Universidade de Brasília – UnB
Faculdade de Ciências e Tecnologias em Engenharia – FCTE
Engenharia de Software

Projeto e Desenvolvimento do software TROPA: Teatro de Operações de Airsoft

Autor: Erick Giffoni Felicíssimo
Orientador: Prof. Dr. Daniel Sundfeld

Brasília, DF
2025

Erick Giffoni Felicíssimo

Projeto e Desenvolvimento do software TROPA: Teatro de Operações de Airsoft

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade de Ciências e Tecnologias em Engenharia – FCTE

Orientador: Prof. Dr. Daniel Sundfeld

Brasília, DF

2025

Erick Giffoni Felicíssimo

Projeto e Desenvolvimento do software TROPA: Teatro de Operações de Airsoft/ Erick Giffoni Felicíssimo. – Brasília, DF, 2025-

96 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Daniel Sundfeld

Trabalho de Conclusão de Curso –
Universidade de Brasília – UnB
Faculdade de Ciências e Tecnologias em Engenharia – FCTE , 2025.

1. . 2. . I. Prof. Dr. Daniel Sundfeld. II. Universidade de Brasília. III. Faculdade de Ciências e Tecnologias em Engenharia. IV. Projeto e Desenvolvimento do software TROPA: Teatro de Operações de Airsoft

CDU 02:141:005.6

Erick Giffoni Felicíssimo

Projeto e Desenvolvimento do software TROPA: Teatro de Operações de Airsoft

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 15 de julho de 2025:

Prof. Dr. Daniel Sundfeld
Orientador

Prof. Dr. Andre Luiz Peron Martins Lanna
Convidado 1

Prof. Dr. John Lenon Cardoso Gardenghi
Convidado 2

Brasília, DF
2025

Este trabalho é dedicado a todos aqueles que, assim como eu, querem fazer do mundo um lugar melhor.

Agradecimentos

A Deus, pelo dom gratuito da vida.

Ao meu amado pai, por tudo que consigo e que não consigo expressar em palavras.

À minha família, a qual sempre amei.

Aos meus amigos, que são poucos.

Aos meus colegas que, cada um à própria maneira, me ajudaram a chegar até aqui.

Aos meus professores, peças essenciais na minha formação.

À minha amada Katriely por, com amor e ternura, acreditar em mim, me apoiar e incentivar.

A mim, pela perseverança, dedicação, humildade, pelos erros e acertos. Pela fé em Jesus, e por acreditar que consigo, mesmo quando os ventos não parecem favoráveis.

Jesus respondeu:
– Eu sou o caminho, a verdade
e a vida; ninguém vem ao Pai
senão por mim.
(Bíblia Sagrada, João 14, 6)

Resumo

O Airsoft é um esporte de simulação tática que tem crescido no Brasil, exigindo cada vez mais ferramentas de apoio à sua organização. Diante desse cenário, este trabalho tem como objetivo o projeto completo e o desenvolvimento do MVP do software chamado TROPA: Teatro de Operações de Airsoft. A aplicação visa oferecer suporte à realização de jogos, com funcionalidades como gestão de operadores, eventos, equipes e bilheteria digital. Para isso, adotou-se uma metodologia baseada nas boas práticas da Engenharia de Software, com foco na definição de requisitos, elaboração de documentação arquitetural e uso de tecnologias modernas como NestJS, React Native com Expo, PostgreSQL, Docker, Prisma e Stripe. O sistema foi estruturado de forma modular e implantado na plataforma Azure. Foi testado por usuários reais, cujos *feedbacks* contribuíram para validar a proposta e identificar oportunidades de melhoria. Os resultados demonstram que os principais requisitos funcionais e não funcionais foram atendidos, ainda que parcialmente em alguns casos, e que o MVP foi bem recebido pelo público-alvo. O trabalho também discute desafios enfrentados e apresenta direções futuras, como a migração para microsserviços, implementação de um *gateway* seguro, integração com redes sociais e lançamento da versão *web*.

Palavras-chave: Airsoft. Tropa. Desenvolvimento de Software.

Abstract

Airsoft is a tactical simulation sport that has been growing in popularity in Brazil, creating a growing demand for tools that support its organization. In this context, this work aims to design and develop the MVP of a software called TROPA: Theater of Airsoft Operations. The application is intended to support the management of games, including functionalities such as player and team management, event scheduling, and digital ticketing. The methodology followed software engineering best practices, emphasizing requirements definition, architectural documentation, and the use of modern technologies such as NestJS, React Native with Expo, PostgreSQL, Docker, Prisma, and Stripe. The system was built with a modular architecture, deployed on the Azure platform, and tested by real users. Their feedback helped validate the proposal and identify areas for improvement. The results show that the main functional and non-functional requirements were met—at least partially—and that the MVP was positively received by the target audience. This work also discusses the challenges encountered and outlines future directions, including a migration to microservices, implementation of a secure gateway, social features integration, and development of a web version.

Key-words: Airsoft. Tropa. Software Development.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface* (Interface de Programação de Aplicações)
AWS	*Amazon Web Services* (Serviços Web da Amazon)
CIA	*Confidentiality, Integrity, Availability* (Confidencialidade, Integridade, Disponibilidade)
DER	Diagrama de Entidade e Relacionamento (*Entity-Relationship Diagram*)
DLD	Diagrama Lógico de Dados (*Logical Data Diagram*)
HTTPS	*Hypertext Transfer Protocol Secure* (Protocolo Seguro de Transferência de Hipertexto)
MQTTS	*Message Queuing Telemetry Transport Secure* (Protocolo Seguro para Transporte de Mensagens)
MVC	*Model–View–Controller* (Modelo–Visão–Controlador)
NIST	*National Institute of Standards and Technology* (Instituto Nacional de Padrões e Tecnologia dos Estados Unidos da América)
ORM	*Object-Relational Mapping* (Mapeador Relacional de Objetos)
SDL	*Security Development Lifecycle* (Ciclo de Vida de Desenvolvimento de Segurança)
SQL	*Structured Query Language* (Linguagem de Consulta Estruturada)
SSE	*Secure Software Engineering* (Engenharia de Software Segura)
STRIDE	*Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege* (Falsificação, Adulteração, Repúdio, Divulgação de Informação, Negação de Serviço, Elevação de Privilégio)
TROPA	Teatro de Operações de Airsoft

Sumário

1 Introdução

Desenvolvimento de Software é um processo iterativo, incremental e contínuo (PRESSMAN, 2011). Ele faz parte dos cinco processos essenciais da Engenharia de Software e é o que transforma subjetividade em "materialidade", uma vez que implementa os requisitos e a arquitetura propostos de um sistema utilizando código. Existem diversas linguagens de programação, tecnologias, ferramentas e metodologias para realizar-se o desenvolvimento de um software, e a escolha de qual (quais) utilizar dependerá de vários fatores, como facilidade de uso, eficiência, adequação funcional e familiaridade, por exemplo.

A Engenharia de Software pode ser entendida como uma abordagem de engenharia que permite o projeto, desenvolvimento, gerenciamento, a manutenção e entrega de software. Essa abordagem é sistemática, metodológica, disciplinada, baseada em processos (PRESSMAN, 2011). Estes são formados por um conjunto de atividades e de tarefas, cada qual atendendo suas responsabilidades de maneira adaptável, flexível e com foco na entrega dentro do prazo e com qualidade.

De acordo com Pressman (2011), a Engenharia de Software é formada por 5 (cinco) processos básicos que, geralmente, acontecem de maneira contínua e iterativa ao longo do projeto: comunicação, planejamento, projeto, desenvolvimento e entrega. Cada iteração produzirá um incremento do software, o qual torna-se mais completo. A descrição dos processos apresenta-se a seguir:

- **Comunicação.** Durante essa etapa o principal objetivo é conversar com o cliente para entender os objetivos e requisitos do projeto;
- **Planejamento.** Organizar o que será feito, quais são as tarefas e atividades, os riscos envolvidos, prazos, cronogramas etc;
- **Projeto.** Modelagem do software em relação à arquitetura, aos componentes e às interfaces de usuário. A ideia é ter uma visão do todo;
- **Desenvolvimento.** É nessa etapa que o projeto será construído, códigos serão feitos e testes executados;
- **Entrega.** O software resultante, ou o incremento dele, será entregue ao cliente, que o usará e fornecerá retorno (*feedback*) para a equipe de engenharia.

O propósito dessa monografia é, a partir dos conceitos da Engenharia de Software e de cada um dos processos dela, fazer o projeto e a construção de um software voltado

para os jogos de *Airsoft*. A Seção 3.1 apresentará com mais detalhes do que se trata esse sistema.

Além disso, um assunto muito importante, muitas vezes negligenciado nos processos de desenvolvimento, é a segurança de software. Nesse sentido, também é do intuito desse trabalho falar sobre o assunto, bem como aplicá-lo no projeto e no desenvolvimento do sistema proposto.

1.1 Objetivo

Projetar todo o sistema e desenvolver o MVP (Mínimo Produto Viável) do nomeado TROPA: Teatro de Operações de *Airsoft*. Esse software irá gerenciar a participação de usuários em operações do esporte, além de permitir a venda, divulgação e organização de eventos relacionados. Haverá, também, outras funcionalidades no sistema, como sistema de pontuação e formação de *ranking*, conclusão de missões e possibilidade de receber premiações.

1.1.1 Objetivos específicos

- Fazer a documentação pertinente do sistema, a saber: documento de visão, de arquitetura, levantamento e modelagem de requisitos;
- Desenvolver o software com as tecnologias adequadas às necessidades conforme a documentação elaborada;
- Desenhar protótipos de baixa fidelidade para a interface de usuário;
- Realizar testes (tanto livres quanto guiados) com usuários;
- Coletar *feedbacks* dos usuários que testarem o sistema.

1.2 Metodologia

A metodologia para cumprir com o objetivo e a proposta desse trabalho contará com as seguintes atividades:

- Documentação do sistema: inclui documento de visão e de arquitetura; levantamento e modelagem de requisitos;
- Desenvolvimento do sistema: etapa de concretização do MVP por meio da programação. Esse processo acontecerá utilizando as melhores práticas, ferramentas e tecnologias de Engenharia de Software adequadas à solução proposta;

- Ajustes finais: período reservado para resolução de quaisquer intercorrências e/ou modificações necessárias na documentação ou no sistema.

Para conseguir cumprir com as atividades da metologia proposta, as diversas diretrizes explicadas por Pressman e Maxim (2021), no livro Engenharia de Software, foram tomadas como referência.

1.3 Organização

O presente trabalho está dividido em vários Capítulos e Seções. O Capítulo 1 apresenta a introdução, os objetivos, a metodologia e a organização do trabalho. A Seção 2.1 discorre sobre a história, as regras, o crescimento e outros aspectos do *Airsoft*. Já a Seção 2.2 trata sobre tecnologias e conceitos relevantes que foram utilizados para o desenvolvimento do trabalho.

O Capítulo 3 mostra os resultados e a discussão sobre o trabalho e o sistema. A Seção 3.1 é o documento de visão do TROPA, apresentando, dentre outros tópicos, o posicionamento e a visão geral do produto. A Seção 3.2 é o documento resultado da engenharia de requisitos do software TROPA. Na Seção 3.3, o documento de arquitetura é evidenciado, o qual apresenta o sistema de forma ampla perante diferentes visões arquitetônicas.

A Seção 3.4 discorre sobre a concretização do MVP TROPA, abordando o que foi desenvolvido. Já a 3.5 apresenta as principais telas da interface de usuário do aplicativo. A Seção 3.6 explica o processo de validação da aplicação, que envolveu testes com usuários. Finalmente, o Capítulo 4 conclui esse trabalho e indica os próximos passos na evolução do TROPA.

2 Desenvolvimento

2.1 Airsoft

2.1.1 Introdução

O *Airsoft* é um esporte de simulação militar que tem crescido mundialmente, proporcionando uma experiência imersiva para entusiastas de táticas militares e operações estratégicas. Caracterizado pelo uso de simulacros de armas de fogo que disparam pequenas esferas plásticas, o Airsoft combina elementos de competição, estratégia e trabalho em equipe. Diferente de outros esportes de combate, como o *Paintball*, o Airsoft se baseia na honestidade dos jogadores, já que os projéteis não deixam marcas visíveis (CANDIDO, 2024).

Esta Seção explora a história do Airsoft, suas regras e funcionamento, os equipamentos utilizados, além de abordar sua regulamentação e crescimento no Brasil. Compreender o contexto desse esporte é fundamental para justificar a importância do software desenvolvido neste trabalho, que busca aprimorar a organização e a experiência dos jogadores.

2.1.2 História do Airsoft

O Airsoft surgiu no Japão na década de 1970, como uma alternativa legal ao uso de armas de fogo, que eram fortemente reguladas no país. As primeiras armas de Airsoft eram réplicas funcionais, mas incapazes de disparar munição real, sendo utilizadas para treino e recreação (PLUS, 2023).

Com o tempo, o esporte se popularizou e começou a se expandir para outros países, como Estados Unidos e Reino Unido, onde surgiram novas modalidades e regulamentações específicas para sua prática. A partir dos anos 2000, o Airsoft ganhou força na América Latina, especialmente no Brasil, onde a comunidade cresceu exponencialmente, impulsionada por eventos temáticos e maior acessibilidade aos equipamentos (COMBAT, 2023a).

No Brasil, a prática do Airsoft foi regulamentada pelo Exército Brasileiro, que estabeleceu normas para a comercialização e uso dos simulacros, exigindo que as armas possuam ponteiras laranja ou vermelha para diferenciação de armamentos reais (AIRSOFTPEDIA, 2023).

2.1.3 Regras e Funcionamento

O Airsoft é um esporte altamente versátil, com diferentes tipos de jogos e regras, dependendo do evento ou do grupo organizador. No entanto, algumas diretrizes são comuns a todas as modalidades:

2.1.3.1 Modalidades de Jogo

Os jogos de Airsoft podem assumir diversos formatos, entre os quais se destacam:

- **Mata-mata**: Equipes competem até que todos os membros do time adversário sejam eliminados.
- **Resgate de reféns**: Um grupo de jogadores deve resgatar um refém, enquanto outro grupo defende o local.
- **Captura de bandeira**: Equipes tentam capturar a bandeira do adversário e levá-la até sua base.
- **Simulações militares**: Jogos mais elaborados, que podem durar horas ou até dias, envolvendo táticas avançadas.

2.1.3.2 Honestidade e Autorregulação

Diferentemente de esportes como o *Paintball*, onde a tinta marca os jogadores atingidos, o Airsoft funciona com base na honestidade dos participantes. Cada jogador deve declarar quando foi atingido e sair da rodada ou aguardar o tempo de respawn, dependendo da regra do jogo.

2.1.3.3 Eliminação e Sinalização

Para garantir a segurança e a clareza durante os jogos, alguns sinais são universalmente aceitos:

- Levantar a mão ao ser atingido para sinalizar a eliminação.
- Uso de coletes vermelhos ou fitas para indicar jogadores fora da partida.
- Comunicação clara com os demais jogadores para evitar disparos desnecessários em jogadores já eliminados.

2.1.4 Equipamentos e Segurança

A prática do *Airsoft* requer o uso de equipamentos específicos para garantir tanto a autenticidade das simulações quanto a segurança dos participantes (AVENTURA, 2023). São listados, nas seções a seguir, os principais componentes.

2.1.4.1 Simulacros de Armas

As réplicas utilizadas no Airsoft, conhecidas como *simulacros*, são classificadas como armas de pressão que disparam esferas plásticas, geralmente de 6 mm de diâmetro. Existem diferentes tipos de mecanismos de propulsão:

- **Elétricas (AEG - *Automatic Electric Gun*)**: Utilizam baterias recarregáveis para acionar um motor que comprime uma mola, liberando ar para propelir o projétil.
- **Gás (GBB - *Gas Blowback*)**: Funcionam com gás comprimido, como propano ou *green gas*, proporcionando um recuo mais realista.
- **Mola (Spring)**: Necessitam ser armadas manualmente a cada disparo, sendo comuns em rifles de precisão (*snipers*).

2.1.4.2 Equipamentos de Proteção Individual (EPIs)

A segurança é primordial no Airsoft. O uso de EPIs adequados é obrigatório para prevenir lesões:

- **Proteção Ocular**: Óculos ou máscaras com lentes de policarbonato resistentes a impactos são essenciais para proteger os olhos de possíveis danos causados pelos projéteis.
- **Proteção Facial**: Máscaras que cobrem o rosto inteiro ajudam a prevenir lesões dentárias e faciais.
- **Vestuário Adequado**: Roupas de manga longa, luvas e joelheiras oferecem proteção adicional contra impactos e abrasões.

2.1.4.3 Regras de Segurança

Para garantir a integridade de todos os participantes, algumas regras de segurança são universalmente adotadas:

- **Transporte Seguro**: As réplicas devem ser transportadas em cases ou bolsas apropriadas, evitando exposição pública que possa causar alarmes ou mal-entendidos (AIRSOFTPEDIA, 2023).
- **Área de Jogo Delimitada**: As partidas devem ocorrer em locais específicos, afastados de áreas públicas, com sinalização adequada para informar sobre a atividade em andamento.

- **Manutenção Regular**: Verificações periódicas nos equipamentos garantem seu bom funcionamento e previnem acidentes.

2.1.5 Airsoft no Brasil

2.1.5.1 Regulamentação Legal

No Brasil, o Airsoft é regulamentado por legislações específicas que visam assegurar a prática segura do esporte. As principais diretrizes incluem:

- **Portaria nº 002-COLOG, de 26 de fevereiro de 2010**: Estabelece normas para a comercialização e uso de armas de pressão, incluindo as de Airsoft, classificando-as como equipamentos controlados pelo Exército Brasileiro (COMBAT, 2023b).
- **Decreto nº 10.030, de 30 de setembro de 2019 (R-105)**: Define os produtos controlados pelo Exército e as responsabilidades dos usuários, comerciantes e colecionadores (AIRSOFT, 2023).

Essas regulamentações determinam que as réplicas devem possuir ponteiras laranja ou vermelhas para diferenciá-las de armas de fogo reais e que sua comercialização é permitida apenas para maiores de 18 anos. Além disso, o transporte deve ser realizado de forma discreta, sempre acompanhado de nota fiscal e em embalagem que impeça o uso imediato do equipamento.

2.1.5.2 Crescimento e Comunidade

Desde sua introdução no país, o Airsoft tem experimentado um crescimento significativo. Fatores que contribuíram para essa expansão incluem:

- **Popularização de Eventos Temáticos**: A realização de eventos e partidas temáticas atraiu entusiastas de simulações militares e jogos de estratégia.
- **Acesso Facilitado a Equipamentos**: O aumento de lojas especializadas e a facilidade de importação tornaram os equipamentos mais acessíveis aos praticantes.
- **Formação de Clubes e Associações**: A criação de grupos organizados promoveu a disseminação de informações, treinamentos e a padronização de regras, fortalecendo a comunidade.

Atualmente, diversas cidades brasileiras possuem campos dedicados à prática do Airsoft, e a comunidade continua a crescer, promovendo eventos de grande escala e fomentando o espírito de camaradagem e disciplina entre os participantes.

2.1.6 Conclusão

O Airsoft se consolidou como uma atividade que combina esporte, estratégia e recreação, oferecendo aos seus praticantes uma experiência imersiva em simulações táticas. No Brasil, apesar dos desafios iniciais relacionados à regulamentação e ao acesso a equipamentos, o esporte encontrou um terreno fértil para seu desenvolvimento, contando com uma comunidade engajada e em constante expansão.

A compreensão das regras, do uso adequado dos equipamentos e da legislação vigente é fundamental para a prática segura e responsável do Airsoft. Além disso, o fortalecimento da comunidade e a promoção de eventos contribuem para a disseminação do esporte e para a integração dos participantes.

2.2 Suporte Tecnológico

O desenvolvimento do software **TROPA: Teatro de Operações de Airsoft** baseou-se em uma ampla gama de tecnologias e conceitos da Engenharia de Software. Este Capítulo apresenta os fundamentos teóricos das linguagens, ferramentas, dos *frameworks* e serviços utilizados, bem como a revisão de literatura que embasa o projeto.

