UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

PEDRO LUÍS RIBEIRO DE SOUZA MARRA

UM SISTEMA PARA GERENCIAR A REVENDA DE INGRESSOS

Ouro Preto, MG
2025

PEDRO LUÍS RIBEIRO DE SOUZA MARRA

UM SISTEMA PARA GERENCIAR A REVENDA DE INGRESSOS

Monografia II apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Joubert de Castro Lima

Ouro Preto, MG
2025

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

Pedro Luís Ribeiro de Souza Marra

UM SISTEMA PARA GERENCIAR A REVENDA DE INGRESSOS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 28 de Agosto de 2025.

Membros da banca

Joubert de Castro Lima (Orientador) - Doutor - Universidade Federal de Ouro Preto
André Luis Almeida (Examinador) - Doutor - IFMG - Instituto Federal Minas Gerais Campus Ouro Preto
Reinaldo Silva Fortes (Examinador) - Doutor - Universidade Federal de Ouro Preto

Joubert de Castro Lima, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 28/08/2025.

Documento assinado eletronicamente por **Joubert de Castro Lima**, **PROFESSOR 3 GRAU**, em 01/09/2025, às 14:13, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0965547** e o código CRC **1000E70B**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591692 - www.ufop.br

Resumo

A pandemia de COVID-19 impulsionou o uso de serviços online, incluindo a venda de ingressos, devido ao fechamento das bilheterias físicas. Essa transição para o ambiente digital trouxe vantagens significativas para consumidores e organizadores de eventos, consolidando o crescimento das plataformas de comercialização de ingressos online, mesmo após o fim da pandemia. Contudo, a existência desses sistemas apenas aumentou um problema que já existia no mercado de venda de ingressos: a falta de um processo confiável para a revenda de ingressos. Tal problema gera fraudes diversas, pois o usuário até consegue realizar a troca de titularidade; entretanto, se expõe a inúmeros riscos no pagamento. Este trabalho propõe uma solução para mitigar o problema e, para tal, implementamos e testamos um sistema confiável e fracamente acoplado às aplicações de venda de ingressos, que permite a revenda de ingressos, superando em parte as limitações impostas até o momento. Até o término desse trabalho, não foram encontrados sistemas que permitem usuários recomprarem ou revenderem ingressos, viabilizando a troca segura de titularidade de ingressos de forma compatível com diferentes plataformas de venda e pagamento online.

Palavras-chave: Segurança da Informação, Fraudes Online, Sistemas Distribuídos, Comércio Eletrônico, Integração de Sistemas, Criptografia Aplicada, Engenharia de Software para Aplicações Web/Mobile, Model-Based Testing.

Abstract

The COVID-19 pandemic boosted the use of online services, including ticket sales, due to the closure of physical box offices. This transition to the digital environment brought significant advantages for both consumers and event organizers, consolidating the growth of online ticketing platforms even after the end of the pandemic. However, the existence of these systems only increased a problem that had already been present in the ticketing market: the lack of a reliable process for ticket resale. Such an issue leads to various types of fraud, since users are able to change ticket ownership but are still exposed to numerous risks during payment. This work proposes a solution to mitigate this problem. To achieve this, we designed and tested a reliable and loosely coupled system that integrates with ticketing platforms and enables ticket resale, partially overcoming the limitations imposed so far. By the conclusion of this work, no systems were found that allow users to safely repurchase or resell tickets, making it possible to securely transfer ticket ownership in a way that is compatible with different online sales and payment platforms.

Keywords: Information Security, Online Fraud, Distributed Systems, Electronic Commerce, System Integration, Applied Cryptography, Software Engineering for Web/Mobile Applications, Model-Based Testing.

Dedico este trabalho à memória de minha avó Neusa, tenho certeza que está torcendo por mim de algum lugar.

Agradecimentos

Agradeço aos meus pais, Gildácio e Neusa, por me fornecerem o suporte e o incentivo necessários para que essa jornada fosse trilhada.

Aos meus irmãos, Mateus, Davi e Miguel, por me inspirarem a ser alguém melhor.

Agradeço à Renata, por ter sido meu ponto de equilíbrio do início ao fim.

À República Arca de Noé, por ter sido meu lar e minha família em Ouro Preto.

Meus sinceros agradecimentos ao Prof. Dr. Joubert Lima, que pacientemente e com enorme experiência me guiou durante este projeto.

Por fim, agradeço à UFOP e ao DECOM pelo ensino de qualidade.

Sumário

1 Introdução

O mercado de ingressos online foi estimado em US$ 81,10 milhões em 2024, e acredita-se que pode chegar a US$ 101,94 milhões em 2029, com uma previsão de crescimento de 4,68% nesse período (Mordor Intelligence, 2024). Juntamente com esse crescimento, também vêm aumentando os casos de golpes envolvendo a venda de ingressos online, como a venda de ingressos inexistentes, sites falsos, venda de ingressos roubados ou usados, ou o uso de métodos de pagamento não confiáveis durante a troca de titularidade.

Segundo o site (ENCYCLOPEDIA.COM, 2024), o comércio de ingressos online surgiu em 1978 nos Estados Unidos. Nessa época, cada ingresso tinha seu assento atribuído, e eram divididos entre vendedores; dessa forma, o comprador poderia ser forçado a comprar assentos ruins, pois não saberia da disponibilidade de assentos melhores com outros vendedores. Diante disso, dois estudantes enxergaram a necessidade de desenvolver um sistema que integrasse a venda desses ingressos, oferecendo acesso a todos os ingressos disponíveis para compra, e assim nasceu a TicketMaster(TICKETMASTER, 2025), empresa pioneira no ramo.

No Brasil, a venda de ingressos online se iniciou por volta dos anos 2000, com foco na venda de ingressos de cinemas (FUNCAP, 2011), mas a popularização e consolidação vieram nos anos 2010 com a expansão dos smartphones, possibilitando a compra de ingressos sem a retirada na bilheteria. Em 2020, com a chegada da pandemia de COVID-19, as empresas enfrentaram dificuldades inicialmente, causadas pelo cancelamento dos eventos. Porém, com a explosão digital acelerada na pandemia e o surgimento de eventos online, as plataformas de ingressos online tornaram-se a única forma de adquirir ingressos, o que possibilitou o conhecimento dessas plataformas e de seus benefícios por um público maior, gerando um grande aumento nas receitas dessas empresas nos anos seguintes (EXAME, 2024).

1.1 Justificativa

Muitas vezes, as pessoas procuram comprar seus ingressos para eventos com bastante antecedência devido aos preços mais acessíveis ou ao risco de esgotamento dos ingressos. No entanto, em alguns casos, a pessoa pode acabar não conseguindo ir ao evento por diversos motivos. Para não perder o dinheiro gasto na compra do ingresso, ela procura transferir seu ingresso para outra titularidade em troca de um pagamento, o que abre margem para golpes.

Por outro lado, quando o evento se aproxima, existe outro grupo de pessoas que procura comprar ingressos de quem os adquiriu nos lotes anteriores e desistiu de ir ao evento. Nesse momento, surgem muitos golpistas com ingressos falsos ou que não enviam ingresso algum após receberem o pagamento.

Cada empresa de venda de ingresso normalmente possui seu próprio sistema de troca de titularidade, entretanto, toda a questão de pagamentos fica fora de tais empresas. As empresas não se interessam em misturar vendas com revendas, pois esses serviços podem competir entre si. Mas e se tais empresas de venda de ingressos pudessem ganhar também com as revendas, em um modelo de negócio que separasse os serviços?

Diante desse cenário, é essencial a criação de um sistema que gerencie todo o fluxo da troca de titularidade de ingressos com segurança, minimizando os riscos de fraudes e gerando interesse por parte das empresas que vendem ingressos.

1.2 Objetivo Geral

Desenvolver um sistema robusto e confiável para centralizar num único ambiente a revenda de ingressos. Tal sistema deve ser fracamente acoplado aos sistemas de venda de ingressos existentes, assegurando a autenticidade dos ingressos e a integridade financeira das transações.

1.2.1 Objetivos Específicos

• Verificação de autenticidade de ingressos: Implementar um mecanismo que valide a autenticidade dos ingressos antes da revenda.

• Certificação de envio e recebimento de ingressos: Desenvolver um mecanismo que certifique que o ingresso teve sua titularidade trocada; portanto, o vendedor e o comprador devem receber as informações necessárias para checagem tanto na empresa vendedora como também no sistema de revenda de ingressos que está sendo proposto.

• Criação de um fluxo de pagamento e comissionamento: Implementar um mecanismo de pagamento que garanta a transferência financeira entre o comprador e o revendedor do ingresso, assim como um método que gere as comissões tanto para a empresa de venda quanto para a empresa de revenda de ingressos.

• Realização de avaliações experimentais dos componentes de forma individualizada e de forma integrada.

• Discussão dos resultados obtidos com os experimentos.

1.3 Estrutura do Trabalho

O restante desta monografia se organiza da seguinte forma: no Capítulo 2 há a fundamentação teórica e a descrição dos principais sistemas de venda ou troca de ingressos no Brasil ou internacionais. Já no Capítulo 3 descrevemos os componentes arquiteturais do sistema de revenda de ingressos. No Capítulo 4 há os experimentos conduzidos para verificar o funcionamento da aplicação desenvolvida, mesmo que na forma de simulações, pois a integração com os sistemas

de pagamento e venda de ingressos foi emulada. Por fim, no Capítulo 5 realizamos as conclusões e apontamos os trabalhos futuros.

2 Fundamentação Teórica e Trabalhos Relacionados

Neste capítulo serão apresentados os fundamentos teóricos que sustentam o desenvolvimento da proposta, abordando conceitos relacionados a aplicativos móveis, segurança em transações digitais e uso de *frameworks* como o React Native. Além disso, serão discutidos trabalhos e soluções já existentes no mercado voltados para a compra e venda de ingressos online, destacando suas características, limitações e contribuições, de modo a situar a pesquisa no contexto atual e evidenciar sua relevância.

2.1 Fundamentação Teórica

Para a realização deste trabalho, usamos o conceito de banco de dados NoSQL e API, assim como diversos *frameworks* e tecnologias. Nesta seção, descrevemos as tecnologias e conceitos adotados para a construção do sistema de revenda de ingressos.

