escalabilidade do projeto, permitindo que componentes complexos sejam formados pela composição de componentes menores, sem que haja dependência direta entre suas lógicas internas, demonstrado na Figura 5.

Figura 5 - Comunicação entre componentes em React por meio de propriedades (props).



Fonte: SAHOO (2023).

Essa abordagem oferece diversos benefícios do ponto de vista da engenharia de software, pois reduz a dependência entre as partes do sistema (módulos, componentes e classes) e promove a reutilização de componentes, uma vez que o mesmo elemento pode ser empregado em diferentes contextos do aplicativo apenas por meio do ajuste das props que recebe, objetivando uma maior flexibilidade ao desenvolvimento. Já a independência funcional permite que alterações em um módulo

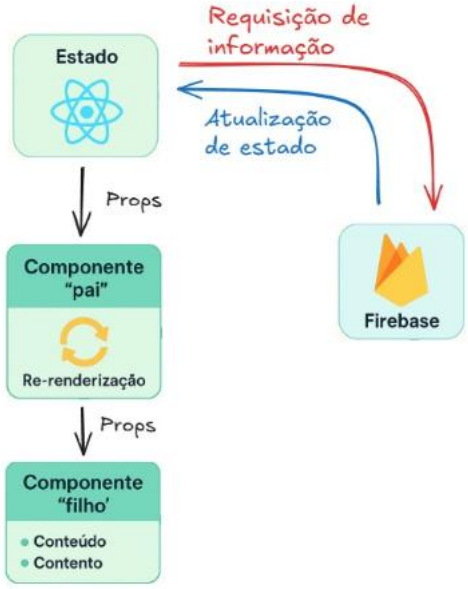
não interfiram no funcionamento dos demais, tornando o processo de manutenção mais seguro e eficiente.

Quando um componente-filho precisa acionar uma atualização no componente-pai, utiliza-se o envio de funções de *callback* através de *props*. Esse mecanismo mantém o fluxo de dados unidirecional, permitindo que o componente-pai permaneça responsável pelo estado enquanto o componente-filho apenas solicita a mudança. Segundo a definição da MDN Web Docs (2023), *"uma função callback é uma função passada a outra função como argumento, que é então invocado dentro da função externa para completar algum tipo de rotina ou ação."*. Dessa forma o componente-pai fornece a função e o componente-filho a executa quando o evento ocorre, preservando a organização e a coerência da comunicação entre módulos.

Essa arquitetura de comunicação é complementada por um gerenciamento eficiente de estado, no qual variáveis reativas são utilizadas para armazenar informações dinâmicas, como listas de eventos, status de login do usuário e conteúdos vindos do banco de dados Firebase, sempre que o estado é atualizado o componente ativo re-renderiza automaticamente os dados, refletindo as mudanças de forma instantânea na interface.

No caso específico deste projeto, as *props* foram amplamente utilizadas para transmitir dados provenientes do banco de dados Firebase, como listas de eventos, estabelecimentos e roteiros turísticos, até os componentes responsáveis por renderizar essas informações na interface, visando que o sistema mantenha um fluxo de dados organizado, coerente e eficiente, garantindo que o usuário visualize sempre as informações atualizadas e integradas em tempo real, conforme as alterações ocorrem no banco de dados.

Figura 6 - Fluxo de dados entre Firebase e componentes via Props.



Fonte: Elaborado pelo autor (2025).

A Figura 6 apresenta o funcionamento do fluxo de dados, destacando a relação entre o banco de dados, o gerenciamento de estado e o repasse de informações entre componentes por meio de *props*. Inicialmente, os dados são armazenados e mantidos no Firebase, que atua como fonte central de informações da aplicação.

Quando o aplicativo realiza uma requisição ao Firebase, os dados retornados são utilizados para atualizar variáveis de estado internas (*states*), permitindo que o sistema mantenha o controle sobre informações dinâmicas, como conteúdos exibidos na interface, status e listas atualizadas. Após a atualização do estado o sistema automaticamente identifica quais componentes dependem dessas informações e realiza a re-renderização apenas das partes necessárias da interface, visando garantir eficiência no processamento.

Para evitar repetição de código e facilitar a organização da aplicação, o sistema faz uso de *hooks* personalizados. De acordo com a documentação oficial do React, hooks permitem reutilizar lógica com estado de forma simples e padronizada, evitando que o mesmo código precise ser reescrito várias vezes (META, 2023). Banks e Porcello (2020) explicam que essa prática melhora a manutenção e o crescimento do sistema, pois mantém a lógica separada da interface, tornando o código mais claro, fácil de ajustar e de expandir conforme o projeto evolui.

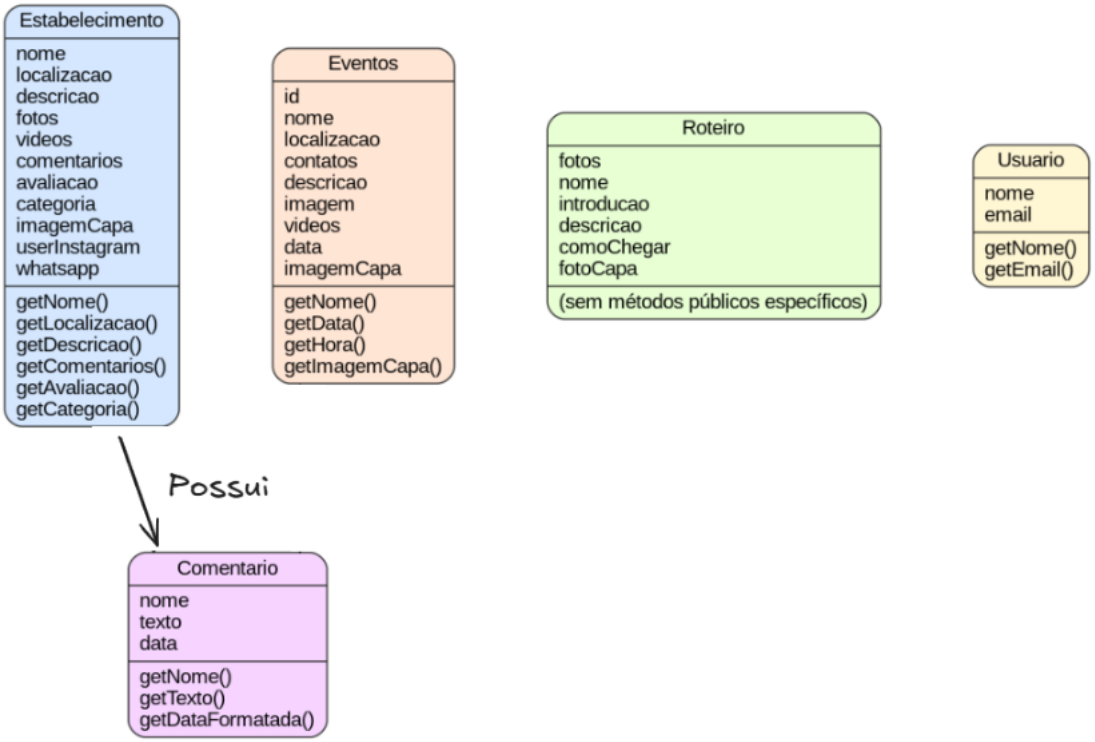
4.6.2 Classes de domínio e organização da camada lógica

Além dos componentes voltados para a interface, o aplicativo também utiliza classes auxiliares que atuam como modelos de dados, responsáveis por representar e organizar informações referentes aos módulos do software centralizando atributos e métodos específicos de cada entidade, realizando operações, conversão de datas, estruturação de listas e padronização de formatos antes que os dados sejam exibidos na interface. De acordo com Abbot *et al*. (2019), a utilização de modelos de dados em aplicações React Native promove uma arquitetura mais clara e escalável, pois ao mover a lógica de manipulação e transformação de dados para classes dedicadas, os componentes tornam-se mais simples e fáceis de compreender, resultando em uma arquitetura clara, modular e menos propensa a erros.

Essas classes funcionam como uma camada intermediária de serviço, posicionada entre a lógica de interface (UI) e a camada de persistência de dados (Firebase), atuando como responsáveis pela estruturação, transformação e padronização das informações manipuladas pelo aplicativo.

Na aplicação as classes foram implementadas com o objetivo de representar entidades do domínio turístico, encapsulando atributos e comportamentos específicos de cada uma delas, centralizando funções como cálculo de média de avaliações, formatação de datas, organização de listas de comentários e padronização de dados provenientes do Firebase, visando garantir que os componentes de interface permaneçam focados apenas na apresentação visual e na interação com o usuário. Esse modelo de abordagem reduz o acoplamento entre camadas, facilita a manutenção e melhora a escalabilidade do sistema, uma vez que a lógica de negócio não é distribuída de forma desordenada pelos componentes da aplicação.

Figura 7 - Esquema de Classes Utilizadas no Aplicativo LagoTur.



Fonte: Elaborado pelo autor (2025).

Dessa forma, o diagrama da Figura 7 evidencia a estrutura da camada de lógica do aplicativo, na qual cada classe representa uma entidade do domínio turístico, organizada de modo a concentrar atributos e comportamentos específicos. Observa-se que a classe *Estabelecimento* mantém relação direta com a classe *Comentario*, refletindo a possibilidade de um estabelecimento possuir múltiplos comentários realizados pelos usuários. As demais classes (Eventos, Roteiro e Usuario) são tratadas como modelos independentes que estruturam e preparam os dados antes de sua utilização pelos componentes.

4.6.3 Estrutura lógica de navegação e organização dos componentes

Desde as etapas iniciais do desenvolvimento, estabeleceu-se como objetivo a criação de uma estrutura de navegação que fosse intuitiva, coerente e funcionalmente segmentada, assegurando que cada tela desempenhasse um papel claramente definido e operasse de forma independente. Buscou-se, ao mesmo tempo, proporcionar transições fluidas entre as funcionalidades do sistema, preservando um controle centralizado e consistente da navegação interna do aplicativo.

Para a definição dessa navegação, adotou-se como principal referência a documentação de arquitetura de compontente demonstrada em Hands on React, a

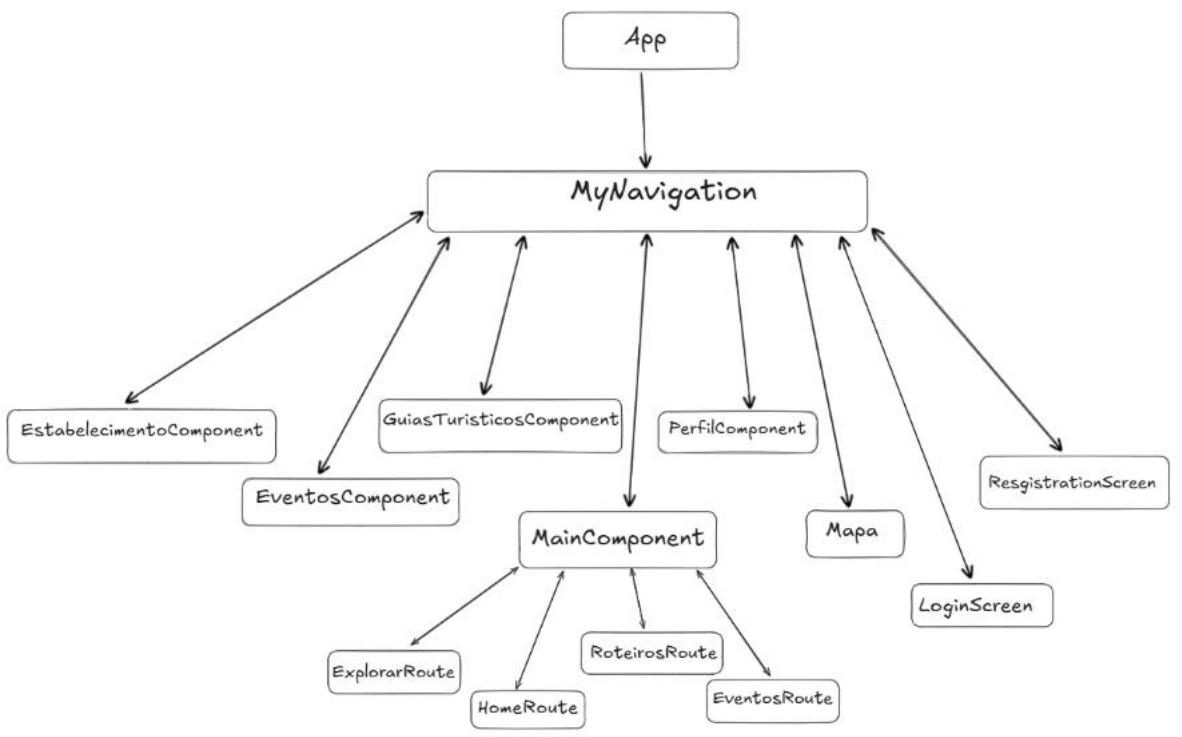
qual apresenta diretrizes, padrões de projeto e exemplos de implementação voltados à construção de fluxos de navegação eficientes em aplicações móveis. Esse material foi fundamental para orientar decisões estruturais, oferecendo modelos consolidados visando favorecer a construção de uma navegação fluida, previsível e tecnicamente alinhada às práticas recomendadas. Também tem como objetivo contribuir para ampliar a flexibilidade e a escalabilidade do sistema, facilitando a incorporação de novas funcionalidades ao longo do desenvolvimento.

Na arquitetura desse projeto, criou-se um componente central chamado de *MainComponent ,* atuando como gerenciador principal, sendo responsável por coordenar o fluxo de navegação, selecionar dinamicamente qual tela deve ser exibida e garantir que o usuário visualize exatamente o que correspondente à sua ação.

A estrutura lógica de navegação foi organizada de forma a garantir clareza, previsibilidade e controle centralizado sobre as telas exibidas ao usuário, o componente *MyNavigation* atua como ponto de entrada do sistema de navegação, sendo responsável por direcionar o usuário às telas apropriadas com base em seu estado atual (como estar logado ou não) e nas ações realizadas, também determina qual interface deve ser apresentada e encaminha o fluxo de navegação de maneira estruturada e sequencial.

O componente *MainComponent* funciona como o módulo de gerenciamento interno da aplicação utilizando a função de navegação recebida via *props* do *MyNavigation* para solicitar a abertura de uma determinada tela (decidida pelo usuário), acionada essa função o *MyNavigation*, que possui todas as rotas declaradas em sua estrutura identifica a tela correspondente e executa a navegação, exibindo ao utilizador a interface solicitada, centralizando o controle das telas principais do aplicativo, permitindo que o mesmo navegue de forma contínua entre os módulos internos sem a necessidade de recarregar o fluxo global de navegação.

Figura 8 - Diagrama da estrutura lógica de navegação do aplicativo LagoTur.

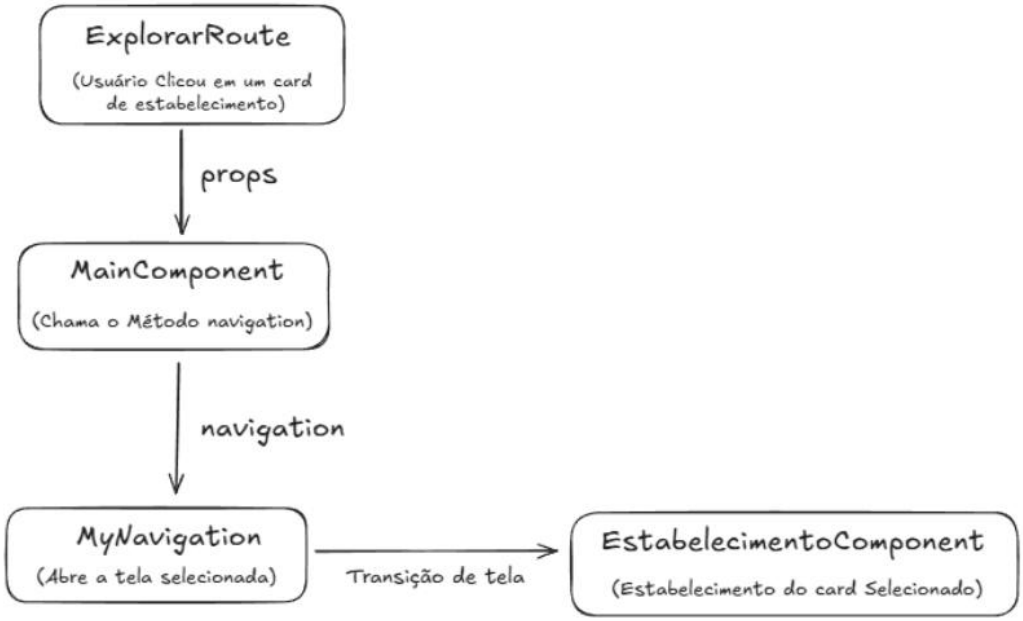


Fonte: Elaborado pelo autor (2025).

Na Figura 8, os fluxos de interação são representados por setas com duas pontas para indicar que existe a possibilidade de navegação bidirecional entre determinadas telas, ou seja, o usuário pode avançar para novas interfaces e quando necessário, retornar à tela anterior. Esse comportamento reflete o funcionamento do padrão *Stack Navigation* que segundo a documentação do React Navigation, “o Stack Navigator organiza as telas em uma estrutura de pilha, onde cada nova navegação empilha uma tela acima da anterior, permitindo a navegação reversa por meio da remoção dessa camada” (META, 2023). Esse padrão está diretamente aplicado ao sistema uma vez que telas secundárias (como *EstabelecimentoComponent* por exemplo*)* são acessadas sobre a pilha principal e podem ser removidas quando o usuário retorna.

Pode-se ter como exemplo de fluxo interno do aplicativo ao clicar em um card presente no componente *ExplorarRoute* (Figura 9) acionando o método de navegação disponibilizado pelo *MainComponent* . Essa solicitação utiliza a função de navegação enviada para ele via *props* pelo *MyNavigation* e solicita a abertura da tela correspondente. O *MyNavigation*, que contém todas as rotas declaradas em sua estrutura, identifica a rota do *EstabelecimentoComponent* e realiza a transição de tela, o fluxo em questão está representado na imagem abaixo:

Figura 9 - Fluxo de Transição de Telas Gerenciado pelo Sistema de Navegação.



Fonte: Elaborado pelo autor (2025).

A Tabela 1 organiza de forma objetiva os principais componentes que compõem o aplicativo LagoTur, descrevendo a função desempenhada por cada um na arquitetura do sistema. Essa síntese permite visualizar como as responsabilidades estão distribuídas entre os módulos, evidenciando a estrutura lógica que sustenta o funcionamento da aplicação e a interação entre suas diferentes partes.

Tabela 1 - Principais componentes e suas funcionalidades.

Componente/Módulo	**Função Principal**
App	Inicializa o aplicativo e carrega o sistema de navegação.
MyNavigation	Gerencia a navegação global e direciona o usuário às telas corretas.
MainComponent	Centraliza dados do Firebase e controla qual rota principal deve ser exibida.
HomeRoute	Tela inicial, apresenta atalhos e acesso às principais funcionalidades.
ExplorarRoute	Lista estabelecimentos e direciona para a tela de detalhes do estabelecimento selecionado.
EventosRoute	Lista eventos cadastrados e permite acessar detalhes individuais do evento selecionado.
RoteirosRoute	Lista roteiros turísticos e envia o usuário ao módulo de guias turísticos.

EstabelecimentoComponent	Exibe informações completas do estabelecimento selecionado.
EventosComponent	Tela de detalhes dos eventos contendo informações de data, hora, localização, descrição, fotos e vídeos.
GuiasTuristicosComponent	Apresenta roteiros turísticos, instruções, detalhes e pontos de visita da rota selecionada.
SugestoesComponent	Tela inicial com estabelecimentos de sugestões para o usuário (Apresentada na *HomeRoute*).
PerfilComponent	Gerencia dados do usuário (nome, e-mail, imagem) e permite edição.
Mapa	Abre localização e rota no Google Maps baseado nas coordenadas.
LoginScreen	Tela de autenticação, direciona à interface principal após login.
RegistrationScreen	Realiza cadastro do usuário e o direciona para a navegação principal.

Fonte: Elaborado pelo autor (2025).

Com essa organização o sistema visa apresentar uma estrutura lógica consistente, previsível e escalável, sustentada por uma arquitetura modular que integra classes de lógica, componentes de interface e um sistema de navegação. Essa abordagem tem como objetivo garantir clareza estrutural, facilitar a manutenção e facilitar adição de novas funcionalidades.

4.7 Camada de dados

A Camada de Dados (*Data Layer*) é responsável pela comunicação direta com o banco de dados da aplicação, realizando operações de consulta, atualização e sincronização das informações armazenadas. Segundo Fowler (2002), a camada de dados deve encapsular os mecanismos de persistência, expondo apenas métodos de leitura e escrita, de forma que o restante da aplicação permaneça independente de detalhes técnicos de armazenamento. Por conta disso essa camada visa garantir que o aplicativo permaneça flexível frente a mudanças no banco de dados ou na forma de acesso aos dados.