2.2.1 TypeScript

O **TypeScript** é uma linguagem de programação criada pela Microsoft em 2012 com o objetivo de superar as limitações do *JavaScript*, de forma a oferecer uma experiência mais robusta para o desenvolvimento de sistemas escaláveis. Essa linguagem é um superconjunto de *JavaScript*, o que significa que todo código *JavaScript* é válido em *TypeScript*, permitindo uma adoção gradual por desenvolvedores (MICROSOFT, 2023a).

O principal diferencial do TypeScript é seu sistema de tipagem estática opcional. Essa característica permite a definição explícita de tipos de variáveis, parâmetros de funções e retornos, aumentando a segurança e reduzindo erros comuns (REINEHR, 2020). Além disso, o TypeScript é orientado a objetos, oferecendo suporte a classes, interfaces, herança e outros conceitos típicos de linguagens como *Java* e *C#*.

Outro conceito importante é o *transpilation*, no qual o código TypeScript é convertido para *JavaScript*, garantindo compatibilidade com navegadores e ambientes de execução (MICROSOFT, 2023a). Em outras palavras, ocorre uma "compilação" do código original, só que para outra linguagem (*JavaScript*) em que possa ser interpretado e executado, ou compilado.

Desde seu lançamento, o TypeScript tem se destacado como uma das linguagens mais utilizadas em projetos de grande escala. Relatórios como o Stack Overflow Developer Survey (OVERFLOW, 2022) apontam o TypeScript como uma das linguagens mais

amadas, consolidando-se como escolha padrão em *frameworks* modernos, como Angular e NestJS.

A tabela 1 faz uma comparação entre essas duas linguagens. No contexto da aplicação TROPA, o TypeScript apresentou mais vantagens de ser utilizado do que o JavaScript.

Tabela 1 – Comparação entre TypeScript e JavaScript

Critério	**JavaScript**	**TypeScript**
Tipagem	Linguagem de tipagem dinâmica.	Linguagem de tipagem estática e opcional, detecta erros em tempo de desenvolvimento.
Escalabilidade	Pode dificultar manutenção em sistemas grandes.	Ideal para projetos escaláveis, com suporte a interfaces, tipos e contratos de código.
IDE e Ferramentas	Suporte básico a sugestões e validações.	Suporte avançado a autocompletar, refatoração e verificação de tipos.
Legibilidade e Robustez	Pode levar a bugs silenciosos por falta de verificação.	Código mais legível e robusto, com validações automáticas pelo compilador.
Compatibilidade	Executado diretamente em navegadores e Node.js.	Compila para JavaScript, mantendo compatibilidade com o ecossistema existente.

No desenvolvimento do software TROPA, o TypeScript foi utilizado tanto no *backend* quanto no *frontend*, garantindo:

- **Manutenção facilitada**: A tipagem estática reduz a ocorrência de erros e melhora a clareza do código.
- **Padronização**: A mesma linguagem foi empregada nas duas camadas, simplificando a integração entre os módulos do sistema.

2.2.2 Framework Backend: NestJS

O **NestJS** é um framework para desenvolvimento de aplicações *backend* em Node.js, projetado com foco em modularidade, escalabilidade e produtividade. Inspirado no Angular, o NestJS adota conceitos como injeção de dependências e decoradores, permitindo uma arquitetura organizada e facilmente extensível (MYSLIWIEC, 2023).

Um dos conceitos centrais do NestJS é a arquitetura modular, que organiza o código em módulos independentes. Essa abordagem promove o desacoplamento, facilitando a manutenção e a escalabilidade. Além disso, o *framework* utiliza decoradores para definir controladores, serviços e *middlewares*, seguindo o padrão MVC (*Model-View-Controller*).

Outro aspecto importante é a compatibilidade com ferramentas como Swagger (documentação de APIs) e RabbitMQ (mensageria), que serão detalhadas posteriormente.

Lançado em 2017 por Kamil Myśliwiec, o NestJS foi desenvolvido para suprir a necessidade de um *framework backend* completo para Node.js, similar ao que Angular representa no *frontend*. Sua popularidade tem crescido exponencialmente, tornando-se uma escolha preferida em projetos corporativos (MYSLIWIEC, 2023).

No TROPA, o NestJS foi utilizado para estruturar o *backend*, gerenciar a comunicação com o banco de dados, lidar com autenticação e autorizações e integrar serviços externos, como e-mails e pagamento.

2.2.3 Framework Frontend: React Native e Expo

O **React Native** é um *framework* desenvolvido pelo Facebook em 2015, que permite o desenvolvimento de aplicativos móveis multiplataforma a partir de uma única base de código. O **Expo** complementa o React Native, oferecendo ferramentas para prototipagem rápida e testes em dispositivos reais (PLATFORMS, 2023).

React Native utiliza o conceito de *bridge*, conectando código JavaScript a componentes nativos do sistema operacional, como botões e barras de navegação. Essa abordagem garante desempenho próximo ao de aplicativos nativos, com a vantagem de um desenvolvimento unificado.

O Expo adiciona funcionalidades como a criação de *builds* automáticas e suporte integrado a bibliotecas nativas, simplificando o fluxo de trabalho do desenvolvedor.

React Native foi criado para superar as limitações de soluções híbridas anteriores, como Apache Cordova. Desde seu lançamento, tornou-se um dos *frameworks* mais populares para desenvolvimento móvel, sendo utilizado por empresas como Airbnb e Uber Eats (PLATFORMS, 2023).

No TROPA, o React Native e o Expo foram utilizados para desenvolver o aplicativo móvel, oferecendo:

- **Praticidade**: O Expo simplificou os testes em dispositivos Android e iOS.
- **Imersão**: A interface foi projetada para proporcionar uma experiência intuitiva e eficiente aos usuários.

2.2.4 Banco de Dados: PostgreSQL

O **PostgreSQL** é um sistema de gerenciamento de banco de dados relacional (SGBDR) *open-source*, conhecido por sua robustez e suporte a funcionalidades avança-

das, como armazenamento de JSON (*JavaScript Object Notation*) e transações complexas (GROUP, 2023).

Bancos de dados relacionais organizam informações em tabelas relacionadas por chaves primárias e estrangeiras. O PostgreSQL, além de suportar esse modelo, oferece:

- Índices avançados para melhorar a performance de consultas.
- Controle de concorrência multiversionamento (MVCC), garantindo transações seguras.
- Suporte a extensões, como PostGIS, para processamento de dados geoespaciais.

Criado em 1986 como parte do projeto POSTGRES na Universidade da Califórnia, Berkeley, o PostgreSQL se consolidou como um dos SGBDs mais populares do mundo, frequentemente utilizado em grandes empresas (GROUP, 2023).

O PostgreSQL armazena as informações críticas do TROPA, incluindo:

- Registros de usuários (operadores, organizadores e equipes).
- Operações e vendas de ingressos.

2.2.5 Mensageria: RabbitMQ

A troca de mensagens entre serviços é um aspecto essencial em sistemas distribuídos. O **RabbitMQ** é um sistema de mensageria robusto que implementa o protocolo AMQP (*Advanced Message Queuing Protocol*). Ele é amplamente utilizado em aplicações que exigem comunicação assíncrona entre diferentes componentes ou serviços.

O RabbitMQ organiza o fluxo de mensagens em filas (*queues*), onde produtores (*producers*) enviam mensagens e consumidores (*consumers*) as processam. Uma de suas principais vantagens é a possibilidade de desacoplar serviços, permitindo que cada componente opere de forma independente (SOFTWARE, 2023a).

Além disso, o RabbitMQ suporta diferentes padrões de troca de mensagens:

- **Fila direta (*direct*):** A mensagem é enviada para uma fila específica.
- **Broadcast (*fanout*):** A mensagem é replicada para todas as filas vinculadas a um *exchange*.
- **Roteamento baseado em padrões (*topic*):** Mensagens são enviadas para filas específicas com base em um padrão de roteamento.

Essas características tornam o RabbitMQ uma ferramenta poderosa para aplicações que requerem alta escalabilidade e resiliência.

O RabbitMQ foi desenvolvido em 2007 pela Rabbit Technologies e, posteriormente, adquirido pela VMware em 2010. Desde então, ele se tornou uma escolha popular em sistemas de grande escala devido à sua estabilidade e suporte a diversos protocolos (SOFTWARE, 2023a).

No TROPA, o RabbitMQ é utilizado para ajudar a gerenciar eventos relacionados a operações de Airsoft, como:

- Notificações para organizadores e operadores.
- Atualizações de status de ingressos.

Essa abordagem garante que o sistema seja escalável e capaz de lidar com picos de demanda de forma eficiente.

2.2.6 Documentação de API: Swagger

A documentação de APIs desempenha um papel crucial no desenvolvimento de software, especialmente em sistemas com múltiplos módulos ou equipes de desenvolvimento. O **Swagger** é uma ferramenta amplamente utilizada para gerar e interagir com documentações de APIs RESTful de forma automatizada e amigável (SOFTWARE, 2023b).

O Swagger utiliza um padrão chamado *OpenAPI Specification* (OAS) para descrever as APIs. Esse padrão permite que desenvolvedores definam os *endpoints*, métodos HTTP, parâmetros de entrada e respostas de forma estruturada. Além disso, o Swagger fornece uma interface gráfica que permite testar os *endpoints* diretamente na documentação, agilizando o desenvolvimento e a validação.

Criado pela SmartBear Software, o Swagger foi introduzido em 2011 como uma solução para simplificar a integração de APIs. Em 2016, sua especificação foi transferida para a Linux Foundation, consolidando-se como o padrão de fato para documentação de APIs (SOFTWARE, 2023b).

No TROPA, o Swagger foi integrado ao backend para documentar os principais *endpoints*, como:

- Registro de usuários (operadores, organizadores).
- Gerenciamento de operações e times.
- Vendas e pagamentos de ingressos.

Essa integração facilita a manutenção e documentação do sistema.

2.2.7 Serviços de E-mail

Os serviços de e-mail desempenham um papel fundamental em sistemas modernos, sendo utilizados para enviar notificações, confirmações e comunicações personalizadas aos usuários. No projeto TROPA, dois serviços foram integrados para atender a essas demandas: **SendGrid** e **Mailchimp**.

2.2.7.1 SendGrid

O **SendGrid** é uma plataforma de envio de e-mails transacionais e de marketing em larga escala. Criada em 2009, foi adquirida pela Twilio em 2019, consolidando-se como uma solução robusta para aplicações que demandam alto volume de e-mails (SENDGRID, 2023).

E-mails transacionais são mensagens automatizadas enviadas em resposta a ações do usuário, como confirmações de cadastro ou atualizações de status. O SendGrid fornece APIs para integração direta com aplicações, oferecendo:

- Alta taxa de entrega (*deliverability*).
- Suporte a métricas e *logs* para monitoramento.
- Ferramentas de autenticação, como DKIM e SPF, para evitar que e-mails sejam classificados como spam.

No TROPA, o SendGrid é utilizado para:

- Enviar notificações de cadastro e atualizações de operações.
- Confirmar compras de ingressos para eventos de Airsoft.

Esse serviço foi escolhido como principal, mas na eventual indisponibilidade dele, o TROPA conta com uma segunda opção: Mailchimp.

2.2.7.2 Mailchimp

O **Mailchimp**, criado em 2001 (MAILCHIMP, 2023), é uma das ferramentas mais populares para campanhas de e-mail e automação de marketing. Ele oferece funcionalidades avançadas de criação de campanhas e segmentação de usuários (MAILCHIMP, 2023).

Diferentemente do SendGrid, que é focado em e-mails transacionais, o Mailchimp se destaca por sua interface gráfica intuitiva e recursos voltados para marketing, como:

- Criação de campanhas personalizadas com *drag-and-drop*.

- Automação de fluxos de e-mail baseados em eventos.
- Segmentação de listas para públicos específicos.

No TROPA, o Mailchimp é a alternativa na ocasião da indisponibilidade ou falha do Sendgrid. Além disso, ele poderá ser utilizado para campanhas promocionais direcionadas aos usuários. Por exemplo, campanhas de lançamento de novas operações ou eventos especiais de Airsoft.

2.2.8 Plataforma de Pagamentos: Stripe

Em aplicações modernas que envolvem transações financeiras, a integração com plataformas de pagamento confiáveis e escaláveis é essencial. No projeto TROPA, a ferramenta escolhida para gerenciar os pagamentos foi a **Stripe**, devido à sua robustez, documentação detalhada e suporte a recursos avançados, como pagamento por cartão de crédito, carteiras digitais e transferência direta (INC., 2023c).

A Stripe é uma plataforma de processamento de pagamentos que permite que empresas aceitem transações online de maneira rápida e segura. Fundada em 2010, a Stripe oferece APIs que abstraem a complexidade dos sistemas de pagamento, incluindo:

- **Criação e gerenciamento de *Payment Intents*:** Um *Payment Intent* é um objeto que encapsula todo o ciclo de vida de uma transação, desde sua criação até a confirmação do pagamento.
- **Segurança integrada:** A Stripe segue padrões como PCI-DSS, garantindo que os dados do cliente sejam manipulados de forma segura.
- **Suporte a múltiplos métodos de pagamento:** Inclui cartões de crédito, débito, carteiras digitais (Apple Pay, Google Pay) e transferências bancárias.

Um dos principais diferenciais da Stripe é o foco em desenvolvedores, oferecendo APIs claras e consistentes para integração, além de bibliotecas em diversas linguagens de programação, incluindo TypeScript (INC., 2023c).

A Stripe foi criada por Patrick e John Collison com o objetivo de simplificar o processo de pagamentos para desenvolvedores e empresas. Desde seu lançamento, tornou-se uma das maiores empresas do setor, sendo adotada por organizações como Amazon, Lyft e Shopify. Atualmente, a Stripe opera em mais de 35 países, processando bilhões de dólares em transações por ano (INC., 2023b).

No TROPA, a Stripe é utilizada para gerenciar o pagamento de ingressos das operações de Airsoft. A integração com a Stripe segue o seguinte fluxo:

1. **Criação do *Payment Intent*:** Quando um operador compra um ingresso, o sistema cria uma intenção de pagamento, vinculando o pagamento ao organizador do evento e calculando as taxas de plataforma.

2. **Webhook para monitoramento:** A confirmação do pagamento é capturada pelo `payment_intent.succeeded`, disparando o processo de geração do ingresso.

3. **Divisão de receitas:** A Stripe permite transferir automaticamente o valor pago ao organizador, descontando uma taxa fixa para a plataforma TROPA.

Essas funcionalidades tornam a Stripe ideal para o projeto, garantindo segurança, escalabilidade e simplicidade no gerenciamento de pagamentos.

2.2.9 Ferramentas Auxiliares

Além das tecnologias principais, diversas ferramentas auxiliares foram empregadas no desenvolvimento do projeto TROPA, contribuindo para a organização, controle e qualidade do software.

2.2.9.1 Controle de Versão: Git e GitHub

O controle de versão é um elemento indispensável em projetos de software, permitindo rastrear alterações no código e facilitar o trabalho em equipe. No TROPA, o **Git** foi utilizado para versionamento do código, enquanto o **GitHub** serviu como repositório remoto.

Criado por Linus Torvalds em 2005, o Git é um sistema de controle de versão distribuído que permite a criação de *branches*, rastreamento de alterações e fusão de códigos (*merge*) (PROJECT, 2023). O GitHub, por sua vez, adiciona funcionalidades como hospedagem de repositórios, controle de permissões e integração com ferramentas de CI/CD (*Continuous Integration/Continuous Deployment*).

No projeto, o Git e o GitHub foram utilizados para:

- Versionar e armazenar o código do sistema.
- Automatizar o deploy utilizando integrações com pipelines.

2.2.9.2 Containerização: Docker e Docker Compose

O Docker foi utilizado para criar ambientes isolados de desenvolvimento, enquanto o **Docker Compose** facilitou o gerenciamento de múltiplos serviços no mesmo ambiente.

Containers são ambientes leves e portáteis que contêm todos os recursos necessários para rodar uma aplicação. O Docker utiliza imagens pré-configuradas para criar esses

ambientes de forma rápida e eficiente. Já o Docker Compose permite orquestrar vários *containers* em conjunto, definindo as configurações de cada serviço em um único arquivo (INC., 2023a).

No TROPA, o Docker e o Docker Compose foram utilizados para:

- Criar containers para o banco de dados PostgreSQL e o RabbitMQ.
- Facilitar a configuração do ambiente local de desenvolvimento.

2.2.9.3 Editor de Código: Visual Studio Code (VSCode)

O **VSCode** foi o editor de código escolhido para o desenvolvimento do projeto, devido à sua leveza, extensibilidade e integração nativa com Git.

Lançado pela Microsoft em 2015, o VSCode é um editor de código *open-source* que suporta múltiplas linguagens e *frameworks*. Ele oferece recursos como depuração (*debugging*), integração com controle de versão e suporte a extensões (MICROSOFT, 2023b).

No TROPA, o VSCode foi utilizado com extensões como `Prettier` e `ESLint` para padronizar o código e evitar erros comuns.

2.2.9.4 Design: Whimsical

O **Whimsical** foi utilizado para projetar alguns elementos de documentação de *software*, como o diagrama de arquitetura 8. Ele também serviu para rascunhar protótipos de baixa fidelidade.

Ferramentas de prototipagem como o Whimsical permitem criar diagramas, *wireframes* e fluxos de forma rápida e colaborativa. Esses protótipos servem como base para o desenvolvimento, garantindo que os requisitos sejam atendidos (WHIMSICAL, 2023).

O Whimsical foi utilizado para:

- Projetar diagramas de documentação;
- Esboçar interfaces básicas das telas.

2.3 Metodologia 5W2H

A metodologia 5W2H é uma ferramenta amplamente utilizada em gestão de projetos, planejamento estratégico e processos de melhoria contínua. Seu nome deriva das iniciais de sete perguntas fundamentais que devem ser respondidas para descrever ou organizar qualquer atividade de maneira clara e objetiva: *What* (o quê), *Why* (por quê), *Where* (onde), *When* (quando), *Who* (quem), *How* (como) e *How much* (quanto custa). Ao

responder essas questões, é possível obter uma visão abrangente da ação a ser realizada, seus responsáveis, recursos, prazos e justificativas (CAMPOS, 2004).

No contexto da Engenharia de Software, o 5W2H pode ser aplicado para estruturar a análise de problemas, a definição de escopos de sistemas e a identificação de requisitos, funcionando como uma ferramenta complementar às demais técnicas de elicitação. Sua simplicidade e clareza tornam o método especialmente útil em fases iniciais do projeto, contribuindo para o alinhamento entre *stakeholders* e desenvolvedores, além de facilitar a documentação e a comunicação técnica do sistema.

Neste trabalho, o 5W2H foi utilizado como instrumento de apoio para descrever e entender o problema que o software TROPA se propõe a resolver. Essa descrição, apresentada na Seção 3.1.3.3, auxilia na identificação das principais motivações e requisitos do sistema, garantindo uma fundamentação prática e objetiva para o desenvolvimento do projeto.

3 Resultados e Discussão

3.1 TROPA: Visão

O objetivo desta Seção é trazer uma visão ampla de alto nível para os requisitos técnicos do sistema TROPA. Ao longo do processo de escrita e desenvolvimento dessa monografia, podem acontecer mudanças na visão do TROPA para que necessidades sejam atendidas.

3.1.1 Escopo

O *software* TROPA: Teatro de Operações de *Airsoft* consiste em uma plataforma cujo principal objetivo é aumentar o engajamento dos jogadores (operadores) de *airsoft* a partir da participação em eventos e jogos, do cumprimento de objetivos e missões, e do posicionamento no *ranking*. Outros objetivos incluem:

- Uso comercial: pessoas ou empresas organizadores(as) farão o controle de vendas e o gerenciamento de eventos e jogos do esporte por meio do sistema; e lojistas poderão expor e vender artigos militares voltados ao esporte;
- Uso recreativo: jogadores, também chamados de operadores, garantirão suas participações no eventos e jogos utilizando o TROPA. Também serão informados sobre diversas questões relativas ao evento/jogo, como mudanças de hora e/ou local, organização de times, início e fim de jogo, entre outras;
- Fazer com que o TROPA tenha uma interface genérica que possa ser customizada de acordo com a vontade de um possível cliente que queira um sistema personalizado;
- De modo semelhante ao item anterior, projetar o TROPA de forma que seja escalável e de fácil implantação e reprodução para que seja usado de forma personalizada por clientes.