2.1.1 Banco de Dados NoSQL

Os bancos de dados NoSQL (Not Only SQL) surgiram como uma alternativa aos bancos relacionais tradicionais, visando atender às demandas de aplicações modernas que exigem alta escalabilidade, flexibilidade e desempenho em tempo real. Diferentemente dos bancos de dados relacionais, os bancos de dados NoSQL não apresentam esquemas rígidos de armazenamento. Esses sistemas possibilitam a organização das informações em distintos formatos, como documentos, pares chave-valor, colunas ou grafos, o que garante maior flexibilidade na adaptação a diferentes tipos de dados e a estruturas dinâmicas. Essa abordagem é particularmente eficaz em cenários de Big Data e aplicações distribuídas, onde o volume, a variedade e a velocidade dos dados tornam os modelos relacionais menos eficientes (MONIRUZZAMAN; HOSSAIN, 2013; HAN et al., 2011).

2.1.2 Application Programming Interface - API

Uma API (Interface de Programação de Aplicações) é uma interface que permite a comunicação entre diferentes sistemas, aplicações ou componentes de software, definindo um conjunto de regras e protocolos para a troca de dados. Atua como uma ponte que facilita a integração entre sistemas, permitindo que uma aplicação utilize funcionalidades ou dados de outra sem precisar compreender seus detalhes internos. Assim, as APIs desempenham um papel crucial no desenvolvimento moderno, possibilitando a criação de ecossistemas conectados e

promovendo a interação eficiente e escalável entre diferentes serviços e plataformas (FIELDING, 2000).

Entre os padrões e tecnologias amplamente utilizados, destaca-se o RESTful (Representational State Transfer), baseado em princípios como a utilização de recursos identificados por URLs e métodos HTTP (GET, POST, PUT, DELETE). Sua popularidade deriva de sua simplicidade e compatibilidade com a web (FIELDING, 2000). Outra tecnologia relevante é o GraphQL, que oferece consultas mais flexíveis e eficientes, sendo ideal para cenários em que o cliente requer maior controle sobre os dados retornados (Facebook, 2015). Além disso, tecnologias como SOAP (Simple Object Access Protocol), que utilizam XML para a troca de mensagens (BOX et al., 2000), e frameworks como gRPC, projetado para alto desempenho em comunicações entre microsserviços (Google, 2015), são escolhidos conforme as necessidades específicas de cada projeto. Esses recursos permitem que desenvolvedores construam integrações robustas e escaláveis, promovendo a interoperabilidade no ecossistema digital.

2.1.3 Firebase

O Firebase é uma plataforma de desenvolvimento de aplicativos criada pelo Google que oferece uma variedade de serviços backend prontos para uso, permitindo que desenvolvedores criem, melhorem e dimensionem aplicações web e mobile com mais agilidade. Dentre suas funcionalidades, destacam-se autenticação de usuários (Firebase Authentication), banco de dados em tempo real (Realtime Database e Firestore), hospedagem de arquivos e sites (Firebase Hosting), funções serverless (Cloud Functions), além de ferramentas de análise, performance, testes e notificações. A proposta central do Firebase é eliminar a complexidade da infraestrutura backend, fornecendo APIs e SDKs integrados que aceleram o ciclo de desenvolvimento e melhoram a experiência do usuário final (FIREBASE, 2025c; GOOGLE, 2025a). Sua integração nativa com o Google Cloud também permite que os projetos evoluam de forma escalável e segura, atendendo desde pequenos aplicativos até soluções empresariais robustas.

2.1.4 Cloud Firestore

O Cloud Firestore é um banco de dados NoSQL em tempo real desenvolvido pelo Google como parte da plataforma Firebase, voltado para o desenvolvimento de aplicações web e mobile modernas. Ele utiliza uma estrutura baseada em documentos e coleções, permitindo o armazenamento flexível de dados em formato JSON, sem necessidade de esquemas fixos. Uma de suas principais características é a capacidade de sincronizar dados automaticamente entre os dispositivos conectados, facilitando a criação de aplicações colaborativas e responsivas em tempo real. Além disso, o Firestore oferece suporte a operações offline, consultas complexas com filtros e ordenações, regras de segurança baseadas em autenticação e escalabilidade horizontal automática, o que o torna uma solução robusta e amplamente adotada em projetos que exigem desempenho e flexibilidade (FIREBASE, 2025a; CLOUD, 2025).

2.1.5 Firebase Authentication

O Firebase Authentication (Firebase Auth) é um serviço da plataforma Firebase que simplifica o processo de autenticação de usuários em aplicações web e mobile, oferecendo uma implementação segura e integrada com outros serviços da plataforma. Ele permite que desenvolvedores implementem, de forma rápida e confiável, diferentes métodos de login, como autenticação por e-mail e senha, número de telefone, autenticação anônima e logins por provedores externos como Google, Facebook, Apple, GitHub e outros. Além disso, o Firebase Auth gerencia sessões de usuários, recuperação de senhas, verificação de e-mails e integração com tokens de autenticação (JWT), facilitando o controle de acesso e a aplicação de regras de segurança em bancos de dados como Firestore e Realtime Database. Com suporte para SDKs em diversas linguagens e frameworks, o Firebase Auth é amplamente utilizado por desenvolvedores que buscam uma solução de autenticação robusta, escalável e fácil de integrar (FIREBASE, 2025b; GOOGLE, 2025b).

2.1.6 React Native

Criado pelo Facebook, o React Native é um framework de código aberto que utiliza JavaScript e a biblioteca React para desenvolver aplicativos móveis nativos para iOS e Android. Ele permite que uma única base de código seja utilizada para múltiplas plataformas, oferecendo um desenvolvimento mais ágil e eficiente. Uma de suas grandes vantagens é a compatibilidade com diversos SDKs modernos, incluindo o Firebase, que oferece integração nativa com o React Native para serviços essenciais como autenticação de usuários (Firebase Auth), banco de dados em tempo real (Firestore), armazenamento na nuvem, notificações push e funções serverless. Essa integração facilita a implementação de funcionalidades robustas com menos esforço de configuração e manutenção, acelerando o ciclo de desenvolvimento. Apesar de ser uma excelente opção para aplicações multiplataforma, o React Native ainda apresenta limitações em comparação com soluções desenvolvidas inteiramente em código nativo, como Swift para iOS ou Kotlin para Android, especialmente em aplicações que exigem alto desempenho gráfico ou uso intensivo de recursos específicos do sistema operacional ((FACEBOOK), 2024; FIREBASE, 2025d).

2.1.7 Metodologia de Testes MBT

O Model-Based Testing (MBT) é uma abordagem de testes de software que utiliza modelos formais para descrever o comportamento esperado do sistema, permitindo a geração sistemática de casos de teste a partir desses modelos (UTTING; PRETSCHNER; LEGEARD, 2012). Uma das ferramentas amplamente utilizadas nesse contexto é o GraphWalker, que possibilita a execução de testes baseados em grafos, explorando caminhos definidos por vértices e arestas que representam estados e transições do sistema (GRAPHWALKER,). Para a construção desses grafos, é comum a utilização do yEd, uma ferramenta de modelagem visual que facilita o desenho dos modelos e permite exportá-los no formato GraphML, padrão aceito pelo GraphWalker para execução dos

testes (YED...,). Dessa forma, a integração entre o yEd e o GraphWalker promove um fluxo de trabalho eficiente no MBT, desde a modelagem gráfica até a automação da execução dos testes.

2.2 Trabalhos Relacionados

Os trabalhos relacionados abrangem sistemas de venda de ingressos para eventos. Nesses sistemas, os organizadores anunciam seus eventos e efetuam as vendas de ingressos de forma segura. Em contrapartida, as empresas que realizam a venda ganham uma porcentagem da transação. Nos próximos parágrafos, são apresentadas as vantagens e desvantagens de cada sistema.

A *Ticketmaster* (TICKETMASTER, 2025) é um dos sistemas de venda de ingressos mais visitados do mundo, segundo o ranking da *SimilarWeb* (SIMILARWEB, 2025). Além de anunciar os eventos, o sistema também fornece ferramentas de marketing para promover tais eventos e monitorar as vendas em tempo real. A *Ticketmaster* oferece mecanismos para efetuar a troca de titularidade dos ingressos e permite devoluções; porém, não transfere a responsabilidade de pagamento para o outro usuário. Portanto, se a compra for cancelada, o ingresso ficará inválido. De uma forma geral, não há o conceito de revenda de ingressos em tal sistema.

Outra plataforma bastante visitada, segundo o ranking da *SimilarWeb* (SIMILARWEB, 2025), é a *Eventbrite* (EVENTBRITE, 2025). Seu funcionamento de vendas é semelhante ao da *Ticketmaster*, mas, no quesito troca de ingressos, a *Eventbrite* não permite a transferência de um ingresso de uma conta para outra. Em vez disso, possibilita que o usuário edite as informações do ingresso, colocando o nome e endereço de e-mail de outra pessoa. Isso não remove o ingresso da conta do usuário que fez a compra, mas o novo participante recebe um e-mail com uma solicitação de reivindicação de ingresso após uma alteração de titularidade ser salva.

Os sistemas *Sympla* (SYMPLA, 2025), *Eventim* (EVENTIM, 2025), *Ticket360* (TICKET360, 2025) e *Ingresse* (INGRESSE, 2025) oferecem o mesmo tipo de serviço de venda de ingresso, mas em esfera nacional. Na *Sympla*, a troca de titularidade é feita como uma autorização de uso do ingresso, mas o pedido fica permanentemente vinculado ao usuário que fez a compra. A *Eventim* fez parceria com a *FanSale* (FANSALE, 2025) para ter um sistema de revenda de ingressos com seguro de compra, mas ainda não está disponível e funcionará apenas para ingressos adquiridos na própria *Eventim*. Já nos sistemas *Ticket360* e *Ingresse*, a troca de titularidade é feita da mesma forma que na *Ticketmaster*. Tais sistemas não recomendam a compra de ingressos fora de seus domínios, devido ao risco de golpes.

StubHub (STUBHUB, 2025), *SeatGeek* (SEATGEEK, 2025), *FanSale* (FANSALE, 2025) e *Pia.jp* (PIA.JP, 2025) são sistemas que permitem ao usuário revender seus ingressos. Eles oferecem garantia de devolução do dinheiro caso o ingresso seja falso e validam os ingressos através de parcerias com as empresas de venda ou exigindo comprovações da compra dos ingressos. Entretanto, nenhuma dessas empresas é brasileira, e a maior parte de seus ingressos é para eventos

no exterior.

A seguir, na Tabela 2.1, é apresentada uma relação das plataformas citadas e as funcionalidades descritas anteriormente.

Plataforma / Funcionalidade	TicketMaster	EventBrite	Sympla	Eventim	Ticket360	Ingresse	Stubhub	Seatgeek	Pia.jp	FanSale	Sistema Proposto
Aplicação de venda de ingressos	X	X	X	X	X	X	X	X	X	X	
Aplicação de revenda de ingressos							X	X	X	X	X
Permite troca de titularidade	X	X	X	X	X	X	X	X	X	X	X
Permite transferência de conta	X	X		X	X	X	X	X	X	X	X
Assegura pagamento							X	X	X	X	X
Plataforma brasileira			X	X	X	X					X
Plataforma estrangeira	X	X					X	X	X	X	

Tabela 2.1 – Tabela de comparação entre plataformas.