A camada de dados, portanto, assume um papel central no funcionamento do LagoTur, garantindo que as informações sejam obtidas, tratadas e entregues de

maneira consistente aos demais módulos da aplicação. Sua estrutura visa organização e independência, permitindo que o sistema evolua sem impacto direto nas outras camadas, mantendo estabilidade e previsibilidade no fluxo de dados.

4.7.1 Integração em nuvem com Firebase e uso de *AsyncStorage*

A integração com o Firebase foi um ponto estratégico para garantir que o aplicativo tivesse persistência de dados em tempo real, autenticação segura e facilidade de gerenciamento das informações, essa ferramenta foi utilizada como backend para armazenar em nuvem diversos tipos de informações como dados de eventos, estabelecimentos, roteiros turísticos, além de gerenciar a autenticação dos usuários, permitindo que qualquer ação realizada no aplicativo fosse refletida instantaneamente na interface. Essa abordagem elimina a necessidade de desenvolver uma infraestrutura própria de servidor, que segundo Kurose e Ross (2017), consiste no conjunto de máquinas, serviços e processos responsáveis por receber requisições dos clientes, processar dados, manter conexões ativas e gerenciar toda a lógica de backend, ao adotar uma solução em nuvem, evita-se a manutenção desse ambiente complexo.

As informações do sistema (como dados de estabelecimentos, eventos, roteiros turísticos e usuários) são mantidas no Firestore, um banco de dados NoSQL (banco de dados não relacional) baseado em documentos, armazenados dentro de coleções, o que permite organizar informações de forma hierárquica e flexível (FIREBASE, 2025). Esse formato foi escolhido pois permite que os dados sejam estruturados em coleções e documentos, o que se adapta muito bem ao modelo de entidades do aplicativo dessa forma cada documento pode representar uma instância dessas entidades, mantendo seus atributos de forma organizada e facilmente acessível.

Quando o aplicativo é executado, realiza consultas diretas ao Firestore, que retorna os dados no formato *JavaScript Object Notation* (JSON), esses dados são então convertidos internamente para as classes-modelo do sistema, garantindo que a estrutura lógica da aplicação permaneça padronizada e fácil de manipular. A comunicação ocorre de maneira assíncrona, ou seja, o aplicativo envia a requisição e continua funcionando normalmente enquanto aguarda a resposta do servidor, permitindo que o aplicativo recupere as informações sob demanda, sem travar a interface do usuário. O Firebase mantém sincronização em tempo real, garantindo

que qualquer alteração feita no banco de dados seja refletida automaticamente no app, essa arquitetura tornou possível que o LagoTur encontre dados complexos de forma simples e rápida.

A primeira etapa da integração foi a configuração do Firebase dentro do projeto (Código-fonte 1), para isso foi criado um módulo de configuração centralizada chamada *firebaseConfig*, responsável por inicializar e exportar a instância da aplicação Firebase, na sua estrutura é definido um objeto de configuração que contém as credenciais fornecidas, esses parâmetros garantem que o aplicativo esteja corretamente vinculado ao projeto na nuvem, possibilitando a utilização de serviços como autenticação de usuários, banco de dados em tempo real (Firestore) e armazenamento de arquivos.

Código-fonte 1 – Módulo de configuração firebaseConfig

```
1  import { getApp, getApps, initializeApp } from "firebase/app";
2  const firebaseConfig = {
3    apiKey: "AIzaSyBl22…",
4    authDomain: "igr-lago-tur.firebaseapp.com",
5    projectId: "igr-lago-tur",
6    storageBucket: "igr-lago-tur.appspot.com",
7    messagingSenderId: "1049…",
8    appId: "1:1049430657076:web…",
9    measurementId: "G-3GTTXD5F8R"
10 };
11 const app = getApps().length === 0 ? initializeApp(firebaseConfig) :
   getApp();
12 export default app;
```

O *firebaseConfig* também implementa um controle que verifica se já existe uma instância ativa do Firebase antes de inicializá-la novamente, esse procedimento é realizado por meio da função *getApps*, que retorna todas as instâncias já existentes, e da condicional que utiliza *initializeApp(firebaseConfig),* para inicializar as configurações do banco de dados apenas quando não há instâncias ativas, essa estratégia visa garantir maior estabilidade ao aplicativo, prevenindo falhas na execução.

Por fim, a instância criada é exportada e reutilizada em outras partes do código, permitindo que módulos distintos como os responsáveis pela navegação e pelas rotas de exibição de dados, possam se conectar ao banco de dados e aos demais serviços

do Firebase de forma padronizada, dessa forma o *firebaseConfig* atua como um ponto único de integração, simplificando o gerenciamento da conexão com o backend e garantindo a escalabilidade da aplicação.

Além da integração em nuvem, o aplicativo também utiliza o *AsyncStorage*, mecanismo de armazenamento local do dispositivo incluído pelo Expo, que permite salvar informações de forma persistente diretamente no celular do usuário. Segundo a documentação oficial do React Native (META, 2023), o *AsyncStorage* funciona como um sistema de armazenamento chave-valor assíncrono e persistente, projetado para manter dados essenciais sem depender de conexão com a internet. Dessa forma, informações como cidade selecionada, credenciais de login e preferências do usuário permanecem disponíveis mesmo em cenários de instabilidade ou ausência de conectividade, esse armazenamento complementa o acesso remoto ao Firestore, com objetivo de garantir maior continuidade de uso e melhoria na experiência do usuário.

O vínculo bem-sucedido do aplicativo com o banco de dados Firebase garante que os usuários possam acessar todas as informações disponibilizadas dentro da plataforma, desde dados turísticos até conteúdos personalizados, essa integração também possibilita que cada utilizador crie e gerencie sua própria conta, registre preferências pessoais, visualize eventos e até realize interações. Sendo assim, o banco de dados atua como um dos principais recursos para funcionamento correto do aplicativo, centralizando e organizando as informações de maneira segura e acessível.

4.7.2 Estrutura dos dados armazenados no Firestore

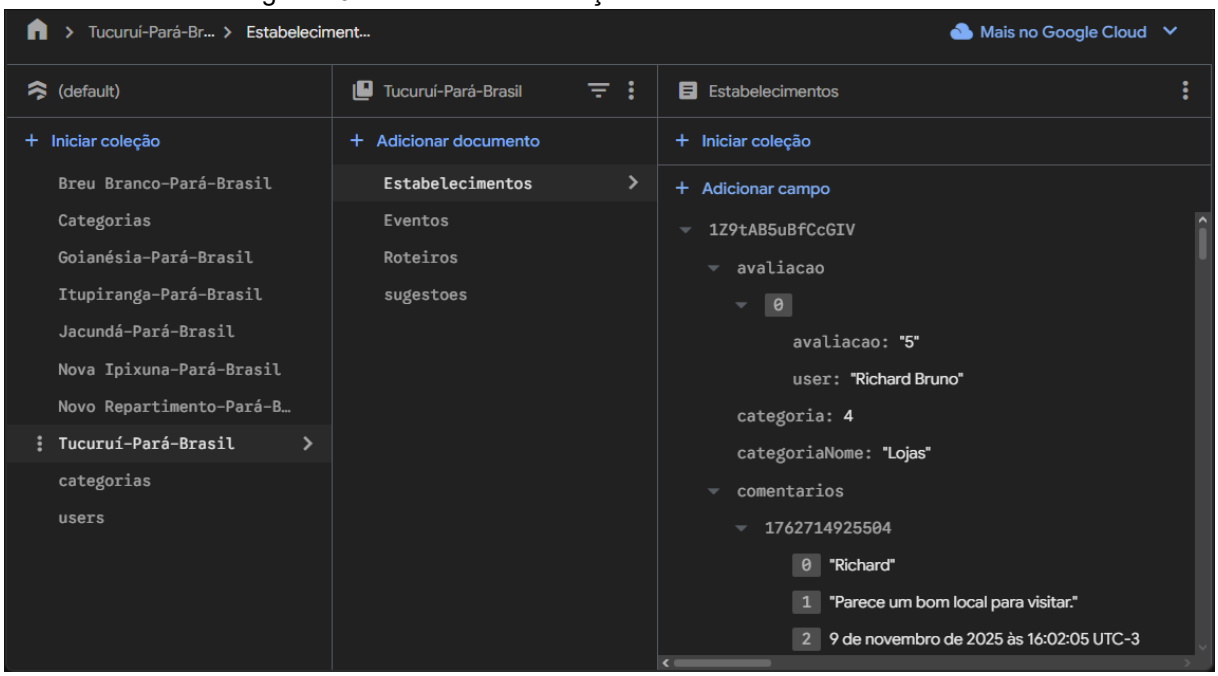
A estrutura dos dados armazenados no Firebase Firestore foram organizadas de maneira a refletir diretamente as entidades do domínio turístico da aplicação, o sistema utiliza coleções como contêineres principais, dentro das quais cada documento representa um item do domínio.

Os dados foram armazenados no Firestore de forma estruturada e segmentada onde cada coleção corresponde a um município cadastrado atuando como uma coleção principal (Figura 10). Dentro de cada coleção o sistema organiza as informações em documentos específicos que representam os tipos de conteúdo manipulados pelo aplicativo, como *“Estabelecimentos”*, *“Eventos”*, *“Roteiros”* e *“sugestoes”*. A divisão em

coleções por cidade e documentos por tipo de conteúdo tem como objetivo tornar o banco escalável, organizado e otimizado para consultas específicas da aplicação.

No interior desses documentos, o sistema acessa dinamicamente um identificador (ID) que atua como chave primária lógica, garantindo unicidade para cada registro inserido. Esse identificador referencia um objeto completo contendo todos os atributos necessários para a representação da entidade (nome, descrição, localização, mídia, avaliações, comentários, entre outros), essa estrutura permite que cada entrada seja tratada como uma unidade independente, facilitando consultas, atualizações seletivas e a recuperação precisa dos dados para exibição ao usuário durante a navegação.

Figura 10 - Estrutura das coleções e documentos no Firestore.



Fonte: Elaborado pelo autor (2025).

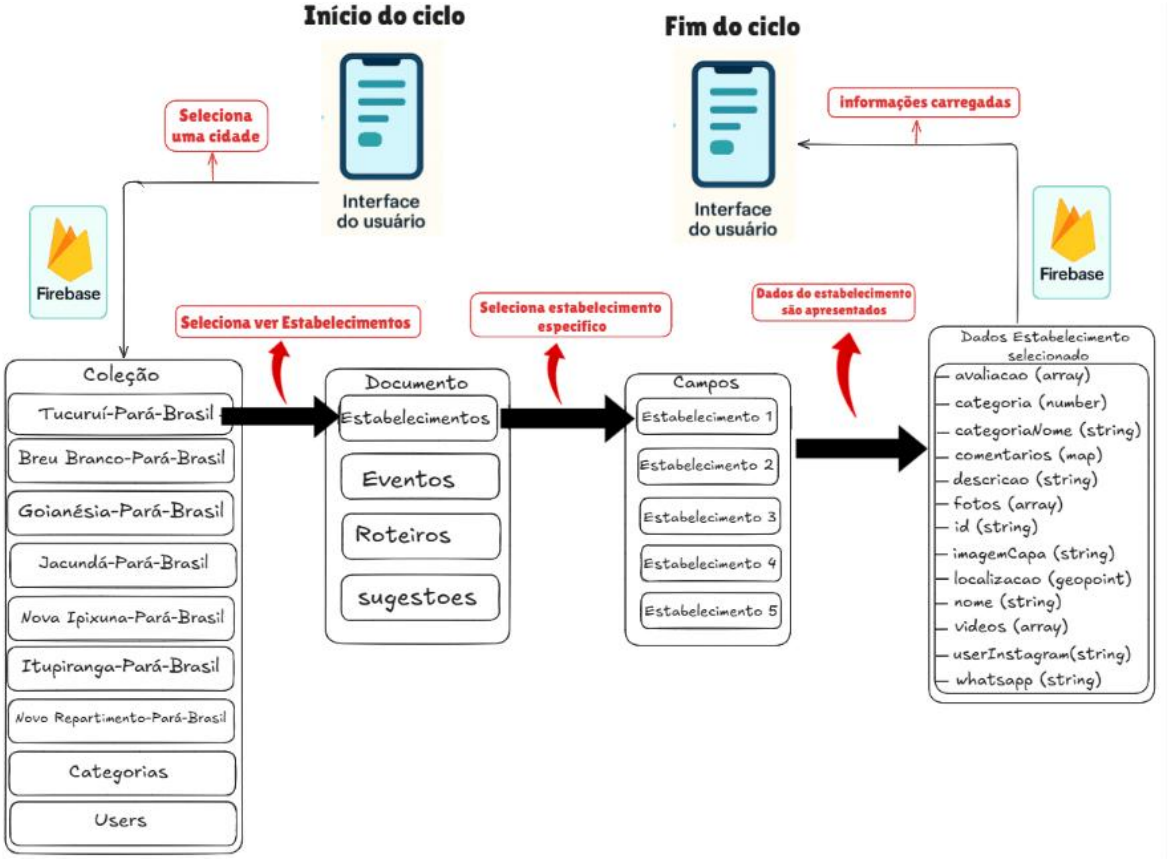
Embora cada documento do Firestore possua uma estrutura própria, contendo informações específicas para o contexto ao qual pertence, todos seguem um padrão organizacional comum, visando garantir que mesmo com diferenças de conteúdo entre documentos da coleção a estrutura interna permaneça coerente o suficiente para que o aplicativo possa consultar e manipular os dados de maneira uniforme. Essa abordagem é essencial para que a camada lógica da aplicação recupere apenas as informações pertinentes ao módulo acessado, ao mesmo tempo em que assegura consistência na leitura dos campos e facilita a manutenção e expansão futura da base de dados.

O fluxo de acesso aos dados no aplicativo segue uma sequência estruturada (Figura 11), iniciando-se quando o usuário seleciona uma das cidades disponíveis na interface, o sistema identifica qual coleção correspondente deverá ser consultada no Firestore e a partir disso os documentos correspondentes podem ser carregados para o usuário.

Como exemplo de requisição de dados entre o aplicativo e o Firestore, considere o cenário em que o usuário seleciona uma cidade e em seguida, opta por visualizar a lista de estabelecimentos disponíveis naquela região. Nesse momento, o sistema consulta o documento “Estabelecimentos” dentro da coleção correspondente ao município escolhido e recupera todos os registros armazenados, cada um identificado por uma chave única que permite sua individualização.

Ao clicar em um estabelecimento específico, o aplicativo utiliza esse identificador para requisitar ao Firestore somente o objeto associado ao item selecionado, dessa forma apenas as informações relevantes são carregados e apresentadas na interface, esse mecanismo tem como objetivo agilizar as consultas, precisão no retorno das informações e consistência na experiência do utilizador. A Figura 11 ilustra o fluxo de funcionamento citado anteriormente, evidenciando como a interação do cliente, a estrutura de armazenamento no banco de dados e a renderização dos processos ocorrem.

Figura 11 - Fluxograma do ciclo de requisição de dados do Firestore.



Fonte: Elaborado pelo autor (2025).

A adoção de coleções segmentadas por município, documentos categorizados por tipo de conteúdo e registros individualizados por identificadores únicos permitiu construir um modelo de dados orientado ao domínio turístico da aplicação. A estrutura adotada tem por objetivo facilitar o acesso eficiente às informações durante a navegação e, também simplificar os processos de expansão, manutenção e inclusão de novos conteúdos no banco, a modelagem empregada visa garantir consistência no armazenamento e recuperação de dados.

4.7.3 Ferramenta auxiliar em Python para inserção de dados no Firestore

Para facilitar e automatizar o gerenciamento das informações utilizadas pelo sistema, foi desenvolvido um programa auxiliar em Python sem vínculo direto com o aplicativo, sendo responsável por inserir novos registros no Firebase Firestore de forma estruturada e padronizada. A criação dessa ferramenta teve como referência conceitual e técnica um tutorial que aborda a relação entre bancos de dados e o tratamento de informações utilizando Python (FREECODECAMP, 2023), além da

documentação oficial do Firebase, que orienta os procedimentos adequados para manipulação de dados no Firestore (GOOGLE, 2024).

O uso de um software dedicado para inserção de informações garante que todos os documentos sigam o mesmo padrão estrutural adotado pelo aplicativo, com o objetivo de evitar inconsistências nos campos, tipos de dados ou organização interna, assegurando que o aplicativo consiga interpretar corretamente todas as entradas, já que a estrutura produzida pelo script corresponde exatamente ao modelo consumido pela camada de dados do sistema.

O script inicia carregando as credenciais de acesso ao Firebase por meio de um arquivo JSON contendo a chave de serviço da aplicação, o funcionamento do programa baseia-se na interação direta com o usuário por meio de entradas no console. Primeiramente, solicita-se o nome da coleção correspondente ao município em que o estabelecimento será inserido e em seguida, o usuário seleciona qual tipo de documento deseja alimentar.

Para o cadastro de estabelecimentos, o programa executa uma função, responsável por coletar todas as informações necessárias. Cada etapa do preenchimento ocorre de forma modular permitindo adicionar múltiplos itens em listas (como fotos e vídeos) ou estruturas mapeadas (como comentários ou avaliações).

O software desenvolvido em Python segue um fluxo estruturado dividido em três etapas principais: inicialização, processamento e envio dos dados ao Firestore (Figura 12). Na primeira fase, o usuário executa o programa, que carrega automaticamente as credenciais do Firebase e estabelece a conexão com a coleção e o documento selecionados, após isso ocorre o processamento das informações inseridas pelo operador no qual o sistema reúne os dados fornecidos, organiza-os no formato esperado pelo Firestore e gera um identificador exclusivo, que funciona simultaneamente como chave do campo e como atributo interno do objeto armazenado.

A abordagem utilizada visa garantir que cada registro seja singularmente identificável, facilitando futuras consultas e integrando-se de maneira direta à lógica de recuperação utilizada no aplicativo móvel. Na etapa de envio de dados, o script cria um novo campo no documento correspondente usando o ID gerado e grava o objeto completo no Firestore. Ao término da operação, o sistema exibe uma mensagem de confirmação, assegurando que o processo foi concluído com êxito e que os dados foram devidamente persistidos no Firestore.

Figura 12 - Fluxo de inserção estruturada de dados no Firestore via script Python.

Fonte: Elaborado pelo autor (2025).

Esse mecanismo de inserção estruturada tem como objetivo automatizar o processo de inserção de dados no Firestore e possibilitar maior controle sobre a organização das informações, assegurando que os dados sejam gravados exatamente no formato esperado pelo aplicativo móvel, visando manter coerência na estrutura interna dos documentos, previsibilidade para consultas e integridade dos dados apresentados ao usuário.

4.8 Camada de apresentação

A Camada de Apresentação (*UI Layer*) é responsável por lidar com a interação direta com o usuário e por transformar os dados do sistema em elementos visuais compreensíveis. Fowler (2002) explica que a função dessa camada é apresentar informações ao usuário e interpretar seus comandos, operando como um intermediário entre a interface e as camadas internas da aplicação. Essa separação tem por objetivo permitir que alterações na lógica interna não comprometam o comportamento da interface.

No sistema, a camada de apresentação assume papel fundamental ao converter os dados consultados no Firestore em elementos visuais organizados e que podem ser compreendidos pelo usuário. Sua função é estruturar a interface de forma que o fluxo de navegação se mantenha intuitivo e consistente, facilitando a percepção das informações exibidas. Nesse sentido, Nielsen (1993) afirma que o sistema deve minimizar a carga de memória do usuário tornando objetos, ações e opções visíveis, destacando que a interface deve evitar que o usuário dependa da memorização de comandos ou etapas para operar o sistema. A aplicação foi

projetada com o objetivo de apresentar informações de forma explícita, garantindo que a exploração de estabelecimentos, eventos e roteiros turísticos ocorra de maneira fluida, sem exigir esforço excessivo cognitivo do utilizador para compreensão das telas.