Além disso, o sistema terá como principal interface com os usuários um aplicativo *mobile*, mas é também uma vontade que tenha acesso por meio de um navegador *web*.

A **principal funcionalidade** da plataforma, entretanto, é a **bilheteria** digital/eletrônica, por meio da qual os usuários organizadores poderão cadastrar, divulgar e gerenciar eventos e jogos de *airsoft*. Isso inclui a venda dos ingressos, a formação das equipes (ou dos exércitos), a leitura da carteira virtual (*QR code*) do operador para validar (*ckeck-in*)) ou fazer a cobrança (*check-out*) do ingresso, os comandos de pista quente/fria,

briefing, início/fim de jogo, a criação e realização de missões, dentre outras possíveis questões.

A **funcionalidade secundária** é o **ranking** de operadores. Essa classificação será ordenada pela quantidade de XP (moeda prata) do operador, sendo que na primeira posição fica quem tem mais moedas prata. O XP é adquirido ao participar de eventos, jogos e missões promovidos na bilheteria, além de outras formas a serem elaboradas (p.ex.: jogador que permanecer no top 10 por Y semanas ganha mais XP). Operadores que estiverem subindo no *ranking* poderão receber prêmios graduais, como *patches*, *dog tag* virtual, *dog tag* física, carteirinha física (representando a carteira virtual do jogador). Também ficam sugeridas as criações de *ranking* de organizações, como a FABE (Federação de *Airsoft* de Brasília e Entorno)(FABE, 2024), *ranking* de equipes entre outros.

A **funcionalidade terciária** é o **espaço loja**, onde os parceiros da plataforma ou lojistas poderão comercializar artigos militares voltados ao esporte. Para isso, os usuários precisarão de uma moeda de troca (sugere-se moeda ouro), a qual poderá ser adquirida pela compra de créditos individuais (esse ato também agrega XP). Esse *e-commerce* contará com calculador de frete e carrinho.

A **funcionalidade quaternária** é o **espaço galeria**, no qual será possível encontrar vídeos e fotos de eventos anteriores, de conclusão de missões, de operadores entre outros.

Há ainda a necessidade de controle e manutenção absolutos do sistema por parte do autor dessa monografia, o qual será o titular da propriedade intelectual do sistema. Para isso deverá existir uma forma de autenticação e autorização que reconheça o autor e permita as atividades necessárias. Para isso sugere-se um serviço isolado dos demais do sistema, a ser denominado “administração”. A real necessidade de implementação disso será discutida nos próximos Capítulos.

O *software* TROPA não:

- Implementará a própria API (Interface de Programação de Aplicações) de pagamentos, mas usará a de terceiros;
- Será responsável pela correta organização dos jogos e eventos, pois essa responsabilidade é do usuário organizador. TROPA é um facilitador e auxiliador do processo;
- Terá responsabilidade legal nenhuma relativa aos jogos/eventos em si, como ajudar em caso de acidentes, por exemplo;
- Fará o frete, o transporte em si de mercadorias vendidas no espaço loja;

3.1.2 Descrição dos Usuários

A saber, são quatro tipos principais de usuários do TROPA, conforme a Tabela 2.

Tabela 2 – Tipos de usuários do sistema TROPA, suas descrições e funções

Tipo de Usuário	**Descrição**	**Função**
Operador	Também conhecido como jogador, é o tipo de usuário mais simples do sistema.	Interagir com a plataforma a partir da participação em jogos e eventos, do posicionamento no *ranking*, da compra de artigos no espaço loja e da publicação de mídia na galeria.
Organizador	Pessoas ou empresas que utilizam o esporte *Airsoft* para fins lucrativos.	Promover e gerenciar jogos e eventos do esporte, participar da publicação de mídia na galeria.
Ranger	Um tipo especial de usuário organizador.	Auxiliar os Organizadores no gerenciamento dos jogos/eventos.
Lojista	Pessoas ou empresas que vendem produtos utilizados pelos praticantes de *Airsoft*.	Vender produtos no espaço loja, mantendo uma boa comunicação com os compradores.
Visitantes	Podem ser os usuários operadores ou um público externo, que não tenha cadastro no TROPA.	Comprar produtos à venda no espaço loja.
Administrador	Autoridade máxima no sistema, que tem permissões para realizar qualquer tipo de ação.	Controlar, gerenciar, moderar a plataforma no que tange aos usuários, às funcionalidades e à implantação do sistema.

3.1.3 Posicionamento

3.1.3.1 Oportunidade de Negócios

A partir de entrevistas do autor desse trabalho com o Presidente da FABE (2024) e da participação em jogos e eventos de *Aisoft*, constatou-se que esse esporte está crescendo muito no Brasil. Existem diversas equipes e organizações que promovem eventos de norte a sul no País, tanto em cidades capitais quanto no interior e em cidades menos populosas.

A FABE, assim como a La Catedral (CATEDRAL, 2024), organizam jogos e eventos locais e nacionais de *Airsoft* com a participação de, em alguns casos, centenas de jogadores. Foi verificada uma dificuldade de contato com os operadores e de gerenciamento do pré e pós-jogos por parte dessas organizações, que utilizam listas e inúmeros grupos de *WhatsApp* para tanto.

Ambas empresas, FABE e La Catedral, foram entrevistadas com relação ao desenvolvimento de uma plataforma que auxilie nos jogos e eventos. Ambas manifestaram grande interesse no que o TROPA se propõe em oferecer, configurando uma boa oportunidade de negócios.

3.1.3.2 Alternativas e Concorrências

Atualmente são desconhecidas ferramentas específicas para o *Airsoft* que se propõem a entregar as mesmas funcionalidade que o TROPA. Uma pesquisa no mercado de aplicativos *web* e *mobile* foi feita, mas poucos resultados que se parecem com a ideia proposta foram encontrados.

O sistema mais promissor encontrado é o WARCAMP. De acordo com a própria página inicial dele WARCAMP (2024), é um "aplicativo para airsoft. Sistema de criação, gerenciamento e gameficação para partidas de airsoft". Esse sistema permite a localização em tempo real, a criação e edição de mapas e o *chat* por voz e texto como funcionalidades que mais se destacam.

Do ponto de vista negocial, o WARCAMP é visto como um potencial parceiro para melhor integrar a experiência de participar em jogos e eventos por meio do TROPA. Entretando, caso o WARCAMP decida implementar funcionalidades como bilheteria ou *ranking*, ele se torna uma concorrência forte, visto que, à princípio, não está previsto a utilização de mapas em tempo real pelo TROPA, funcionalidade que a outra ferramenta apresenta.

Além disso, existem outros sítios na *web* com propósito de gerenciar e organizar eventos em geral. A ferramenta AIRSOFTZONE (AIRSOFTZONE, 2024) é voltada para o mundo do esporte em questão, porém é só uma bilheteria básica, não representando uma alternativa forte ao TROPA.

Também podem ser considerados outros sistemas de bilheteria para eventos como alternativas não fortes, como Eventiza (EVENTIZA, 2024), Sympla (SYMPLA, 2024), Bilheteria Digital (DIGITAL, 2024), Eventbrite (EVENTBRITE, 2024), entre outros. Porém, essas plataformas não são focadas em eventos específicos como o TROPA, mas sim de forma genérica.

3.1.3.3 Descrição do Problema com 5W2H

O problema a ser resolvido pode ser entendido a partir do 5W2H (Seção 2.3), como mostra a Tabela 3. Esta mostra como, onde e quando será feito o sistema, e mais informações.

Tabela 3 – 5W2H

O quê?	Plataforma *mobile* e *web*
Quem?	Erick Giffoni Felicíssimo
Onde?	Brasília - DF
Quando?	1º semestre de 2025
Por quê?	Tem boa oportunidade de negócio e interesse por parte de empresas organizadoras
Como?	Pela aplicação de disciplinas da Engenharia de Software para a concretização do sistema
Quanto custa?	Aproximadamente R$ 30,000

Tabela 4 – Posição do Produto

O problema da	Dificuldade de contato com os operadores e de gerenciamento do pré e pós-jogos
Afeta	Organizações promotoras dos jogos e operadores
Uma boa solução seria	Um aplicativo *mobile* que permita a venda de ingressos e o gerenciamento dos jogos, a formação de equipes, o reconhecimento de bons operadores por meio de um *ranking*, um espaço para venda de artigos militares voltados ao esporte, e um espaço para publicação de fotos e vídeos dos eventos
Cujo impacto é	O aumento do engajamento dos operadores na participação de jogos/eventos de *Airsoft*

3.1.3.4 Instrução de Posição do Produto

O posicionamento desse sistema enquanto produto é evidenciado pela Tabela 4. Esta mostra o problema a ser resolvido, a solução e seu impacto.

3.1.4 Visão Geral do Produto

Uma plataforma *mobile* e *web* que permita organizações venderem ingressos, gerenciarem e controlarem jogos de *Airsoft*, formarem exércitos para os jogos e mais. Operadores poderão formar de equipes e terão o devido reconhecimento de bons jogadores por meio de um *ranking*. Além disso, haverá um espaço para que lojistas vendam de artigos militares voltados ao esporte, e um espaço para publicação de fotos e vídeos dos eventos pelos operadores e organizadores.

Existem ferramentas cujas funcionalidades são parecidas com as propostas pelo TROPA, mas não há, até o momento da pesquisa, ferramentas com os mesmos recursos. O sistema mais parecido é o WARCAMP, porém este não tem bilheteria, nem *ranking*, loja nem galeria.

TROPA se destaca por cuja ideia ter sido concebida no ambiente em que se insere, o esporte *Airsoft*, sendo específica e projetada para isso. As funcionalidades pensadas para a plataforma são resultado de entrevistas formais e informais com diversos interessados, principalmente a FABE.

3.1.4.1 Estimativa de Custos

Essa estimativa será dividida em: custo de desenvolvimento; e custo de operação. Como o desenvolvimento será feito durante esse TCC, o custo dele é zero. Então resta o custo de operação.

3.1.4.1.1 Custos de Operação

O custo de operação vai depender de quais tecnologias serão utilizadas para fazer a implantação do sistema. À princípio, pretende-se utilizar a AWS (Serviços Web da Amazon) para implantar um ou mais serviços. Estima-se a utilização de:

- 6 instâncias EC2 tipo t3.small;
- 1 *Amazon RDS for PostgreSQL* tipo db.m1.small com 1TB de armazenamento;
- 1 VPC (*Virtual Private Cloud*) com 1 *Gateway Load Balancer endpoint*.

Todos esses recursos alocados na região de São Paulo. Foi utilizada a calculadora de preços da AWS (AWS, 2024), a qual calcula em Dólares Americanos.

- AWS EC2: USD 97.24 / mês;
- AWS *RDS for PostgreSQL*: USD 600.79 / mês;
- AWS VPC: USD 288.83 / mês.

Custo de Operação Total: USD 986.86 por mês, aproximadamente.

3.1.4.2 Propriedade Intelectual

A Propriedade Intelectual dessa solução de *software* será inteiramente e exclusivamente do autor desse Trabalho de Conclusão de Curso.

3.1.5 Recursos do Produto

- Bilheteria: usuários organizadores poderão cadastrar, divulgar e gerenciar eventos e jogos de *Airsoft*. Operadores comprarão ingressos e participarão dos jogos;
- *Ranking*: funcionalidade que fornece reconhecimento ("fama") aos melhores operadores. Será ordenada pela quantidade de XP;
- Espaço loja: onde os parceiros da plataforma ou lojistas poderão comercializar artigos militares voltados ao esporte;
- Espaço Galeria: organizadores e operados poderão publicar fotos e vídeos de jogos/eventos;

3.1.6 Restrições do Produto

- Acesso à *Internet*;
- Dispositivo móvel (ex.: celular) com capacidade de armazenamento e processamento suficientes para usar o aplicativo TROPA ou para acessá-lo por meio de um navegador *web* (ex.: Firefox);
- Indisponibilidade de recursos, como a bilheteria, por exemplo;

3.2 TROPA: Requisitos

Conforme mencionado na Seção 3.1, duas empresas organizadoras de jogos de *Airsoft* demonstraram interesse comercial no TROPA, a FABE (FABE, 2024), de Brasília - DF, e a La Catedral (CATEDRAL, 2024), de Belo Horizonte - MG.

Para o levantamento dos requisitos foram utilizadas as técnicas de: entrevista aberta; *brainstorming*; e introspecção. Cerca de 4 entrevistas aconteceram com o Presidente da FABE, e uma entrevista informal por meio de um aplicativo de mensagens com o Proprietário da La Catedral.

Somado a isso e às anotações das entrevistas, fez-se uma sessão de *brainstorming* bem como uma introspeção. O resultado desse levantamento conta com 23 requisitos, incialmente:

- R1: CRUD (Criar, Ler, Atualizar, Deletar) de usuário;
- R2: Perfis de usuários, que determinam as permissões do usuário no sistema, tais como Operador, Organizador, *Ranger*, Parceiro/Lojista e Administrador;

- R3: *QR Code* do usuário, que serve como um identificador. Será utilizado em diversas funcionalidades, tais como a validação de ingressos;
- R4: CRUD de equipe;
- R5: Administrar o sistema utilizando o próprio sistema, pode ser um *front-end* específico para isso);
- R6: CRUD de jogos;
- R7: CRUD de eventos;
- R8: CRUD de missões e geração de *QR Code*;
- R9: Venda de ingressos para jogos e eventos. Necessitará da utilização de APIs de pagamento de terceiros;
- R10: Validação de ingresso por parte do organizador por meio da leitura de *QR Code* dos operadores;
- R11: Divulgação de eventos;
- R12: Divulgação de jogos;
- R13: Formação de exércitos para os jogos/eventos. Exército se refere a um conjunto de operadores com um mesmo objetivo. Geralmente os jogos contam com dois exércitos identificados por cores diferentes, como amarelo e azul;
- R14: Sistema de XP, para ganho e para perda;
- R15: Visualização e organização do *ranking*.
- R16: *Reset* do *ranking*, que ocorrerá em períodos de tempo fixos, potencialmente anual;
- R17: Premiação. Distribuição de prêmios aos jogadores que permaneceram no topo do *ranking* por um determinado período de tempo, potencialmente o mesmo tempo do *reset*. Os prêmios podem ser definidos a qualquer momento e a distribuição deles pode ser de responsabilidade de um usuário organizador (ex.: FABE decide fabricar e distribuir prêmios com visibilidade no TROPA) ou do próprio TROPA enquanto negócio;
- R18: CRUD de lojistas/parceiros;
- R19: CRUD de produtos;
- R20: Carrinho de compras;

Figura 1 – Épicos e Histórias de Usuário

Épico	ID	Eu, como	desejo	para que	Prioridade	Estimativa	ID requisito
Usuário	1	Operador, Organizador, Lojista, Administrador	possuir uma conta de usuário	eu consiga utilizar as demais funcionalidades da plataforma	Must	10	R1
							R2
	2	Operador	possuir um QRCode	eu consiga me identificar nos jogos e eventos	Must	4	R3
	3	Operador	criar uma equipe	eu consiga participar de eventos e jogos com minha equipe	Must	3	R4
	4	Administrador	utilizar o sistema	administrar usuários, permissões, taxas de cobrança, entre outras questões	Must	7	R5
Bilheteria	5	Organizador	organizar jogos e eventos pela plataforma	eu os possa gerenciar efetivamente	Must	10	R6
							R7
							R8
							R9
							R10
	6	Organizador	divulgar jogos e eventos	o número de operadores participantes aumente	Must	1	R11
							R12
	7	Organizador	organizar exércitos	os jogos/eventos tenham uma dinâmica adequada para os operadoes	Must	1	R13
Ranking	8	Operador	que exista um sistema de ranking	eu consiga visualizar a minha performance e competir com outros operadores	Should	5	R14
							R15
	9	Operador	que o ranking zere dentro de um determinado período de tempo	eu tenha a oportunidade de subir no ranking	Should	3	R16
	10	Operador	receber prêmios	eu me sinta recompensado pelo meu esforço	Could	5	R17
Loja	11	Lojista	ter meus produtos na plataforma	eu consiga aumentar as minhas vendas	Would	5	R18
	12	Operador	comprar produtos pela plataforma	eu consiga participar dos jogos e eventos com bons equipamentos e utilizar minhas recompensas	Would	3	R19
							R20
							R21
							R22
Galeria	13	Operador, Organizador	postar vídeos e imagens	eu divulgue minhas atividades e habilidades	Could	4	R23

- R21: Cálculo de frete;
- R22: Sistema de pagamento para processar as transações. Necessitará da utilização de APIs de pagamento de terceiros.
- R23: Postagem de mídias, como fotos e vídeos, na galeria.

3.2.1 Histórias de Usuário

Os requisitos foram então organizados e relacionados em histórias de usuário. Estas possuem: ID; prioridade, seguindo a técnica MoSCoW; estimativa em horas; e ID do requisito relacionado.

À princípio foram elencadas 13 histórias de usuário, as quais foram agrupadas em épicos conforme a funcionalidade (ex.: bilheteria) que descrevem. A Figura 1 mostra a relação épico -> história -> ID do requisito.

3.2.2 Casos de Uso

Para identificação dos casos de uso pertinentes ao TROPA, tomou-se como referência parte da abordagem explicada por Reinehr (2020). Tomando como ponto de partida a seguinte pergunta, os casos de uso de cada tipo de ator do sistema foram identificados: “Quais são as tarefas principais que um ator deverá executar?”

Usuário Operador

- Cadastrar;
- Participar em jogos/eventos;
- Criar equipe;
- Entrar para uma equipe;
- Interagir com o *ranking*;
- Comprar produtos;
- Publicar fotos ou vídeos.

Usuário Organizador

- Cadastrar;
- Organizar jogos/eventos;
- Divulgar jogos/eventos;
- Interagir com o *ranking*;
- Publicar fotos ou vídeos.

Usuário Lojista

- Cadastrar;
- Vender produtos;
- Publicar fotos ou vídeos.

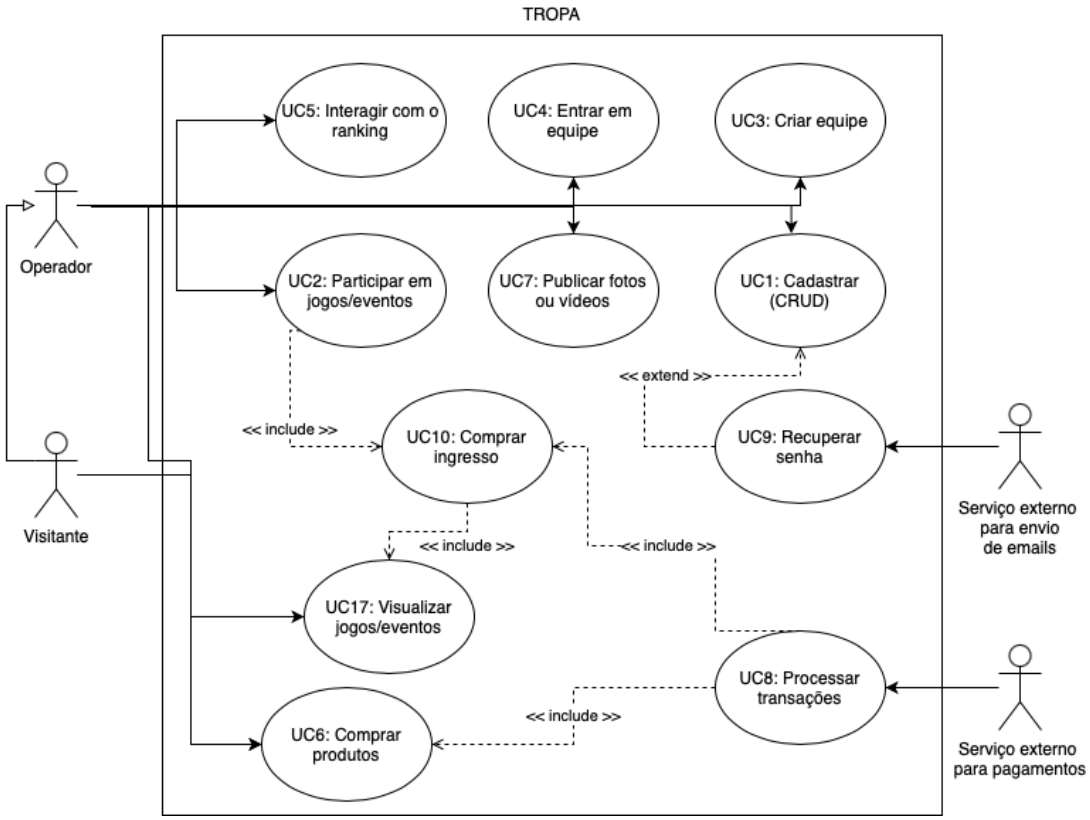
Usuário Administrador

- Cadastrar;
- Fazer a moderação de conteúdo publicado;
- Gerenciar o sistema.