De uma forma geral, os sistemas de troca de titularidade de ingressos funcionam para ingressos adquiridos na própria empresa de venda. O comprador quase sempre fica responsável pelo ingresso, mesmo após a troca de titularidade. O único sistema de revenda encontrado no Brasil ainda é uma promessa - *FanSale* (FANSALE, 2025); entretanto, o mesmo possuirá integração com uma única plataforma de venda. Os demais sistemas para revendas de ingressos são para o mercado estrangeiro.

O sistema para revenda de ingressos sendo proposto tenta atenuar esta limitação do atual mercado de ingressos online no Brasil, com uma proposta que também deve ser benéfica para as empresas de venda. Em linhas gerais, o sistema de revenda de ingressos deste trabalho deve ser gratuito, de fácil acoplamento nas plataformas de venda de ingressos, capaz de gerar renda através das taxas pagas ao revender ingressos e capaz de unificar eventos de variadas plataformas de venda em uma loja centralizada que pode alcançar o interesse de novos usuários.

O sistema também deve ser benéfico para os usuários, oferecendo a possibilidade de compra de ingressos com a segurança de que são legítimos, pois são enviados diretamente das plataformas de venda de ingressos, e que os devidos pagamentos serão recebidos, assim como as trocas de titularidade efetuadas tanto na plataforma de venda como também no sistema de revenda.

3 Desenvolvimento

Este capítulo descreve o processo de desenvolvimento do sistema para revenda de ingressos. O desenvolvimento foi realizado utilizando o framework *React Native*, juntamente com o *Firebase Authentication* e o *Cloud Firestore* para gerenciar a publicação e consumo de informações diversas. Ao longo do capítulo, são apresentados trechos de código exemplificando as implementações, seguidos por explicações sobre as decisões de projeto, assim como as tecnologias empregadas.

3.1 Metodologia de Desenvolvimento

Nessa sessão serão apresentados os métodos de coleta e análise de requisitos, o planejamento das funcionalidades, a arquitetura do sistema, bem como as tecnologias escolhidas para garantir eficiência, segurança e usabilidade.

3.1.1 Arquitetura do Sistema

O aplicativo utiliza uma arquitetura baseada em Backend-as-a-Service (BaaS), onde o cliente desenvolvido em React Native se comunica diretamente com o banco de dados Firestore, um banco NoSQL em nuvem fornecido pelo Firebase, por meio do SDK react-native-firebase. Esse modelo elimina a necessidade de um servidor backend dedicado, transferindo responsabilidades como autenticação, armazenamento e lógica de negócios para a infraestrutura do Firebase. Os dados são enviados no formato JSON, incluindo as informações dos ingressos. Esta arquitetura foi escolhida por proporcionar uma escalabilidade horizontal, facilitando a expansão do sistema conforme a demanda de usuários.

Para que um ingresso esteja disponível para revenda, a plataforma de venda de ingressos deve adicionar uma função JavaScript que envia um JSON contendo as informações do ingresso e do seu titular. O JSON é enviado de forma segura e armazenado no Cloud Firestore, dentro de uma coleção indexada para a loja de ingressos onde o ingresso foi comprado e em uma subcoleção indexada para o documento do usuário que deseja revendê-lo. Após a publicação, o ingresso pode ser acessado pela loja de revenda de ingressos. A Figura 3.1 ilustra todo o processo de publicação, onde os usuários 1 e 2, oriundos de plataformas de venda de ingressos distintas, querem trocar a titularidade de seus ingressos e obterem os devidos pagamentos/recebimentos.

Na Figura 3.2, há a continuidade do processo de revenda de ingressos, onde um terceiro usuário se interessa por um dos ingressos publicados para revenda. A Loja de revenda de ingressos possui uma função JavaScript que recebe todos os ingressos publicados, podendo filtrá-los de várias formas (Ex. artista, preço, localidade, etc.). Quando o usuário 3 finaliza o processo de

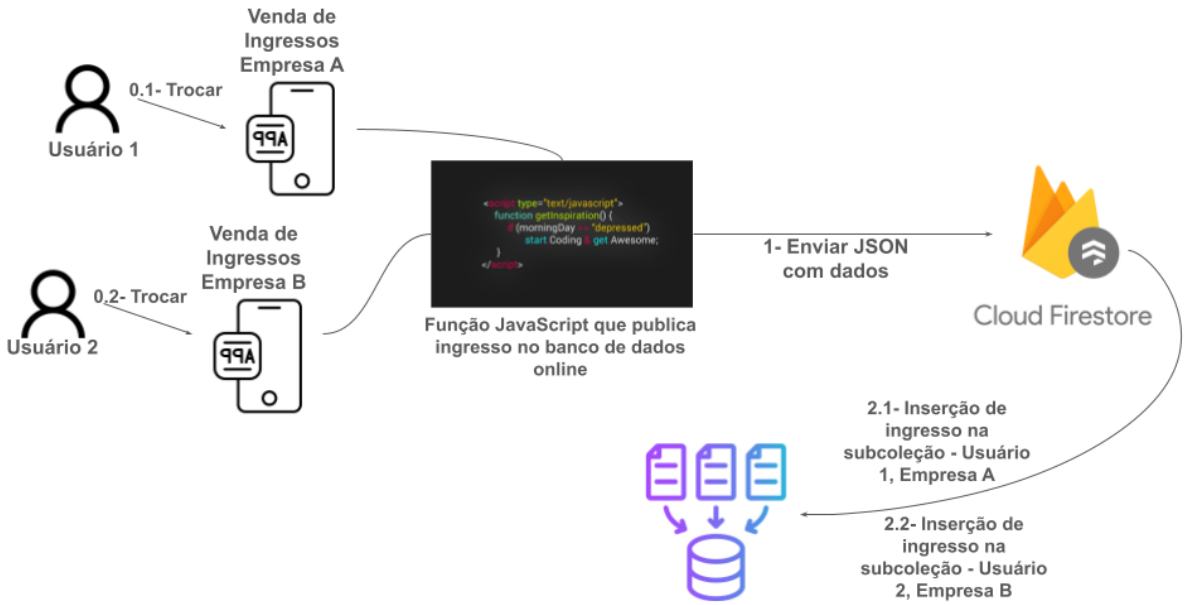

Figura 3.1 – Fluxo de mensagens do processo de revenda de ingressos.

preenchimento dos dados do novo titular do ingresso, formas de pagamento, etc., uma nova função JavaScript é chamada para validar o pagamento junto à API de serviços financeiros, e então os dados do ingresso são atualizados e movidos para outra subcoleção indexada ao documento do usuário que acaba de comprar o ingresso, permitindo que as aplicações de venda de ingressos e os usuários 1 e 2 possam perceber a troca de seus ingressos com êxito e segurança.

Caso os usuários 1 e 2 ainda não tenham completado sua conta na Loja de revenda de ingressos, um email com link é enviado, pois tais usuários precisam informar contas para recebimento e demais informações cadastrais. De uma forma geral, o pagamento do usuário 3 é creditado na conta da Loja de revenda de ingressos e depois, deduzidas taxas e comissões, é devidamente transferido para a conta do usuário 1 ou 2, a depender do ingresso revendido.

A comunicação entre o aplicativo e os serviços do Firebase é protegida por múltiplos mecanismos que atuam em conjunto. Todas as requisições trafegam sob HTTPS/TLS, garantindo a confidencialidade e a integridade dos dados em trânsito. Quando o usuário realiza login, o Firebase emite um ID Token JWT assinado, que comprova sua identidade e é enviado em cada requisição; esses tokens expiram periodicamente e podem ser renovados de forma segura por meio de refresh tokens. O acesso aos dados no Firestore é controlado pelas Security Rules, que verificam o token do usuário a cada leitura ou escrita para autorizar apenas operações compatíveis com as permissões definidas. Além disso, todos os dados armazenados são automaticamente criptografados pela infraestrutura do Google Cloud, e o sistema conta com mecanismos adicionais de proteção contra abuso e acesso não autorizado, garantindo segurança tanto no transporte quanto no armazenamento e no controle de acesso.

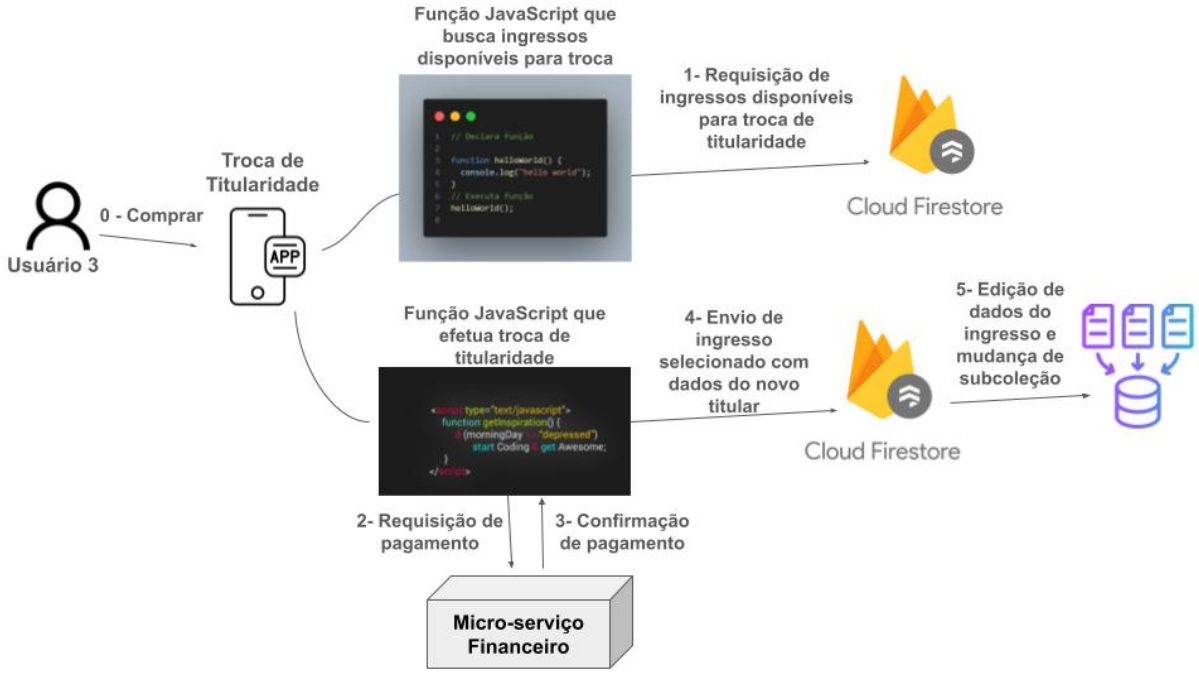

Figura 3.2 – Fluxo de mensagens do processo de compra de ingressos disponíveis para revenda.