4.8.1 Planejamento, modelagem e implementação da camada de apresentação

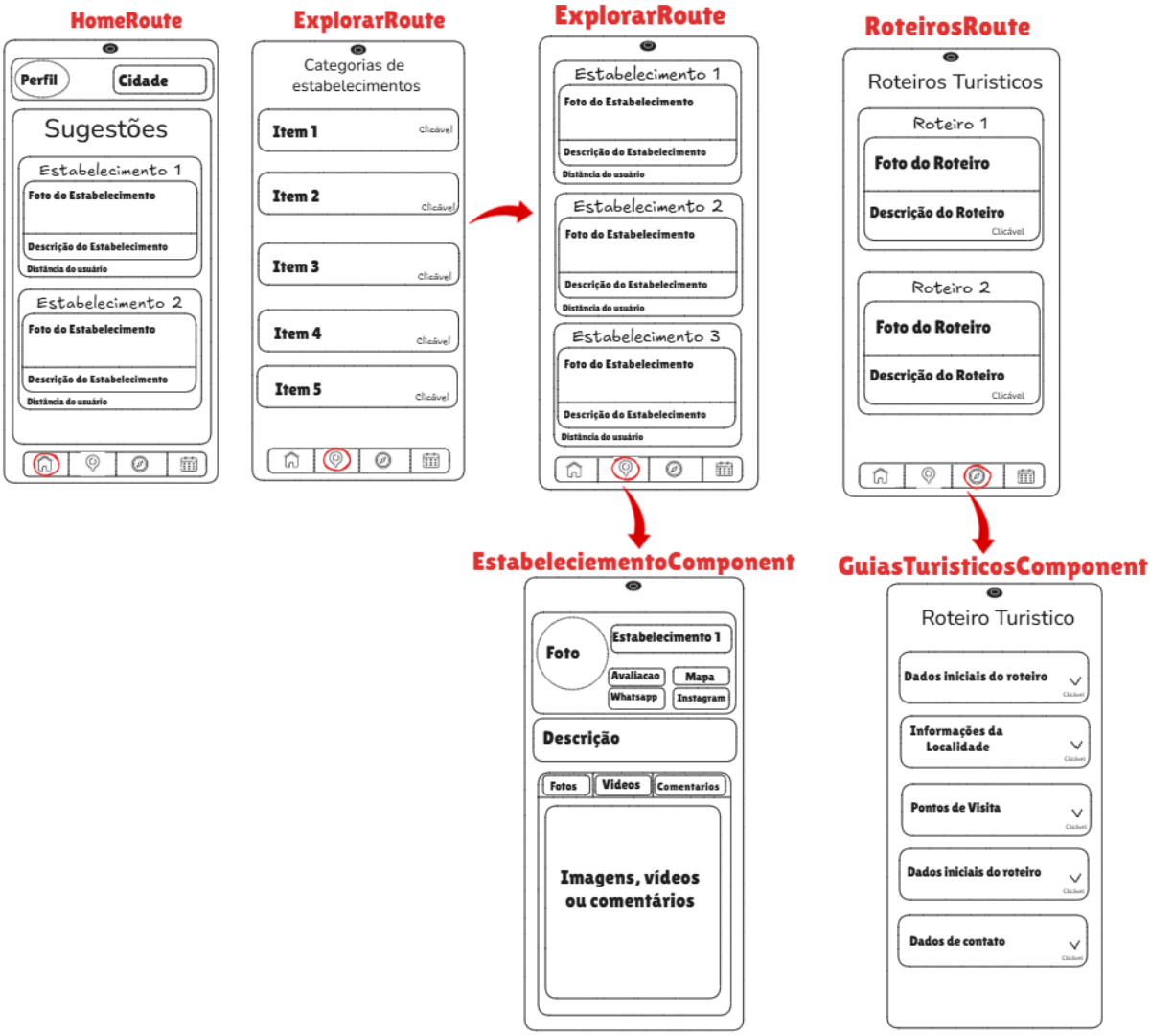
Um requisito fundamental da engenharia de software empregada no desenvolvimento do LagoTur foi a atenção dedicada à experiência do usuário (UX) e à interface do usuário (UI), elementos essenciais para garantir uma interação eficiente e intuitiva. Segundo Norman (2013), uma boa experiência do usuário depende da capacidade do sistema de apresentar suas funcionalidades de forma clara, reduzindo atritos cognitivos e facilitando a tomada de decisões durante a navegação. Tendo isso em vista, a camada de apresentação deve oferecer feedback imediato e manter consistência visual para que o usuário compreenda o funcionamento da aplicação sem esforço excessivo.

Para o planejamento e desenvolvimento da camada de apresentação do aplicativo, buscou-se adotar práticas consolidadas de design de interface, visando garantir clareza visual, coerência estrutural e facilidade de uso. A etapa inicial concentrou-se na definição de padrões estéticos e comportamentais que orientariam a construção de cada tela do sistema, nesse processo o site Dribbble desempenhou papel fundamental como fonte de inspiração sendo uma plataforma internacional amplamente utilizada por designers e desenvolvedores, reunindo portfólios, modelos visuais, componentes de UI e tendências contemporâneas de design. Sua função nesse contexto foi oferecer referências estilísticas e funcionais que auxiliaram na definição de layouts, organização de elementos e interações que pudessem ser adaptadas para a realidade do aplicativo.

Além disso, utilizou-se o Excalidraw como ferramenta de apoio para prototipação visual e organização estrutural das telas (Figura 13), sendo um ambiente de desenho interativo voltado para esboços rápidos, fluxogramas e wireframes, permitindo que ideias iniciais fossem representadas graficamente antes da implementação, ele foi essencial para mapear a disposição dos componentes, definir áreas de interação, estruturar a hierarquia de informações e antecipar a experiência do usuário, permitindo visualizar previamente a organização dos

elementos na tela (cards, listas, botões e seções de conteúdo) possibilitando o ajuste, avaliação e validação da navegação.

Figura 13 - Wireframe das informações das telas no Excalidraw.



Fonte: Elaborado pelo autor (2025).

Esses esboços serviram como guia para o posicionamento das *views* (contêiner visual) durante o desenvolvimento, assegurando que cada componente de interface fosse estruturado de forma lógica e coerente com a experiência desejada para o usuário. Essa etapa foi fundamental para alinhar requisitos funcionais ao design visual, visando reduzir ambiguidades e facilitando a implementação das telas. Ao centralizar decisões estéticas e funcionais, houve uma otimização do processo de codificação do protótipo que evitaram retrabalhos e fizeram com que a camada de apresentação fosse construída alinhada ao fluxo de navegação do aplicativo.

Somado a isso, o uso combinado dessas ferramentas contribuiram para que a camada de apresentação fosse desenvolvida de forma planejada, fundamentada e orientada por princípios contemporâneos de UI/UX. Como destaca Norman (2013), interfaces eficazes são resultado de decisões intencionais sobre visibilidade, feedback e consistência, fatores diretamente influenciados pelo uso de protótipos e referências visuais. Dessa forma, o planejamento da UI do LagoTur buscou não apenas atender às necessidades funcionais do sistema, mas também promover uma experiência clara, agradável e consistente ao usuário, fortalecendo a navegabilidade entre as telas do sistema.

Por fim, a integração entre planejamento, modelagem e implementação resultou em uma camada de apresentação que visa entregar para os usuários as informações da maneira planejada. A combinação entre padrões de usabilidade, organização modular de componentes e navegação estruturada foi implementada com o objetivo de possibilitar a criação de um ambiente visual que facilita a descoberta de informações e contribui diretamente para a experiência turística proposta pelo LagoTur. Dessa forma, a UI Layer se consolida como elemento essencial para a compreensão do sistema e para a satisfação do usuário durante a interação com o aplicativo.

4.8.2 Estrutura lógica e desenvolvimento da camada de apresentação

A construção da camada de apresentação envolveu a definição de padrões de navegação, organização e criação de layouts para os componentes visuais, com objetivo de garantir que a interação com as demais funcionalidades implementadas ocorresse de forma intuitiva e previsível. Para alcançar esse resultado, adotou-se uma abordagem centrada no usuário, com foco na clareza da informação, na fluidez das transições e na facilidade de exploração do conteúdo turístico disponibilizado pelo sistema.

No processo de implementação, buscou-se estabelecer um conjunto de componentes reutilizáveis, como cards de estabelecimentos e eventos, botões padronizados, barras de navegação, caixas de conteúdo e elementos interativos. Essa prática não apenas contribuiu para a uniformidade visual em todas as seções do aplicativo, mas também aumentou a eficiência no desenvolvimento. Ao criar estruturas modulares, foi possível aplicar os mesmos padrões de desenvolvimento da camada

de apresentação em diferentes componentes, mantendo consistência entre telas e reduzindo a duplicação de código.

A lógica de comunicação entre os componentes da camada de apresentação foi modelada para favorecer clareza e separação de responsabilidades, as rotas recebem os dados já estruturados pelo *MainComponent* , que atua como intermediário entre a camada de dados e a interface, evitando que cada tela tenha que realizar consultas diretas ao Firestore, dessa forma a UI Layer permanece focada exclusivamente em exibir informações e responder às ações do usuário.

A etapa de desenvolvimento também demandou decisões específicas relacionadas à acessibilidade e responsividade, visando garantir que a interface se adaptasse adequadamente a diferentes tamanhos de tela e dispositivos. Aspectos como hierarquia tipográfica, espaçamento entre elementos, áreas de toque ampliadas e contraste de cores foram planejados para assegurar legibilidade e interação confortável ao utilitário final.

No desenvolvimento da camada de apresentação a estilização da interface foi realizada por meio do *StyleSheet* do React Native, recurso utilizado para criar estilos estruturados em objetos JavaScript. Segundo a documentação de desenvolvimento, o StyleSheet permite definir estilos em cascata, porém usando sintaxe baseada em objetos (META, 2023). Dessa forma, elementos como *View*, *Text*, *Image* (imagem) e *ScrollView* (rolagem de tela) atuam de maneira análoga às tags HTML (*Hypertext Markup Language*) que é a estrutura visual da tela, enquanto o StyleSheet é responsável por determinar como esses elementos serão exibidos, controlando cores, espaçamentos, tamanhos, fontes e demais propriedades visuais representados no Código-fonte 2, sendo esse o início da folha de estilização do componente ExplorarRoute.

O uso da biblioteca *React Native Paper* desempenhou papel fundamental, pois forneceu componentes estruturados (como cards, botões, barra de navegação, dentre outros*)* já configurados com padrões modernos, reduzindo o esforço de estilização manual. Complementarmente a biblioteca *React Native Vector Icons* foi utilizada para compor a interface com ícones consistentes e escaláveis, garantindo clareza sem comprometer o desempenho. A combinação dessas ferramentas permitiu criar telas visualmente identificaveis, tendo como objetivo manter o código organizado, reutilizável e alinhado às boas práticas de design de interface presentes no ecossistema React Native.

Código-fonte 2 - Criação de elementos e estilização de tela

```
export const styles = StyleSheet.create({
    container: {
        flex: 1,
        backgroundColor: '#E6F5E6',
    },
    searchContainer: {
        flexDirection: 'row',
        backgroundColor: '#fff',
        borderRadius: 30,
        padding: 15,
        marginHorizontal: 20,
        marginVertical: 20,
        alignItems: 'center',
        shadowColor: "#000",
        shadowOffset: { width: 0, height: 2 },
        shadowOpacity: 0.1,
        shadowRadius: 5,
        elevation: 2,
    },
```

Por fim, todo o processo de construção da camada de apresentação foi conduzido de forma iterativa, permitindo validações contínuas conforme os dados reais do Firestore eram integrados às telas, essa estratégia possibilitou ajustes rápidos na estrutura visual, correção de conflitos de layout e refinamento da experiência de navegação, com o objetivo de garantir que o resultado final estivesse alinhado tanto aos requisitos técnicos quanto às expectativas de usabilidade do público-alvo. A interface desenvolvida para a aplicação consolida-se como um pilar essencial do sistema, traduzindo dados complexos em interfaces organizadas e acessíveis, que visam facilitar a exploração das funcionalidades.

4.9 Utilização do framework Expo no desenvolvimento do LagoTur

A construção do aplicativo foi fortemente apoiado pelo uso do Expo que fornece um conjunto de ferramentas, bibliotecas integradas e serviços que agilizam significativamente o ciclo de desenvolvimento. De acordo com a documentação oficial do Expo, o framework foi projetado para permitir que desenvolvedores construam aplicações nativas usando apenas JavaScript e React, eliminando a necessidade de lidar diretamente com configurações nativas complexas (EXPO, 2024). Essa filosofia

de simplificação foi essencial para o LagoTur, pois possibilitou que o foco do projeto permanecesse na lógica do aplicativo, em sua interface e na integração com o Firebase, sem a necessidade de configurações durante as etapas iniciais de desenvolvimento.

A utilização do Expo também contribuiu para padronizar o ambiente de desenvolvimento, garantindo maior estabilidade no processo de construção da camada de apresentação e da lógica interna. O sistema de módulos do Expo, que inclui bibliotecas prontas para uso reduziu o tempo de implementação e evitou problemas comuns relacionados à compatibilidade nativa. De acordo com o Expo Documentation (2024), suas bibliotecas internas seguem padrões de atualização contínua e são projetadas para funcionar de maneira consistente em múltiplas plataformas, o que favoreceu diretamente a previsibilidade e o controle da interface no desenvolvimento.

No processo de desenvolvimento do LagoTur diversas bibliotecas e módulos do Expo foram incorporadas para viabilizar funcionalidades centrais do aplicativo, dentre os principais utilizados estão representados na Tabela 2.

Tabela 2 - Principais módulos e bibliotecas do Expo utilizados no LagoTur.

Expo Location	Utilizada para obter a localização atual do usuário, permitindo calcular distâncias até estabelecimentos, gerar rotas e integrar o aplicativo com serviços externos de mapas.
MediaLibrary e ImagePicker	Possibilitaram a seleção e manipulação de imagens diretamente no dispositivo do usuário, recurso essencial para telas que exibem conteúdos visuais, além de permitir futuras expansões como upload de imagens.
Clipboard	Responsável pelo compartilhamento e cópia rápida de informações e links, agregando praticidade à interação do usuário.
Expo Managed Workflow	Simplifica o gerenciamento de dependências e builds, permitindo focar na lógica e na interface do aplicativo.
Expo Linking	Permite abrir links diretamente em outros aplicativos instalados no dispositivo, sendo essencial no

	processo de abrir informações diretamente no Google Maps, Instagram ou Whatsapp.

Fonte: Elaborado pelo autor.

Esse processo de incrementação e importação de bibliotecas internas do dispositivo de forma direta do Expo reduziu significativamente a necessidade de intervenções manuais em estruturas Android e iOS, fazendo com que módulos essenciais (como abertura de links no dispositivo, manipulação de mídia, notificações e gestão de permissões) fossem incorporados ao LagoTur de maneira facilitada. A compatibilidade entre Expo e bibliotecas JavaScript possibilitou a conexão com o Firestore sem dependências nativas adicionais, o que otimizou o processo de implementação dessa ferramenta.

Outro recurso essencial no processo de desenvolvimento foi o *Expo Go*, aplicativo oficial da plataforma, destinado a facilitar a visualização e o teste de projetos em tempo real. Essa ferramenta permite executar o aplicativo diretamente em um dispositivo físico ou emulador por meio da leitura de um QR Code ou acesso a uma URL. Conforme descrito pela própria Expo (EXPO, 2024) o Expo Go fornece um ambiente de execução capaz de atualizar instantaneamente a aplicação sempre que o código é alterado, eliminando o tempo de recompilação e permitindo ciclos de teste muito mais rápidos. No desenvolvimento do LagoTur, essa funcionalidade foi determinante para acelerar as etapas de prototipagem, estilização e validação de componentes, garantindo um fluxo de trabalho dinâmico, iterativo e eficiente, especialmente durante o ajuste visual das telas e a testagem das interações entre rotas e componentes.

Por fim, o uso do Expo se mostrou fundamental para o gerenciamento e a evolução do projeto ao longo do tempo, permitindo que novas funcionalidades fossem adicionadas de maneira incremental sem comprometer o restante do sistema, além de fornecer ferramentas para depuração, logs e testes em múltiplos dispositivos. Essa combinação de praticidade e robustez tornou o Expo uma peça central no processo de desenvolvimento do LagoTur.

5 DESENVOLVIMENTO

5.1 Funcionalidades de autenticação e gerenciamento de usuários

O sistema de cadastro e login foi planejado com o objetivo de permitir que o usuário tivesse suas interações e preferências registradas dentro do aplicativo, essa funcionalidade visa não apenas facilitar a experiência do usuário, mas também possibilitar o armazenamento de informações relacionadas a comentários e avaliações realizadas, garantindo maior personalização no uso da aplicação.

Para o desenvolvimento inicial da tela de registro, foi criado o componente *RegistrationScreen* que contém as importações das bibliotecas necessárias para o funcionamento da tela de cadastro, além de funções responsáveis por capturar os dados fornecidos pelo usuário, validar as informações inseridas e armazená-las no banco de dados.

Com o objetivo de garantir segurança e personalização no uso do aplicativo, foi implementado um sistema de autenticação por meio do Firebase Authentication, utilizando a biblioteca *firebase/auth*. Esse recurso está vinculado ao principal componente da tela de registro, permitindo que cada usuário possua credenciais individuais de acesso, realizando o tratamento dos dados do usuário e enviando-os para o Firebase representado no Código-fonte 3.

A autenticação foi estruturada de forma a possibilitar tanto o cadastro de novos usuários quanto o login em contas já existentes, funcionando de maneira integrada ao banco de dados Firestore e ao armazenamento local do dispositivo, essa integração assegura maior confiabilidade no acesso às informações, além de oferecer uma experiência mais prática e personalizada para cada usuário.

Código-fonte 3 - Função de registro de tela e inicialização cadastro no Firebase

```
const RegistrationScreen = ({navigation}) => {
    const [fullName, setFullName] = useState('')
    const [email, setEmail] = useState('')
    const [password, setPassword] = useState('')
    const [confirmPassword, setConfirmPassword] = useState('')
    const db=getFirestore(app, "(default)");
    const auth = getAuth(app);
    const onFooterLinkPress = () => {
        navigation.reset({
            index: 0,
```

```
            routes: [{ name: "MainComponent" }],
        });
    }
    const criaUserNoBanco = async (uid)=>{
        try {
            await setDoc(doc(db, "users", uid), {
                nome: fullName,
                uid: uid,
                imagemPerfil: imageNewUserDefault,
                email: email
            });
        } catch (e) {
        }
    }
```

Uma das principais funcionalidades implementadas na classe de registro do aplicativo está no componente responsável pela criação de contas. Nessa etapa, as funções *criaUserNoBanco* e *createUserWithEmailAndPassword* (Código-fonte 4) foram desenvolvidas para serem utilizadas em cadastros de novos usuários a partir das informações fornecidas sendo essas o e-mail, senha e nome.

Os dados do usuário (ID, nome, e-mail e uma imagem de perfil padrão) são preparados pela função *criaUserNoBanco*, que também se encarrega de salvar essas informações na memória local do dispositivo.

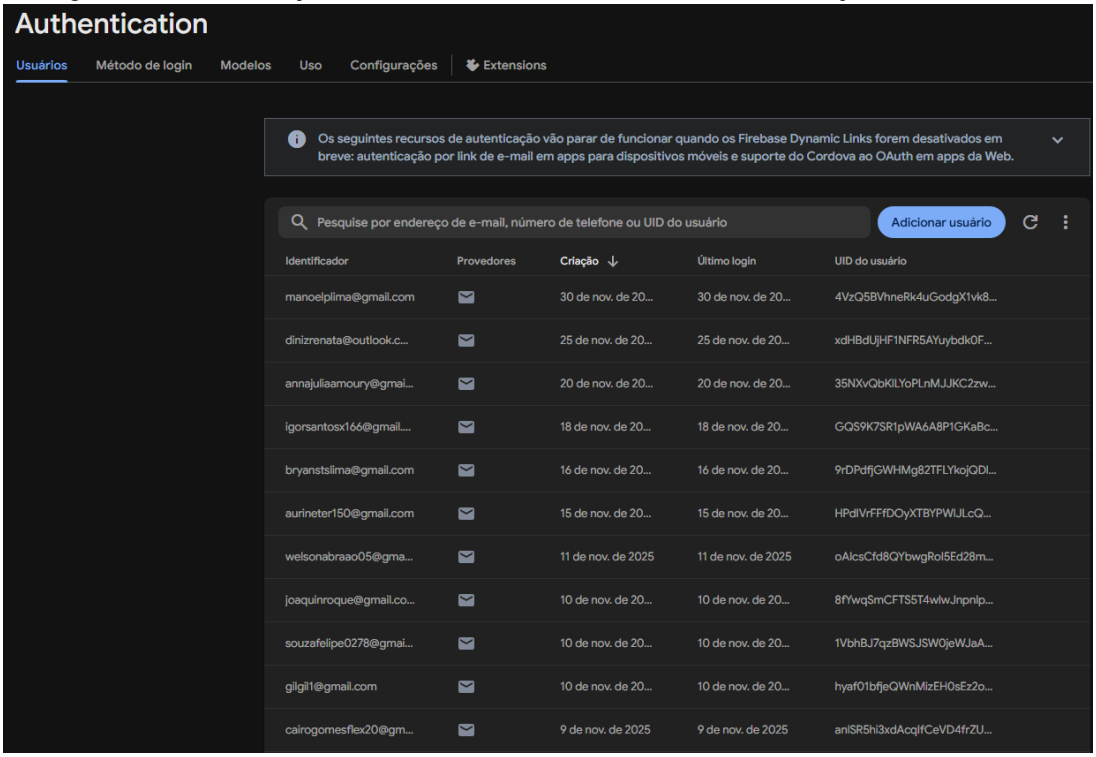
Código-fonte 4 - Função de registro de tela e inicialização cadastro no Firebase

```
createUserWithEmailAndPassword(auth, email, password)
.then(async (response) => {
    const uid = response.user.uid
    await criaUserNoBanco(uid);
    const objetoUser = {uid :uid, nome: fullName, email: email,
imagemPerfil: imageNewUserDefault};
    await AsyncStorage.setItem("user", JSON.stringify(objetoUser));
    navigation.reset({
        index: 0,
        routes: [{ name: "MainComponent" }],
    });
})
.catch((error) => {
    alert(error)
```

```
14 });
```

Em seguida os dados são utilizados pela função *createUserWithEmailAndPassword*, que registra o usuário no Firestore (Figura 14), cria o objeto de cadastro, armazena as informações localmente para permitir login automático e redireciona o usuário para a tela principal do aplicativo, caso existam informações duplicadas no banco de dados ou incorretas na tela de criação, o aplicativo retorna uma mensagem de erro informando o problema.