Usuário Visitante

- Visualizar jogos/eventos;

Figura 2 – Diagrama de Casos de Uso do Usuário Operador



- Comprar produtos.

Serviço externo para pagamentos

- Processar pagamento da loja e da bilheteria.

Serviço externo para envio de e-mails

- Notificar sobre atualizações diversas;
- Dar suporte à recuperação de senha.

A partir disso os casos de uso foram identificados e os diagramas de caso de uso foram elaborados. A Figura 2 mostra o diagrama de casos de uso do usuário operador.

De forma semelhante, a Figura 3 representa o diagrama de casos de uso do usuário organizador.

O diagrama de casos de uso para o usuário administrador (Figura 4) contém menos casos, porem um deles, o UC19 (3.2.3.8), que será detalhado adiante, é bem complexo.

Por fim, conforme a Figura 5, o último diagrama de casos de uso relevante ao TROPA, evidenciando o papel do serviço externo para envio de e-*mails.*

Figura 3 – Diagrama de Casos de Uso do Usuário Organizador

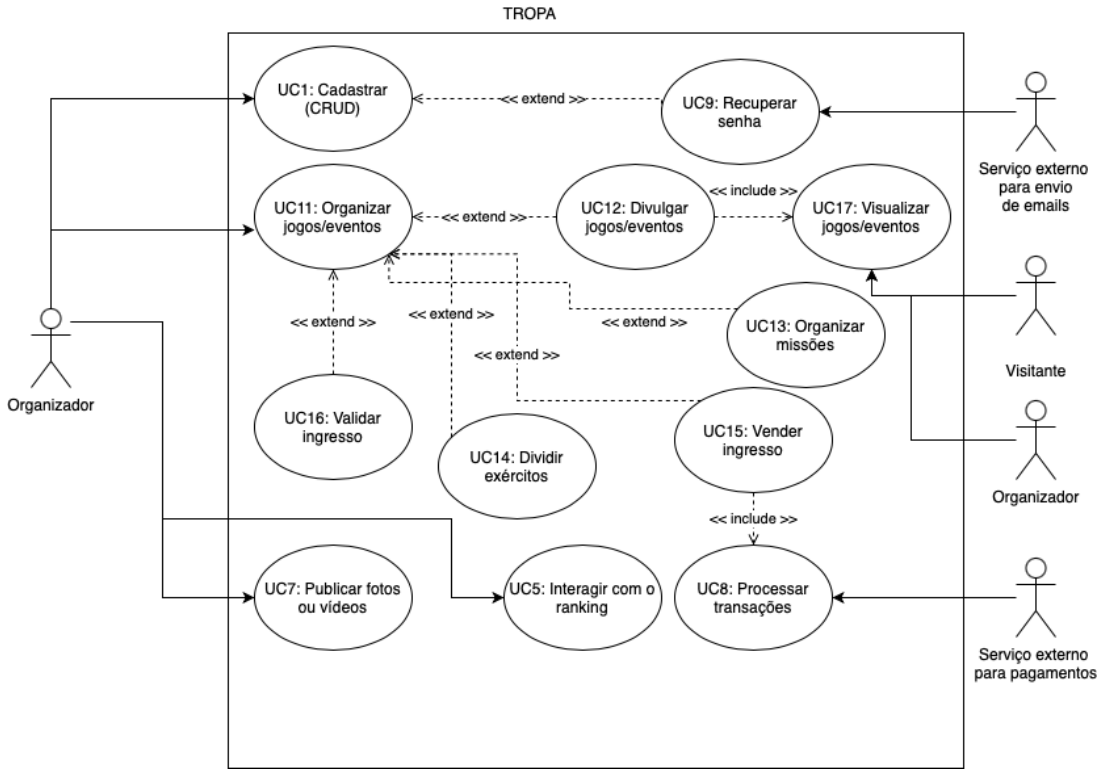


Figura 4 – Diagrama de Casos de Uso do Usuário Administrador

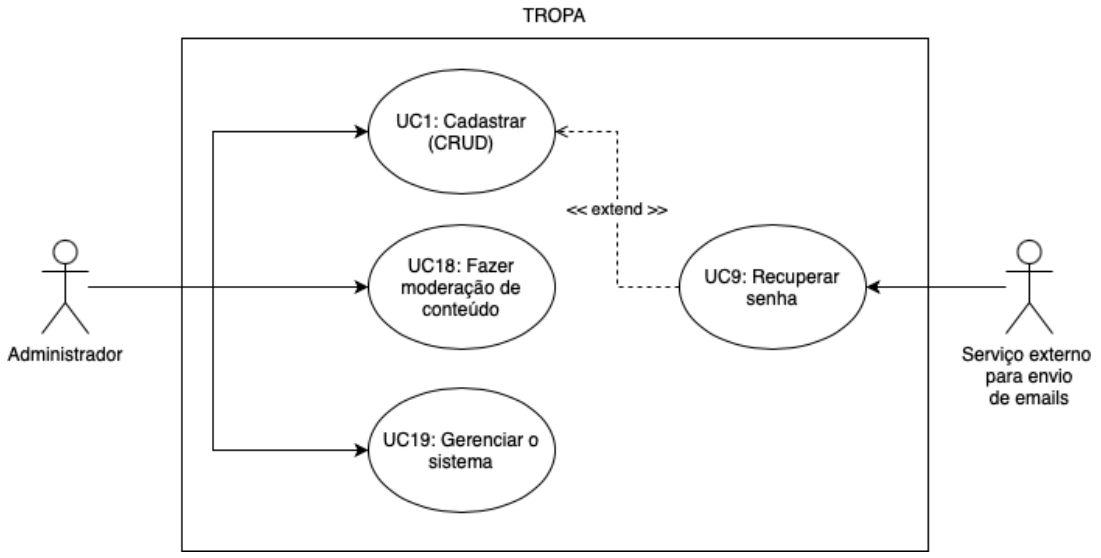
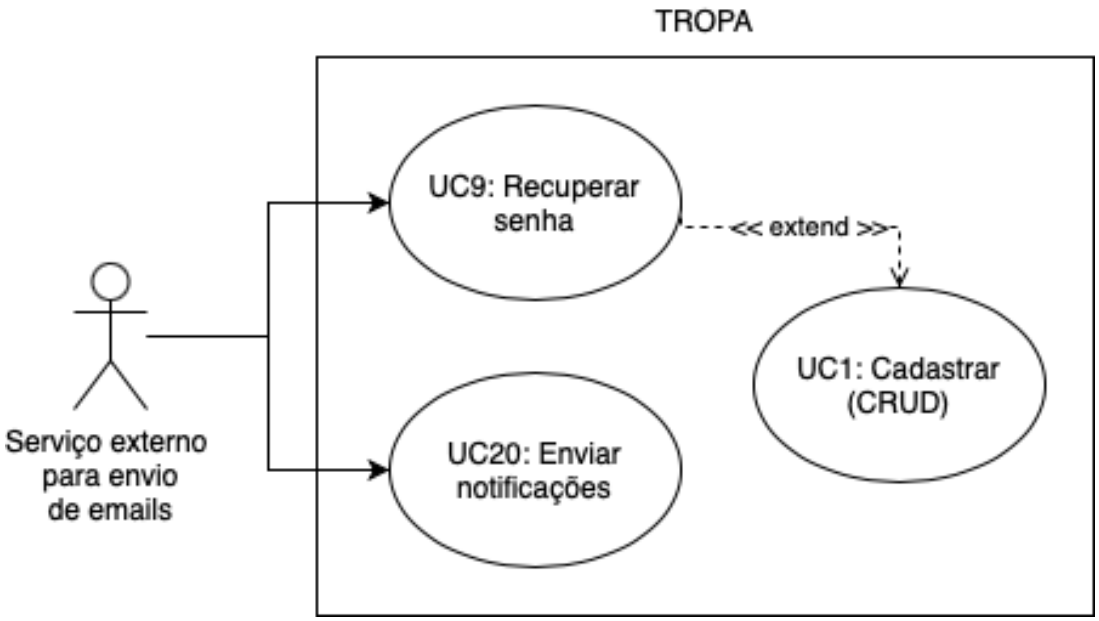


Figura 5 – Diagrama de Casos de Uso do Serviço Externo para Envio de E-mails



3.2.3 Especificação dos Casos de Uso

Os casos de uso julgados como mais complexos serão especificados para que seja obtido um melhor entendimento sobre o funcionamento do TROPA. O modelo de especificação utilizado é uma mescla do que propõem Reinehr (2020) e Cockburn em Pressman e Maxim (2021).

Obs.: onde ler "Sistema" ou "Aplicativo", entenda como o *software* TROPA.

3.2.3.1 UC 2: Participar em jogos/eventos

Identificador do caso de uso: UC 2

Ator primário: Operador

Ator(es) secundário(s): Organizador e Serviço Externo para Pagamentos

Descrição: O Operador deseja participar dos jogos/eventos promovidos na bilheteria do sistema. Para isso ele precisará visualizar os jogos disponíveis e fazer a compra do seu ingresso. No dia e horário do evento ele fará o *check-in* com o Organizador por meio da leitura do *QR Code*. O Operador poderá ver todos os detalhes do jogo, bem como participar de missões e premiações, caso existam.

Pré-condições: Usuário Operador já fez o *login* no TROPA

Gatilho: Operador entra na tela de bilheteria

Fluxo Principal

1. (Ator) Busca por eventos/jogos disponíveis

2. Encontra o desejado, visualiza informações e prossegue para compra (UC 10 3.2.3.2)
3. (Sistema) Recebe o pedido de compra do ingresso e utiliza o serviço externo para processar a transação financeira (3.2.3.4)
4. (Sistema) Salva ingresso do operador na base de dados
5. (Ator) Comparece ao evento com sua carteira virtual (*QR Code*) aberta no aplicativo TROPA
6. (Organizador) Lê credenciais do Operador e valida a entrada dele no jogo
7. (Operador) Participa das missões e premiações, caso existam

Fluxo Alternativo

1. (Ator) Decide por pedir o reembolso do valor do ingresso antes da data e hora do evento
2. (Sistema) Utiliza o serviço externo para processar a transação financeira (UC 8 3.2.3.4)
3. (Ator) Recebe o dinheiro de volta em forma de moedas ouro, para posterior utilização, ou na conta bancária informada

Fluxo de Exceção

- 3.1 (Serviço externo) Notifica o Sistema de falha na transação
- 3.1 (Sistema) Toma as devidas medidas e informa ao usuário a falha
- 6.1 (Sistema) Acusa erro na leitura da carteira virtual do Operador
- 6.2 (Organizador) Entra manualmente com o código QR no sistema para validar o ingresso do Operador

3.2.3.2 UC 10: Comprar Ingresso

Identificador do caso de uso: UC 10

Atores primários: Operador

Ator(es) secundário(s): Serviço externo para pagamentos

Descrição: Refere-se a toda vez que um Operador decidir comprar um ingresso para um jogo/evento.

Pré-condições: Operador já fez o *login* no sistema e já escolheu qual evento/jogo deseja participar (UC 17 3.2.3.3)

Gatilho: Operador clica em "finalizar compra"

Fluxo Principal

1. (Ator) Informa ao sistema informações sobre a compra, como quantidade de ingressos e forma de pagamento, por exemplo. As formas de pagamento disponibilizadas serão: moeda ouro; pix; crédito; débito
2. (Sistema) Recebe o pedido de compra do ingresso e utiliza o serviço externo para processar a transação financeira (UC 8 3.2.3.4)
3. (Sistema) Salva ingresso do operador na base de dados

Fluxo Alternativo

1. (Ator) Ao invés de clicar para finalizar a compra, clica em "cancelar"
2. (Sistema) Não inicia o processamento da transação

Fluxo de Exceção

- 2.1 (Serviço externo) Notifica o Sistema de falha na transação
- 2.1 (Sistema) Toma as devidas medidas e informa ao usuário a falha

3.2.3.3 UC 17: Visualizar jogos/eventos

Identificador do caso de uso: UC 17

Ator primário: Operador ou Visitante

Ator(es) secundário(s): Não há

Descrição: Um Operador ou Visitante poderá apenas visualizar os jogos/eventos disponíveis para compra de ingressos na plataforma TROPA

Pré-condições: Usuário já fez o *login* no sistema

Gatilho: Usuário clica na "aba" de bilheteria da plataforma

Fluxo Principal

1. (Ator) Escolhe um evento ou usa a barra de busca para encontrá-lo, depois clica nele
2. (Sistema) Devolve as informações sobre o evento

Fluxo Alternativo

1. (Ator) Decide ir para outra "aba" senão bilheteria

Fluxo de Exceção

- 1.1 (Sistema) Informa ao usuário que um erro aconteceu ao carregar as informações e pede para tentar novamente depois

3.2.3.4 UC 8: Processar Transações

Identificador do caso de uso: UC 8

Ator primário: Serviço externo de pagamentos

Ator(es) secundário(s): Operador ou Visitante

Descrição: Qualquer tipo de compra feita na plataforma TROPA, seja da bilheteria ou da loja, será efetivamente realizada por meio de um serviço de terceiros, o qual será definido mais adiante. Esse cenário lidará tanto com cobranças quanto com possíveis devoluções.

Pré-condições: Usuário já fez o *login* no sistema; usuário clica em "finalizar compra" ou em "pedir reembolso".

Gatilho:

Fluxo Principal

1. (Sistema) Envia um pedido de transação monetária ao serviço externo
2. (Serviço externo) Recebe o pedido, faz o processamento e retorna o resultado ao sistema

Fluxo Alternativo: não se aplica

Fluxo de Exceção

- 1.1 (Sistema) Apresenta indisponibilidade ou erro de conexão
- 1.2 (Serviço externo) Não recebe o pedido
- 2.1 (Serviço externo) Apresenta indisponibilidade ou erro de conexão
- 2.2 (Sistema) Não recebe a resposta, então informa ao usuário que um erro aconteceu com o serviço externo

3.2.3.5 UC 11: Organizar jogos/eventos

Identificador do caso de uso: UC 11

Ator primário: Organizador

Ator(es) secundário(s): Não há

Descrição: A organização de um jogo/evento de *Airsoft* utilizando o sistema TROPA compreende atividades de gerenciamento, por parte do Organizador. Definição de quantidade de ingressos disponíveis, preços, data, hora, local, divisão das equipes e dos Operadores em exércitos, dar início e fim do jogo são exemplos das várias atribuições do Organizador.

Pré-condições: Organizador já fez o *login* no sistema e está pronto para organizar eventos/jogos

Gatilho: qualquer ação dentro da plataforma que envolva a elaboração e/ou edição de um evento/jogo

Fluxo Principal

1. (Ator) Faz uma ação para criar novo evento/jogo
2. (Sistema) Pede as informações necessárias
3. (Ator) Preenche todas as informação referentes ao jogo/evento
4. (Sistema) Confirma criação do evento e o salva na base de dados

Fluxo Alternativo

- 2.1 (Ator) Desiste de criar o evento
- 3.1 (Ator) Desiste de criar o evento
- 4.1 (Ator) Faz edições no evento/jogo, tais como divisão de exércitos (UC 14), alteração de local, comanda o início e o fim do jogo, cria e edita missões (UC 13), lê a carteira virtual dos Operadores inscritos (UC 16), entre outras

Fluxo de Exceção

- (Sistema) Apresenta algum tipo de erro que impeça a realização de qualquer um dos itens dos fluxos acima

3.2.3.6 UC 16: Validar ingresso

Identificador do caso de uso: UC 16

Ator primário: Organizador

Ator(es) secundário(s): Operador, Aplicativo

Descrição: Para confirmar a participação do Operador no evento/jogo, o Organizador precisará ler a carteira virtual (*QR Code*) do Operador e validar o ingresso

Pré-condições: Operador já comprou o ingresso na bilheteria do sistema; a data, hora e o local do jogo/evento são no momento presente, não no futuro

Gatilho: Organizador usa a funcionalidade de ler *QR Code* por meio do aplicativo

Fluxo Principal

1. (Ator) Lê a carteira virtual do Operador
2. (Sistema) Faz a validação do ingresso
3. (Organizador e Operador) Recebem notificação de ingresso validado

Fluxo Alternativo: não se aplica

Fluxo de Exceção

- 1.1 (Aplicativo) Apresenta erro na leitura
- 1.2 (Ator) Entra manualmente com o código QR
- 2.1 (Sistema) Apresenta erro na validação do ingresso
- 2.2 (Aplicativo) Exibe mensagem de erro correspondente

3.2.3.7 UC 18: Fazer moderação de conteúdo

Identificador do caso de uso: UC 18

Ator primário: Administrador

Ator(es) secundário(s): Usuário

Descrição: Usuários, como Operador, Lojista e Organizador, poderão publicar conteúdo, tal como mídias digitais, na plataforma. Pode ser que esse conteúdo seja inapropriado para o TROPA, então o Administrador fará a moderação correta, seja exclusão da informação ou outra

Pré-condições: Usuário fez uma publicação na plataforma

Gatilho: Sistema emitiu alerta de conteúdo inapropriado; ou o Administrador encontrou o conteúdo por conta própria

Fluxo Principal

1. (Ator) Revisa o conteúdo. Sendo necessário, faz a exclusão
2. (Sistema) Notifica quem fez a publicação sobre o ocorrido

Fluxo Alternativo

1. (Ator) Coloca a publicação com visibilidade apenas para o publicante (Usuário), enquanto este é notificado de revisar o conteúdo
2. (Usuário) É notificado e faz a revisão. Fluxo volta ao principal

Fluxo de Exceção

- 2.1 (Usuário) Entre com pedido de recurso pois não concorda com o julgamento do Administrador
- 2.2 (Ator) Faz uma nova revisão. Fluxo volta ao principal

3.2.3.8 UC 19: Gerenciar o sistema

Identificador do caso de uso: UC 19

Ator primário: Administrador

Ator(es) secundário(s): Não há

Descrição: A tarefa de gerenciamento do sistema é de responsabilidade do Administrador. Este não é um usuário comum, como os outros, mas sim uma autoridade máxima para controlar o sistema. Esse resolverá questões de permissão de usuário, moderação de conteúdo (3.2.3.7), edição, bloqueio/desbloqueio de eventos/jogos, de usuários, de lojas/produtos, adição/remoção de Administradores, enfim, diversas ações dentro do TROPA

Pré-condições: O sistema está em funcionamento, implantado

Gatilho: Não se aplica

Fluxo Principal

1. (Ator) Faz o *login* na parte do sistema específica para Administradores
2. (Ator) Desempenha funções diversas tais como as descritas acima

Tabela 5 – Requisitos de Qualidade

Característica	Sub-característica	ID	Requisito para o TROPA
Funcionalidade	Aptidão (*suitability*)	RNFQ 1	Funcionalidades específicas do sistema (ex.: venda de ingresso) deverão ser implementadas por um conjunto de funções apropriadas
	Interoperabilidade	RNFQ 2	O sistema deverá ser capaz de interagir com: os sistemas operacionais mais comuns aos dispoitivos celulares: Android e iOS; e com o navegadores *web* mais comuns: Google Chrome, Firefox e Safari
Confiabilidade	Tolerância a falhas	RNFQ 3	Caso uma falha aconteça em uma das funcionalidades do sistema, as outras não devem ser impactadas
	Recuperabilidade	RNFQ 4	O sistema deve se recuperar de uma falhas no prazo máximo de 20 minutos
Usabilidade	Aprendibilidade	RNFQ 5	Usuários devem aprender com facilidade as formas de se utilizar o sistema
	Operacionabilidade	RNFQ 6	Usuários devem utilizar o sistema com facilidade
Eficiência	Tempo de resposta	RNFQ 7	O tempo de esposa deve ser aceitável para cada funcionalidade proposta
Manutenibilidade	Analisabilidade	RNFQ 8	A identiicação de falhas no sistema deve ser imediata
	Mutabilidade	RNFQ 9	Eventuais mudanças devem ser pouco trabalhosas
	Testabilidade	RNFQ 10	Mudanças feitas devem ser simples de testar
Portabilidade	Adaptabilidade	RNFQ 11	Mudanças de ambiente de implantação devem ser simples de fazer

Fluxo Alternativo: não se aplica

Fluxo de Exceção

- 1.1 (Sistema) Não consegue autenticar ou autorizar o Administrador, por algum motivo

3.2.4 Requisitos Não-Funcionais (RNFs)

3.2.4.1 Requisitos de Qualidade

A partir das características e sub-características definidas pela ISO/IEC 9126, como foi mostrado por Burgués e Franch (2000), foram escolhidas para maior foco pelo TROPA aquelas que fazem sentido ao que esse sistema propõe. A Tabela 5 mostra as características, sub-características e os requisitos de qualidade para o TROPA. O identificador deles será RNFQ: requisito não-funcional de qualidade.