3.1.2 Estrutura de um Aplicativo típico de Venda de Ingressos

Para simular todo o processo de venda, foi desenvolvido um aplicativo para uma loja fictícia chamada *BuyTickets*, com o objetivo de ilustrar o funcionamento típico dentro de plataformas de venda de ingressos. O aplicativo possui duas abas principais: a aba “Destaques”, que exibe eventos fictícios, e a aba “Ingressos”, onde são listados os ingressos já comprados pelo usuário.

Na Figura 3.3, é exibido um exemplo do processo de venda, no qual o usuário João Silva coloca à venda seu ingresso para o evento “Mostra de Cinema Independente”. Note que, ao selecionar o ingresso, são exibidas as informações do titular, e após clicar sobre o botão vender, surge um modal para que o usuário insira o valor de venda. Perceba também que, ao disponibilizar um ingresso para venda, uma tag “À Venda” é exibida no ingresso, e ao clicar sobre ele, o usuário tem a opção de cancelar a revenda.

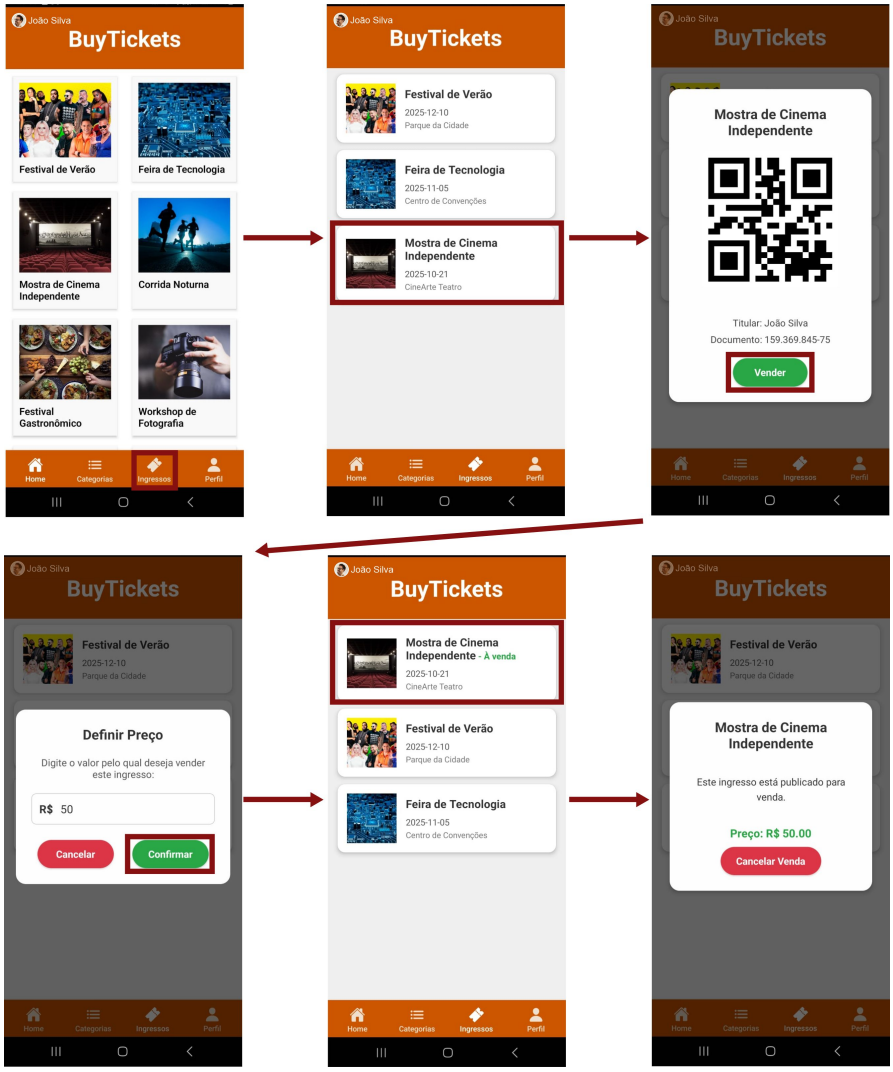

Figura 3.3 – Aplicativo de venda de ingressos.

Ao confirmar a venda, uma função JavaScript é acionada para enviar um objeto JSON ao banco de dados NoSQL localizado no Cloud Firestore da loja de troca de titularidade. Esse JSON contém as seguintes informações: ID do usuário, e-mail, CPF, token da empresa de venda, metadados do ingresso e o valor desejado para a troca. A Listagem 3.1 apresenta o código relacionado a essa função.

```
const handleConfirmSale = async () => {
    if (!ticketPrice || isNaN(parseFloat(ticketPrice))) {
      Alert.alert("Erro", "Por favor, insira um preco valido.");
      return;
    }

    const price = parseFloat(ticketPrice);
    if (price <= 0) {
      Alert.alert("Erro", "O preco deve ser maior que zero.");
      return;
    }
```

```
12
13      setSellLoading(true);
14      closePriceModal();
15
16      try {
17        const db = firestore();
18
19        if (selectedTicketForSale.isFromFirestore) {
20          const ticketRef = doc(db, 'ingressos_a_venda',
                selectedTicketForSale.id);
21
22          await updateDoc(ticketRef, {
23            isForSale: true,
24            status: 'a_venda',
25            price: price,
26            timestampPublicacao: firestore.FieldValue.serverTimestamp()
27          });
28        } else {
29            const perfisRef = collectionGroup(db, 'perfisUsuarios');
30            const querySnapshot = await getDocs(query(perfisRef,
                where('cpf', '==', selectedCpf)));
31
32
33            if (querySnapshot.empty) {
34              Alert.alert("Erro", "Nenhum usuario encontrado com este
                    CPF. \n\nFaca o cadastro no app TrocaTickets.io e tente
                    novamente",);
35              setSellLoading(false);
36              return;
37            }
38
39            const perfilDoc = querySnapshot.docs[0];
40            const authUidDoDono = perfilDoc.data().authUid ||
                'uid-desconhecido';
41
42          await addDoc(collection(db, 'ingressos_a_venda'), {
43            idOriginal: selectedTicketForSale.id,
44            title: selectedTicketForSale.title,
45            description: selectedTicketForSale.description,
46            location: selectedTicketForSale.location,
47            date: selectedTicketForSale.date,
48            image: selectedTicketForSale.image,
49            ticketCode: selectedTicketForSale.ticketCode,
50            vendedorId: authUidDoDono,
51            cpfVendedor: selectedCpf,
52            isForSale: true,
53            status: 'a_venda',
```

```
            price: price,
            storeToken: APP_STORE_TOKEN,
            timestampPublicacao: firestore.FieldValue.serverTimestamp(),
          });
      }

      Alert.alert("Sucesso", "Ingresso publicado para venda!");
      loadTickets();
    } catch (error) {
      console.error("Erro ao vender ingresso:", error);
      Alert.alert("Erro", error.message);
    } finally {
      setSellLoading(false);
    }
};
```

Listagem 3.1 – Implementação da função que disponibiliza ingresso para revenda.

Para que o ingresso seja enviado para revenda, o usuário deve estar cadastrado no sistema de revenda de ingressos, pois o sistema utiliza o CPF para indexar o ingresso publicado ao perfil, além de precisar dos dados de conta bancária para envio do pagamento. Caso o usuário não tenha cadastro no Aplicativo para Revenda de Ingressos, não é possível publicá-lo para revenda e uma mensagem é exibida solicitando que o cadastro seja realizado.

3.1.3 Estrutura do Aplicativo de Revenda de Ingressos

Para o aplicativo de revenda de ingressos escolhemos o nome *TrocaTickets.io*. Caso seja a primeira vez do usuário acessando o aplicativo, ele deverá acessar a tela de cadastro e inserir seus dados para criar uma conta no *TrocaTickets.io*. Esses dados são nome, e-mail e número de documento, e posteriormente os dados de sua conta bancária. Esse processo é mostrado na Figura 3.4, com o cadastro de um usuário fictício.

Após a publicação do ingresso para revenda, ele fica imediatamente disponível no *TrocaTickets.io*. Como mostrado na Figura 3.5, o usuário fictício, após efetuar o login com sua conta cadastrada, pode acessar a aba “Home”, onde um conjunto de funções JavaScript é acionado para buscar os ingressos disponíveis para revenda no Cloud Firestore. Além disso, ao acessar a aba “Ingressos”, é possível visualizar os ingressos pertencentes ao usuário, como no exemplo da 3.5, onde o usuário fictício pode visualizar seu ingresso, anteriormente publicado para venda.

Para adquirir um ingresso, basta clicar no ingresso desejado, escolher a opção de compra, inserir os dados necessários, escolher a forma de pagamento e confirmar a troca de titularidade clicando em “Compra”. Nesse momento, a API de serviços financeiros é acionada e o sistema fica aguardando a resposta sobre o pagamento. Caso seja negativa, é exibida uma mensagem pedindo para que o usuário tente novamente, e, caso seja positiva, a troca é confirmada e a

Figura 3.4 – Processo de criação de conta no *TrocaTickets.io*.