Figura 14 - Autenticação de usuários e armazenamento de informações no Firebase.



Fonte: Elaborado pelo autor (2025).

O componente de login de usuário chamado de *LoginScreen* funciona semelhante a tela de registro, importando as mesmas bibliotecas e funções com poucas diferenças, todavia essa parte é responsável por implementar a tela de autenticação de usuários do aplicativo. Sua função principal é permitir que o usuário acesse o sistema por meio da inserção de e-mail e senha, realizando a validação das credenciais através do serviço de autenticação do Firebase Authentication.

Para isso, o componente mantém estados locais que armazenam o e-mail e a senha informados, e utiliza a função *signInWithEmailAndPassword* para validar as credenciais, representada no Código-fonte 5. Caso o login seja bem-sucedido, os dados do usuário são buscados no banco de dados Firestore e armazenados na

memória do dispositivo, garantindo a persistência da sessão mesmo após o fechamento do aplicativo. Se o usuário já tiver uma sessão salva, é utilizado um recurso do React Native chamado *useEffect* que é amplamente utilizado no desenvolvimento do sistema pois permite executar uma ação automática quando a tela carrega ou quando algum dado muda, utilizada nesse caso para redirecionar automaticamente para a tela principal sem necessidade de novo login.

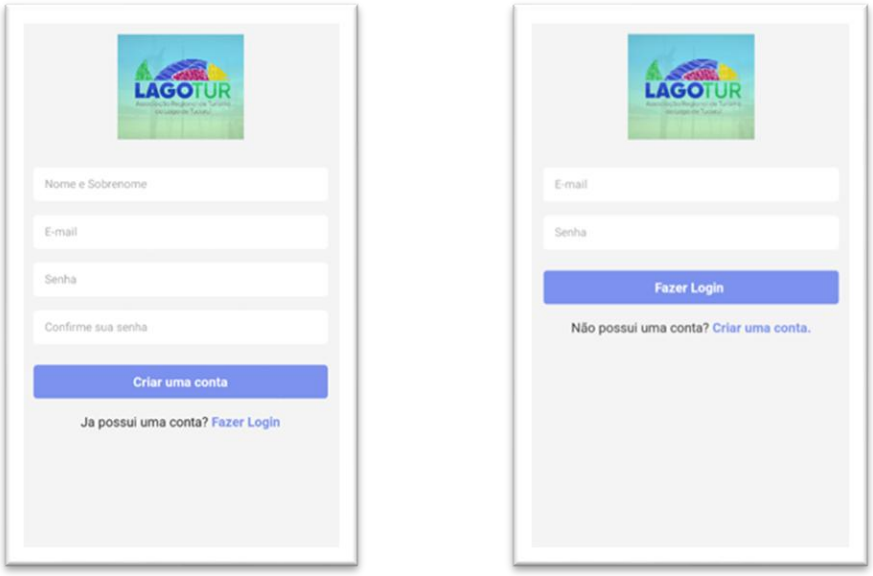
Código-fonte 5 - Função signWithEmailAndPassword para autenticação de usuário

```
const onLoginPress = () => {
    signInWithEmailAndPassword(auth, email, password)
    .then(async (response) => {
        const user = response.user
        const docRef = doc(DB, "users", user.uid);
        try {
            const docSnap = await getDoc(docRef);
            if (docSnap.exists()) {
                const usuario = docSnap.data();
                await
AsyncStorage.setItem("user",JSON.stringify(usuario));
                navigation.navigate("MainComponent")
            } else {
            }
        } catch (error) {
        }

    })
    .catch(error => {
        alert('Falha no login')
    })
}
```

Os componentes de cadastro e login de usuário são responsáveis por renderizar as telas de autenticação seguindo os padrões recomendados pela documentação do Firebase, garantindo padronização e facilitando tanto a compreensão do usuário quanto possíveis alterações no código. Além disso, a tela aplica os estilos definidos e exibe a logomarca da Lago Tur, reforçando a identidade visual do aplicativo representados na Figura 15. Por fim, a classe é exportada permitindo que outras partes do sistema acessem e utilizem suas funcionalidades de forma integrada.

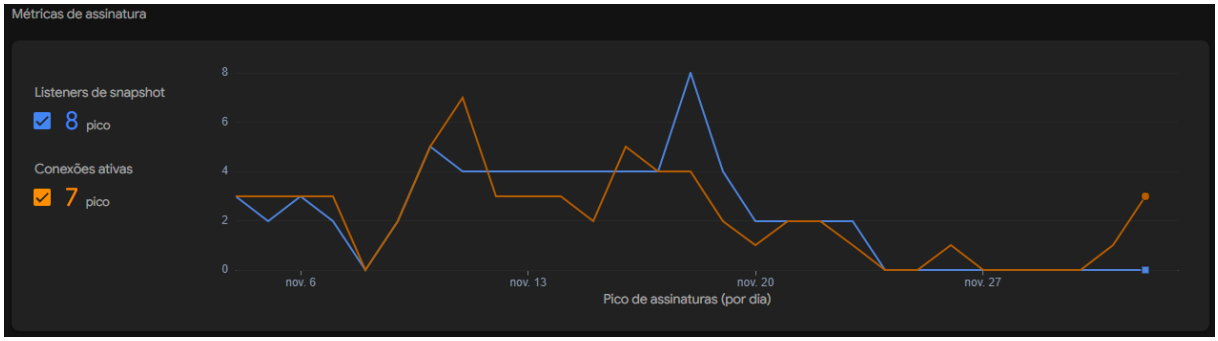
Figura 15 - Telas de cadastro e login de usuário.



Fonte: Elaborado pelo autor (2025).

Essas implementações permitem futuras expansões, como a personalização da experiência, o armazenamento de preferências e o registro do histórico de interações no aplicativo. Além disso, a autenticação possibilita acompanhar o engajamento de cada usuário, ampliando as perspectivas para análises de usabilidade (Figura 16).

Figura 16 - Acompanhamento de usuários ativos por período no Firebase.



Fonte: Elaborado pelo autor (2025).

Dessa maneira, o sistema de cadastro e login consolidou-se como uma etapa fundamental para a utilização do aplicativo, pois garante que cada usuário tenha acesso seguro às suas próprias informações, além de permitir a personalização da experiência de navegação. Assim, a autenticação não apenas tem como objetivo cumprir o papel de validar credenciais, mas também atua como a base para oferecer uma experiência mais personalizada, consistente e voltada às necessidades de cada usuário.

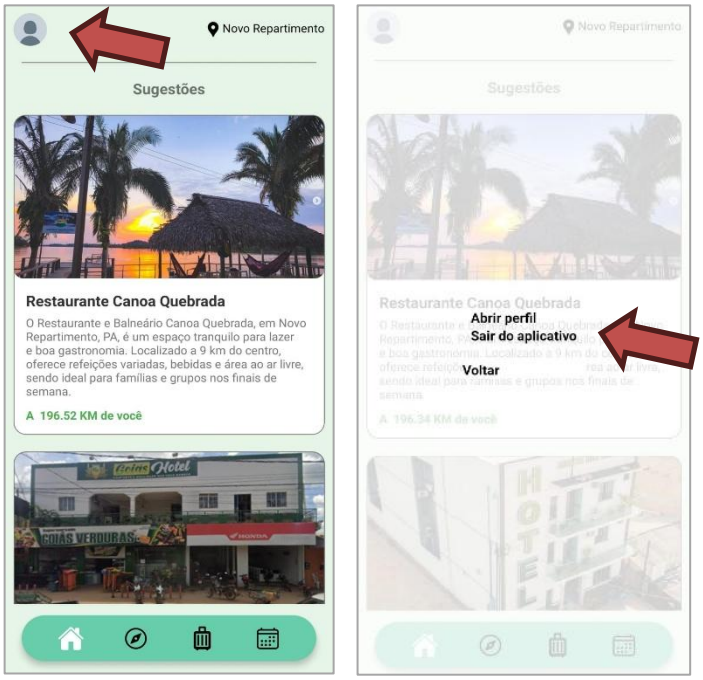
5.1.1 Perfil de usuário e alteração de dados cadastrais

A gestão do perfil de usuário no aplicativo foi projetada para permitir que cada indivíduo personalize e mantenha suas informações pessoais atualizadas, garantindo maior confiabilidade e adequação na experiência de navegação. Essa funcionalidade envolve a integração entre o armazenamento local do dispositivo, o banco de dados Firestore e o serviço de armazenamento de arquivos do Firebase, de forma a assegurar persistência e sincronização entre os dados exibidos na interface e aqueles registrados no servidor.

No banco de dados, as informações são administradas por meio da integração com o Firebase Authentication, responsável pela autenticação dos usuários e gerenciamento de credenciais de acesso, enquanto o Firestore Database armazena dados complementares, como nome e imagem de perfil. O Firebase Storage é utilizado para o armazenamento das imagens de perfil enviadas pelos usuários, vinculadas ao documento de cada um por meio de sua identificação única, cada alteração realizada, seja no nome, senha ou foto é automaticamente refletida no Firestore, garantindo consistência e centralização dos dados.

O ponto de acesso inicial ao perfil ocorre por meio do componente Header, que disponibiliza a imagem de perfil do usuário (caso não seja adicionada será uma imagem padrão) e o botão para acessar o menu lateral representados na Figura 17.

Figura 17 - Telas para acesso ao perfil de usuário.



Fonte: Elaborado pelo autor (2025).

Essa implementação realizada no componente Menu permite a abertura da tela de perfil ou a execução do processo de logout que ao ser selecionada, chama o método *signOut* do Firebase Authentication, garantindo que os dados locais sejam removidos e o usuário redirecionado para a tela de login (Código-fonte 6).

Código-fonte 6 - Componente Menu para gerenciar logout ou acesso ao perfil

```
const Menu = ({abrirPerfilMethod, goToLogin}) =>{
    const auth = getAuth(app);
    const abrePerfil = ()=>{
        abrirPerfilMethod();
    }
    const signOutPress = () =>{
        signOut(auth).then( async () => {
            await AsyncStorage.clear();
            goToLogin();
        }).catch((error) => {
        });
    }
```

No *PerfilComponent* concentram-se as operações de alteração e atualização de dados cadastrais, essas informações carregam inicialmente os dados salvos no *AsyncStorage*, na renderização de tela desse componente o usuário pode estar realizando as seguintes ações:

- Alterar imagem de perfil: Utilizando a biblioteca *expo-image-picker* para selecionar uma nova foto na galeria, após a escolha o arquivo é enviado ao Firebase Storage, sendo gerada uma URL de acesso público que é salva no Firestore para o ID do usuário em questão, além de substituir a informação em memória local.
- Modificar o nome de usuário: Através de uma função é possível estar alterando as informações de nome e sobrenome, isso ocorre tanto no Firestore quanto no *AsyncStorage*, assegurando que a alteração seja refletida em futuras sessões.
- Redefinir senha: Com uso de uma função para alterar a senha, a nova senha escolhida pelo utilitário validade é definida, reforçando a segurança dos acessos e atualizando a nova senha no Firebase para logins futuros.

Esse processo evidencia a sincronização entre banco de dados remoto e armazenamento local, permitindo que o usuário tenha seus dados imediatamente

refletidos na interface. Além disso, ao armazenar a foto e o nome no Firestore, essas informações podem ser utilizadas em diferentes partes do aplicativo.

Figura 18 - Tela de alteração de dados do usuário.

Fonte: Elaborado pelo autor (2025).

Dessa forma, a modelagem do perfil de usuário e a possibilidade de alteração de dados cadastrais (Figura 18) não apenas fortalecem a experiência personalizada, mas também garantem que o sistema mantenha a consistência das informações em diferentes dispositivos e sessões de uso, visando criar um ambiente dinâmico e centrado no usuário.

5.2 Funcionalidades de interface e navegação interna

O conceito central por trás da navegação do aplicativo foi organizar as telas em dois níveis distintos, permitindo maior controle e clareza na transição entre funcionalidades, sendo esses a navegação principal (externa) e navegação interna (sub-telas).

A navegação principal do aplicativo é gerenciada pelo componente *MyNavigation*, que atua como o núcleo do sistema de roteamento sendo responsável por registrar todas as telas principais disponíveis, organizar a hierarquia e disponibilizar o objeto navigation (Código-fonte 7), possibilitando que cada tela execute comandos de avançar, retornar ou acessar diretamente outra tela específica.

O *MyNavigation* assegura que qualquer tela registrada possa ser acessada a partir de outra, promovendo consistência e flexibilidade no fluxo do aplicativo, fornecendo a estrutura necessária para que outros componentes, como

o *MainComponent* e suas respectivas sub-telas sejam exibidas de forma correta dentro da hierarquia de navegação.

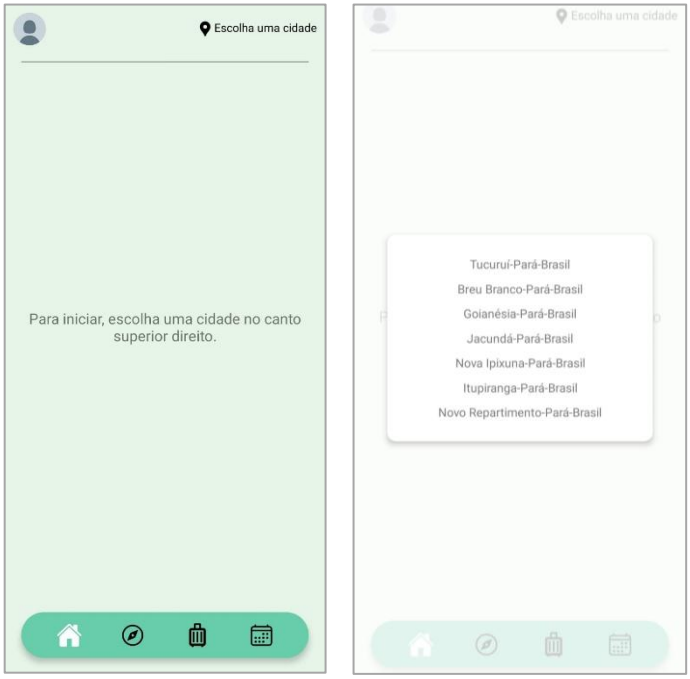
Código-fonte 7 - Pilha de navegação principal do componente MyNavigation

```
return (
  <Stack.Navigator
    initialRouteName={initialRoute}
    screenOptions={{
      header: (props) => <CustomNavigationBar {...props} />,
    }}
  >
    <Stack.Screen name="Login"
      options={{ headerShown: true, headerTitle: "Login" }}
      component={LoginScreen} />
    <Stack.Screen name="MainComponent"
      options={{ headerShown: false }}
      component={MainComponent} />
    <Stack.Screen name="EstabelecimentoComponent"
      options={{ headerShown: true, headerTitle: "Estabelecimento" }}
      component={EstabelecimentoComponent} />
    <Stack.Screen name="Mapa"
      options={{ headerShown: true, headerTitle: "Mapa" }}
      component={Mapa} />
    <Stack.Screen name="EventosComponent"
      options={{ headerShown: true, headerTitle: "Eventos" }}
      component={EventosComponent} />
    <Stack.Screen name="Registration"
      options={{ headerShown: true, headerTitle: "Registro de usuário"
}}
      component={RegistrationScreen} />
    <Stack.Screen name="Perfil"
```

```
      options={{ headerShown: true, headerTitle: "Perfil" }}
      component={PerfilComponent} />
    <Stack.Screen name="Sugestoes"
      options={{ headerShown: true, headerTitle: "Sugestões" }}
      component={SugestoesComponent} />
    <Stack.Screen name="GuiasTuristicos"
      options={{ headerShown: true, headerTitle: "Guias Turísticos" }}
      component={GuiasTuristicosComponent} />
  </Stack.Navigator>
);
```

Foi desenvolvido um componente denominado Header (Código-fonte 8) que desempenha um papel essencial na organização e na usabilidade do aplicativo, servindo como ponto de referência para o usuário em todas as telas, apresentando o título da seção atual e disponibilizando botões de ação localizados na parte superior da interface durante a navegação principal (Figura 19), contendo também ícones que podem direcionar para funções específicas orientando o usuário durante processos utilização do sistema, servindo de apoio à navegação e gestão de conteúdo, garantindo que o utilizador consiga transitar de forma simples entre os recursos disponibilizados pelo sistema além de servir como guia pois disponibiliza informações no topo de cada tela.

Figura 19 - Componente Header na parte superior da tela para auxiliar na navegação.



Fonte: Elaborado pelo autor (2025).

Além disso, o Header também desempenha papel fundamental na troca entre cidades dentro do aplicativo, permitindo que o usuário adapte a experiência de navegação de acordo com sua localização ou interesse, sendo implementada por meio de um modal (janela que aparece por cima da interface principal do aplicativo) para seleção de cidades, acessado quando o usuário clica sobre o botão para selecionar a cidade.

Ao escolher um município cadastrado no aplicativo a função *escolheCidade* é chamada, onde o nome selecionado é salvo no *AsyncStorage* e a coleção que contém as informações da seleção é acionada. O nome da cidade deve estar exatamente igual ao criado na coleção do projeto no Firestore para que as informações da cidade selecionada carreguem corretamente.

Código-fonte 8 - Componente Header para seleção de cidades na tela inicial

```
1 const Header = ({ image, nomeColecao, mudaColecao, abrirPerfilMethod,
  goToLogin }) => {
2     const [modalMenu, setModalMenu] = useState(false);
3     const [modalLocal, setModalLocal] = useState(false);
4     const cidades = [
5         “Tucuruí-Pará-Brasil”
6         “Breu Branco-Pará-Brasil”
7         “Goianésia-Pará-Brasil”
```

```
        "Jacundá-Pará-Brasil"
        "Nova Ipixuna-Pará-Brasil"
        "Itupiranga-Pará-Brasil"
        "Novo Repartimento-Pará-Brasil"
    ];
    const escolheCidade = async (cidade) => {
        await AsyncStorage.setItem("nomeColecao", cidade);
        console.log("Header cidade="+cidade);
        mudaColecao(cidade);
        setModalLocal(false);
    }
```

A navegação interna é gerenciada pelo *MainComponent* (Código-fonte 9) que é o componente responsável por reunir e exibir seções principais do aplicativo, nesse modo de transição não ocorre mudança de pilha, mas sim alternância entre conteúdos dentro de uma mesma tela principal, o componente mantém um estado interno denominado *currentScreen*, que armazena qual sub-tela está ativa no momento, cada vez que o usuário interage com a seleção de tela, o valor de currentScreen é atualizado e a interface correspondente é renderizada dinamicamente.

O *MainComponent* concentra o gerenciamento de estados através de *hooks* como *useState* (armazena um valor e atualiza a tela sempre que esse valor mudar*)* e useEffect, o que garante que as informações sejam processadas de maneira integrada com os demais dados da aplicação, essa estratégia não apenas simplifica a comunicação entre o banco e a interface, mas também assegura consistência na exibição das informações, já que todas as telas dependem das variáveis globais controladas por esse componente que não se limita a uma função de busca, mas atua como um mediador entre o Firestore e os componentes visuais, controlando a forma como as sugestões são tratadas, armazenadas e disponibilizadas no fluxo de navegação do aplicativo.

Código-fonte 9 - Componente de gerenciamento de telas MainComponent

```
const renderScreen = () => {
  switch (currentScreen) {
    case 'Home':
      return(
        <Home
          sugestoes={sugestoes}
          user={user}
```

```
          localUser={localUser}
          nomeColecao={nomeColecao}
          navigation={navigation}
          mudaColecao={mudaColecao}
          abrirPerfilMethod={abrirPerfil}
          goToLogin={goToLogin}
          estabelecimentos={estabelecimentos}
        />
      );
    case 'Explorar':
      return (
        <ExplorarRoute
          categorias={categorias}
          estabelecimentos={estabelecimentos}
          localUser={localUser}
          navigation={navigation}
        />
      );
    case 'Eventos':
      return <EventosRoute eventos={eventos} localUser={localUser} />;
    case "Roteiros":
      return (
        <Roteiros
          roteiros={roteirosTuristicos}
        />
      );
    default:
      return null;
  }
};
```

A partir desse estado definido pelo usuário sendo selecionado no currentScren, o aplicativo consegue alternar dinamicamente entre diferentes sub-telas como Home, *ExplorarRoute*, EventosRoute e Roteiros, exibindo o conteúdo correspondente a cada um desses componentes de acordo com a escolha do usuário, o *MainComponent* possui sub-telas, que são renderizados da seguinte forma:

- *HomeRoute*: Nesta tela são exibidas informações de destaque, sugestões e atalhos para as principais funcionalidades do app, servindo como um painel inicial de navegação. A *HomeRoute* integra-se diretamente com o sistema de

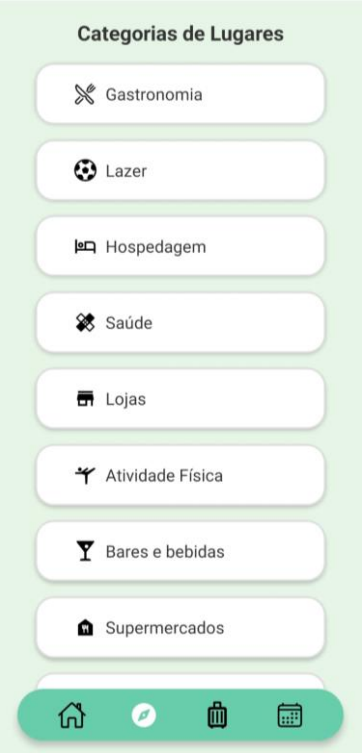
navegação interna, permitindo que o usuário acesse outras seções, como *ExplorarRoute*, EventosRoute e Roteiros (Figura 20).