3.2.4.2 Requisitos de Segurança

3.2.4.2.1 Justificativa

Apesar de Pressman (2011) e outros autores apresentarem a segurança como um atributo de qualidade, o autor dessa monografia prefere a abordagem de que segurança e qualidade são assuntos diferentes, mas que se completam, como foi abordado por Naik e Tripathy (1959).

Nesse sentido, a segurança aqui é fundamentada pela disciplina da Segurança Cibernética. Esta é entendida, segundo Kremer et al. (2019), como a proteção de sistemas de computadores contra roubo ou dano ao *hardware*, *software* ou à informação relacionados

a tais sistemas, bem como contra à perturbação, disrupção, desordem ou direcionamento incorreto dos serviços fornecidos por aqueles sistemas.

Mais ainda, a segurança cibernética também pode ser entendida como a garantia de 3 (três) propriedades importantíssimas para a informação, para os serviços e para a infraestrutura de tecnologia da informação (KREMER et al., 2019). Tais propriedades são conhecidas como a tríade *CIA* de segurança cibernética, a qual traduz-se em: confidencialidade; integridade; e disponibilidade. É importante ter noção do que significa cada propriedade da CIA para conseguir estabelecer os requisitos de segurança do TROPA.

- A **confidencialidade** estabelece que partes não confiáveis sejam incapazes de vazar ou inferir informações de uma mensagem (ROSSOW; JHA, 2019). Para ACADEMY (2023), o termo se refere a limitar o acesso e a divulgação de informação para usuários autorizados, e prevenir o acesso por pessoas não autorizadas.
- A**integridade** de um sistema, de acordo com Naik e Tripathy (1959), refere-se à habilidade ou capacidade dele de resistir a ataques contra a segurança. Para ACADEMY (2023), significa preservar dados sem corrupção de qualquer informação transmitida para ou inserida no sistema. Também significa assegurar que o transmissor é quem diz ser.
- A **disponibilidade** estabelece que os usuários e interessados de um sistema devem ser capazes de acessá-lo com a garantia de que ele estará disponível, ou seja, funcionando corretamente e atendendo às demandas solicitadas (ROSSOW; JHA, 2019). E como afirma ACADEMY (2023), um sistema que está indisponível quando precisa-se dele é quase tão inútil à não existência do tal.

3.2.4.2.2 Os requisitos

A partir desse entendimento, foi possível estabelecer o que caracterizam os requisitos de segurança de confidencialidade, integridade e disponibilidade para o TROPA. O identificador deles será RNFSec: requisito não-funcional de segurança.

- **RNFSec 1** - Confidencialidade: nem todas as informações guardadas e tramitadas pelo TROPA serão públicas. Portanto somente usuários autorizados poderão visualizar esses dados. Estes, em caso de invasão no sistema, não devem ser compreendidos pelo invasor;
- **RNFSec 2** - Integridade: o TROPA deverá ser capaz de resistir a alguns ataques de segurança, como DDoS (Negação de Serviço Distribuído, injeção SQL, injeção de comando e XSS (*Cross-Site Scripting*)). Também deve assegurar que dados armazenados permaneçam sem corrupção nem adulteração;

Figura 6 – SIG dos Requisitos Não-Funcionais de Qualidade

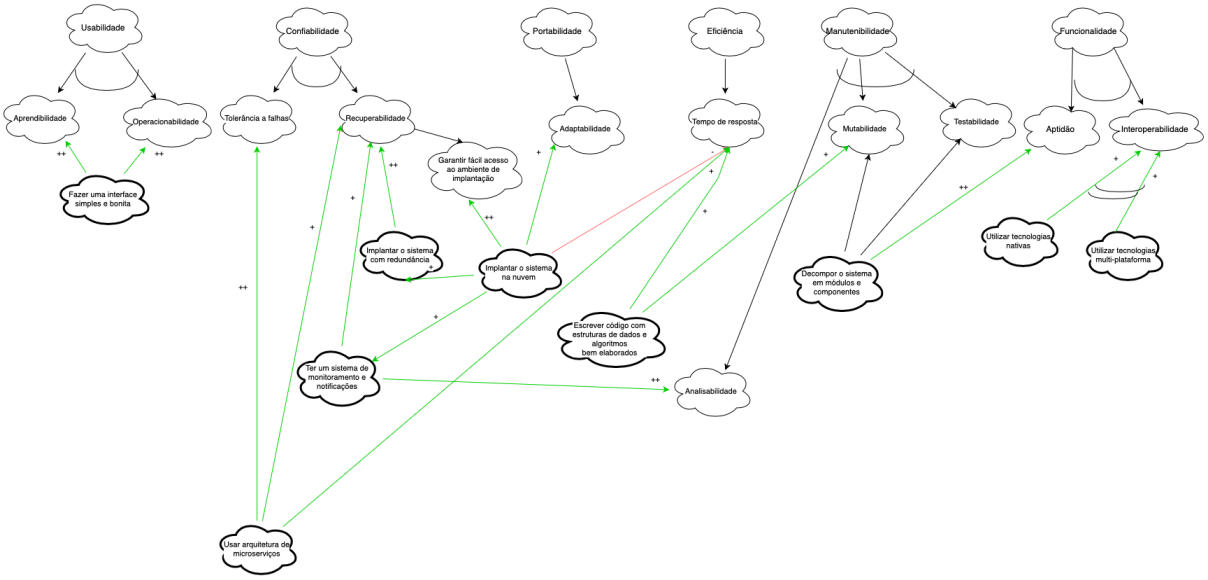
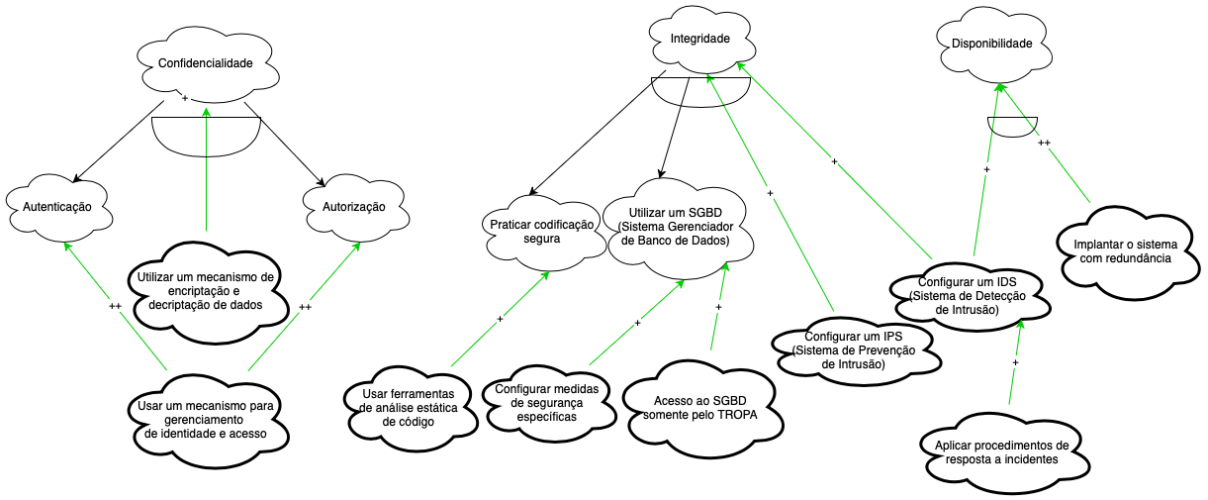


Figura 7 – SIG dos Requisitos Não-Funcionais de Segurança



- **RNFSec 3** - Disponibilidade: o sistema deverá se recuperar contra qualquer coisa/pessoa que tente provocar indisponibilidade total ou parcial.

3.2.4.3 Modelagem dos Requisitos Não-Funcionais

Para melhor entender como conseguir atender aos requisitos não-funcionais, foi feita a modelagem utilizando o *framework* NFR (SILVA, 2004), proposto por Chung (2006). A Figura 6 mostra o grafo (SIG) para os requisitos de qualidade.

De forma semelhante, a Figura 7 mostra o grafo para os requisitos de segurança.

3.3 TROPA: Arquitetura

3.3.1 Introdução

Este é o documento de arquitetura do *software* TROPA: Teatro de Operações de *Airsoft*, uma plataforma *mobile* e *web* para a organização, venda, a participação e o gerenciamento de jogos/eventos do esporte, além de outras finalidades conforme evidenciado no Documento de Visão (Seção 3.1).

3.3.1.1 Propósito

Prover uma visão geral compreensiva do sistema por meio de diferentes visões arquitetônicas para ilustrar diferentes aspectos do *software*. A intenção é capturar e comunicar as decisões de arquitetura que foram feitas, tomando como base a Visão e os Requisitos (seções 3.1 e 3.2) do sistema.

O público-alvo desse documento é composto por:

- Analistas de Requisitos: verificarão se a arquitetura atende aos requisitos definidos;
- Arquitetos de *Software*: projetarão a arquitetura;
- Desenvolvedores: implementarão o sistema de acordo com a arquitetura; e
- Engenheiros *Devops*: implantarão o sistema considerando a arquitetura.

3.3.1.2 Escopo

O escopo desse documento é referente apenas ao *software* TROPA, o que inclui seu *back-end* e *front-end*. A arquitetura apresentada é um reflexo dos requisitos e da visão do sistema, e influencia as decisões que serão tomadas a partir daqui, como a escolha das tecnologias e o projeto dos componentes, por exemplo.

3.3.1.3 Referências

A arquitetura do TROPA faz referência aos seguintes documentos, localizados nas suas seções:

- **Documento de Visão**, Seção 3.1; e
- **Documento de Requisitos**, Seção 3.2.

3.3.2 Objetivos e Restrições

3.3.2.1 Objetivos

Os requisitos de qualidade (Seção 3.2.4.1) e de segurança (Seção 3.2.4.2) desempenharam papel crucial nas decisões sobre a arquitetura do TROPA. Formas pelas quais tais requisitos podem ser atingidos são evidenciadas pelas operacionalizações (nuvens com bordas escuras) ilustradas nos grafos NFR da Seção 3.2.4.3.

Os principais requisitos de qualidade e de segurança que apresentam impacto significativo na arquitetura do sistema TROPA são: interoperabilidade (RNFQ2); tolerância a falhas (RNFQ3); recuperabilidade (RNFQ4); analisabilidade (RNFQ8); mutabilidade (RNFQ9); adaptabilidade (RNFQ11); integridade (RNFSec2); e disponibilidade (RNFSec3).

Tais requisitos apontam para uma arquitetura:

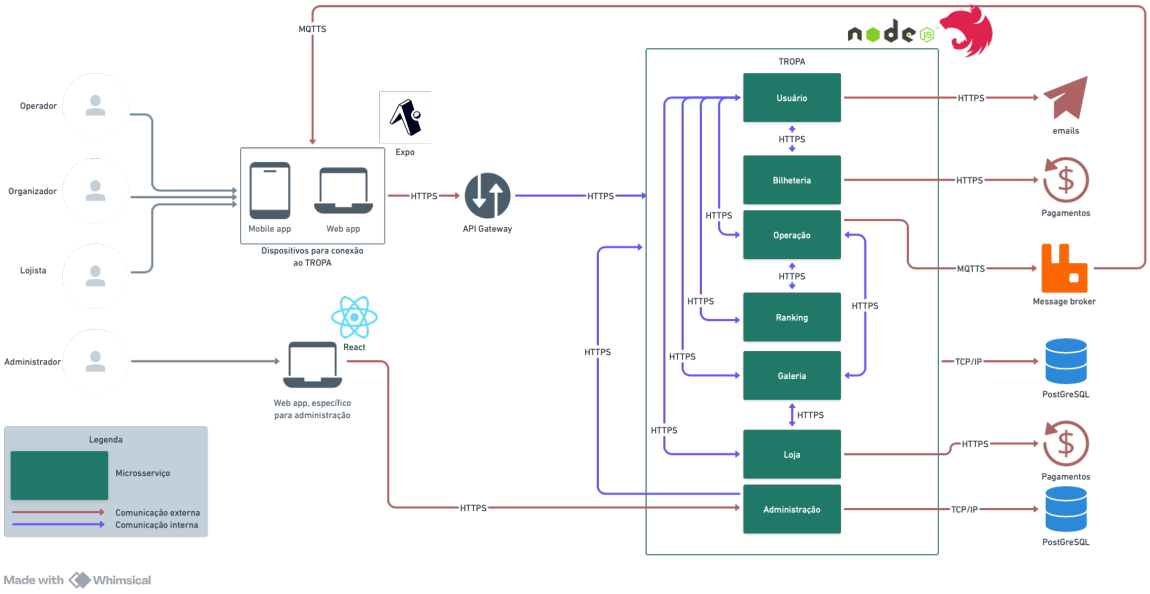
- Construída com tecnologias adequadas para proporcionar aos usuários um aplicativo móvel funcional;
- Orientada a serviços ou microsserviços, isolados, modularizados e independentes;
- Implantada em nuvem, sendo fácil e rápida a manutenção;
- Com escalabilidade flexível e baixa sensibilidade à mudanças de ambiente de implantação;
- Encapsulada, de forma que seja garantida a privacidade dos serviços em relação ao mundo exterior.

3.3.2.2 Restrições

Algumas restrições se aplicam para a arquitetura do TROPA de forma a colaborarem com os objetivos listados anteriormente. São elas:

- Os serviços do TROPA não aceitarão requisições de qualquer número IP, mas somente dos que eles conhecem;
- A implantação em nuvem deve ser feita utilizando recursos presentes no país em que o sistema funcionará, nesse caso no Brasil;
- A comunicação entre os serviços deve: ser criptografada; utilizar protocolos adequados; acontecer somente quando necessário;
- Serão utilizados dois serviços externos ao TROPA, a saber: um para envio de e-mails; e um para realizar os pagamentos.

Figura 8 – Diagrama relacional da arquitetura



3.3.3 Representação da Arquitetura

A arquitetura do TROPA é distribuída, sendo seu estilo orientado a serviços, mais especificamente a microsserviços: unidades lógicas funcionais independentes e isoladas. A Figura 8 mostra a arquitetura geral do sistema.

Conforme evidenciado, os usuários interagem com o sistema por meio de um aplicativo *mobile* e *web*. Exceto pelo usuário administrador, que interage por meio de uma aplicação *web* específica.

O sistema TROPA utilizará um *gateway/load balancer* para rotear as requisições dos usuários ao serviço correspondente, bem como para balancear a carga de várias requisições. Essa API também serve para esconder os serviços do sistema da *Internet*, colaborando com a segurança.

São 7 os microsserviços do TROPA, representados dentro do retângulo verde. Eles se comunicam com um banco de dados e, alguns deles, entre si. Para o microsserviço da administração há um banco de dados adicional isolado do outro por segurança. Caso uma invasão aconteça no banco dos serviços, não haverá acesso ao da administração, e vice-versa.

O TROPA se comunica com 2 serviços externos, um para processar pagamentos e outro para envio de e-mails. Há também um *broker* MQTTS para envio de mensagens da Operação para os usuários.

3.3.3.1 Tecnologias

As tecnologias utilizadas pelo sistema são descritas a seguir, mas mais detalhes estão nas seções 2.2 e **??**.

- Expo: *framework* para desenvolvimento de aplicativos para *Android*, *iOS* e *web*;
- React-Native: *framework* de interfaces de usuário para aplicativos *Android* e *iOS*;
- React: biblioteca para interfaces *web* baseadas em componentes;
- NestJS: *framework* para construção de aplicações NodeJS escaláveis e eficientes;
- Prisma: ORM (Mapeador Relacional de Dados) NodeJS e TypeScript com modelos de dados intuitivos, migrações automáticas e mais;
- PostgreSQL: sistema gerenciador de bancos de dados (SGBD) relacionais robusto;
- RabbitMQ: *broker* de mensagens maduro e confiável;

3.3.4 Visão Lógica

O sistema TROPA está decomposto em dois subsistemas, *front-end* e *back-end*, cada um com os devidos pacotes/módulos necessários.

3.3.4.1 *Front-end*

Este é o responsável por prover a interface do usuário e por comunica-se com o *back-end*. O *front-end* será *mobile*, mas também *web*. A Figura 9 mostra os pacotes desse subsistema.

Conforme evidenciado, os pacotes são:

- *Assets*: armazena as mídias usadas, sejam fotos ou vídeos;
- *Pages*: responsável palas páginas da aplicação;
- *Styles*: os estilos e as tipografias de textos e demais elementos gráficos ficam aqui;
- *Routes*: configura as rotas do aplicativo;
- *Components*: responsável pela criação de componentes de interface reutilizáveis à aplicação.