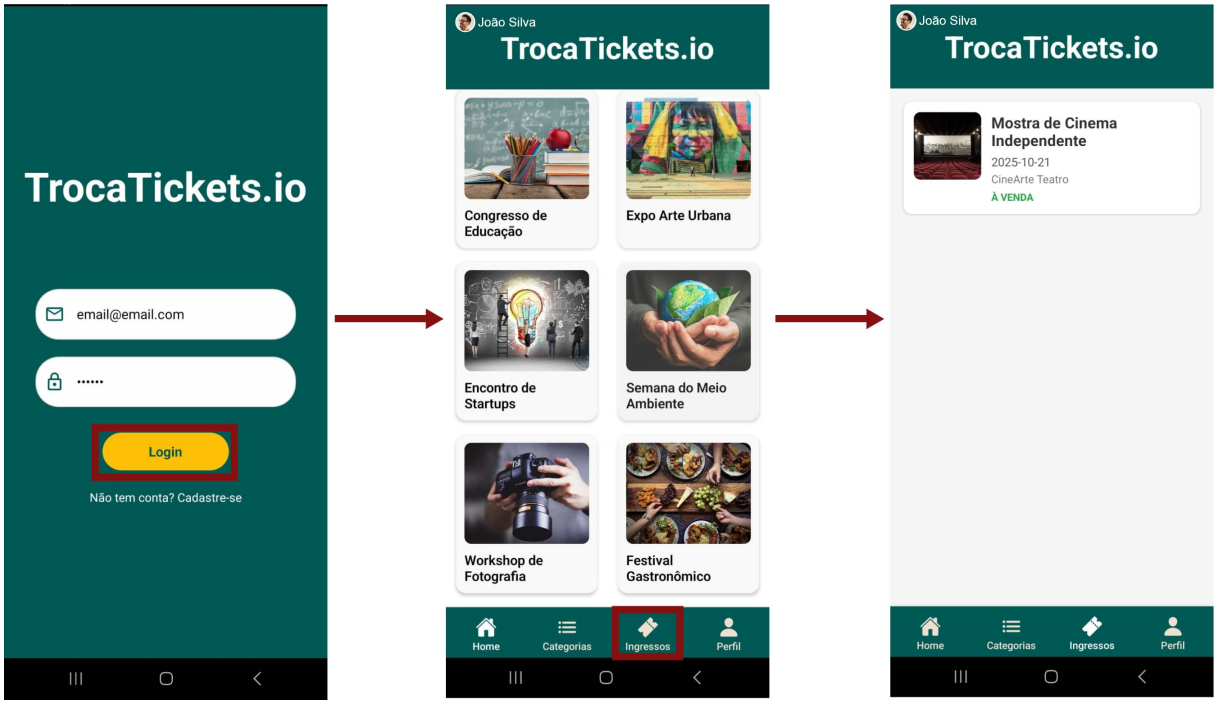

Figura 3.5 – Processo de login e visualização de ingressos pertencentes ao usuário.

alteração dos dados de titularidade do ingresso é efetuada, incluindo o pagamento entre as partes e o comissionamento da plataforma de venda.

As funções utilizadas nesse processo estão no código ilustrado pela Listagem 3.2. Para simular a API de serviços financeiros, foi implementado um servidor que retorna uma resposta após um atraso de 3 segundos. A chamada dessa API também está presente no código ilustrado pela Listagem 3.2.

```
useEffect(() => {
   const unsubscribe = auth().onAuthStateChanged(async (currentUser)
      => {
     if (!currentUser) {
       setLoading(false);
       return;
     }

     try {
       const userProfileRef = firestore()
         .collection('artifacts')
         .doc('default-app-id')
         .collection('users')
         .doc(currentUser.uid)
         .collection('perfisUsuarios')
         .doc(currentUser.uid);

       const doc = await userProfileRef.get();

       if (doc.exists) {
         setUser(doc.data());
       } else {
         console.warn("Perfil do usuario nao encontrado");
       }
     } catch (error) {
       console.error("Erro ao carregar perfil:", error);
     }
   });

   return unsubscribe;
 }, []);

 const paymentMethods = [
   { label: 'Cartao de Credito', value: 'cartao' },
   { label: 'PIX', value: 'pix' }
 ];

 const handleFinalizarCompra = async () => {
   if (!nome.trim() || !documento.trim()) {
     Alert.alert('Erro', 'Por favor, preencha todos os campos');
```

```
40        return;
41      }
42
43      if (!metodoPagamento) {
44        Alert.alert('Erro', 'Por favor, selecione um metodo de
              pagamento');
45        return;
46      }
47
48      setPaymentLoading(true);
49
50      try {
51        const paymentData = {
52          ticketId: ticket.id,
53          valor: ticket.price,
54          metodo: metodoPagamento,
55          comprador: {
56            nome,
57            documento
58          }
59        };
60
61        const paymentResult = await ApiBanco(paymentData);
62
63        if (paymentResult.success) {
64          await updateTicketAfterPayment();
65
66          Alert.alert(
67            'Sucesso',
68            'Pagamento realizado com sucesso! Ingresso reservado para
                  voce.',
69            [{ text: 'OK', onPress: () => navigation.goBack() }]
70          );
71        } else {
72          throw new Error('Falha no pagamento');
73        }
74      } catch (error) {
75        console.error('Erro no processo de pagamento:', error);
76        Alert.alert('Erro', 'Ocorreu um erro ao processar seu pagamento.
              Tente novamente.');
77      } finally {
78        setPaymentLoading(false);
79      }
80    };
81
82    const updateTicketAfterPayment = async () => {
83      try {
```

```
84        await firestore()
85          .collection('ingressos_a_venda')
86          .doc(ticket.id)
87          .update({
88            isForSale: false,
89            status: 'vendido',
90            comprador: {
91              nome,
92              documento
93            },
94            cpfVendedor: user.cpf, /
95            metodoPagamento,
96            dataVenda: firestore.FieldValue.serverTimestamp()
97          });
98
99        await firestore()
100         .collection('ingressos_vendidos')
101         .add({
102           ...ticket,
103           isForSale: false,
104           comprador: { nome, documento },
105           cpfVendedor: user.cpf,
106           metodoPagamento,
107           dataVenda: firestore.FieldValue.serverTimestamp()
108         });
109
110       console.log('Ticket atualizado com sucesso apos pagamento');
111     } catch (error) {
112       console.error('Erro ao atualizar ticket:', error);
113       throw error;
114     }
115   };
```

Listagem 3.2 – Conjunto de funções que efetuam troca de titularidade do ingresso.

Todo esse processo de compra está representado na Figura 3.6, onde o usuário fictício Jorge Lima visualiza sua aba “Ingressos’ ainda vazia, navega para a aba “Home” e seleciona o ingresso para “Mostra de Cinema Independente”, disponibilizado por outro usuário fictício, escolhe a opção de compra, preenche seus dados, escolhe a opção de pagamento e confirma a compra.

Após a confirmação da API de serviços financeiros, a troca de titularidade é realizada, e o ingresso fica disponível na aba “Ingressos”. Como ilustra a Figura 3.7, ao clicar sobre o ingresso, o usuário pode visualizar informações da empresa de venda de ingressos, a qual o sistema reconhece por meio do token único de identificação da loja presente no ingresso. Caso o usuário clique em “Acessar Ingresso”, ele é redirecionado para a plataforma de venda de ingressos

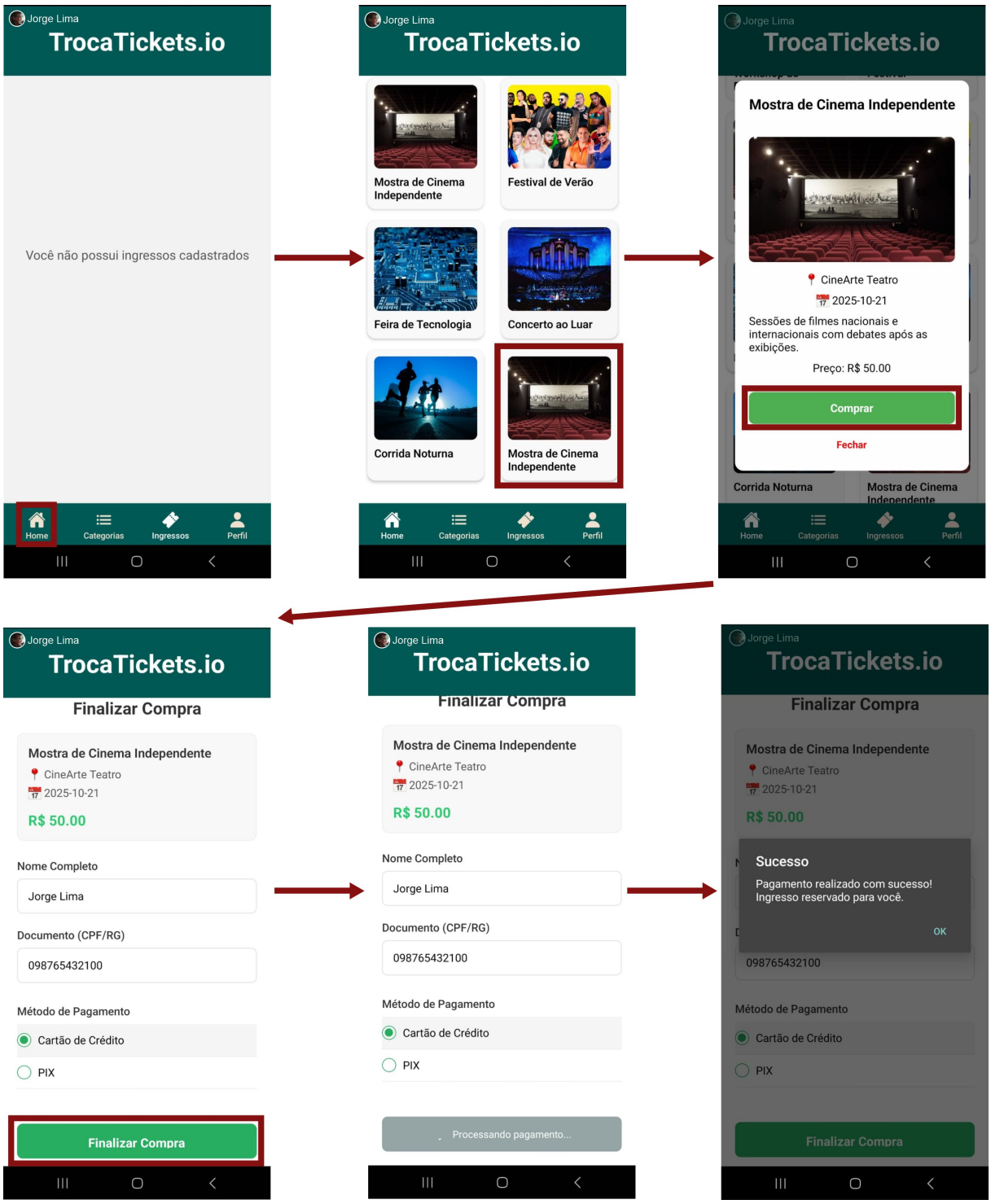

Figura 3.6 – Processo de compra de ingresso no *TrocaTickets.io*.

ou, caso não a tenha instalado, para a loja de aplicativos do sistema operacional.