Figura 20 - Tela inicial de Sugestões do aplicativo.

Fonte: Elaborado pelo autor (2025).

- *ExplorarRoute*: Nessa tela os itens são organizados em uma lista de categorias que após selecionada apresenta ao usuário cards relacionado a seleção. Esse componente recebe dados do *MainComponent* como a lista de estabelecimentos a ser amostrada ao utilizador após seleção de um card, utiliza componentes auxiliares para renderizar cada item de forma estruturada (Figura 21).

Figura 21 - Tela de categoria de lugares.



Fonte: Elaborado pelo autor (2025).

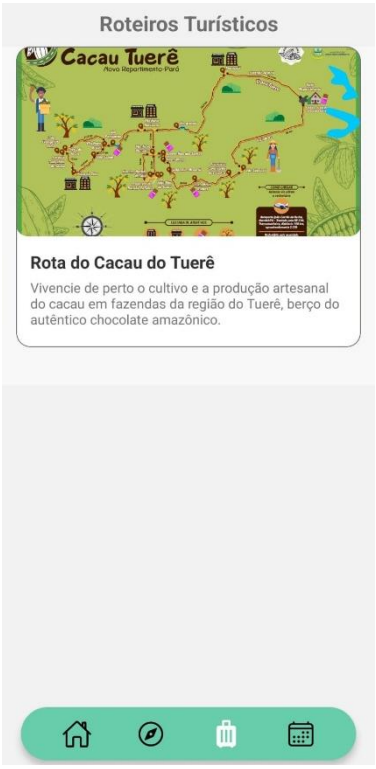
- EventosRoute: Cada evento é apresentado em um card interativo, que direciona o usuário à tela de detalhes de evento ao ser clicado. Informações preliminares so evento são mostradas ao usuário em uma lista animada (Figura 22).

Figura 22 - Tela de eventos locais.

Fonte: Elaborado pelo autor (2025).

- RoteirosRoute: Responsável por renderizar uma lista de cards contendo informações preliminares dos roteiros turisticos disponíveis para a localidade, recebe essa lista do banco de dados e exibe ao usuário (Figura 23).

Figura 23 - Tela de Roteiros Turísticos.



Fonte: Elaborado pelo autor (2025).

O *MainComponent* utiliza para a navegação e interação visual do aplicativo um menu flutuante (Figura 24), implementado pelo componente chamado *FloatingMenu*. Esse menu de navegação segue os princípios já apresentados, indicando claramente ao usuário em qual tela ele se encontra, os botões apresentam ícones que alteram sua aparência e cor de acordo com a tela ativa, oferecendo feedback visual imediato e tornando a experiência mais intuitiva.

Figura 24 - Menu flutuante nas telas de navegação princial do sistema.

Fonte: Elaborado pelo autor (2025).

O menu de navegação é posicionado de forma flutuante e fixa na parte inferior da tela, garantindo que o usuário tenha acesso rápido a todas as seções principais do aplicativo. Dessa forma o FloatingMenu se integra à navegação principal do

aplicativo, permitindo alternância entre as telas sem depender da pilha de navegação principal controlada pelo *MyNavigation*. Ele se configura como um elemento central para a experiência do sistema visando facilitar a exploração de funcionalidades.

Toda a navegação do aplicativo foi projetada e implementada com foco na clareza, usabilidade e facilidade de manuseio pelo usuário. A estrutura de navegação apresenta rotas e ícones autoexplicativos, hierarquia entre telas e feedback visual que orienta o usuário em cada ação. Essas decisões de design visam reduzir a curva de aprendizagem, garantir uma interação intuitiva e permitir que o usuário compreenda rapidamente o funcionamento da barra de navegação e dos fluxos internos da interface.

5.3 Estrutura e exibição dos conteúdos informacionais

5.3.1 Implementação da interface de detalhes de sugestões

A interface responsável por apresentar sugestões ao usuário foi estruturada como o componente *SugestoesComponent* que possui trecho mostrado no Código-fonte 10, projetado para exibir uma lista de locais recomendados conforme localização selecionada pelo utilizador dentro do aplicativo representados na Figura 25. O objetivo principal desse componente é fornecer uma navegação intuitiva e rápida para quaisquer card de sugestão selecionado pelo usuário.

Antes de apresentar qualquer informação ao usuário, o componente executa um processamento interno responsável por montar dinamicamente a lista final de sugestões, selecionando apenas aqueles cujo identificador está presente na lista de sugestões retornada pelo banco de dados. Essa estratégia almeja evitar renderizações desnecessárias e manter o componente eficiente ao trabalhar somente com os dados relevantes para o utilizador do sistema.

Código-fonte 10 - Cards interativos do SugestoesComponent

```
1 <FlatList
2   style={{ marginBottom: 90, paddingHorizontal:10,
3     paddingBottom: 10, marginTop: 20 }}
4   data={sugestoes!==null && sugestoes!==undefined ?
5     buildObjectsSugestoes(sugestoes): null}
6   keyExtractor={(_, index)=> index}
7   renderItem={(i)=>{
8     const item = i.item;
```

```
 9     console.log("C=SugestoesComponent item=", item)
10     return(
11       <Card onPress={()=>clickCard(item.id)} mode="outlined"
12       style={styles.estiloCard}>
13           {item.fotos && item.fotos.length > 0 && (
14               <Card.Cover style={styles.estiloCardCover}
15               source={{ uri: item.fotos[0] }} />
16           )}
17           <View style={styles.cardContent}>
18               <Text style={styles.estiloTitulo}>
19                 {item.nome}</Text>
20               <Text style={styles.estiloDescricao}>
21                 {item.descricao}</Text>
22               {localUser!==undefined && localUser!==null &&
23               item.localizacao !==undefined && item.localizacao
!==null ?
24                 <Text style={styles.estiloDistancia}>
25                     A  {getDistanceBetweenPoints(localUser.latitude,
26                     localUser.longitude, item.localizacao.latitude,
27                     item.localizacao.longitude, "kilometers")} KM de
você
28                 </Text>
29                 : null
30               }
31           </View>
32       </Card>
33     )
34   }}
35 />
```

Cada sugestão é apresentada ao usuário por meio de um Card, estruturado de forma semelhante aos itens exibidos no componente *ExplorarRoute*, essa padronização visual reforça a consistência da interface e evidencia o uso de reciclagem de componentes. Ao interagir com qualquer item sugerido, o usuário é imediatamente direcionado para a tela de detalhes do estabelecimento selecionado, mantendo o fluxo de navegação intuitivo e coerente com o restante da aplicação.

Figura 25 - Renderização da lista de sugestões na interface do usuário.

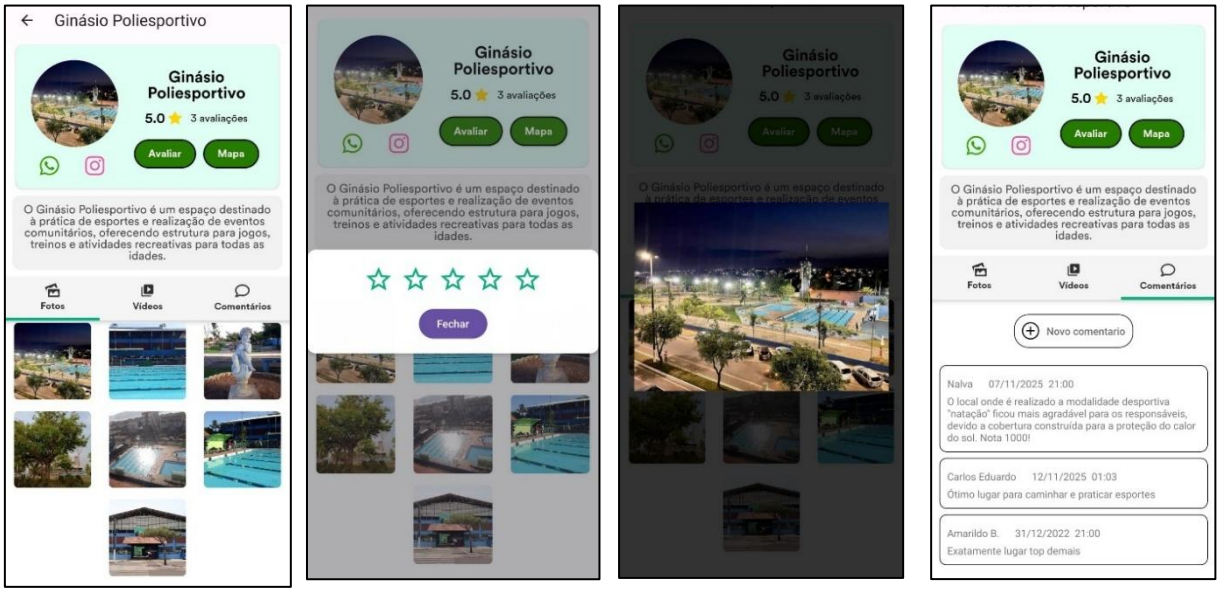
Fonte: Elaborado pelo autor (2025).

Dessa forma, o *SugestoesComponent* foi projetado para ser leve, reutilizável e de fácil integração com outras telas do aplicativo. A combinação entre filtragem eficiente, apresentação visual organizada e interação direta oferece uma experiência fluida, auxiliando o usuário a encontrar rapidamente locais de interesse e fortalecendo o papel da aplicação como ferramenta de orientação e descoberta de serviços e atrações disponíveis na região.

5.3.2 Implementação da interface de detalhes de estabelecimentos

A interface desenvolvida para exibição dos detalhes dos estabelecimentos foi chamada de *EstabelecimentoComponent* (Figura 26), formulada com o objetivo de apresentar, em uma única tela, todas as informações essenciais relacionadas ao estabelecimento selecionado pelo usuário, para alcançar essa organização a implementação seguiu o princípio da componentização no qual cada parte funcional da interface é construída como um componente independente, sendo posteriormente integrado ao componente principal.

Figura 26 - Telas de detalhes de estabelecimentos, interface e componentes visuais.



Fonte: Elaborado pelo autor (2025).

A componentização é aplicada de forma estratégica fazendo uso de componentes internos para manipular conteúdos específicos, como imagens, vídeos e comentários (Tabela 3), sendo importados e renderizados conforme a aba selecionada pelo usuário.

Tabela 3 - Componentes utilizados para exibir informações em estabelecimentos.

Imagens	Organiza e exibe as fotos do estabelecimento.
Videos	Apresenta miniaturas com reprodução em tela ampliada.
Comentarios	Lida com a visualização e o registro de novas opiniões no Firestore.

Fonte: Elaborado pelo autor.

Cada componente apresentado possui responsabilidades claramente definidas, dessa forma o *EstabelecimentoComponent* atua como um controlador, chamando e exibindo cada elemento conforme necessário, enquanto os componentes secundários tratam os dados e a renderização específica de suas funções.

Essa tela também exibe botões que direcionam o utilizador do sistema para outros aplicativos ou site, como é o caso dos botões de Whatsapp, Instagram e Mapa que direcionam o usuário para apps externos, visando facilidade para entrar em contato ou obter informações a respeito do estabelecimento. Também foi implementada uma funcionalidade de avaliação que exibe a média das avaliações do local além de permitir que o usuário dê uma nota para o local.

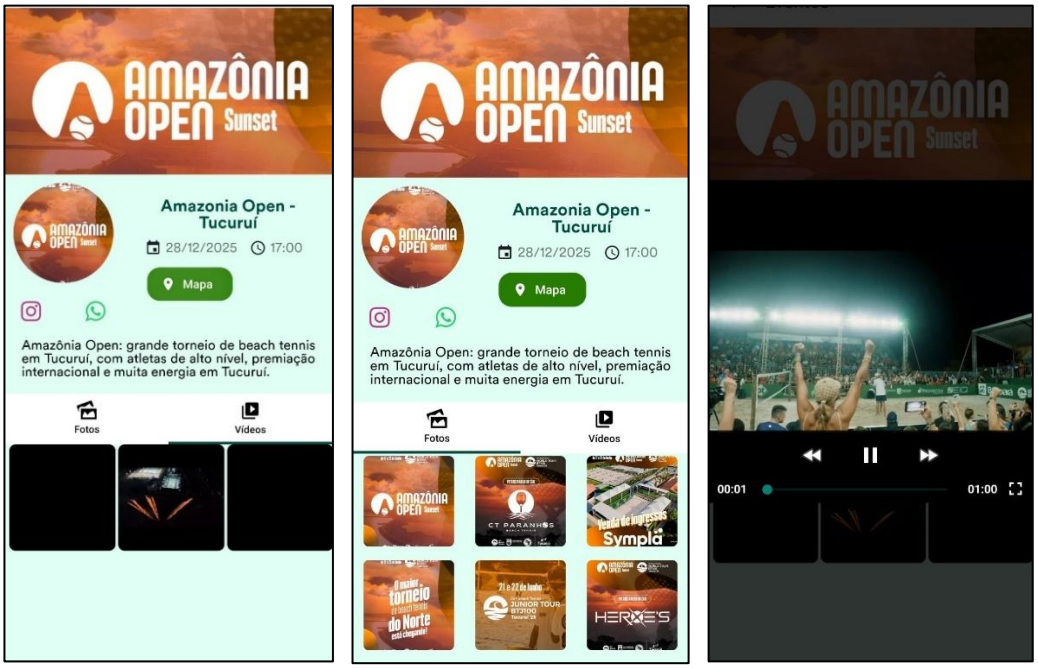
A modelagem adotada, aliada à implementação no código tem como objetivo viabilizar a escalabilidade e usabilidade do sistema, assegurando que novas informações possam ser incorporadas sem comprometer a consistência dos dados já existentes, assim o aplicativo visa tornar-se uma ferramenta dinâmica, capaz de fornecer aos usuários uma experiência fluida e confiável na busca por serviços e locais de interesse.

O botão de Mapa presente nessa tela direciona o usuário para a tela do componente de mapa, que visa direcionar as informações de estabelecimento e usuário para o Google Maps.

5.3.3 Implementação da interface de detalhes de eventos

A interface responsável pela exibição detalhada dos eventos foi implementada através do componente denominado *EventosComponent* (Figura 27), sendo projetado para integrar-se à tela destinada à apresentação das informações completas de cada evento, fornecendo ao usuário meios de interação direta. Seu desenvolvimento tomou como referência o componente previamente utilizado para exibir detalhes de estabelecimentos, adotando os mesmos princípios de organização lógica e reaproveitamento de elementos funcionais.

Figura 27 - Tela de detalhes de eventos, interface e elementos visuais.



Fonte: Elaborado pelo autor (2025).

A construção do *EventosComponent* seguiu uma abordagem orientada à componentização, possibilitando o reaproveitamento de módulos já estabelecidos no

projeto, como os componentes responsáveis por renderizar vídeos e imagens. Essa estratégia promoveu a reciclagem de código, reduzindo duplicações, mantendo padrões visuais, com objetivo de garantir uma manutenção mais simples e eficiente ao longo do ciclo de vida da aplicação, a lógica interna foi adaptada para atender à natureza específica dos eventos, mas preservou a estrutura modular já consolidada.

Durante a definição das funcionalidades essenciais dessa interface, concluiu-se que não seria necessária a inclusão de seções de comentários ou avaliações por parte dos usuários, pois os eventos cadastrados são voltados para datas futuras ou estão em andamento, não há informações prévias suficientes para formar opiniões consistentes, o que tornaria tais módulos pouco relevantes. Em vez disso, optou-se por priorizar informações consideradas fundamentais, como os meios de contato disponibilizados pela organização da atração, além da apresentação da data, horário e demais detalhes essenciais para o planejamento e participação do público.

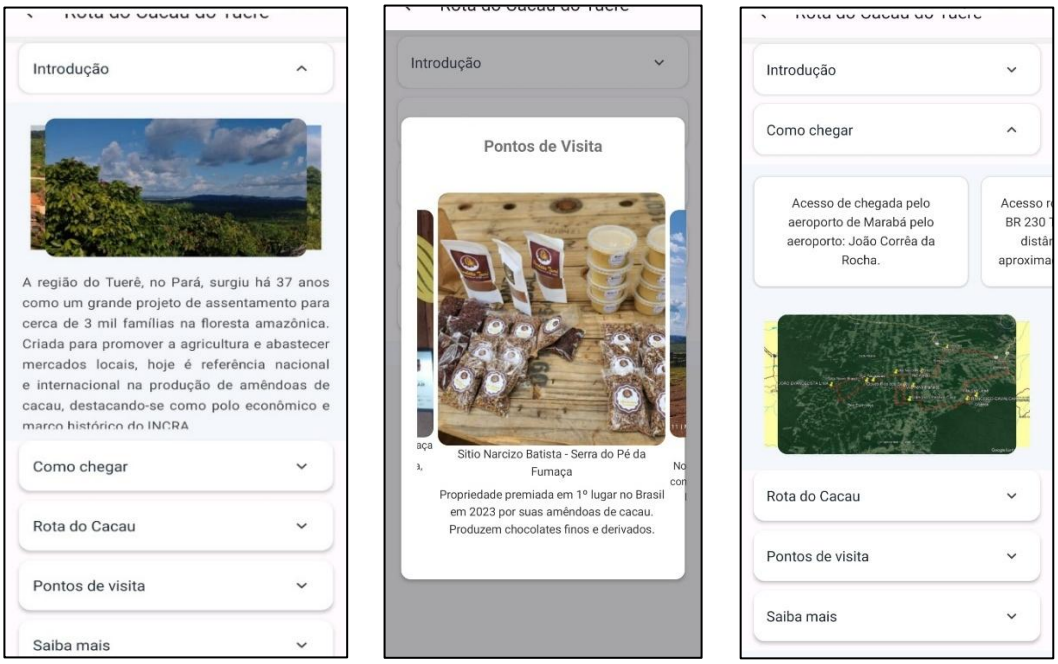
A interface também foi projetada para oferecer recursos de interação direta com os organizadores, incorporando botões que redirecionam o usuário para o WhatsApp e para o Instagram do responsável pelo evento. Essa funcionalidade amplia a possibilidade de comunicação imediata, favorecendo o engajamento e permitindo a resolução rápida de dúvidas. Por fim, o *EventosComponent* integra um mecanismo de abertura do mapa de navegação a partir das coordenadas registradas para o local do evento, garantindo ao usuário acesso facilitado a rotas e orientações geográficas.

Dessa forma, a interface não apenas apresenta informações estáticas, mas também disponibiliza ferramentas que apoiam o deslocamento, a comunicação e a experiência completa do usuário.

5.3.4 Implementação da interface de detalhes de Roteiros Turísticos

A interface responsável por exibir os detalhes dos roteiros turísticos foi implementada no componente denominado *GuiasTuristicosComponent* (Figura 28), sendo estruturado para apresentar informações completas e organizadas sobre cada roteiro, utilizando um modelo padronizado que permite a exibição de dados entre diferentes pontos turísticos. A construção da interface partiu de uma estrutura modular, baseada em carrosséis de imagens e elementos de navegação, garantindo ao usuário acesso intuitivo e simplificado às informações essenciais de cada rota turística.

Figura 28 - Telas de detalhes dos Roteiros Turísticos e elementos visuais..



Fonte: Elaborado pelo autor (2025).

A estrutura inicial do componente foi desenvolvida utilizando a lógica de componentização e controle de estados individuais, permitindo que cada seção de conteúdo funcione de forma independente e responsiva. A aplicação faz uso de estados internos para controlar conteúdos expansíveis, enquanto exibe elementos visuais necessários para manter a interface coerente com o restante da aplicação. Além disso, o componente incorpora carrosséis dinâmicos que permite a apresentação de imagens de forma fluida e interativa, visando reforçar o aspecto visual da experiência turística.

O conteúdo é exibido de forma hierárquica, onde o usuário pode acessar explicações introdutórias, orientações de deslocamento, observações extras e descrições completas dos pontos visitados ao longo do roteiro. Um modal adicional foi implementado especificamente para visualizar os pontos de visita, apresentando imagens e descrições detalhadas em um ambiente separado.