Figura 9 – Diagrama de pacotes do *front-end*

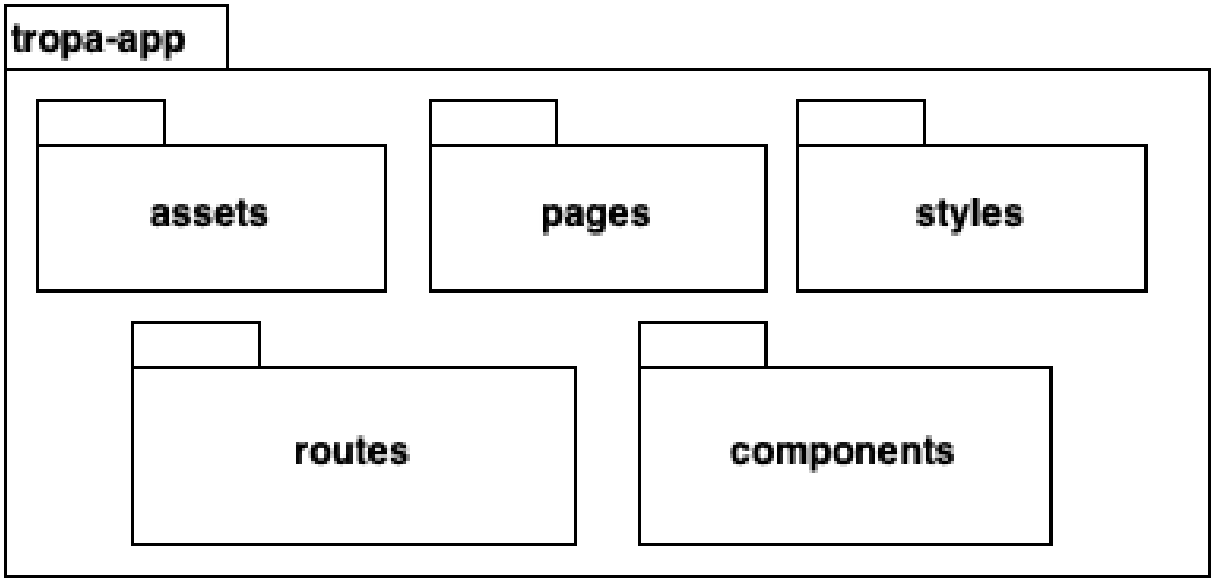
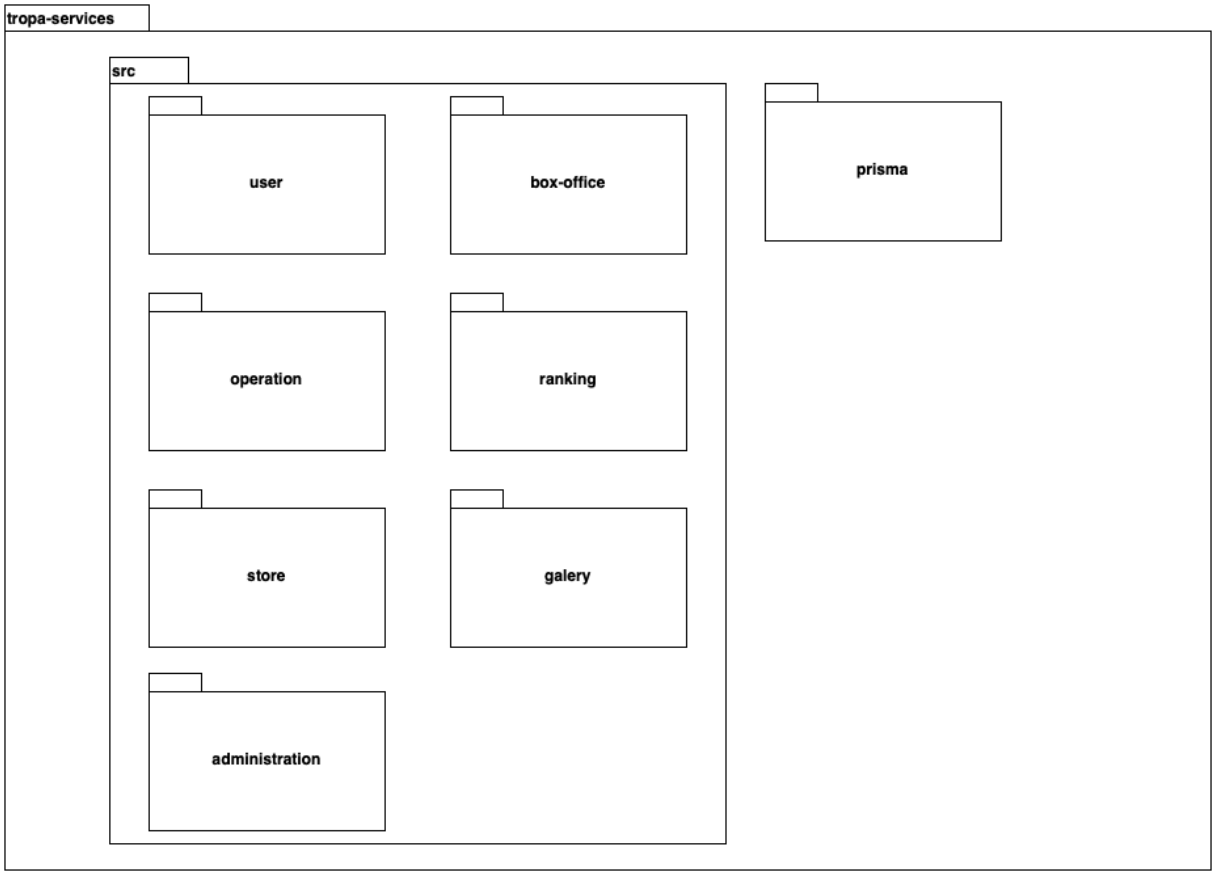


Figura 10 – Diagrama de pacotes do *back-end*



3.3.4.2 *Back-end*

Este é o subserviço responsável pela lógica negocial e de dados do sistema. A Figura 10 mostra os pacotes do *back-end*:

Conforme evidenciado, os pacotes são:

- Prisma: configura a comunicação com o banco de dados;
- *User*: responsável pela implementação e pelo controle do microsserviço dos usuários, exceto pelo usuário administrador;
- *Box-office*: responsável pela implementação e pelo controle do microsserviço da bilheteria;
- *Operation*: responsável pela implementação e pelo controle do microsserviço das operações, ou seja, dos jogos/eventos de *Airsoft*;
- *Ranking*: responsável pela implementação e pelo controle do microsserviço do *ranking*, conforme a pontuação dos usuários;
- *Store*: responsável pela implementação e pelo controle do microsserviço da loja;
- *Galery*: responsável pela implementação e pelo controle do microsserviço da galeria;
- *Administration*: responsável pela implementação e pelo controle do microsserviço da administração, exclusivo de usuários administradores. Esse pacote tem um subpacote Prisma próprio.

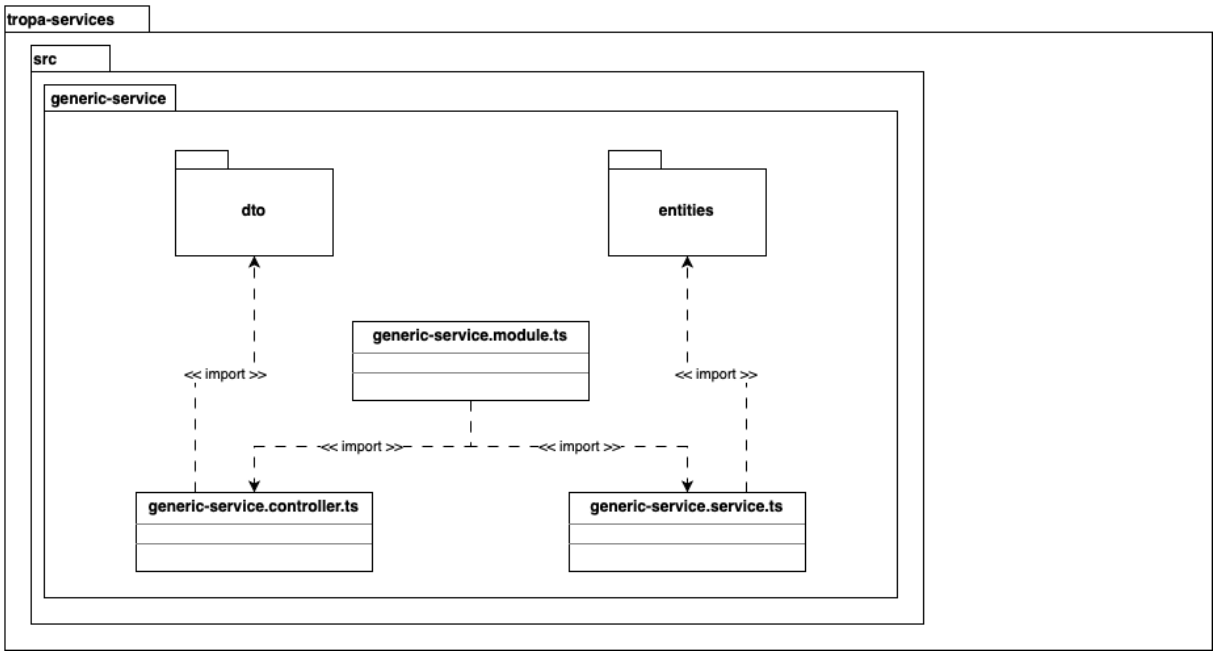
3.3.4.2.1 Pacotes dos microsserviços

Os pacotes do *back-end* mostrados anteriormente seguem um padrão de arquitetura parecido com o MVP (*Model, View, Controller*), exceto pelo pacote Prisma. Tendo em vista que essa mesma organização estrutural segue para cada um dos microsserviços, a Figura 11 apresenta um serviço genérico "*generic-service*" para ilustrar os pacotes de cada microsserviço.

Como mostrado, há dois pacotes "dto" e "*entities*", bem como 3 classes, "*generic-service.module.ts*", "*generic-service.controller.ts*" e "*generic-service.service.ts*":

- DTO: é uma camada de validação de dados de requisições, garantindo acurácia e consistência nos dados que trafegam a rede do sistema. Ela é usada pelo controlador;
- *Entities*: as entidades atuam, junto às "*services*", como camada de modelo do MVC. As entidades definem as estruturas de dados, representando objetos reais armazenados no banco de dados;

Figura 11 – Diagrama de pacotes genérico para os microsserviços do TROPA



- *generic-service.module.ts*: módulo serve como ponto de entrada do microsserviço, especificando quem é seu controlador e quem é seu serviço;
- *generic-service.controller.ts*: o controlador atua como as camadas *controller* e *view* do MVC, recebendo requisições, comunicando-se com a camada modelo, e retornando a resposta ao requisitante;
- *generic-service.service.ts*: esta atua como a camada *model* (modelo) do MVC, fazendo a lógica negocial e interagindo com os dados da aplicação.

Além dos pacotes apresentados, tanto do *back-end* quanto do *front-end*, é importante mostrar as principais classes do TROPA. Para isso, a Figura 12 representa o diagrama de classes.

Tal diagrama foi elaborado a partir do DLD (Figura 18) do sistema. Nota-se que vários tipos de usuários tem atributos em comum, por isso há uma classe pai Usuário, da qual outras herdam. Os relacionamentos mostrados no DLD se traduzem para as agregações e composições do diagrama de classes.

3.3.5 Visão de Processos

Para essa visão, optou-se por mostrar de forma genérica como ocorre a comunicação no sistema. Para isso foi feito o diagrama de comunicação, representado pela Figura 13.

Muitas requisições são feitas para suprir as necessidades do usuário. Via de regra, todas começam chamando a controladora do microsserviço adequado. Essa controladora

Figura 12 – Diagrama de classes

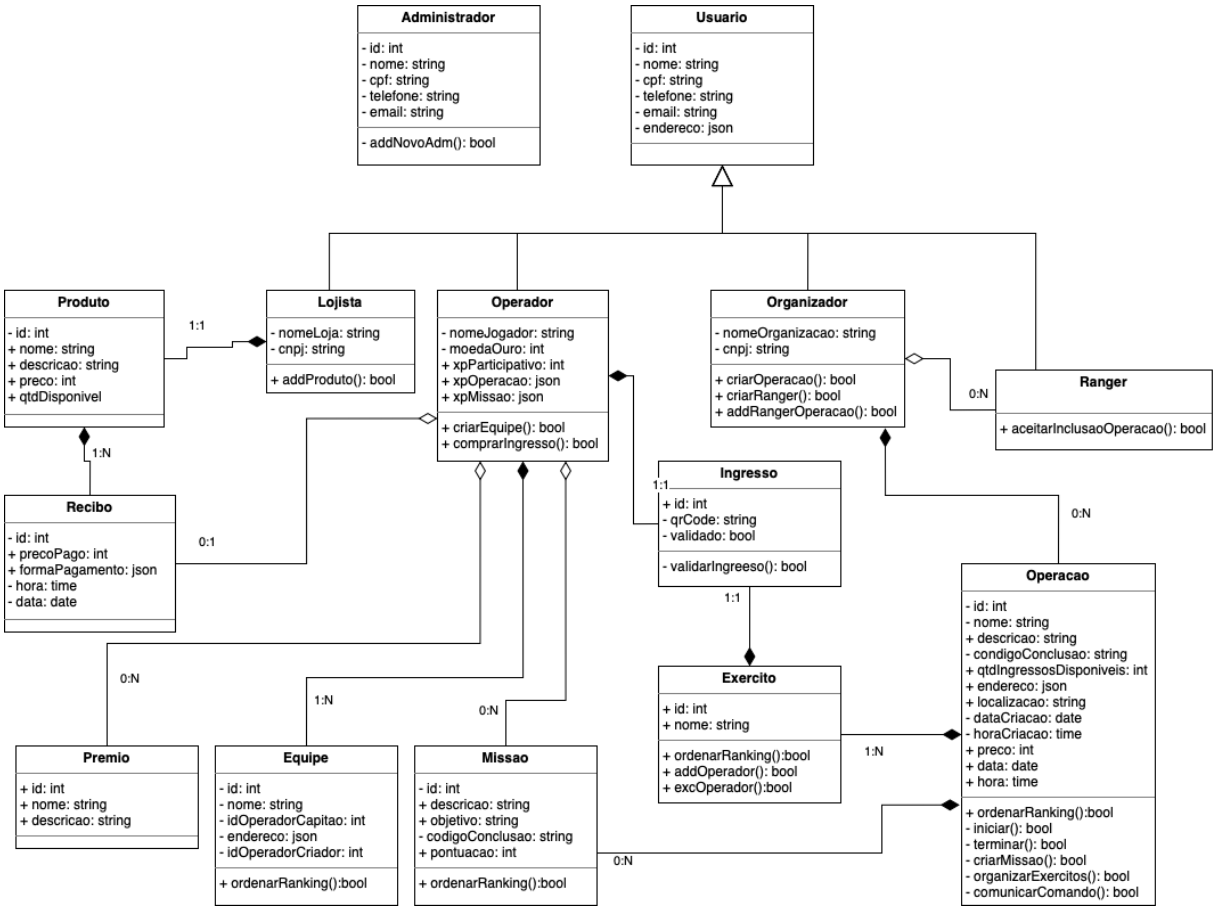
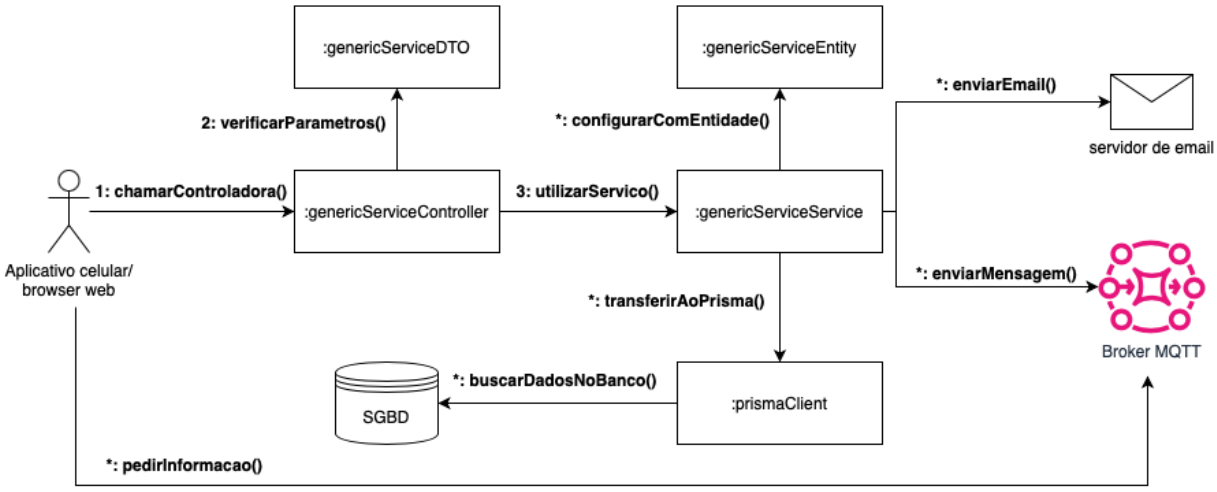


Figura 13 – Diagrama de comunicação genérico para o TROPA



verifica se os dados da requisição estão corretos. Caso positivo, ela chama a camada de serviço. Esta pode tanto se comunicar com as entidades, para depois pedir ao "cliente-Prisma" um dado armazenado pelo SGBD (Sistema Gerenciador de Bancos de Dados), quando enviar um e-mail ou uma mensagem relacionada a alguma operação de *Airsoft* por meio do *broker*.

3.3.6 Visão de Implantação

A implantação do sistema TROPA, parte do *front-end*, será tanto por um aplicativo móvel, o qual os usuários baixarão nos dispositivos celulares deles, quanto utilizando dois servidores *web*: um para a aplicação em si; e outro para o painel de administração. Já a parte do *back-end*, como mostra a Figura 14, se dará por meio: de um servidor dedicado para cada microsserviço; de outro para cada banco de dados; de um terceiro servidor para o *broker* de mensagens escolhido; e da alocação de um *gateway/load balancer*. Todos esses recursos em nuvem.

Os microsserviços de usuário (*user*) e administração (*administration*) se comunicam com todos os demais. Cores diferentes foram usadas para destacar as linhas de comunicação de cada um. O microsserviço de operação (*operation*) se comunica tanto com os de galeria (*galery*) e *ranking*, quanto com o *broker*, para enviar mensagens aos operadores durante uma operação. Ainda, a galeria também se comunica com a loja (*store*).

3.3.7 Visão de Implementação

O *software* TROPA está dividido em 4 camadas, como mostra a Figura 15.

A camada em que acontece a interface com o usuário é a do *front-end*. O fluxo de informação segue para o *gateway*, que irá direcionar o caminho para o microsserviço do *back-end* apropriado. Esse último faz a lógica negocial, comunica-se com a camada de dados e redireciona o fluxo da informação para a primeira camada.

A mesma ideia pode ser representada por meio dos componentes do sistema, os quais são ilustrados na Figura 16. O usuário interage com o ponto de entrada do *front-end*, o App.tsx, por meio de uma interface no aplicativo celular.

O *Gateway* direciona a requisição para um dos microsserviços. Estes estão representados em seus respectivos módulos por meio do arquivo "-module.ts", que é a quem os expõe para comunicação. Há ainda um cliente Prisma que é responsável por fazer a ponte entre os microsserviços e o banco de dados.

De modo semelhante acontece o fluxo para o usuário administrador. Este interage com uma aplicação *web* específica, a qual utiliza o microsserviço da administação ("administration.module.ts"). Este pode comunicar-se com qualquer um dos microsserviços do sistema, e também com seu próprio banco de dados por meio do cliente Prisma específico.