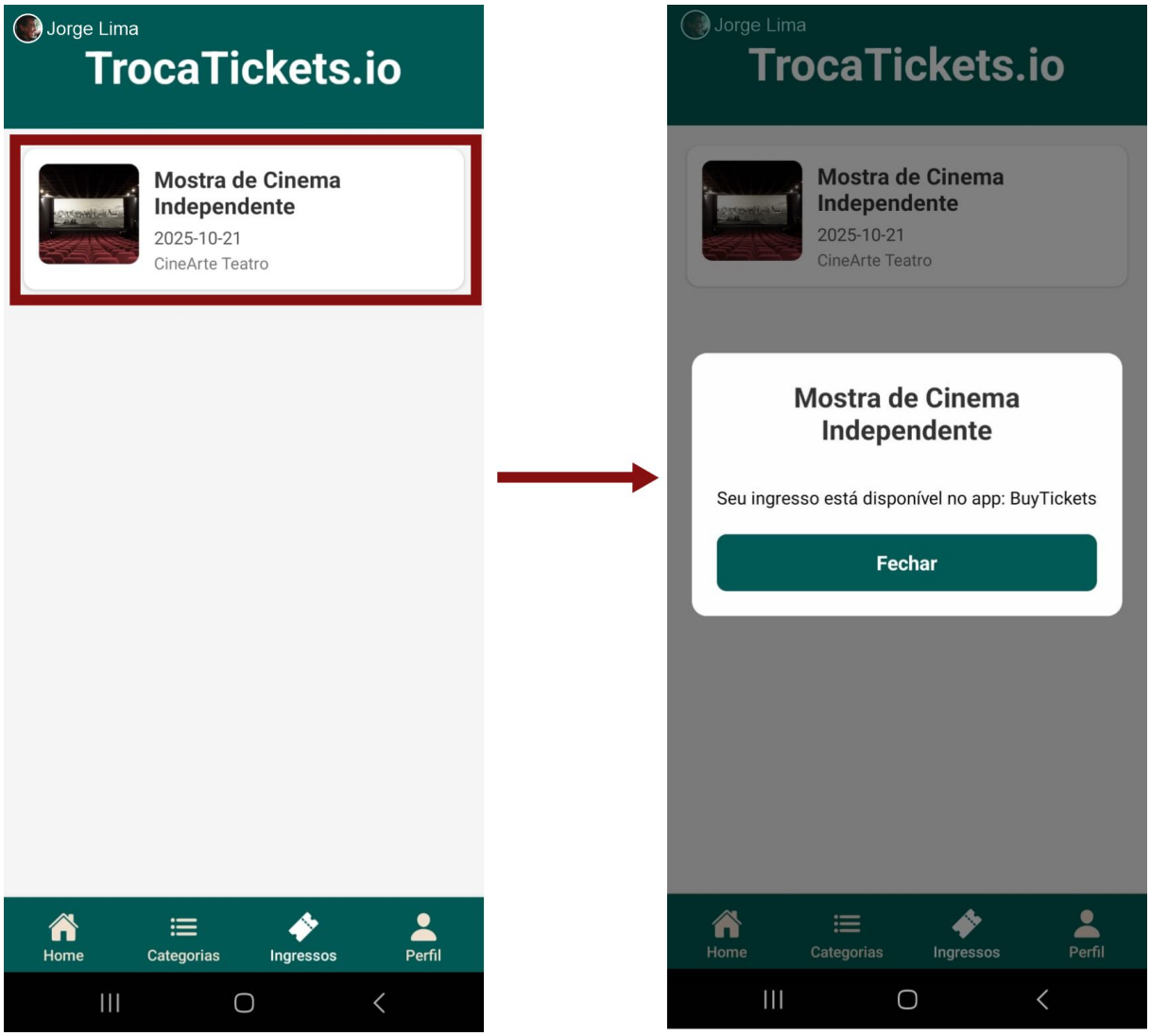

Figura 3.7 – Ingresso disponível no *TrocaTickets.io*.

Ao abrir o aplicativo da plataforma de venda de ingressos, o usuário comprador precisa fazer login em sua conta. Note que a *Loja Virtual para Venda de Ingressos* pode criar uma conta, pois as informações do comprador do ingresso sob revenda foram repassadas à mesma pelo sistema de revenda de ingressos. O ingresso é sempre vinculado ao usuário por meio de seu número de documento e fica disponível na aba “Ingresso”. Esse processo é apresentado na Figura 3.8, onde também é exibida a visualização do ingresso já com a nova titularidade. Por outro lado, o ingresso fica indisponível para troca na aba “home” do aplicativo *TrocaTickets.io*, assim como na aba “Ingressos” da loja BuyTickets, como ilustra a Figura 3.9.

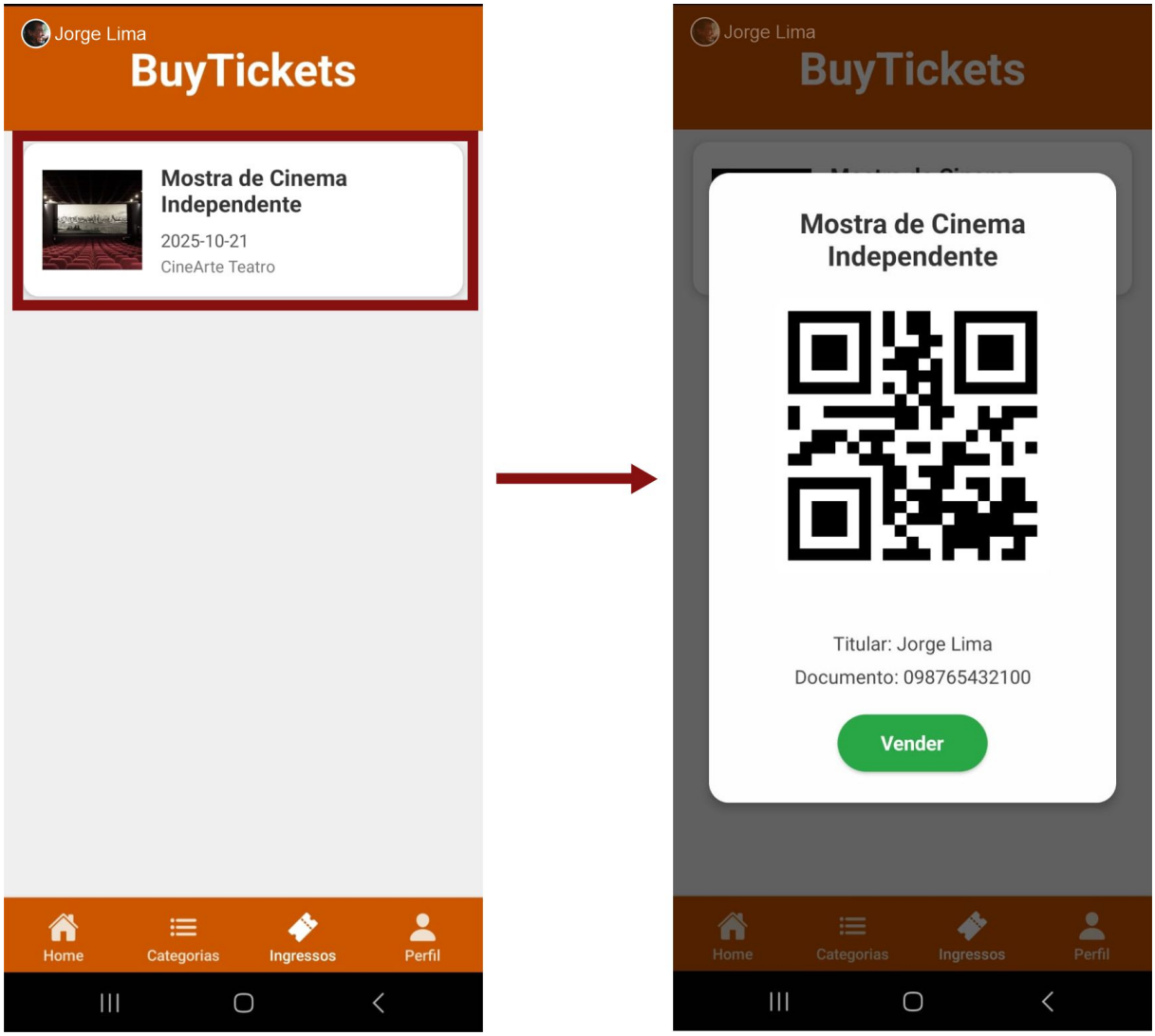

Figura 3.8 – Ingresso sendo acessado pelo novo titular no BuyTickets.

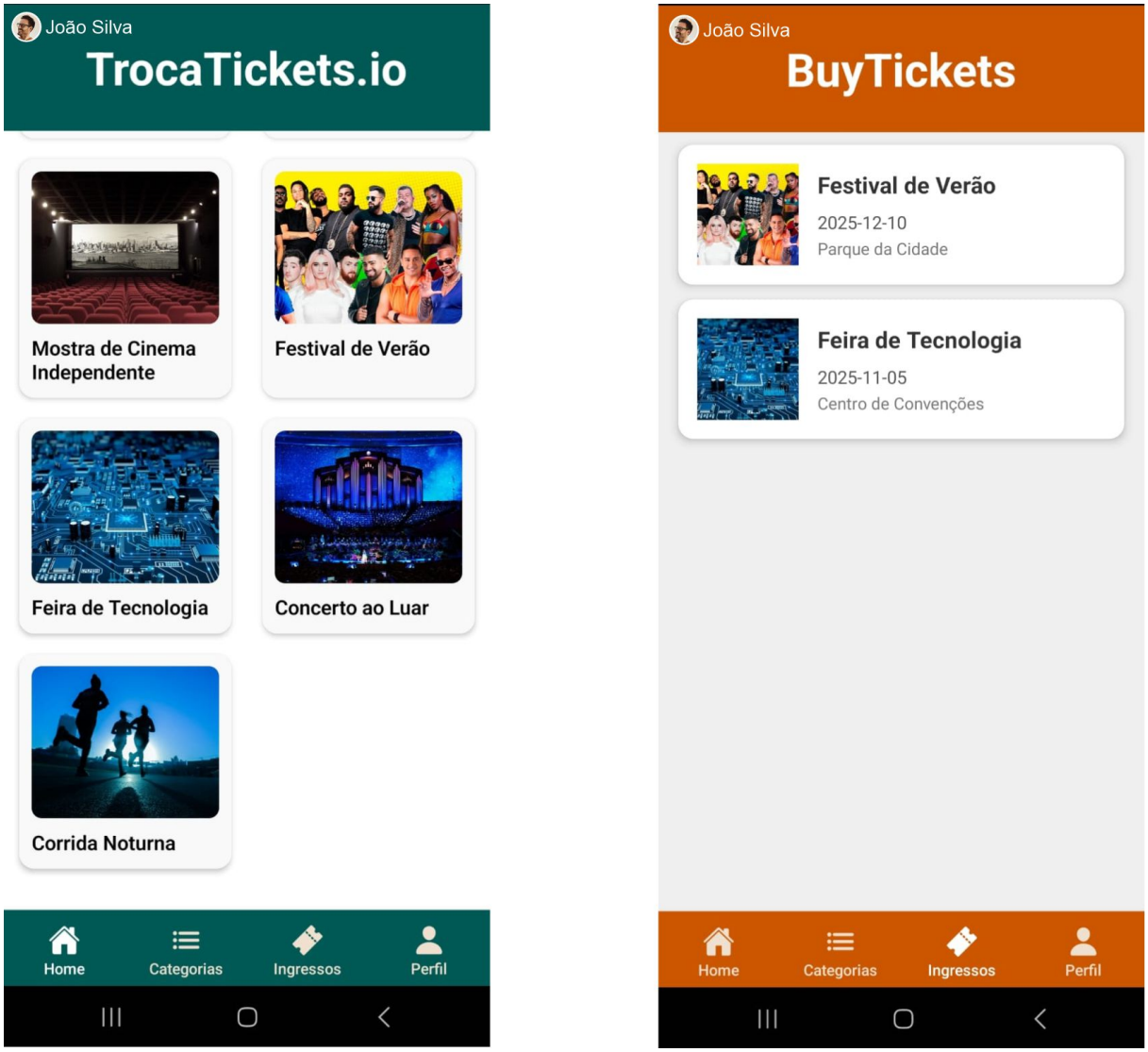

Figura 3.9 – Ingresso indisponível para troca no *TrocaTickets.io* e na aba de ingressos no BuyTickets.