O desenvolvimento desse componente teve como foco oferecer ao usuário uma interface informativa, acessível e tecnicamente eficiente. A estrutura adotada tem por objetivo facilitar o consumo dos dados turísticos cedidos pela LagoTur, permitindo que visitantes compreendam como chegar à localidade. A combinação de elementos utilizados visa garantir que o usuário navegue pelas informações sem sobrecarga visual, preservando a estética e o desempenho da aplicação.

A IGR LagoTur forneceu ao projeto o material oficial da Rota do Cacau do Tuerê, um roteiro turístico previamente estruturado, documentado e validado pela própria instituição. Esse conteúdo serviu como referência direta para definir a organização interna dos roteiros dentro do aplicativo, orientando a forma de apresentação das informações, a hierarquia dos dados, a descrição dos pontos turísticos e o padrão narrativo utilizado.

A disponibilização de um roteiro real produzido, revisado e autenticado por um órgão oficial de turismo conferiu maior credibilidade ao protótipo e garantiu que a aplicação fosse construída a partir de parâmetros concretos, e não apenas hipotéticos no que se refere a esse roteiro cadastrado para a cidade de Novo Repartimento. Esse diferencial faz uma distinção do LagoTur para os aplicativos de turismo da região, pois oferece um roteiro turístico pronto, fortalecendo a utilidade prática da ferramenta e sua aderência às necessidades reais dos visitantes e gestores.

A arquitetura aplicada na lógica do componente e seu design tem como principal objetivo garantir que os dados sejam apresentados com consistência, além de facilitar futuras expansões e adaptações. Dessa forma, o componente disponibiliza recursos visuais e textuais que tornam a descoberta dos roteiros mais clara e funcional.

5.4 Geolocalização e tratamento de localização no aplicativo

A geolocalização desempenha um papel central no funcionamento do aplicativo, uma vez que possibilita a personalização da experiência do usuário a partir de sua posição atual. Por meio desse recurso o sistema consegue identificar a localização geográfica do dispositivo (latitude e longitude) e oferecer informações contextuais, além de calcular distâncias e traçar rotas até os destinos de interesse. Para a implementação, foi utilizada a biblioteca de localização do Expo (*expo-location*), a qual permite realizar a captura das coordenadas do dispositivo (Figura 42), visando garantir a interatividade e a relevância dos conteúdos apresentados.

A parte do sistema responsável por obter e tratar a localização do usuário está concentrada no *MainComponent* que faz o acionamento de uma função que solicita a permissão do usuário para acessar a localização do dispositivo, se autorizado o sistema captura as coordenadas geográficas do dispositivo.

Os dados capturados são armazenados internamente na memória do dispositivo, utilizando uma variável chamada *location*, em seguida essas

informações são organizadas em um objeto chamado *localUser* que reúne as informações de latitude e longitude do usuário e é repassado para outras partes do sistema, representados no Código-fonte 11. Dessa forma, cada tela do aplicativo consegue usar a localização de maneira adequada ao seu contexto seja para exibir lugares próximos, calcular distâncias ou sugerir opções personalizadas.

Código-fonte 11 - Função para acessar a localização do dispositivo

```
const funcaoLocation = async () => {
  let { status } = await Location.requestForegroundPermissionsAsync();
  if (status !== 'granted') {
    return;
  }
  let location = await Location.getCurrentPositionAsync({});
  setLocation(location);
};
```

Para realizar cálculos entre pontos distintos foi criada uma função localizada no componente *ExplorarRoute* que desempenha um papel essencial na integração da geolocalização ao aplicativo (Código-fonte 12), permitindo calcular a distância aproximada entre o usuário e os locais cadastrados no sistema, recebendo como parâmetros as coordenadas geográficas de dois pontos distintos (posição do dispositivo e lugar desejado). Essa função aplica a fórmula matemática baseada na Lei dos Cossenos Esférica que transforma os dados para determinar a distância real entre os pontos no globo terrestre, o resultado pode ser retornado em milhas ou convertido em quilômetros tornando a informação mais acessível ao usuário final.

Código-fonte 12 - Função que realiza o cálculo entre dois pontos distintos.

```
export function getDistanceBetweenPoints(latitude1, longitude1,
latitude2, longitude2, unit = 'miles') {
    let theta = longitude1 - longitude2;
    let distance = 60 * 1.1515 * (180/Math.PI) * Math.acos(
        Math.sin(latitude1 * (Math.PI/180)) * Math.sin(latitude2 *
(Math.PI/180)) +
        Math.cos(latitude1 * (Math.PI/180)) * Math.cos(latitude2 *
(Math.PI/180)) * Math.cos(theta * (Math.PI/180))
    );

    if (unit === 'miles') {
```

```
9           return parseFloat(distance.toFixed(2));
10      } else if (unit === 'kilometers') {
11          return parseFloat((distance * 1.609344).toFixed(2));
12      }
13  }
```

Uma das principais formas de utilização da geolocalização ocorre em diferentes componentes do aplicativo, que fazem uso da posição atual do usuário para oferecer informações personalizadas e contextualizadas, entre os elementos que utilizam essas informações destacam-se:

- *SugestoesComponent*: Esse componente recebe um objeto com as coordenadas do usuário e, juntamente com os dados dos estabelecimentos armazenados no banco. A função é utilizada para calcular a distância aproximada entre o usuário e cada local, essas informações são exibidas diretamente na interface, permitindo que o usuário visualize em tempo real a que distância cada estabelecimento se encontra de sua posição atual. É importante destacar que o componente *HomeRoute* se utiliza dessas informações geradas pelo *SugestoesComponent* para exibi-las de forma organizada ao usuário (Figura 29), podendo facilitar na escolha de locais mais próximos.

Figura 29 - Card com visualização da distância entre o usuário e o local visualizado.

Fonte: Elaborado pelo autor (2025).

- *ExplorarRoute*: A tela de exploração também se beneficia da geolocalização, utilizando as coordenadas do usuário para apresentar os estabelecimentos e pontos de interesse de forma organizada e contextualizada, os resultados exibidos na tela têm relevância direta com a posição atual do usuário (Figura 30).

Figura 30 - Tela de Explorar Estabelecimentos após selecionar categoria.

Fonte: Elaborado pelo autor (2025).

- EventosRoute: Os eventos utilizam a localização do dispositivo junto aos pontos de interesse para calcular a distância entre ambos. Esse processo emprega a função de calculo para determinar as distâncias aproximadas e apresentar essas informações de forma clara ao usuário final (Figura 31).

Figura 31 - Tela de eventos locais e distância do usuário até o local.

Fonte: Elaborado pelo autor (2025).

- Mapa: O componente Mapa recebe como parâmetros a localização do usuário e de um determinado estabelecimento ou ponto de interesse (Figura 32), com

base nesses dados ele calcula a distância entre os dois pontos utilizando a função getDistanceBetweenPoints e gera uma rota personalizada que pode ser aberta diretamente no Google Maps. Dessa forma, o usuário não apenas visualiza a proximidade do destino, mas também obtém instruções detalhadas para se deslocar até ele, tornando o planejamento da visita mais intuitivo e eficiente.

Figura 32 - Tela para abrir localização de destino no Google Maps do dispositivo.

Fonte: Elaborado pelo autor (2025).

De forma geral, todos os componentes que utilizam a geolocalização seguem uma mesma lógica pois capturam a posição atual do usuário, cruzam esses dados com as informações armazenadas no banco (estabelecimentos ou eventos), e por fim exibem o resultado de forma visual na interface. Essa integração garante que a localização seja utilizada tanto para fins informativos (mostrar distâncias) quanto para fins de navegação (abrir rotas).

Em síntese, a geolocalização atua como um elemento central na arquitetura do aplicativo, permitindo que todas as funcionalidades relacionadas à proximidade e à navegação sejam realizadas. Ao integrar a posição atual do usuário com os dados de estabelecimentos, eventos e pontos de interesse, o sistema consegue fornecer informações contextuais, calcular distâncias em tempo real e gerar rotas detalhadas, visando tornar a interação mais intuitiva e prática. Essa abordagem tem como objetivo melhorar a experiência do usuário, tornando-a mais dinâmica, relevante e evidenciando como a combinação entre dados geográficos,

cálculos de distância e interface visual contribuindo para tornar o sistema capaz de adaptar suas respostas às necessidades e à localização de cada indivíduo.

5.5 Sistema de comentários e avaliações dos estabelecimentos

O sistema de comentários e avaliações implementado no aplicativo LagoTur constitui um dos elementos centrais de interação do usuário com o conteúdo turístico disponibilizado, permitindo que visitantes registrem opiniões, descrevam experiências e atribuam notas aos estabelecimentos, enriquecendo o ecossistema informacional da plataforma e promovendo um ambiente colaborativo. Essas interações constituem uma camada adicional de credibilidade e engajamento, já que os dados inseridos pelos usuários complementam as informações institucionais cadastradas no Firestore, dessa forma o fluxo de feedback transforma cada estabelecimento em um repositório de percepções públicas, contribuindo para uma visão mais realista e dinâmica da cena turística local.

A implementação desse sistema começa na classe *Comentario*, responsável por estruturar semanticamente cada comentário tendo como base os atributos fundamentais (nome do autor, texto e data) e fornecendo métodos utilitários, como a conversão da data do Firestore em um formato legível ao usuário. A presença de modos para tratar essas informações assegura que a camada de apresentação receba um objeto já preparado para exibição, evitando que lógica de formatação seja misturada à interface, garantindo que cada comentário seja um objeto autônomo e padronizado dentro da aplicação.

Os comentários são carregados dentro do aplicativo por meio da classe *Estabelecimento*, que converte as estruturas brutas recebidas do Firestore em objetos da classe *Comentario*, essa conversão automática ocorre no momento da criação do estabelecimento, quando o construtor percorre o mapa de comentários armazenado no banco e cria instâncias individuais para cada entrada. Além disso, essa classe também é responsável por processar as avaliações numéricas, calculando dinamicamente a média das notas atribuídas pelos usuários, esse processo garante que a média de avaliações exibidas na interface seja sempre atualizada.

A interação direta do usuário com o sistema ocorre no componente *Comentarios*, responsável tanto pela exibição quanto pela inserção de novos comentários, fazendo a captura o conteúdo digitado pelo usuário e enviando a

informação para o Firestore por meio de uma função para adicionar o novo comentário em sequência de acordo com a data, o Código-fonte 13 mostra como isso acontece.

Código-fonte 13 - Função para adicionar novo comentário em sequencia

```
async function adicionarComentarioSequencial(texto) {
    try{
        const cidade = await AsyncStorage.getItem("nomeColecao");
        const ref = doc(db, cidade, "Estabelecimentos");
        const snap = await getDoc(ref);
        const usuario = await AsyncStorage.getItem("user");
        if(usuario!==null) {
            let usuarioJson = JSON.parse(usuario);
            if (!snap.exists()) return;
            const date = Date.now();
            const timestamp = Timestamp.now();
            updateComentarios(texto, usuarioJson.nome, timestamp,
date);
            await updateDoc(ref, {
                [`${idEstabelecimento}.comentarios.${date}`]:
[usuarioJson.nome, texto, timestamp]
            });
        } else{
            console.error("usuário não encontrado")
        }
    }
    catch(e){
        console.error(e);
    }
}
```

Ao adicionar um novo comentário é utilizado o identificador único do estabelecimento e partir disso são criadas dinamicamente novas chaves no documento correspondente, mantendo todas as interações organizadas e associadas ao ID correto, representados na Figura 50. Juntamente a isso foi realizado a criação de um método para atualizar a interface localmente de imediato (*updateComentarios*), garantindo que o comentário recém-adicionado seja exibido sem necessidade de recarregar a página, por fim a utilização da memória local do dispositivo permite identificar o autor da interação.

Figura 33 - Aba de comentários na tela de detalhes de estabelecimentos

Fonte: Elaborado pelo autor (2025).

No componente principal de exibição das informações dos estabelecimentos, as abas de navegação organizam fotos, vídeos e comentários, integrando os elementos multimídia aos feedbacks produzidos pelos usuários.

No desenvolvimento da parte responsável pelas avaliações, optou-se pela manipulação dentro do componente principal de exibição das informações dos estabelecimentos, nessa etapa foi definido que a mesma iria ocorrer por estrelas, com utilização da biblioteca *react-native-star-rating-widget* que permite aos usuários atribuir notas ao item avaliado, juntamente com uma função responsável por enviar essas informações ao Firestore que armazena cada nota como um objeto contendo o nome do usuário e o valor dado, representado no Código-fonte 14.

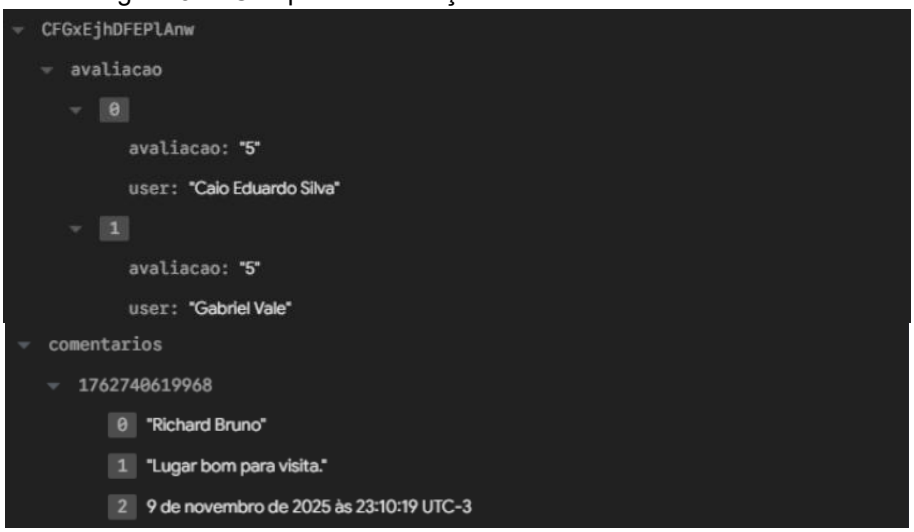
Código-fonte 14 - Função para inserir e atualizar avaliação de usuário

```
1 const atualizarAvaliacao = async () => {
2     let avaliacaoExistente = estabelecimento.avaliacao.find(obj =>
  obj.user === usuario.uid);
3     if (avaliacaoExistente) {
4         alert("Você já avaliou esse estabelecimento");
5     } else {
6         const estabelecimentoCopia = estabelecimento;
```

```
        await estabelecimentoCopia.avaliacao.push({ user:
usuario.nome, avaliacao: avaliacao });
        let idLocal = newLocal.id;
        try {
            await updateDoc(doc(db, cidade, "Estabelecimentos"), {
                [$[idLocal].avaliacao]: estabelecimentoCopia.avaliacao
            });
        } catch (e) {
        }
    }
    hideModal();
}
```

No Firestore, os comentários são salvos como pares chave–valor (maps), onde a chave é um número baseado em timestamp (data e hora que o usuário comentou) e o valor é uma lista contendo autor, texto e data (Figura 52), esse formato permite inserção sequencial, ordenação eficiente e identificação única de cada registro, tendo em vista que cada chave será diferente da próxima gerada. Já as avaliações são armazenadas em arrays (múltiplos valores em um único campo) de objetos, possibilitando cálculos agregados e análises de densidade das avaliações, essa modelagem também permite que o aplicativo recupere apenas os dados necessários no momento da exibição, otimizando o desempenho e reduzindo consultas redundantes.

Figura 34 - Campos de avaliação e comentários no Firestore.



Fonte: Elaborado pelo autor (2025).

No conjunto, o sistema de comentários e avaliações do LagoTur cria um ambiente de interação social, onde usuários não apenas consomem informações, mas

também contribuem para a formação contínua da base de conhecimento turístico da região. O fluxo foi projetado para ser responsivo, intuitivo e seguro, garantindo que cada ação do usuário resulte em uma interação fluida e tecnicamente consistente, essa funcionalidade reforça o compromisso do aplicativo com a construção participativa de conteúdo e com a melhoria constante da experiência dos visitantes.

5.6 Participação do projeto no II SIMTEC e contribuições para o desenvolvimento

Em setembro de 2024, o projeto foi apresentado no II SIMTEC – Simpósio de Tecnologia do IFPA Campus Tucuruí (Figura 53), ocasião em que a aplicação foi demonstrada em sua forma de protótipo inicial. Nessa etapa o desenvolvimento ainda se encontrava em fases preliminares, com diversos componentes estruturais, funcionais e conceituais em construção. Assim, muitas das funcionalidades planejadas, modelos de interação e soluções técnicas exibidas na apresentação passaram por revisões e aprimoramentos ao longo das etapas posteriores de desenvolvimento.

A participação no SIMTEC teve papel fundamental para a validação preliminar da proposta, pois permitiu apresentar a visão central do sistema, demonstrar sua motivação e confirmar a pertinência do projeto perante a comunidade acadêmica e profissionais da área. Mesmo com o protótipo em estágio inicial, a ideia principal da aplicação já estava consolidada, possibilitando receber críticas construtivas, sugestões e percepções técnicas que contribuíram diretamente para o avanço do trabalho.

Figura 35 - Apresentação do protótipo no II Simpósio de Técnologia.

Fonte: Elaborado pelo autor (2025).

Além disso, a apresentação possibilitou o fortalecimento do alinhamento com a Instância Governamental Regional LagoTur, reforçando a relevância social, turística e tecnológica da ferramenta. As discussões originadas no evento contribuíram para direcionar melhorias, redefinir prioridades e ajustar a abordagem de implementação, servindo como inspiração e base para a continuidade do desenvolvimento.

Dessa forma, a participação no SIMTEC configurou-se como uma etapa estratégica para a evolução do projeto, funcionando como espaço de validação, divulgação e refinamento conceitual, além de consolidar parcerias essenciais para o prosseguimento do trabalho.

5.7 Testes e validação do aplicativo LagoTur

A etapa de testes e validações do aplicativo foi fundamental para assegurar que o protótipo desenvolvido atendesse de forma eficaz aos requisitos funcionais, operacionais e de usabilidade estabelecidos no planejamento inicial. Durante esse processo buscou-se avaliar não apenas o desempenho técnico da aplicação mas também a experiência real de uso, verificando como usuários compreendem, navegam e interagem com os elementos da interface.

Para isso, foram conduzidos diferentes ciclos de testes, abrangendo desde inspeções internas realizadas no ambiente de desenvolvimento até experimentações práticas com usuários reais em dispositivos físicos. Essa abordagem progressiva

permitiu identificar comportamentos inesperados e validar a eficiência da arquitetura adotada, constituindo uma etapa determinante para a confirmação da viabilidade do aplicativo e para a preparação das próximas fases evolutivas do projeto.

5.7.1 Testes técnicos no ambiente de desenvolvimento

Os primeiros testes do LagoTur foram realizados diretamente no ambiente de desenvolvimento, utilizando o Expo Go para executar o aplicativo em tempo real em dispositivos físicos e emuladores, o que permitiu observar de forma imediata como as telas eram exibidas, se a navegação estava coerente e se os componentes respondiam corretamente às ações do usuário (como toques em cards, troca de abas e abertura de mapas externos). As alterações no código refletiam no sistema instantaneamente, o que facilitou o ajuste fino de layout, cores, textos, espaçamentos e comportamentos interativos ainda nas fases iniciais de construção do protótipo.

Os mecanismos de registro de atividades (*log*) disponíveis no JavaScript tiveram papel central no processo de depuração, permitindo registrar mensagens no console de desenvolvimento, exibindo valores de variáveis, respostas de funções e o estado de determinados componentes em pontos estratégicos da execução do código. No contexto do LagoTur, foram usadas para acompanhar o retorno das consultas ao Firebase, conferir se as listas da aplicação estavam sendo montadas corretamente, verificar se os dados de localização estavam chegando no formato esperado e identificar eventuais falhas silenciosas na comunicação entre componentes e com o banco de dados.

O uso sistemático desse recurso ajudou a entender o fluxo interno da aplicação, tendo em vista que as mensagens em diferentes trechos do código possibilitou mapear e detectar inconsistências no desenvolvimento da lógica do sistema.

O ciclo de testes e ajustes no ambiente de desenvolvimento, combinando execução via Expo Go e análise por meio de *logs*, foi essencial para estabilizar o protótipo antes de levá-lo para usuários reais, muitos problemas estruturais foram identificados e corrigidos nessa fase, evitando que chegassem à etapa de validação externa. Com isso, a aplicação pôde avançar para os testes práticos em dispositivos físicos com uma base mais consistente, reduzindo o número de falhas críticas e

permitindo que o foco dos testes com usuários se voltasse mais à experiência de uso do que a erros básicos de funcionamento.

5.7.2 Avaliação e testes de desempenho em diferentes dispositivos

Os testes em dispositivos físicos foram realizados após a estabilização das principais funcionalidades do protótipo, com o objetivo de avaliar o comportamento do aplicativo em condições reais de uso. Dessa forma optou-se por distribuir o aplicativo exclusivamente para smartphones Android, uma vez que o processo de instalação e distribuição nesse sistema é mais simples, rápido e menos burocrático do que em iOS, essa decisão permitiu focar na observação da experiência de uso, sem que questões de publicação em lojas oficiais se tornassem um impedimento técnico ou administrativo.