Figura 14 – Diagrama de implantação do sistema TROPA

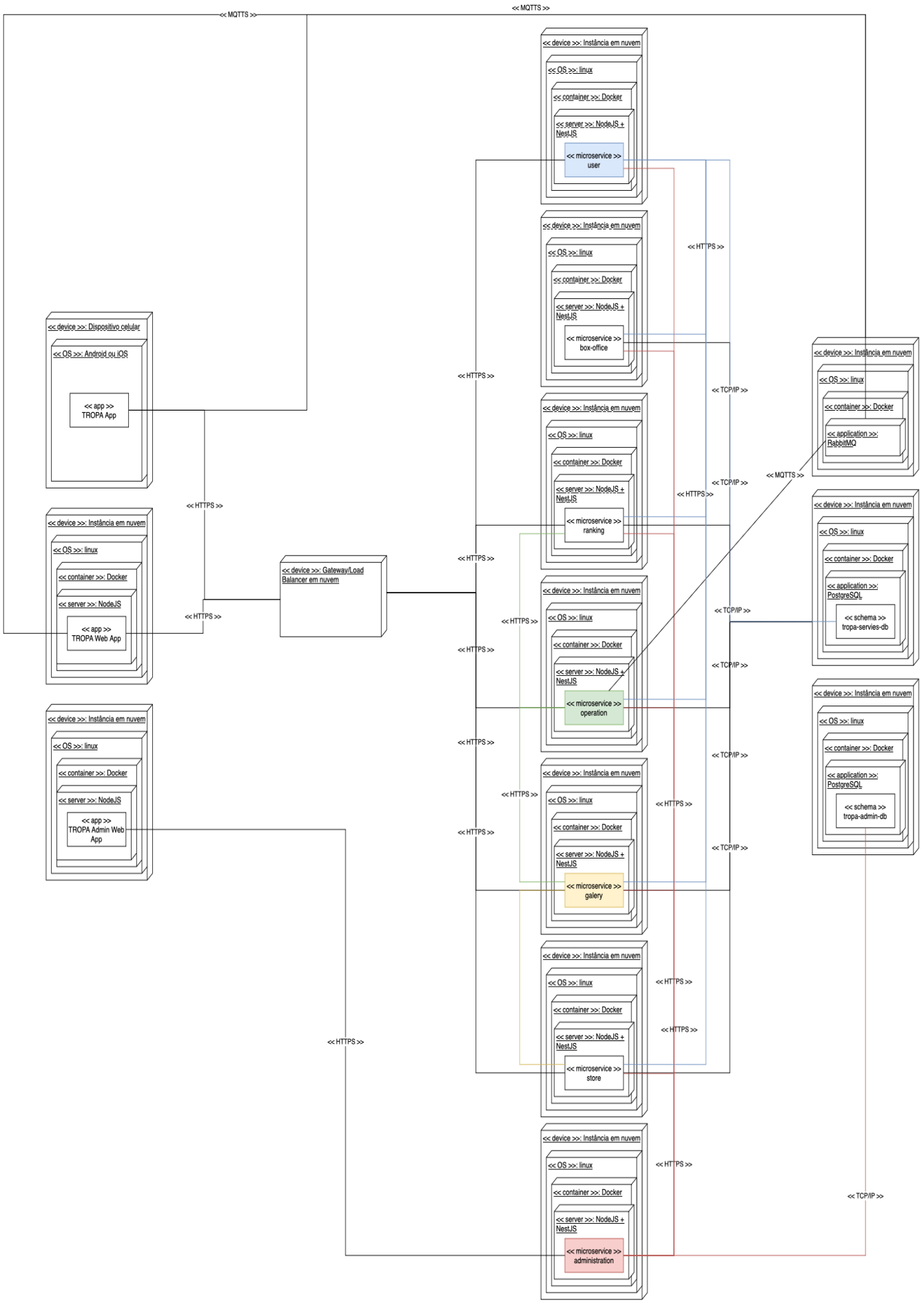


Figura 15 – Camadas do TROPA

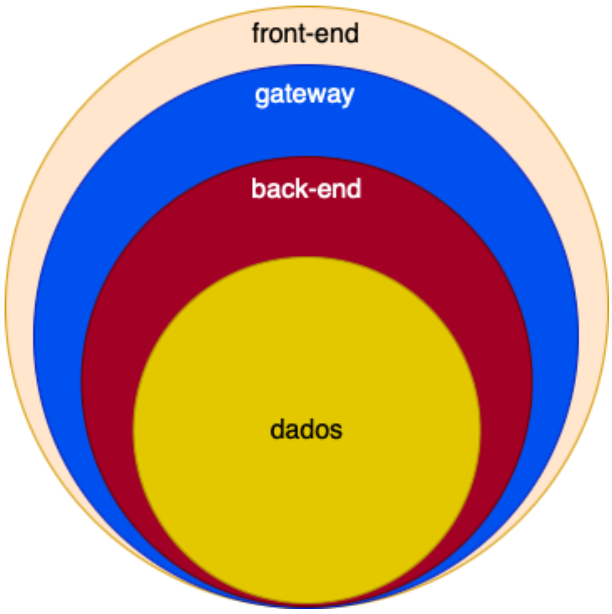


Figura 16 – Diagrama de componentes

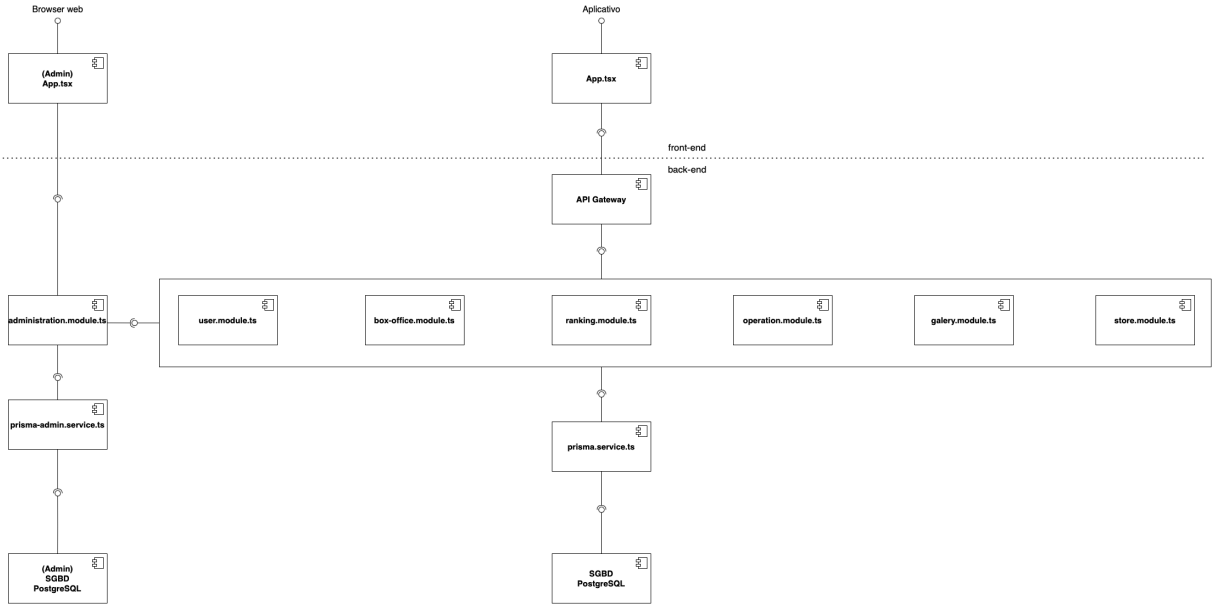
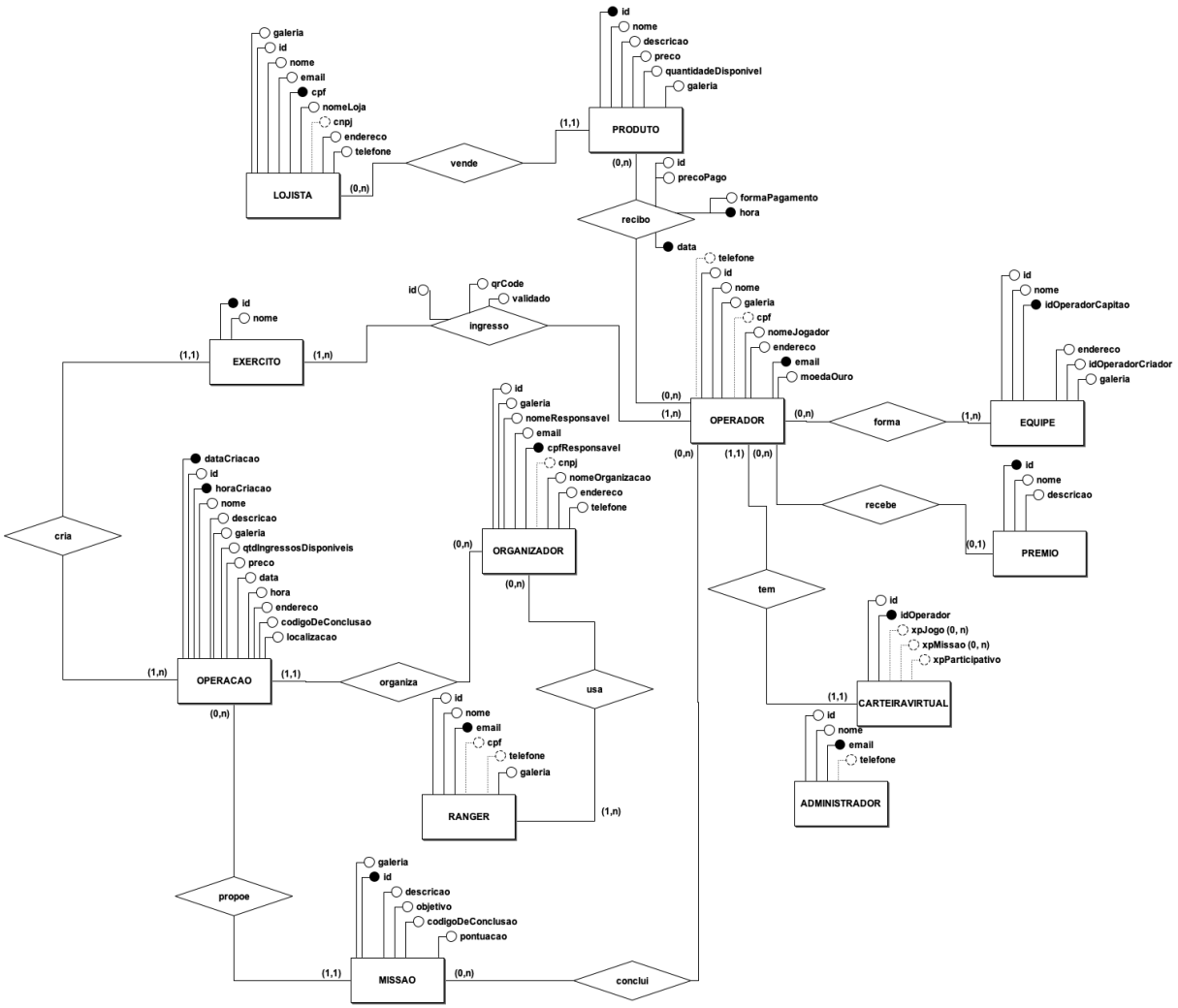


Figura 17 – DE-R do TROPA



3.3.8 Visão de Dados

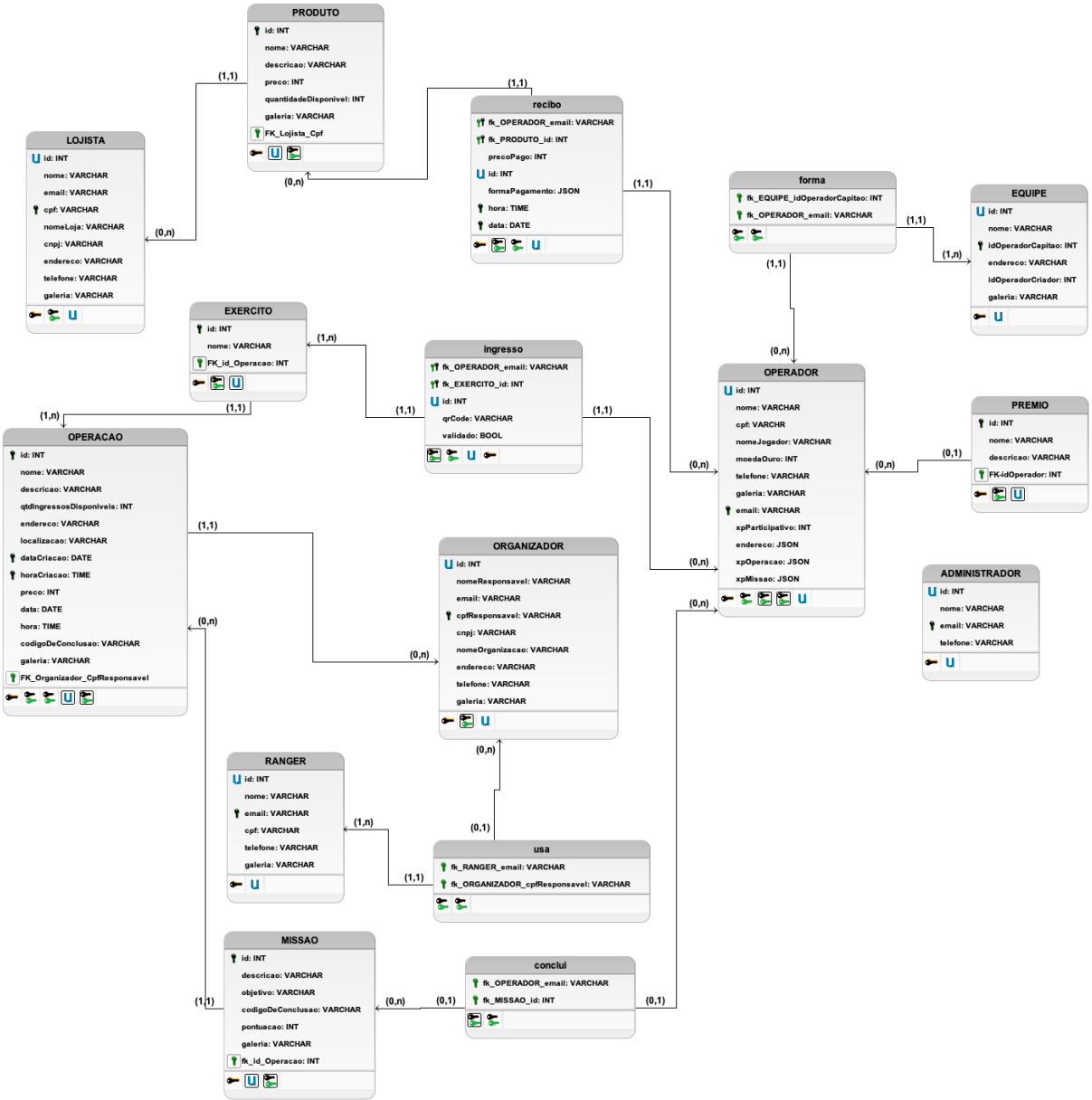
O TROPA precisará persistir dados dos usuários, dos jogos/eventos (agora nomeados como Operação, para simplificar), dos produtos a serem vendidos e mais. Para iniciar a modelagem disso, foi feito o DE-R (Diagrama Entidade Relacionamento), mostrado na Figura 17.

A partir disso gerou-se o DLD (Diagrama Lógico de Dados), o qual representa uma perspectiva mais próxima da implementação dos dados que serão armazenados. Veja a Figura 18.

3.3.9 Qualidade

A arquitetura do sistema TROPA foi pensada e projetada de forma que atendesse aos requisitos de qualidade (Seção 3.2.4.1). Sendo assim, a seguir é apresentada, para cada requisito contemplado, uma descrição de como a arquitetura contribui-lhe.

Figura 18 – DLD do TROPA



- RNFQ 1 - Aptidão: conforme demonstrado pelos diagramas de pacote, comunicação e componentes, o sistema está modularizado o suficiente para atender esse requisito;
- RNFQ 2 - Interoperabilidade: a escolha das tecnologias a serem utilizadas para a construção do sistema obedeceu a esse requisito;
- RNFQ 3 e 11 - Tolerância a falhas e Adaptabilidade: a arquitetura de microsserviços contribui para que falhas em uma funcionalidade não impactem outra. Também facilita a manutenção em ambientes de implantação;
- RNFQ 9 e 10 - Mutabilidade e Testabilidade: a modularização do sistema e a escolha de tecnologias permite a simplicidade nas eventuais mudanças bem como ao testá-las.

3.3.10 Segurança

De forma semelhante aos requisitos de qualidade, a arquitetura do sistema TROPA também foi pensada e projetada para que contribuísse aos requisitos de segurança (Seção 3.2.4.2). Sendo assim, a seguir é apresentada, para cada requisito contemplado, uma descrição de como a arquitetura contribui-lhe.

- RNFSec 1, 2 e 3 - Confidencialidade, Integridade e Disponibilidade: requisitos parcialmente atendidos por causa da utilização de um *Gateway/Load Balancer*. Este esconde os microsserviços da *Internet*, e balanceia as requisições ao sistema.

3.4 TROPA: MVP

O mínimo produto viável definido ao *software* **TROPA** compreendeu os microsserviços de usuário, bilheteria e operação, os quais representam o *core* do sistema de forma que a principal funcionalidade (definida em 3.1.1) fosse concretizada. Somado a isso, foram desenvolvidos módulos complementares para realizar a autenticação, a autorização, o envio de e-mails, de notificações e a realização dos pagamentos.

Com isso, o TROPA já suporta que usuários operadores, organizadores e *rangers* desempenhem suas principais macro-tarefas no aplicativo, a saber, respectivamente: compra de ingressos para participar em operações de *Airsoft*, formação de times (equipes); organização, venda e gerenciamento de operações; auxílio no gerenciamento de operações.

3.4.1 Implementação

Para que o **MVP TROPA** fosse implementado, tanto *backend* quanto *frontend* precisaram ser desenvolvidos, sendo que sofreram leves adaptações em relação ao que foi

projetado no documento de arquitetura (Seção 3.3). A título do primeiro, a Tabela 6 evidencia alguns dos *endpoints* desenvolvidos.

Tabela 6 – Alguns *endpoints* do *backend*

Módulo	**Método**	**Endpoint**
App	GET	/
	GET	/health/db
User	POST	/user/operator
	POST	/user/organizer
	POST	/user/ranger
	GET	/user
	GET	/user/operators
	GET	/user/organizers
	GET	/user/rangers
	GET	/user/operator/{id}
	PATCH	/user/operator/{id}
	GET	/user/organizer/{id}
	PATCH	/user/organizer/{id}
	GET	/user/organizer/refreshCode/{refreshCode}
	GET	/user/ranger/{id}
	PATCH	/user/ranger/{id}
	PATCH	/user/operator/{id}/connect
	PATCH	/user/operator/{id}/disconnect
	PATCH	/user/ranger/{id}/connect
	PATCH	/user/ranger/{id}/disconnect
	DELETE	/user/{id}
Team	POST	/user/team
	GET	/user/teams
	GET	/user/team/{id}
	PATCH	/user/team/{id}
	DELETE	/user/team/{id}
BoxOffice	POST	/box-office
	GET	/box-office/success

Módulo	**Método**	**Endpoint**
	GET	/box-office/cancel
	GET	/box-office/all
	GET	/box-office/{id}
	PATCH	/box-office/checkin/{id}
	POST	/box-office/webhook
Operation	POST	/operation
	GET	/operation
	GET	/operation/mine
	GET	/operation/citystate
	GET	/operation/name
	GET	/operation/{id}
	PATCH	/operation/{id}
	DELETE	/operation/{id}
	POST	/operation/operationimage/{id}
	GET	/operation/operationimage/{id}
	POST	/operation/{id}/command
	POST	/operation/{id}/accomplished
	GET	/operation/conclusionCode/{id}
Army	POST	/operation/army
	GET	/operation/army/{operationId}
	PATCH	/operation/army/{id}
	DELETE	/operation/army/{id}
	POST	/operation/army/rearrange
Auth	POST	/auth/login

Para o microsserviço de usuários (***User***), foi desenvolvido o CRUD completo para operadores, organizadores, *rangers* e também para times, que são compostos exclusivamente por dois ou mais operadores. Há ainda outras rotas que desempenham funções específicas, como, por exemplo, as com sufixos *connect* e *disconnect*. Elas servem para, respectivamente:

- No caso dos operadores: vinculá-los e desvinculá-los a um time;

- No caso dos *rangers*: vinculá-los e desvinculá-los a uma organização, representada por um usuário organizador (*organizer*).

O microsserviço da bilheteria (***box-office***) é responsável pela compra/venda de ingressos para as operações. Nesse sentido, o *endpoint* "/box-office" recebe um pedido de compra vindo do *frontend*, e devolve a este informações relacionadas a uma "intenção de pagamento". Com isso, o aplicativo consegue direcionar o usuário operador a pagar pelo ingresso de uma operação. Nas ocasiões de sucesso e falha dessa compra, o *backend* recebe eventos vindos do serviço de pagamentos externo por meio do *endpoint* "/box-office/webhook". Assim, a decisão de criar (sucesso) ou não (falha) um ingresso (*ticket*) para o operador é tomada e ele é avisado.

Para o microsserviço de operações (***operation***) também há um CRUD completo, inclusive para a divisão dos operadores participantes em exércitos. O *endpoint* "/auth/login" realiza a autenticação dos usuários para cada uma das rotas protegidas (indicadas com um pequeno cadeado à direita) da aplicação, sendo parte do módulo de autenticação: complementar aos microsserviços do *software* TROPA.

Em relação ao ***frontend***, foram implementadas interfaces correspondentes às funcionalidades do MVP TROPA. Para o usuário operador, a tela inicial do aplicativo mostra as operações à venda na cidade em que ele escolher como filtro. A tela de perfil segue um padrão para os usuários, visto que existem informações semelhantes entre eles, mas difere em alguns aspectos: para o operador e *ranger*, são apresentadas as operações passadas e futuras dele dentro de uma "carteira"; para o organizador, além das operações passadas e futuras, é possível criar novos eventos.

Há também a tela dos times existentes na plataforma e, para os operadores, a possibilidade de criar times. Além disso, vários outros elementos visuais e componentes fazem parte da lógica implementada no *frontend*.

3.4.2 Módulos complementares

O Nestjs, *framework* escolhido para o *backend* TROPA, recomenda fortemente o uso de módulos como uma forma efetiva de organizar os componentes de uma aplicação (MYSLIWIEC, 2023). Embora seja possível executar módulos do Nestjs individualmente, como será feito na implantação desse trabalho, uma vez que os microsserviços desenvolvidos são módulos, os componentes complementares a seguir são compartilhados entre vários outros módulos do *backend*. O autor desse trabalho chama os módulos a seguir de "complementares" porque são fundamentais para auxiliar na lógica das funcionalidades do TROPA, ou seja, eles complementam a implementação dos componentes essenciais do sistema.

3.4.2.1 Autenticação: *auth*

Todos os usuários do TROPA são autenticados, inicialmente, por e-mail e senha ao utilizarem o *endpoint* "/auth/login" , mencionado anteriormente. Isso retorna ao *frontend* um *token* de acesso que contém três informações: número identificador do usuário; perfil do usuário; recurso que o usuário pode acessar no sistema. Para requisitar os *endpoints* protegidos do sistema, é obrigatório o uso do *token* de acesso no cabeçalho da requisição.