4 Análise Experimental com Teste Baseado em Modelo

Neste capítulo, detalhamos a aplicação da metodologia de Teste Baseado em Modelo (MBT) para realizar uma análise aprofundada da robustez e da corretude funcional do aplicativo "TrocaTickets.io". O objetivo não é comparar diferentes técnicas de teste, mas sim utilizar o MBT como uma ferramenta poderosa para validar a aplicação de forma abrangente, explorando cenários de uso complexos que dificilmente seriam previstos em testes manuais ou modulares. A metodologia de aplicação do MBT foi inspirada no trabalho de Akpinar et al. (AKPINAR et al., 2020).

4.1 Objetivos da Análise

A análise experimental visa atingir os seguintes objetivos específicos:

- Aplicar a metodologia de MBT para validar os principais fluxos de usuário do "TrocaTickets.io";
- Avaliar a capacidade do MBT em identificar falhas de estado e de integração entre as diferentes telas e componentes da aplicação;
- Medir a cobertura de teste alcançada em relação ao modelo funcional do sistema;
- Analisar a eficiência do próprio processo de teste, como o tempo de geração de cenários a partir do modelo;
- Avaliar se a aplicação cumpre com os requisitos centrais propostos nesse projeto.

4.2 Metodologia de Aplicação do MBT

A validação do sistema foi conduzida exclusivamente através da abordagem de Teste Baseado em Modelo, seguindo as etapas descritas abaixo.

4.2.1 Modelagem do Sistema

O primeiro passo consistiu na criação de um modelo formal que representa os fluxos de usuário da aplicação. Utilizando a ferramenta yEd, os principais fluxos, como cadastro, publicação, compra e visualização de ingressos, foram mapeados em um grafo direcionado. Neste modelo:

- Os **vértices (nós)** representam os estados da aplicação, que correspondem às principais telas ou componentes da interface (ex: ‘v_TelaLogin’, ‘v_TelaHome’, ‘v_TelaFinalizarCompra’).
- As **arestas (transições)** representam as ações do usuário (ex: ‘e_ClicarBotaoComprar’) ou eventos do sistema que disparam mudanças de estado.

A Figura 4.1 ilustra o modelo de grafo desenvolvido para representar o processo de login, para análise.

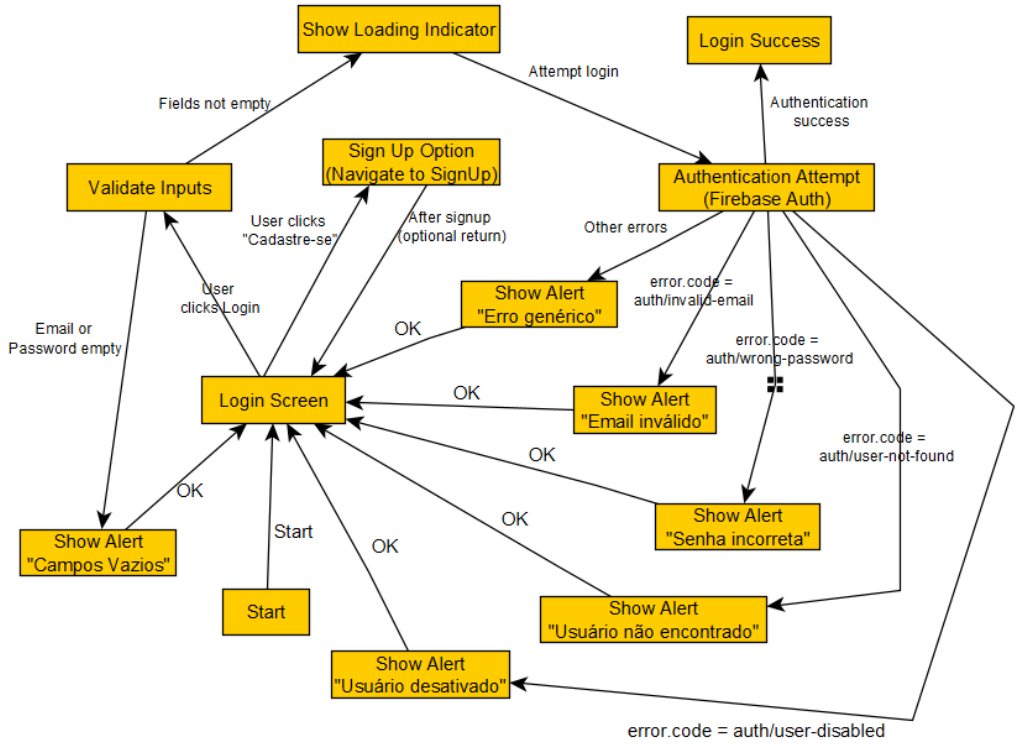

Figura 4.1 – Modelo de grafo funcional utilizado para a análise do processo de login do "Troca-Tickets.io".

4.2.2 Ferramentas e Ambiente de Teste

O ambiente de teste foi configurado com o seguinte conjunto de ferramentas:

- **GraphWalker:** Ferramenta central de MBT, responsável por ler o modelo do grafo e executar os testes, atuando como o “cérebro” do processo (AKPINAR et al., 2020).
- **React Native Testting Library:** Framework de automação de testes para aplicações móveis, que serviu como a ponte entre os comandos de teste gerados pelo GraphWalker (em Java) e as interações com o aplicativo “TrocaTickets.io” (em React Native).
- **yEd Graph Editor:** Ferramenta utilizada para desenhar e exportar o modelo do sistema no formato `graphml`, compatível com o GraphWalker.

4.2.3 Geração e Execução dos Cenários de Teste

O ambiente foi configurado com modelos desenvolvidos no yEd Graph Editor para as telas e processos que desempenham o foco principal desse projeto, como login, cadastro de usuário, publicação de ingressos para revenda, exibição dos ingressos disponíveis e troca de titularidade do ingresso.

Com o ambiente configurado, o GraphWalker foi instruído a percorrer o modelo para os cenários de teste automaticamente. Foi utilizado um gerador de caminho aleatório (`random`) com uma condição de parada baseada na cobertura de 100% das arestas do modelo (`edge_coverage(100%)`). Esta configuração força o executor a navegar pela aplicação até que todas as ações possíveis (arestas) tenham sido testadas pelo menos uma vez, garantindo uma alta cobertura do modelo.

4.3 Métricas de Análise

Para avaliar a eficácia da aplicação do MBT e a robustez do sistema, foram definidas as seguintes métricas:

- **Cobertura do Modelo:** Percentual de vértices e arestas do grafo que foram cobertos durante a execução dos testes.
- **Análise de Falhas Encontradas:** Classificação e descrição dos erros ou comportamentos inesperados identificados durante a execução dos cenários gerados pelo GraphWalker.
- **Performance da Geração de Testes:** Análise do tempo de geração da interface de teste pelo GraphWalker em função da complexidade do modelo (soma do número de vértices e arestas).

4.4 Resultados e Análise

A execução dos testes gerou dados quantitativos e qualitativos que permitiram uma análise detalhada da aplicação.

4.4.1 Cobertura do Modelo Atingida

A execução dos cenários com a condição de parada ‘edge_coverage(100%)‘ garantiu, como esperado, a cobertura completa do modelo funcional, conforme detalhado na Tabela 4.1. Isso significa que todos os estados definidos e todas as transições possíveis entre eles foram exercitados durante os testes.

Tabela 4.1 – Cobertura do modelo de teste.

Elemento do Modelo	Total	Cobertos	Cobertura (%)
Vértices (Estados)	76	76	100%
Arestas (Ações)	100	100	100%

4.4.2 Falhas Críticas Identificadas

A principal vantagem da abordagem foi a descoberta de falhas que ocorriam em sequências de interação não triviais. A Tabela 4.2 descreve alguns dos erros encontrados.

Tabela 4.2 – Exemplos de falhas encontradas pela análise de MBT.

Tipo de Falha	Descrição do Cenário
Falha de Estado	Ao tentar comprar um ingresso, retornar à tela anterior e selecionar outro ingresso, o sistema mantinha o preço do primeiro item selecionado no checkout.
Erro de Navegação	Após um logout, pressionar o botão "voltar" do dispositivo permitia o acesso a uma versão em cache da tela "Meus Ingressos", sem autenticação.
Condição de Corrida Simulada	Se um vendedor cancelasse a venda de um ingresso (no app "BuyTicket") no momento em que um comprador está inserindo dados para compra ou processo de pagamento (no "TrocaTickets.io"), ao confirmar a compra a aplicação do comprador gerava um erro não tratado.

4.4.3 Análise de Performance do GraphWalker

Foi observado que o tempo necessário para o GraphWalker gerar a fonte do teste aumenta de forma consistente com o aumento da complexidade do grafo (quantidade de vértices e arestas), conforme o gráfico da Figura 4.2. Este dado é útil para estimar o esforço de teste em projetos futuros.