Para compor o grupo de testadores foram selecionadas 25 pessoas com perfis variados de residentes das cidades da região do Lago de Tucuruí, usuários que já haviam visitado mais de um município da área e outros com pouco ou nenhum contato prévio com a região. Essa diversidade foi importante para avaliar tanto a percepção de quem já conhece o território (e consegue julgar a utilidade prática do aplicativo em um contexto real de deslocamento) quanto de quem o utilizaria como apoio em uma primeira visita. Cada participante recebeu o aplicativo para uso em seu próprio dispositivo e foi instruído a explorar as funcionalidades de forma livre, simulando situações reais de consulta a estabelecimentos, eventos e roteiros turísticos.

Durante os testes foram observados aspectos como desempenho geral em diferentes modelos de smartphones, tempo de resposta da interface, adaptação a diferentes tamanhos de tela e precisão dos recursos de geolocalização, especialmente no cálculo de distâncias e na integração com mapas externos.

Ao final do período de uso, cada participante respondeu a um formulário estruturado no Google Forms, relatando sua experiência com o aplicativo, possibilitando coletar percepções sobre o funcionamento geral do sistema.

Embora o banco de dados utilizado nessa fase seja um banco de teste (construído a partir de informações públicas disponíveis na internet e ainda não validado oficialmente pela IGR LagoTur), as informações disponíveis cumpriram o objetivo de avaliar o protótipo do ponto de vista de usabilidade, comportamento técnico em dispositivos reais e aceitação inicial da proposta, os feedbacks obtidos

nessa etapa se tornaram insumos diretos para o refinamento do sistema e para o planejamento das próximas melhorias.

A realização dos testes práticos com usuários reais mostrou-se fundamental para validar o funcionamento do protótipo em um cenário próximo ao real, permitindo identificar tanto os pontos fortes quanto os aspectos que demandam aperfeiçoamento. A partir das percepções coletadas, foi possível reunir evidências concretas sobre a usabilidade, o desempenho em diferentes dispositivos e a pertinência das funcionalidades implementadas, gerando insumos objetivos para o planejamento de melhorias futuras, a IGR LagoTur avaliou positivamente esta versão preliminar, reconhecendo o protótipo como um passo consistente na direção da ferramenta que se espera disponibilizar aos turistas da região do lago, ao mesmo tempo em que reforçou a importância de dar continuidade ao ciclo de evolução do sistema a partir dos feedbacks obtidos.

6 RESULTADOS

A seção de Resultados apresenta os produtos finais obtidos ao longo do desenvolvimento do aplicativo LagoTur, evidenciando como os requisitos definidos na fase de planejamento foram transformados em funcionalidades operacionais. Nesta etapa são demonstrados o comportamento real da aplicação e a forma como os dados são processados e exibidos. Além disso são analisados o desempenho do sistema, a experiência do usuário (UX) e a correspondência entre as funcionalidades e os objetivos estabelecidos pela IGR LagoTur e pela Universidade Federal do Pará.

6.1 Análise de desempenho e resultados operacionais do app LagoTur

O desenvolvimento do sistema avançou de forma consistente desde as etapas iniciais de planejamento até a etapa de validação do protótipo inicial, as diretrizes propostas pela IGR LagoTur foram atendidas dentro da aplicação, tendo em vista os recursos disponibilizados pela organização, o que reflete diretamente nos resultados alcançados ao longo do projeto. A adoção de uma arquitetura apoiada no React Native para o desenvolvimento multiplataforma e no Firebase como solução em nuvem, contribuiu significativamente para a solidez operacional do sistema.

A adoção do React Native como tecnologia central demonstrou impacto direto nos resultados operacionais do aplicativo, especialmente no desempenho, na produtividade e na eficiência do desenvolvimento. O modelo de componentização do framework, amplamente utilizado permitiu reduzir significativamente o tempo de implementação e evitar duplicação de código. Essa arquitetura modular resultou em um desenvolvimento mais ágil, estruturado e escalável, refletindo diretamente na maturidade técnica alcançada pelo protótipo final do LagoTur.

O Firebase desempenhou papel central nos resultados operacionais do aplicativo, com utilização do Firestore que permitiu organizar dados de maneira escalável e expansível, respeitando as particularidades de cada município da região do lago. A sincronização em tempo real foram implementadas com sucesso visando uma melhor experiência para o usuário. A integração entre autenticação, armazenamento e banco de dados resultou em um ambiente de interação com responsividade, onde as informações são carregadas e atualizadas de forma frequente, contribuindo diretamente para a experiência de uso.

Apesar do desempenho satisfatório observado durante os testes, foram identificados pontos importantes para aprimoramento, especialmente no que diz respeito à ampliação de conteúdos e a uma construção real e de qualidade profissional para a base de dados, de modo a tornar a experiência ainda mais completa para o turista. Além disso foram apontados ajustes visuais relacionados à usabilidade (como melhorias na disposição de elementos, refinamento de contrastes e otimizações na legibilidade) bem como a necessidade de melhorias na implementação do sistema e adição de outras funcionalidades que podem estar vindo a ajudar os turistas enquanto estiverem visitando a região do lago.

A IGR LagoTur encontra-se atualmente em busca de financiamento para viabilizar a evolução estrutural e operacional do aplicativo, visando transformá-lo em uma ferramenta profissional e alinhada às necessidades do turismo regional. Os recursos almejados destinam-se à melhoria das interfaces de design, possibilitando uma experiência visual mais moderna, intuitiva e eficiente, além da implementação de um banco de dados robusto e profissional, capaz de suportar maior volume de informações e garantir maior confiabilidade.

Com essa infraestrutura ampliada, espera-se integrar ao sistema roteiros turísticos reais da localidade, bem como cadastrar de forma estruturada e contínua estabelecimentos, pontos de visitação e eventos oficiais das cidades atendidas. Esse investimento é fundamental para assegurar que o aplicativo alcance maturidade técnica e operacional, consolidando-se como uma plataforma oficial de suporte ao turismo na região do Lago de Tucuruí.

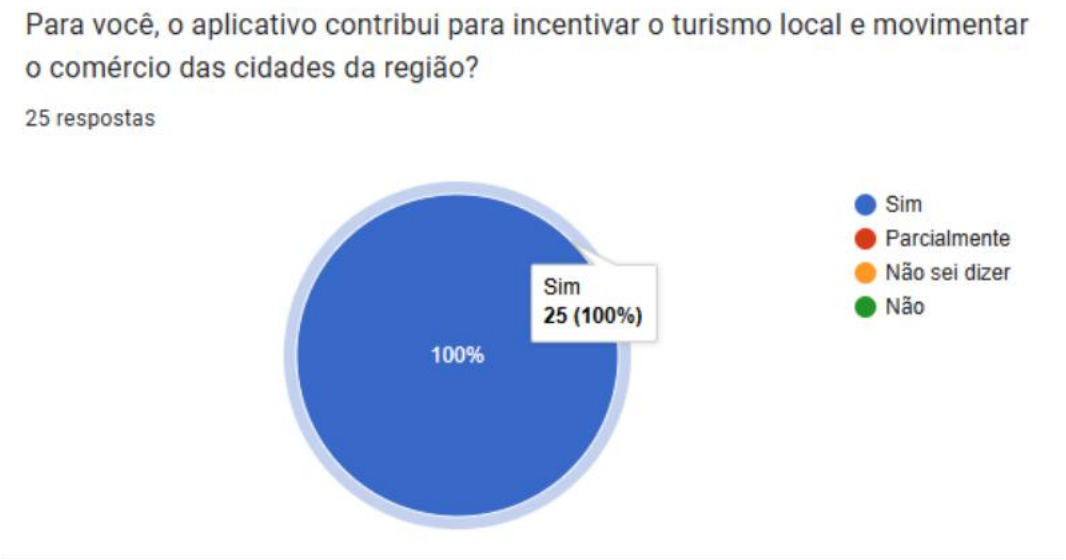
De forma geral, a aplicação do modelo tecnológico adotado permitiu a construção de um protótipo funcional, coerente com os requisitos planejados e alinhado às expectativas da instituição solicitante. Os resultados obtidos até esta etapa confirmam que as escolhas arquiteturais, as tecnologias empregadas e o direcionamento metodológico foram adequados para o escopo do projeto, estabelecendo uma base sólida para as próximas fases de refinamento, expansão de funcionalidades e evolução contínua do aplicativo.

6.2 Resultados de usabilidade

Os testes práticos com usuários reais foram realizados a partir do uso do protótipo do aplicativo LagoTur, seguido do preenchimento de um formulário de avaliação de usabilidade. De forma geral os 25 participantes consideraram a proposta

do sistema altamente relevante para o apoio ao turismo na região do Lago de Tucuruí, indicando que a centralização de informações sobre estabelecimentos, eventos e roteiros em um único ambiente atende a uma necessidade real de visitantes e moradores, fazendo com que os resultados de compreensão e usabilidade da ferramenta estejam alinhados as diretrizes passadas em conjunto com a IGR Lago Tur (Gráfico 1).

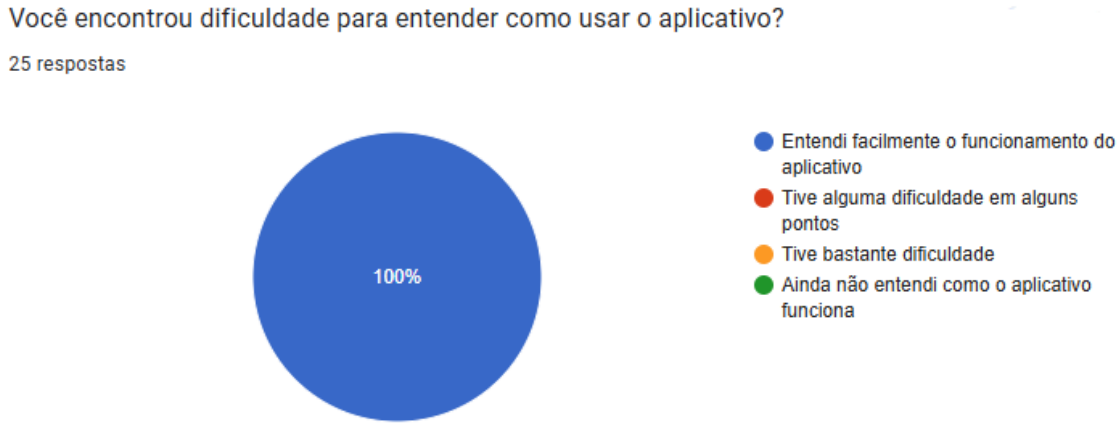
Gráfico 1 - Percentual de usabilidade da aplicação de acordo com usuários dos testes.



Fonte: Elaborado pelo autor (2025).

A maioria dos usuários relatou não ter encontrado dificuldades para compreender o funcionamento do sistema (Gráfico 2), o que indica que os fluxos de interação e a estrutura dos módulos foram adequadamente assimilados. Isso se torna importante tendo em vista que a facilidade na implementação e facilidade de compreensão do utilitário do sistema foi um objetivo esperado desde o inicio do projeto.

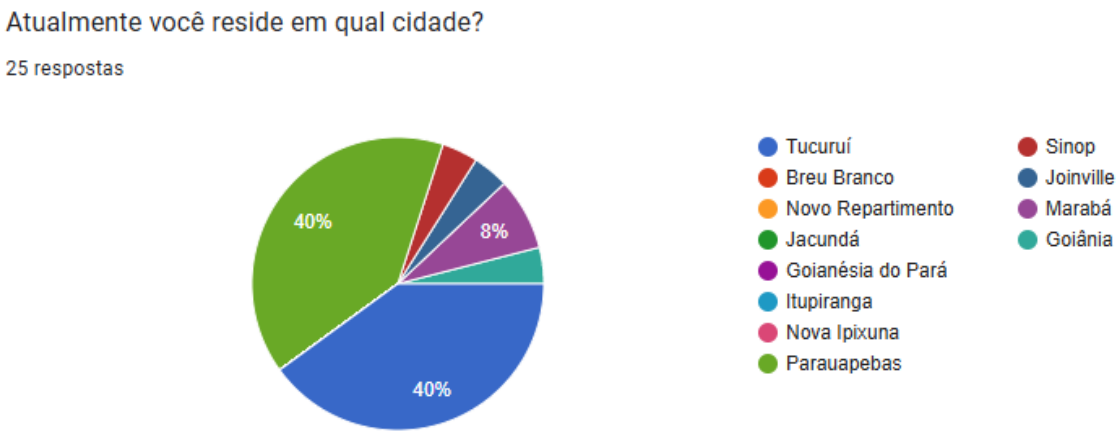
Gráfico 2 - Percentual de dificuldade de uso do LagoTur segundo usuários.



Fonte: Elaborado pelo autor (2025).

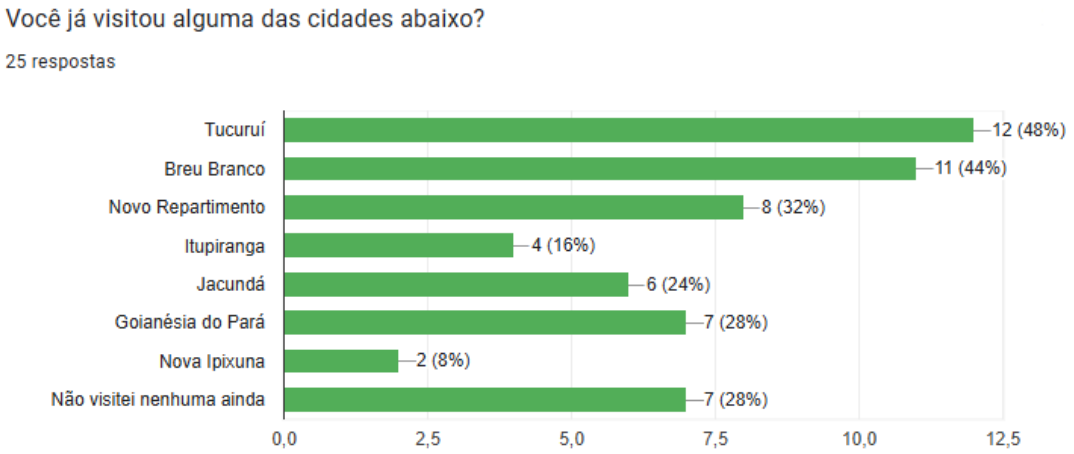
Participaram da pesquisa indivíduos residentes em diferentes cidades e em outros estados (Gráficos 3 e 4), ampliando assim a diversidade do público envolvido na avaliação, essa estratégia teve como finalidade aumentar a abrangência do estudo e fortalecer a validação do protótipo em sua fase inicial, garantindo que as percepções coletadas refletissem uma variedade maior de perfis de usuários e contextos de uso.

Gráfico 3 - Percentual de distribuição dos participantes da pesquisa por cidade de residência.



Fonte: Elaborado pelo autor (2025).

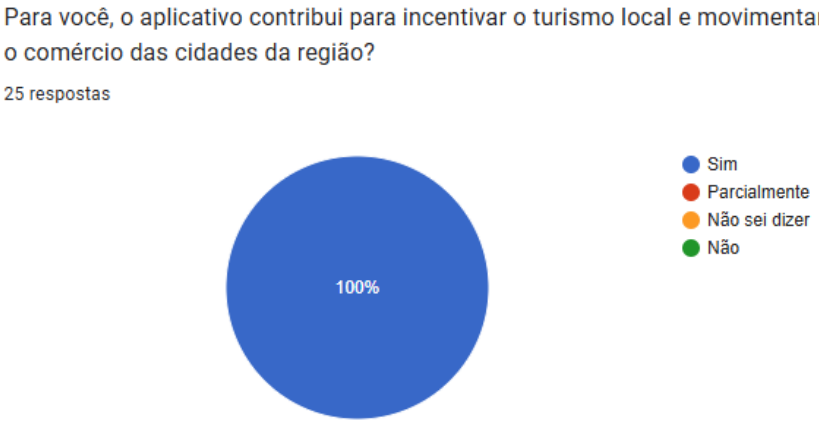
Gráfico 4 - Percentual das cidades visitadas dos participantes da pesquisa.



Fonte: Elaborado pelo autor (2025).

Ademais, esses participantes responderam também a respeito de sua interação com as cidades da região do lago, para assim ser possível validar o grau de interação desses contribuintes com a localidade estudada. Aqueles que participaram da pesquisa e ainda não tinham contato com a região informaram ter bom entendimento da ferramenta e dos itens dispostos no sistema, relatando em sua maioria resultados positivos (Gráfico 5).

Gráfico 5 - Percentual de contribuição da aplicação segundo os participantes da pesquisa.



Fonte: Elaborado pelo autor (2025).

Os testes práticos com usuários reais constituíram uma etapa essencial para validar a premissa central do aplicativo LagoTur e compreender seu desempenho em um cenário próximo ao uso real. A execução desses testes permitiu observar não apenas a aceitação geral do protótipo, mas também o modo como os participantes interagiram com a interface, navegaram entre as funcionalidades e interpretaram os elementos informativos disponibilizados. A partir dessas avaliações, foi possível

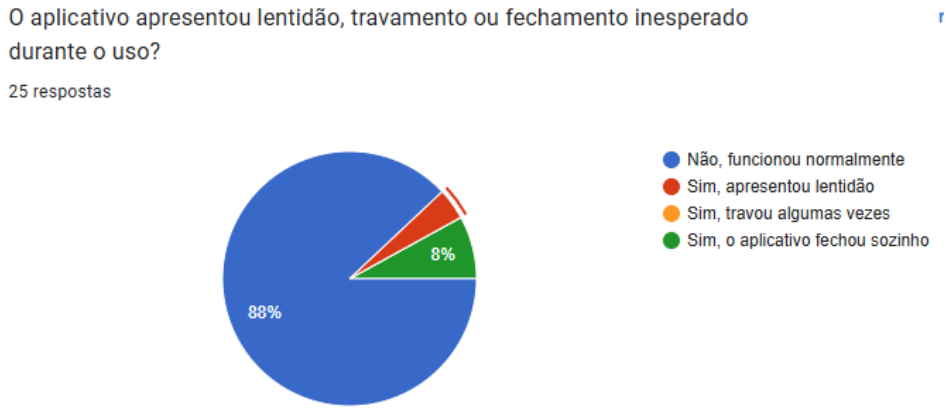
confirmar que o modelo conceitual adotado se mostrou aceitável e compreensível para diferentes perfis de usuários, o processo possibilitou avaliar o comportamento da aplicação em múltiplas configurações de dispositivos e sistemas operacionais, fornecendo evidências importantes sobre sua estabilidade, desempenho e responsividade.

Os dados coletados por meio do formulário aplicado após o uso reforçam que esta etapa representa o ponto de partida para ciclos contínuos de refinamento, permitindo identificar pontos fortes do protótipo e aspectos que poderão ser aprimorados nas versões subsequentes.

6.3 Limitações e pontos de melhoria

Durante os testes práticos realizados com usuários reais, foi possível identificar algumas limitações que, embora não comprometam o funcionamento geral do protótipo, apontam caminhos importantes para as próximas etapas de desenvolvimento (representado no Gráfico 7). Um dos pontos observados diz respeito ao comportamento inesperado do aplicativo em determinados dispositivos, a análise desses casos indicou que fatores como variações de hardware (dispositivo antigo com pouca capacidade de processamento), gerenciamento agressivo de memória e restrições impostas por sistemas de otimização nativos podem ter contribuído para esses incidentes, além disso em alguns dispositivos a aplicação apresentou fechamento repentino.

Gráfico 6 - Percentual de funcionamento nos dispositivos testados.



Fonte: Elaborado pelo autor (2025).

Essa problemática ocasionou no contato com esses participantes visando entender sob quais condições o sistema apresentou falha, fez com que estudos fossem realizados para compreender o motivo disso. Esses relatos foram catalogados

de forma detalhada e servirão de base para correções e ajustes de desempenho nas versões subsequentes do LagoTur.

Outro aspecto identificado como ponto de aprimoramento refere-se à qualidade, consistência e abrangência dos dados exibidos no aplicativo, embora o protótipo utilize informações reais, os testes reforçaram a necessidade de um banco de dados mais robusto, padronizado e validado diretamente pela IGR LagoTur. A implementação futura de uma base de dados oficial e continuamente atualizada fortalecerá a credibilidade da plataforma, ampliará a confiabilidade das informações e proporcionará uma experiência mais completa ao turista.

Além desses fatores, os resultados obtidos revelaram a importância de preparar infraestrutura para um sistema multilíngue, pois o público-alvo inclui turistas nacionais e internacionais, a possibilidade de alternar o idioma da interface foi apontada pelos participantes como uma funcionalidade desejável e relevante. Essa necessidade já está em estudo para implementação futura, e sua identificação só foi possível graças às respostas coletadas nos formulários e às observações realizadas durante os testes práticos. A adoção de um modelo multi-idiomas tende a tornar o aplicativo mais inclusivo e alinhado com o propósito central da ferramenta.

De maneira geral, as limitações observadas não invalidam o funcionamento do protótipo, mas evidenciam oportunidades significativas de evolução, representando um indicativo natural do estágio de desenvolvimento em que o sistema se encontra e contribui de forma direta para a consolidação de melhorias, orientado pelas percepções dos próprios usuários.

6.4 Síntese dos resultados

Os testes realizados com usuários reais permitiram observar que o protótipo do LagoTur alcançou resultados compatíveis com as expectativas definidas no planejamento inicial demonstrando boa capacidade de adaptação a diferentes perfis de usuários e apresentou funcionamento consistente nas funcionalidades essenciais, como navegação, exibição de listas, carregamento de informações por município, uso de galeria multimídia e interação por meio de comentários e avaliações.

Além disso os dados obtidos nas respostas do formulário revelaram que a interface foi percebida como clara e intuitiva pela maior parte dos avaliadores, reforçando que o planejamento da camada de apresentação atendeu adequadamente aos requisitos de usabilidade definidos. Embora tenham sido identificados pontos a

melhorar, tais aspectos se apresentam como oportunidades naturais de aperfeiçoamento após a etapa de prototipagem. Assim, os resultados obtidos consolidam não apenas a viabilidade técnica do projeto, mas também sua relevância prática enquanto ferramenta de apoio ao turismo regional, validando a continuidade do desenvolvimento em direção às próximas versões da aplicação.

Com base nos resultados positivos obtidos durante a fase de testes e na validação inicial do protótipo, a IGR LagoTur iniciou tratativas com autoridades municipais da região para viabilizar o financiamento necessário à implementação da primeira versão funcional e completa do aplicativo. Essa etapa é fundamental, pois o avanço para um produto final envolve demandas técnicas e operacionais que excedem o escopo do protótipo acadêmico, como a ampliação da base de dados oficial com informações verificadas, aprimoramentos de design profissional, expansão da infraestrutura de armazenamento, garantia de suporte multiplataforma (Android e iOS), além dos procedimentos burocráticos e financeiros relacionados à publicação em lojas de aplicativos e de capacidade de armazenamento de informações na nuvem. Dessa forma o processo de captação de recursos e formalização institucional tornou-se um passo estratégico para transformar o LagoTur em uma ferramenta plenamente operacional, garantindo sua sustentabilidade e consolidando seu papel enquanto solução tecnológica de apoio ao turismo regional.

7 CONCLUSÃO

Ao longo deste trabalho, foi proposta, planejada e desenvolvida uma aplicação móvel voltada à divulgação de informações turísticas, estabelecimentos e eventos das cidades da região do lago, fruto da parceria entre a Universidade Federal do Pará e a Instância de Governança Regional LagoTur. Desde o início, o objetivo central foi disponibilizar, em um único aplicativo, conteúdos que hoje se encontram dispersos em redes sociais, sites isolados ou mesmo apenas em conhecimento informal.

Do ponto de vista tecnológico, o projeto demonstrou que a combinação entre React Native, Expo e Firebase foi adequada para a proposta, a arquitetura em camadas (apresentação, lógica e dados), a componentização da interface e o uso de classes de modelo para entidades viabilizou consultas dinâmicas, atualização em tempo quase real e uma comunicação consistente entre banco de dados, classes e componentes.

Concentrar múltiplos conteúdos em uma só plataforma mostrou-se tecnicamente viável e conceitualmente compatível com os objetivos traçados em parceria com a IGR LagoTur, foi possível concluir que a ideia central do sistema foi validada com sucesso nesta fase de prototipagem, demonstrando capacidade de entregar, em um único ambiente digital, a experiência unificada que se pretendia oferecer ao público-alvo, estabelecendo uma base sólida para evoluções futuras do projeto.

Os testes realizados com usuários reais foram fundamentais para validar a premissa da aplicação, os participantes puderam explorar cidades, visualizar listas de estabelecimentos, eventos e roteiros, interagir com o sistema, relatando feedbacks positivos do sistema, as respostas apontaram melhorias desejáveis, como incremento de conteúdo, refinamentos visuais e a necessidade de suporte multilíngue para melhor atendimento a turistas estrangeiros, o protótipo desenvolvido cumpre o papel de validar o conceito inicial do projeto.

Do ponto de vista acadêmico, o trabalho entregou não apenas o código-fonte de uma aplicação funcional, mas também uma documentação detalhada de arquitetura, modelagem de dados, camadas de software, fluxo de navegação, testes e resultados. Do ponto de vista prático, a UFPA e a IGR LagoTur passam a dispor de uma base concreta sobre a qual podem planejar novos ciclos de desenvolvimento e ampliação do escopo funcional.

Em termos de continuidade, a IGR LagoTur já sinaliza a intenção de buscar parcerias para transformar o protótipo em um produto efetivamente lançado nas lojas de aplicativos. O protótipo apresentado neste trabalho, portanto, representa a primeira etapa de um projeto que pode ser mais amplo, em que a tecnologia é utilizada como instrumento de desenvolvimento regional, fortalecimento do turismo e valorização dos municípios do entorno do Lago de Tucuruí. Dessa forma, conclui-se que o LagoTur atingiu os objetivos estabelecidos para esta fase de desenvolvimento, validando ideias e centralizando informações turísticas da região do lago, utilizando uma arquitetura moderna, tecnologias adequadas e uma abordagem centrada no usuário.

REFERÊNCIAS

ABBOTT, D.; DJIRDEH, H.; ACCOMAZZO, A.; SHOEMAKER, S. **Fullstack React Native:** Create beautiful mobile apps with JavaScript. 5. ed. Independently published, 2019.

BANKS, A.; PORCELLO, E. **Learning React:** Modern Patterns for Developing React Apps. 2. ed. Sebastopol: O'Reilly Media, 2020.

BARBOSA, F. F. O turismo como um fator de desenvolvimento local e/ou regional. **Caminhos de Geografia**, Uberlândia, v. 6, n. 14, p. 107-114, fev. 2005.

BERNARDINO, S.; BARROS, L. J. R. Firebase e uso nas aplicações Android e IOS. **Revista Interface Tecnológica**, v. 21 n. 1 p. 279–287 jun. 2024. https://doi.org/10.31510/infa.v21i1.1978

BERRINGTON, J. Databases. **Anaesthesia & Intensive Care Medicine**, [*S. l.*], v. 15, n. 2, p. 59-61, 1 fev. 2014. DOI https://doi.org/10.1016/j.mpaic.2013.12.002. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S1472029913003147?. Acesso em: 11 ago. 2025.

BRASIL. **Decreto-Lei nº 55, de 18 de novembro de 1966**. Define a política nacional de turismo, cria o Conselho Nacional de Turismo e a Empresa Brasileira de Turismo, e dá outras providências. Legislação, Brasília, DF. Disponível em: http://www2.camara.leg.br/legin/fed/declei/1960-1969/decreto-lei-55-18-novembro-1966-371224-norma-pe.html.

BRASIL. **Lei nº 11.771, de 17 de setembro de 2008**. Dispõe sobre a Política Nacional de Turismo, define as atribuições do Governo Federal no planejamento, desenvolvimento e estímulo ao setor turístico, revoga a Lei nº 6.505, de 13 de dezembro de 1977, o Decreto-Lei nº 2.294, de 21 de novembro de 1986, e dispositivos da Lei nº 8.181, de 28 de março de 1991; e dá outras providências. Legislação, Brasília, DF, 2008. Disponível em: http://www2.camara.leg.br/legin/fed/lei/2008/lei-11771-17-setembro-2008-580751-norma-pl.html.

BRASIL. Ministério do Turismo. **Mapa do Turismo Brasileiro**. Disponível em: https://www.mapa.turismo.gov.br/mapa/init.html#/home. Acesso em: 13 ago. 2025.
BRASIL. Ministério do Turismo. **Programa de Regionalização do Turismo – Diretrizes**. Brasília, 2013. Disponível em: https://regionalizacao.turismo.gov.br/images/pdf/PROGRAMA_DE_REGIONALIZACAO_DO_TURISMO_-_DIRETRIZES.pdf. Acesso em: 14 ago. 2025.

BRASIL. Ministério do Turismo. **Programa de Regionalização do Turismo – Roteiros do Brasil:** Módulo Operacional 3 – Institucionalização da Instância de Governança Regional. Brasília: Ministério do Turismo, 2007. Disponível em: https://www.gov.br/turismo/pt-br/centrais-de-conteudo-/publicacoes/programa-de-regionalizacao-do-turismo/modulos-operacionais-do-programa-de-regionalizacao/modulo_operacional_3_institucionalizacao_da_instancia_de_governanca_regional.pdf. Acesso em: 3 set. 2025.

BRASIL. Secretaria de Comunicação Social. **Brasil é segundo país que mais cresceu no turismo internacional**. Gov.br: Portal da Secretaria de Comunicação Social, 15 jul. 2025. Disponível em: https://www.gov.br/secom/pt-br/assuntos/noticias/2025/07/brasil-e-segundo-pais-que-mais-cresceu-no-turismo-internacional. Acesso em: 5 set. 2025.

BRASIL: Ministério do Turismo. **Regiões Turísticas**. 30 maio 2017. Disponível em: https://regionalizacao.turismo.gov.br/index.php?option=com_content&view=article&id=91&Itemid=273. Acesso em: 7 set. 2025.

BRITO, H. *et al*. Javascript in mobile applications: React native vs ionic vs nativescript vs native development. In: IEEE. **2018 13th Iberian Conference on Information Systems and Technologies (CISTI).** [S.l.], 2018. p. 1–6.

COMISSÃO MUNDIAL DE BARRAGENS (CMB). **Estudo de caso da Usina Hidrelétrica de Tucuruí (Brasil):** relatório final da fase de escopo. Rio de Janeiro: Laboratório Interdisciplinar de Meio Ambiente (LIMA)/COPPE/UFRJ, 1999. 42 p.

COUTINHO, G. L. **A Era dos Smartphones: Um estudo Exploratório sobre o uso dos Smartphones no Brasil**. 2014. 60 p. Monografia (Bacharel em Publicidade e Propaganda) - Universidade de Brasília, Brasília, 2014.

DIAS, S.; AFONSO, V. A. Impact of mobile applications in changing the tourist experience. **European Journal of Tourism, Hospitality and Recreation**, v. 11, n. 1, p. 113–120, 2021. DOI: https://doi.org/10.2478/ejthr-2021-0011

DRIBBBLE. Plataforma de portfólios e referências visuais para design de interfaces. Disponível em: https://dribbble.com. Acesso em: 27 ago. 2024.

EL-KASSAS, W. S.; ABDULLAH, B. A.; YOUSEF, A. H.; Wahba, A. M. Taxonomy of Cross-Platform Mobile Applications Development Approaches. **Ain Shams Engineering Journal**, v. 8, n. 2, p. 163–190, 2015. DOI: 10.1016/j.asej.2015.08.004

EMBRATUR. **Brasil registra recorde histórico em receita do turismo internacional no 1º semestre de 2025**. Embratur, 25 jul. 2025. Disponível em: https://embratur.com.br/2025/07/25/brasil-registra-recorde-historico-em-receita-do-turismo-internacional-no-1o-semestre-de-2025/. Acesso em: 14 set. 2025.

EXCALIDRAW. Excalidraw – quadro branco online para criação de esboços, fluxogramas e wireframes. Disponível em: https://excalidraw.com. Acesso em: 27 jul. 2024.

EXPO. **Core Concepts**. Expo Documentation, 2025. Disponível em: https://docs.expo.dev/core-concepts/. Acesso em: 15 set. 2025.
EXPO. Expo Documentation. 2024. Disponível em: https://docs.expo.dev. Acesso em: 29 jul. 2024.

FIREBASE. **Cloud Firestore Documentation**. Google, 2025. Disponível em: https://firebase.google.com/docs/firestore. Acesso em: 14 fev. 2025.

FLANAGAN, David. ***JavaScript: The Definitive Guide***. 7. ed. Sebastopol: O'Reilly Media, 2020.

FOWLER, M. **Modularizing React Applications with Established UI Patterns**. 2023. Disponível em: https://martinfowler.com/articles/modularizing-react-apps.html. Acesso em: 05 out. 2025.

FOWLER, M. **Patterns of Enterprise Application Architecture.** Boston: Addison-Wesley, 2002.

FREECODECAMP. **Como começar com o Firebase usando Python**. 2023. Disponível em: https://www.freecodecamp.org/portuguese/news/como-comecar-com-o-firebase-usando-python/. Acesso em: 27 jan. 2025.

FUNÇÃO CALLBACK. **MDN Web Docs**, 19 nov. 2023. Disponível em: https://developer.mozilla.org/pt-BR/docs/Glossary/Callback_function. Acesso em: 17 out. 2025.

GOOGLE. **Cartões**. Android Developers. Disponível em: https://developer.android.com/design/ui/wear/guides/m2-5/components/cards?hl=pt-br. Acesso em: 19 nov. 2025.

GOOGLE. **Documentation**: Understand Firebase projects. Disponível em: https://firebase.google.com/docs/projects/learn-more. Acesso em: 17 set. 2025.

GOOGLE. **Firebase:** Adicionar dados ao Firestore – Python. 2024. Disponível em: https://firebase.google.com/docs/firestore/manage-data/add-data?hl=pt-br. Acesso em: 27 jan. 2025.

GRETZEL, U.; FESENMAIER, D. R.; O'LEARY, J. T. The transformation of consumer Behaviour. **Journal of Consumer Research**. p. 9-18, 2006. DOI: 10.1086/673522

HANDSONREACT. Component Architecture. Disponível em: https://handsonreact.com/docs/component-architecture. Acesso em: 29 out. 2025.

HEITKÖTTER, H.; HANSCHKE, S.; MAJCHRZAK, T. A. Evaluating cross-platform development approaches for mobile applications. In: **International Conference on Web Information Systems and Technologies**. Springer, Berlin, Heidelberg, 2012. p. 120-138.

INSTITUTO DE DESENVOLVIMENTO FLORESTAL E DA BIODIVERSIDADE DO ESTADO DO PARÁ (IDEFLOR-Bio). **Área De Proteção Ambiental do Lago De Tucuruí**. Disponível em: https://ideflorbio.pa.gov.br/area-de-protecao-ambiental-do-lago-de-tucurui/. Acesso em: 18 set. 2025.

JÚNIOR, H. B. P.; LARANJO, R. B.; ALMEIDA, J. R. **Os benefícios do desenvolvimento multiplataformas**. 2020. Trabalho de Conclusão de Curso (Graduação em Sistema de Informação) - Universidade de Uberaba, 2020. Disponível em: http://dspace.uniube.br:8080/jspui/handle/123456789/1534. Acesso em: 22 set. 2025.

KOMPERLA, V. *et al* . React: A detailed survey. **Indonesian Journal of Electrical Engineering and Computer Science**, [s. l.], v. 26, ed. 3, p. 1710-1717, 3 jun. 2022. DOI 10.11591/ijeecs.v26.i3.pp1710-1717.

KUROSE, J. F.; ROSS, K. W. **Redes de Computadores e a Internet:** Uma Abordagem Top-Down. 6. ed. São Paulo: Pearson, 2017.

LARMAN, C. **Utilizando UML e padrões:** uma introdução à análise e ao projeto orientados a objetos. Porto Alegre: Bookman, 2000. 492 p. ISBN 8573076518

LIMA, F. É. J. F. Regionalização no turismo: uma revisão integrativa. **Revista Iberoamericana de Turismo-RITUR**, v. 15, n. 2, p. 84-97, maio 2025. DOI: 10.28998/ritur.V15.N2.A2025.pp84-97.18780.

MEISTERLABS. MindMeister – Online mind mapping and brainstorming. Disponível em: https://www.mindmeister.com. Acesso em: 27 jul. 2024.

META. **React Documentation**. 2023. Disponível em: https://react.dev. Acesso em: 12 ago. 2024.

META. **React Native – StyleSheet**. 2023. Disponível em: https://reactnative.dev/docs/stylesheet. Acesso em: 29 jan. 2025.

META. **React Native:** Create native apps for Android and iOS using React. 2025. Disponível em: https://reactnative.dev/. Acesso em: 4 out. 2025.

NETTO, A. P.; TRIGO, L. G. G. **Cenários do Turismo Brasileiro**. São Paulo: Aleph, 2009. 214 p.

NGUYEN, A. **Enhancing the TrekBuddy - Travel Mobile Application - with React Native, Firebase Integration, and EAS Expo Deployment**. 2025. 81 p. Thesis (Bachelor of Business Information Technology) - Haaga-Helia University of Applied Sciences, Finland, 2025. Disponível em: https://urn.fi/URN:NBN:fi:amk-2025051411874. Acesso em: 02 out. 2025.

NIELSEN, Jakob. **Usability Engineering**. San Francisco: Morgan Kaufmann, 1993.

OLIVEIRA, T.; PEREIRA, M. L. Instância de Governança da Região Turística da Costa Doce, Rio Grande do Sul: um estudo de caso. **Revista Paranaense de Desenvolvimento**, Curitiba, v. 41, n. 138, p. 53-69, jan./jun. 2020.

PEÑA, G. Governo do Pará entrega a nova orla de Tucuruí e fortalece o turismo na região. **Agência Pará**, 16 maio 2024. Disponível em: https://www.agenciapara.com.br/noticia/55203/governo-do-para-entrega-a-nova-orla-de-tucurui-e-fortalece-o-turismo-na-regiao. Acesso em: 25 set. 2025.

PRESSMAN, R. S**. Engenharia de Software: uma abordagem profissional**. 7. ed. Porto Alegre: AMGH, 2011.

RABAHY, W. A. Análise e perspectivas do turismo no Brasil. **Revista Brasileira de Pesquisa em Turismo**, São Paulo, v. 14, n. 1, p. 1-13, jan./abr. 2020. DOI: 10.7784/rbtur.v14i1.1903

REACT NATIVE. **Environment setup**. React Native Documentation, 2025. Disponível em: https://reactnative.dev/docs/environment-setup. Acesso em: 5 out. 2025.

REACT NATIVE. **About the New Architecture**. Disponível em: https://reactnative.dev/architecture/landing-page. Acesso em: 07 out. 2025.

REACT. **Reusing Logic with Custom Hooks**. Disponível em: https://react.dev/learn/reusing-logic-with-custom-hooks. Acesso em: 15 nov. 2024.

SAARIKOSKI, Anna. **Expo-sovelluskehyksen hyödyntäminen React Native -sovelluskehityksessä**. 2025. Thesis (Data processing training) – Hämeen ammattikorkeakoulu, Hämeenlinna, 2025. Disponível em: https://urn.fi/URN:NBN:fi:amk-202505048920. Acesso em: 10 out. 2025.

SAHOO, A. **Understanding Component Structure in React:** The Building Blocks of Your UI. Medium, 2023. Disponível em: https://medium.com/@asutosh.98.sahoo/understanding-component-structure-in-react-the-building-blocks-of-your-ui-0069f54b4e67. Acesso em: 17 jul. 2025.

SANTOS, B. J. V. *et al*. Potencial do turismo da pesca amadora com base na infraestrutura disponível na Amazônia Oriental, Pará, Brasil. **OBSERVATÓRIO DE LA ECONOMÍA LATINOAMERICANA**, Curitiba v.23, n.8, p. 01-21. 2025. DOI: 10.55905/oelv23n8-137

SOMMERVILLE, I. **Engenharia de software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011. 529p.

SOUZA, C. L.; CAÑETE, V. R. Pesca esportiva e pesca artesanal: lazer e sobrevivência na Hidrelétrica de Tucuruí (PA). **Revista Brasileira de Ecoturismo (RBEcotur)**, São Paulo, v. 8, n. 5, p. 614-633, nov. 2015-jan. 2016. DOI: 10.34024/rbecotur.2015.v8.6435.

STEFANOV, S. **React:** Up and Running. 1. ed. Sebastopol: O'Reilly Media, 2016.

THEOHARIDOU, M.; MYLONAS, A.; GRITZALIS, D. A Risk Assessment Method for Smartphones. **Information Security and Privacy Research: 27th IFIP TC 11 International Information Security Conference (SEC 2012), Proceedings**. Berlim; Heidelberg: Springer, 2012. p. 443-456. DOI: 10.1007/978-3-642-30436-1_36.

TUCURUÍ (PA). Prefeitura Municipal de Tucuruí. Disponível em: https://tucurui.pa.gov.br/. Acesso em: 01 out. 2025.

VAGHELA, Jay. **React Native Card**. Scaler, 16 jan. 2024. Disponível em: https://www.scaler.com/topics/react-native-card/. Acesso em: 07 dez. 2025.

WANG, D.; PARK, S.; FESENMAIER, D. R. The Role of Smartphones in Mediating the Touristic Experience. **Journal of Travel Research**, [S. l.], v. 51, n. 4, p. 371-387, jul. 2012. DOI https://doi.org/10.1177/00472875114263

YONGHUI L. *et al*. ReuNify: A Step Towards Whole Program Analysis for React Native Android Apps. In: **Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE 2023).** Echternach, Luxembourg: IEEE/ACM, 2023. DOI: 10.1109/ASE56229.2023.00113

ZIMMER, D. **Acoplamento, Coesão e Encapsulamento em Organizações.** Target Teal, 17 ago. 2020. Disponível em: https://targetteal.com/blog/acoplamento-coesao-encapsulamento/. Acesso em: 25 out. 2025.