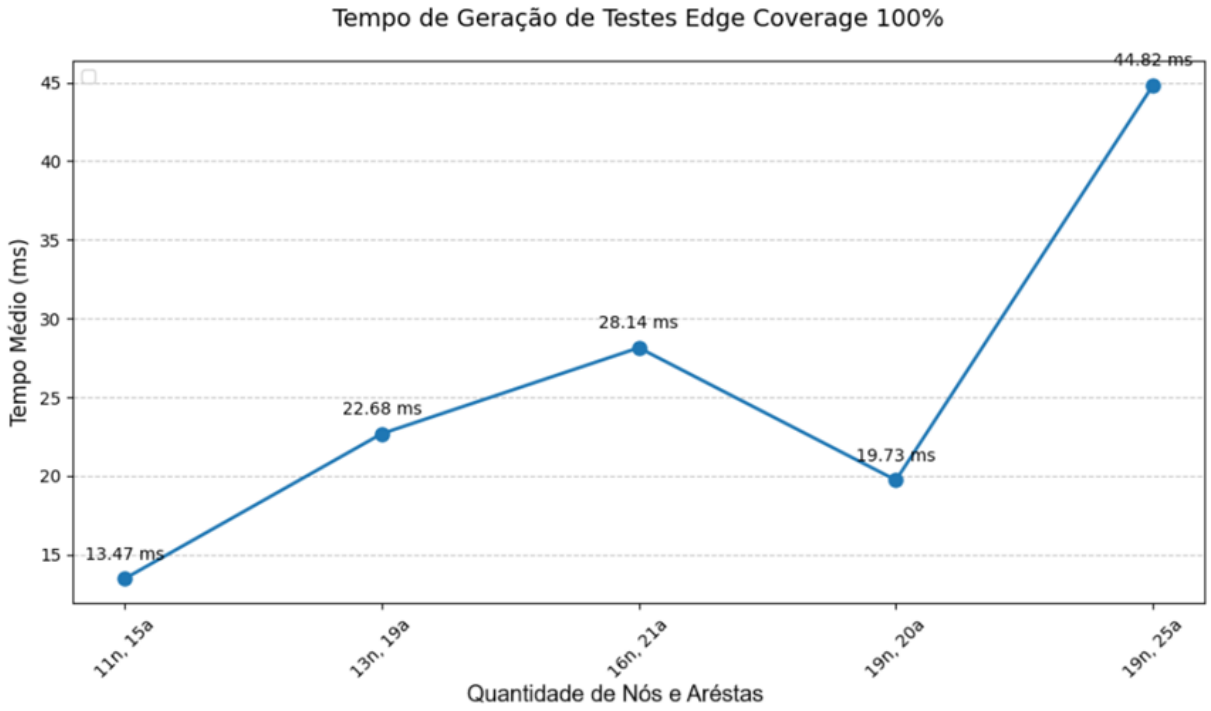

Figura 4.2 – Tempo de geração de testes.

4.5 Discussão

A aplicação da metodologia de Teste Baseado em Modelo provou ser eficaz para a validação do sistema “TrocaTickets.io”. A capacidade de gerar automaticamente milhares de cenários de teste distintos, incluindo caminhos não convencionais, permitiu a identificação de falhas críticas de estado e de navegação que seriam de difícil detecção por meio de testes manuais, e também garantiu que o sistema cumpra com os requisitos principais propostos na ideia do projeto.

Conclui-se que o MBT não apenas validou a funcionalidade principal da aplicação, mas também serviu como uma ferramenta de garantia de qualidade de alto nível, aumentando significativamente a confiança na robustez e na estabilidade do software.

5 Conclusão

Neste trabalho de conclusão de curso foi proposto, implementado e testado um sistema para gerenciar o processo de revenda de ingressos, serviço ainda ausente no Brasil, conforme mostraram os trabalhos correlatos. Neste documento há a descrição da arquitetura do sistema, detalhes sobre seus componentes, incluindo os emulados, tais como plataforma de venda de ingressos e API para simulação de transações financeiras. Um planejamento de testes ajudou a detectar bugs e os corrigir a tempo.

5.1 Desafios e Soluções

Entre os principais desafios enfrentados, destaca-se o planejamento de uma arquitetura robusta que funcionasse com a ideia central do projeto, fornecendo integração com plataformas de vendas variadas com confiabilidade e segurança. Outro desafio foi a garantia de uma comunicação segura e eficaz para o armazenamento, gerenciamento dos dados e autenticação dos usuários, superado pela integração com o *Firebase* (FIREBASE, 2025c).

5.2 Considerações Finais

O desenvolvimento do aplicativo seguiu uma arquitetura robusta e modular, garantindo a segurança das transações de ingressos. A combinação de tecnologias como *React Native*, *Firebase Authentication* e *Cloud Firestore* resultou em um sistema escalável e seguro, alinhado às necessidades modernas de segurança e experiência do usuário.

Além disso, o modelo de testes MBT se mostrou bem eficiente na detecção de falhas de funcionamento da aplicação, detectando problemas não notados anteriormente nos testes manuais e, assim, possibilitando a correção antes que o projeto fosse finalizado. Com a execução dos testes, também foi possível notar que a aplicação cumpre os requisitos centrais propostos no projeto: desenvolver uma aplicação para efetuar troca de ingressos online com segurança.

Pretende-se desenvolver em trabalhos futuros:

- Fidelizar um sistema de pagamentos com uma API de serviços financeiros;
- Investir na usabilidade da aplicação para oferecer melhor experiência aos usuários;
- Evoluir o sistema de autenticação com mais critérios para fornecer mais segurança;
- Integrar um sistema de Blockchain que armazene em um único bloco e de forma imutável as informações oriundas das plataforma de venda de ingressos e as informações do

"TrocaTickets.io". Tal componente permitirá auditorias por parte das plataformas de venda parceiras, o que assegura maior transparência na revenda de ingressos.

Referências

AKPINAR, P. et al. Web application testing with model based testing method: Case study. In: *Proceedings of the 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*. IEEE, 2020. p. 1–5. Disponível em: <https://www.researchgate.net/publication/343953812_Web_Application_Testing_With_Model_Based_Testing_Method_Case_Study>.

BOX, D. et al. *Simple Object Access Protocol (SOAP) 1.1*. 2000. Disponível em: <https://www.w3.org/TR/2000/NOTE-SOAP-20000508/>.

CLOUD, G. *Cloud Firestore Overview*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://cloud.google.com/firestore/docs/overview>.

ENCYCLOPEDIA.COM. *Ticketmaster*. 2024. Disponível em: <https://www.encyclopedia.com/social-sciences-and-law/economics-business-and-labor/businesses-and-occupations/ticketmaster?utm_source=chatgpt.com>.

EVENTBRITE. *Eventbrite*. 2025. Acesso em: 24 jul. 2025. Disponível em: <https://www.eventbrite.com/>.

EVENTIM. *Eventim*. 2025. Acesso em: 24 jul. 2025. Disponível em: <https://www.eventim.com/>.

EXAME. *A empresa dele ficou 18 meses com receita zero, agora vai movimentar R$ 1 bilhão com ingressos*. 2024. Disponível em: <https://exame.com/negocios/a-empresa-dele-ficou-18-meses-com-receita-zero-agora-vai-movimentar-r-1-bilhao-com-ingressos/>.

Facebook. *GraphQL: A query language for your API*. 2015. Disponível em: <https://graphql.org/>.

(FACEBOOK), M. *React Native Documentation*. 2024. Acesso em: 01 ago. 2025. Disponível em: <https://reactnative.dev>.

FANSALE. *FanSale*. 2025. Acesso em: 24 jul. 2025. Disponível em: <https://www.fansale.com/>.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doutorado) — University of California, Irvine, 2000. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>.

FIREBASE. *Cloud Firestore Documentation*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://firebase.google.com/docs/firestore>.

FIREBASE. *Firebase Authentication Documentation*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://firebase.google.com/docs/auth>.

FIREBASE. *Firebase Documentation*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://firebase.google.com/docs>.

FIREBASE. *Firebase for React Native*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://rnfirebase.io>.

FUNCAP. *Criador do Ingresso.com fala sobre desafios de empreender na internet brasileira*. 2011. Disponível em: <https://www.funcap.ce.gov.br/2011/10/19/criador-do-ingressocom-fala-sobre-desafios-de-empreender-na-internet-brasileira/>.

Google. *gRPC: A high-performance, open-source universal RPC framework*. 2015. Disponível em: <https://grpc.io/>.

GOOGLE. *Firebase - Build apps that users love*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://firebase.google.com>.

GOOGLE. *Firebase Authentication - Authenticate users simply and securely*. 2025. Acesso em: 01 ago. 2025. Disponível em: <https://firebase.google.com/products/auth>.

GRAPHWALKER. <http://graphwalker.github.io>. Acesso em: 03 set. 2025.

HAN, J. et al. *Survey on NoSQL database*. 2011. Acesso em: 01 ago. 2025. Disponível em: <https://ieeexplore.ieee.org/document/6108107>.

INGRESSE. *Ingresse*. 2025. Acesso em: 24 jul. 2025. Disponível em: <https://www.ingresse.com/>.

MONIRUZZAMAN, A. B. M.; HOSSAIN, S. A. *NoSQL database: New era of databases for big data analytics—Classification, characteristics and comparison*. 2013. Acesso em: 01 ago. 2025. Disponível em: <https://www.researchgate.net/publication/260244031_NoSQL_Database_New_Era_of_Databases_for_Big_data_Analytics_-_Classification_Characteristics_and_Comparison>.

Mordor Intelligence. *Global Online Event Ticketing Market - Industry*. 2024. Acesso em: 11 ago. 2024. Disponível em: <https://www.mordorintelligence.com/pt/industry-reports/global-online-event-ticketing-market-industry>.

PIA.JP. *Pia.jp*. 2025. Acesso em: 24 jul. 2025. Disponível em: <https://ticket.pia.jp/>.

SEATGEEK. *SeatGeek*. 2025. Acesso em: 24 jul. 2025. Disponível em: <https://seatgeek.com/>.

SIMILARWEB. *Top Websites Ranking*. 2025. Acesso em: 24 jul. 2025. Disponível em: <https://www.similarweb.com/top-websites/>.

STUBHUB. *StubHub*. 2025. Acesso em: 24 jul. 2025. Disponível em: <https://www.stubhub.com/>.

SYMPLA. *Sympla*. 2025. Acesso em: 24 jul. 2025. Disponível em: <https://www.sympla.com.br/>.

TICKET360. *Ticket360*. 2025. Acesso em: 24 jul. 2025. Disponível em: <https://www.ticket360.com.br/>.

TICKETMASTER. *Ticketmaster*. 2025. Acesso em: 24 jul. 2025. Disponível em: <https://www.ticketmaster.com/>.

UTTING, M.; PRETSCHNER, A.; LEGEARD, B. *Practical Model-Based Testing: A Tools Approach*. [S.l.]: Elsevier, 2012.

YED Graph Editor. <https://www.yworks.com/products/yed>. Acesso em: 03 set. 2025.