Uma vez que uma rota protegida é requisitada, entra em cena a guarda "JwtAuthGuard" do módulo *auth*. Guardas (*guards*) do Nestjs são classes de responsabilidade única: determinar se uma requisição será aceita ou não, dependendo de condições programadas (MYSLIWIEC, 2023). A "JwtAuthGuard" do TROPA é uma guarda que estende a guarda de autenticação padrão do Nestjs (*AuthGuard*), utilizando uma estratégia baseada em JWT (*Json Web Token*) para: validar o *token* de acesso da requisição; e popular o objeto "usuário" (*user*) da requisição com o conteúdo decodificado do JWT.

Com isso, todas as rotas que precisam ser protegidas são decoradas com "@UseGuards(JwtAuthGuard)" e, automaticamente, não serão acessíveis caso a requisição vinda do *frontend* não fornecer um JWT válido. Nesse caso a aplicação retorna uma resposta com *status* 401, informando que o requisitante não está autorizado.

3.4.2.2 Autorização: *authorization*

A autorização dos usuários no *software* TROPA é feita por meio de regras definidas em código. O módulo *authorization* conta com um serviço para fazer tais verificações: o "AuthorizationService", onde se encontram as regras. Este está injetado em cada microsserviço do sistema em que há a necessidade de verificar se determinada requisição pode ou não receber as informações que deseja, em totalidade ou parcialidade.

Para tornar isso possível, cada acesso restrito chama o serviço de autorização ainda no nível da controladora do microsserviço requisitado. Em outras palavras, cada *endpoint* em que há tal necessidade usa o "AuthorizationService" para assegurar que o usuário requisitante tem acesso total, parcial ou negado sobre a informação que deseja antes de chamar a classe de serviço em questão. Nos casos em que o acesso é total, toda a informação é retornada. Caso seja parcial, apenas dados considerados "públicos" ou não-sensíveis são fornecidos. Já quando o acesso é negado, uma resposta com código 403, proibido (*forbidden*), é retornada.

Há ainda uma configuração no TROPA que barra requisições vindas de servidores desconhecidos. Isso é feito com a "IpAuthorizationGuard", uma guarda customizada e habilitada globalmente em todo o *backend*. Isto é, ela funciona para todos os *endpoints* obrigatoriamente. Essa guarda verifica, para cada requisição, se o requisitante está na lista de servidores conhecidos do TROPA. Caso não esteja, uma resposta com *status* 403

(*forbidden*) é retornada, mas caso esteja a requisição é aceita, então entram em cena os outros mecanismos de segurança mencionados anteriormente.

3.4.2.3 Envio de e-mails: *mail*

As mensagens eletrônicas do TROPA são enviadas a partir do endereço "contato@tropairsoft.com". Para isso, dois serviços externos estão configurados, um principal e outro secundário, respectivamente: Sendgrid e Mailchimp. A escolha sobre qual deles utilizar se dá em tempo de execução, assim que o *backend* é inicializado, por meio de uma variável de ambiente.

Esse módulo foi feito utilizando o padrão de projeto *Factory*. Dessa forma, módulos que utilizam o serviço de e-mails TROPA não sabem se estão chamando um objeto da classe "SendgridService" ou da "MailchimpService". Ao invés disso, eles utilizam a fábrica "MailServiceFactory", a qual retorna uma instância de um daqueles serviços. Ambas "SendgridService" e "MailchimpService" implementam a interface "IMailService" do módulo *mail*.

Além disso, as informações dos e-mails enviados pelo TROPA, como conteúdo da mensagem e assunto, estão definidas em um arquivo específico, em Português. No futuro, caso o sistema precise enviar mensagens em outros idiomas, basta criar um arquivo com o conteúdo na língua desejada e fazer algumas alterações em código.

3.4.2.4 Pagamentos: *payment*

O módulo de pagamentos do TROPA também seguiu o padrão *Factory*, mas por enquanto está configurado apenas com o serviço Stripe. Sendo assim, na prática, não é uma fábrica, mesmo estando estruturada como uma. Para lidar com compra/venda de ingressos na aplicação, intenções de pagamentos são criadas e retornadas ao *frontend*, onde o pagamento realmente acontece. No *backend*, eventos de sucesso e falha dessas compras são tradados de forma a criar ou não um ingresso para o operador.

Além disso, o TROPA usa um recurso do Stripe chamado Stripe Connect. Por meio dele, usuários organizadores são encaminhados, no momento em que se registram com o TROPA, a fornecerem dados relativos a como querem receber pelas vendas do ingressos diretamente com o Stripe. O TROPA não lida com essas informações, mas sim cria as chamadas "contas conectadas" do Stripe para cada organizador. No momento em que acontece um pagamento de um ingresso, o Stripe automaticamente transfere os valores para a conta conectada do organizador, e uma taxa de serviço é depositada na conta do TROPA.

3.4.3 Atualizações no projeto e Diferenças na implementação

O projeto da arquitetura de um *software* é fundamental, mas nem sempre é seguido totalmente à risca, principalmente quando é feito antes da fase de desenvolvimento, como foi o caso desse trabalho. Nesse sentido, há algumas diferenças entre o que fora projetado na Seção 3.3 e o que realmente foi implementado.

Além disso, frequentemente acontecem atualizações no projeto (*design*) do sistema e na arquitetura durante o ciclo de vida de desenvolvimento. Isso também foi verdade para o TROPA em alguns aspectos, ora por conta de uma adequação a uma tecnologia, ora para dar mais robustez a alguma funcionalidade do sistema.

A seguir serão apresentadas algumas atualizações de projeto e diferenças de implementação.

3.4.3.1 Comunicação entre microsserviços

Durante o planejamento da arquitetura, foi previsto que a comunicação entre serviços aconteceria por meio de chamadas HTTP, simulando um ambiente de microsserviços distribuídos. No entanto, como os módulos foram implementados em um único projeto monolítico modularizado usando **NestJS**, a comunicação foi feita via *injeção de dependência*, aproveitando os recursos do próprio framework para realizar chamadas diretas entre os módulos internos da aplicação.

Essa abordagem trouxe benefícios como menor latência, maior simplicidade de implementação e ausência da sobrecarga de protocolos HTTP, além de facilitar os testes unitários e de integração. Por outro lado, ela representa um acoplamento maior entre os componentes do sistema, o que pode ser revisto em versões futuras, caso se deseje migrar para uma arquitetura de microsserviços distribuídos, com comunicação assíncrona ou via API Gateway.

3.4.3.2 Diagramas de pacotes e estruturação em pastas

A estrutura proposta nos diagramas de pacotes iniciais diferiu da organização real do código-fonte. Durante o desenvolvimento, optou-se por uma abordagem mais alinhada com o estilo do NestJS, que adota a organização por módulos independentes, cada um contendo seus próprios controladores, serviços e entidades.

Além disso, a divisão em pastas refletiu melhor a separação de responsabilidades e a modularização do sistema, facilitando tanto a leitura quanto a manutenção do código. Os pacotes foram reestruturados para refletir melhor os domínios de negócio, como *box-office*, *auth*, *notification*, *payment*, entre outros.

3.4.3.3 Sistema de notificações *push*

O projeto inicial previa que o sistema de notificações usaria exclusivamente o **RabbitMQ** para comunicação assíncrona entre serviços e publicação de eventos. Embora o RabbitMQ tenha de fato sido utilizado, principalmente para envio de mensagens internas no backend, a implementação do sistema de notificações *push* para o aplicativo móvel exigiu a integração com o serviço **Expo Notifications**, próprio do ecossistema do *React Native* com *Expo*.

Assim, foi necessário implementar uma fila intermediária e um serviço de orquestração que recebesse eventos via RabbitMQ e os transformasse em notificações compreendidas pelo Expo. Essa solução híbrida possibilitou a comunicação eficaz entre o backend e os dispositivos móveis dos operadores.

3.4.3.4 Implantação do sistema

Durante o planejamento da arquitetura, optou-se pela utilização da nuvem **Microsoft Azure** para hospedar o *backend* do sistema, aproveitando os créditos estudantis (US$ 100) fornecidos pelo programa *GitHub Student Developer Pack*.

Essa decisão foi motivada pela integração nativa com *containers* Docker, suporte à execução de aplicações Node.js e a disponibilidade de serviços gerenciados como bancos de dados relacionais.

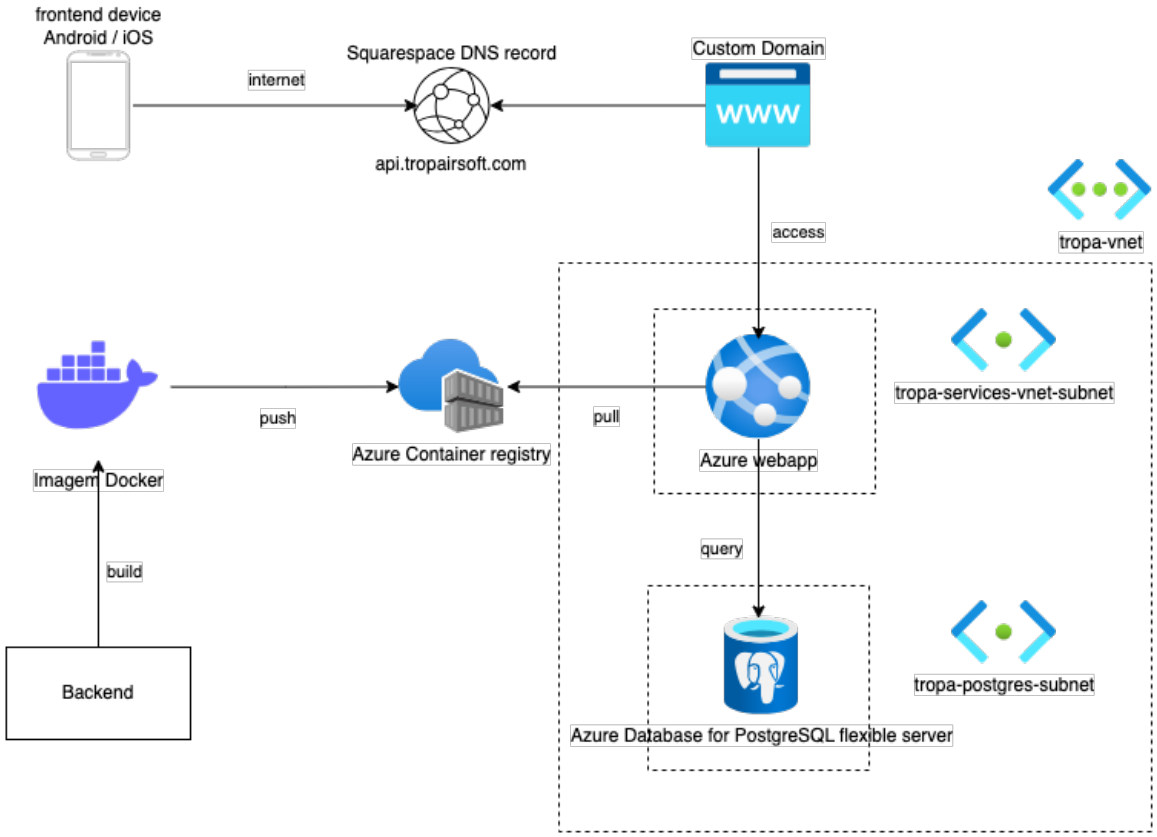
A Figura 19 ilustra como, na prática, a arquitetura do MVP TROPA foi implantada. Ela também destaca os principais recursos de nuvem utilizados e o fluxo de comunicação entre os componentes.

A aplicação backend foi construída em uma imagem Docker customizada e publicada no **Azure Container Registry (ACR)**. Em seguida, um **Azure Web App for Containers** faz o *pull* da última versão (*latest*) da imagem e a executa, com integração a uma **Virtual Network (VNet)** privada.

O serviço Web App encontra-se conectado à **sub-rede** `tropa-services-vnet-subnet` da VNet, permitindo acesso seguro à instância de banco de dados PostgreSQL hospedada na própria Azure. Essa instância foi provisionada como **Azure Database for PostgreSQL - Flexible Server**, com conectividade privada e hospedada na **sub-rede** `tropa-postgres-subnet`. Somente o Web App foi autorizado a se comunicar com a base de dados.

A resolução de nomes DNS para o endereço interno do banco foi viabilizada via **Private DNS Zone**, vinculada à VNet. Para permitir a comunicação segura entre a aplicação e a base, foram configurados **grupos de segurança de rede (NSGs)** permitindo explicitamente o tráfego na porta 5432 entre as sub-redes da aplicação e do banco de dados.

Figura 19 – Diagrama de arquitetura e recursos de implantação do MVP TROPA



O processo de *deployment* da aplicação foi automatizado por meio de *scripts bash* e um *Dockerfile* específico para produção, seguindo boas práticas de CI/CD. Esse processo inclui a construção da imagem, publicação no ACR e reinício automático da aplicação hospedada.

3.5 O aplicativo

À partir do desenvolvimento do MVP, foi desenvolvida a interface de usuário do *frontend*, concretizado por meio do aplicativo *mobile* **tropaapp**. À seguir, as principais telas da aplicação são evidenciadas.

A Figura 20 mostra a principal tela do aplicativo, o ponto de entrada do usuário. Lá ele visualiza e busca por operações para participar.

Ao escolher e clicar em uma operação, são exibidos mais detalhes sobre ela, como mostra a Figura 21.

Para cada tipo de usuário, a tela de perfil se diferencia em alguns elementos. A Figura 22 exibe a tela de perfil de operador, enquanto a Figura 23 mostra a de organizador, e a Figura 24 a de *ranger*.

Para os usuários operadores existe a tela da carteira virtual, como mostra a Figura

Figura 20 – Tela de Operações

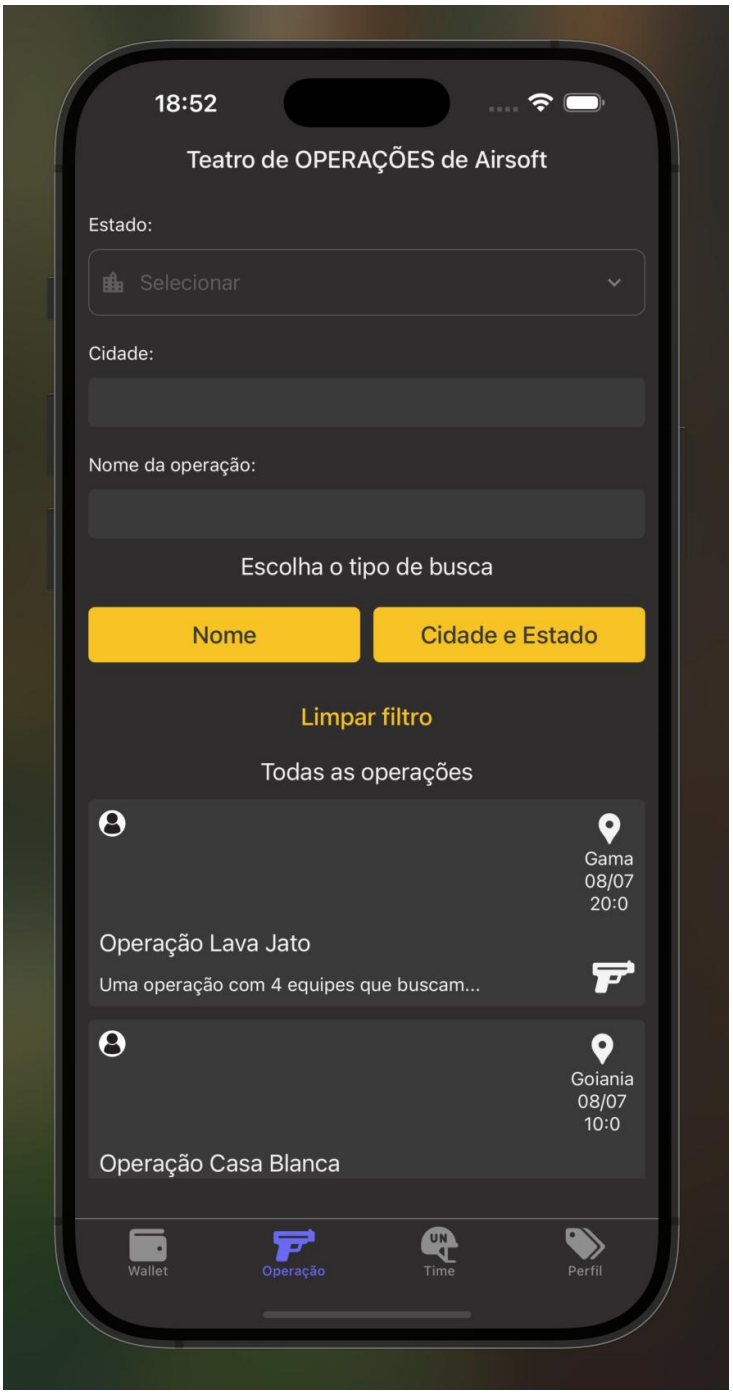


Figura 21 – Tela de Visualização de uma Operação

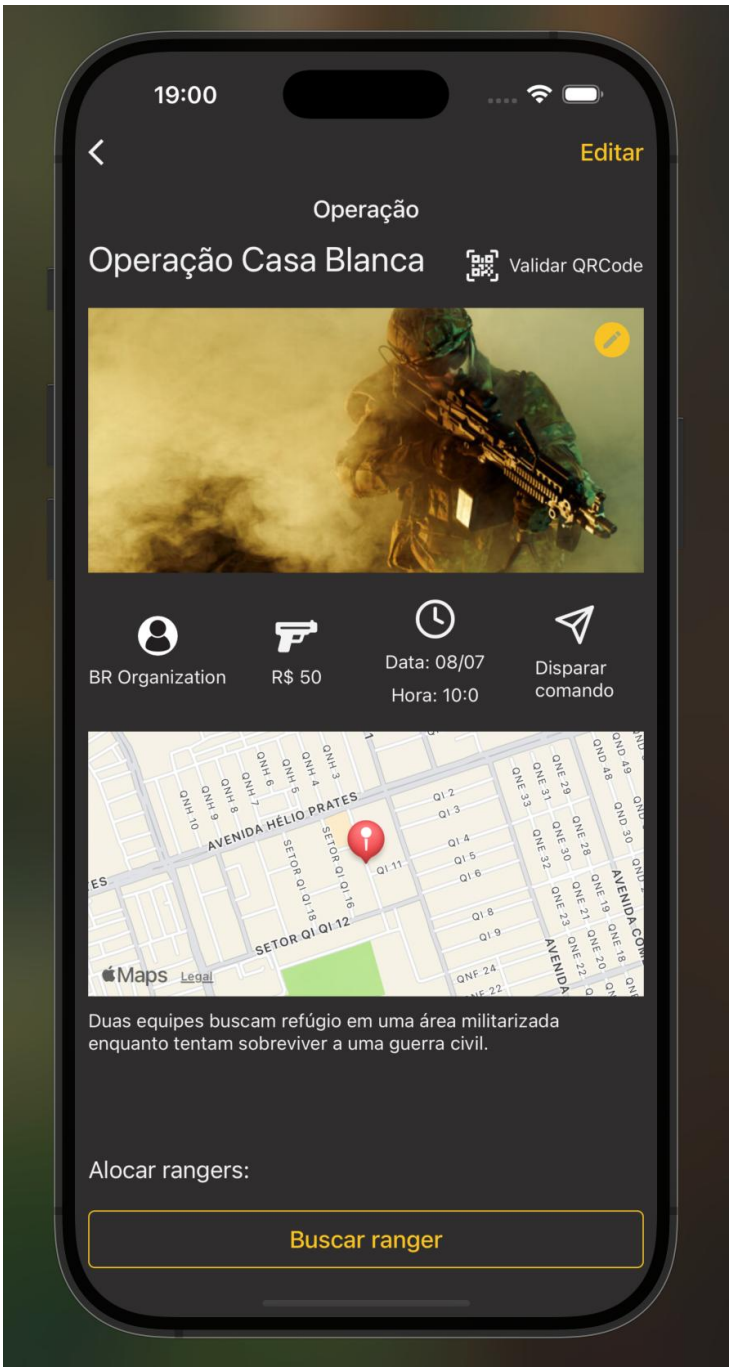
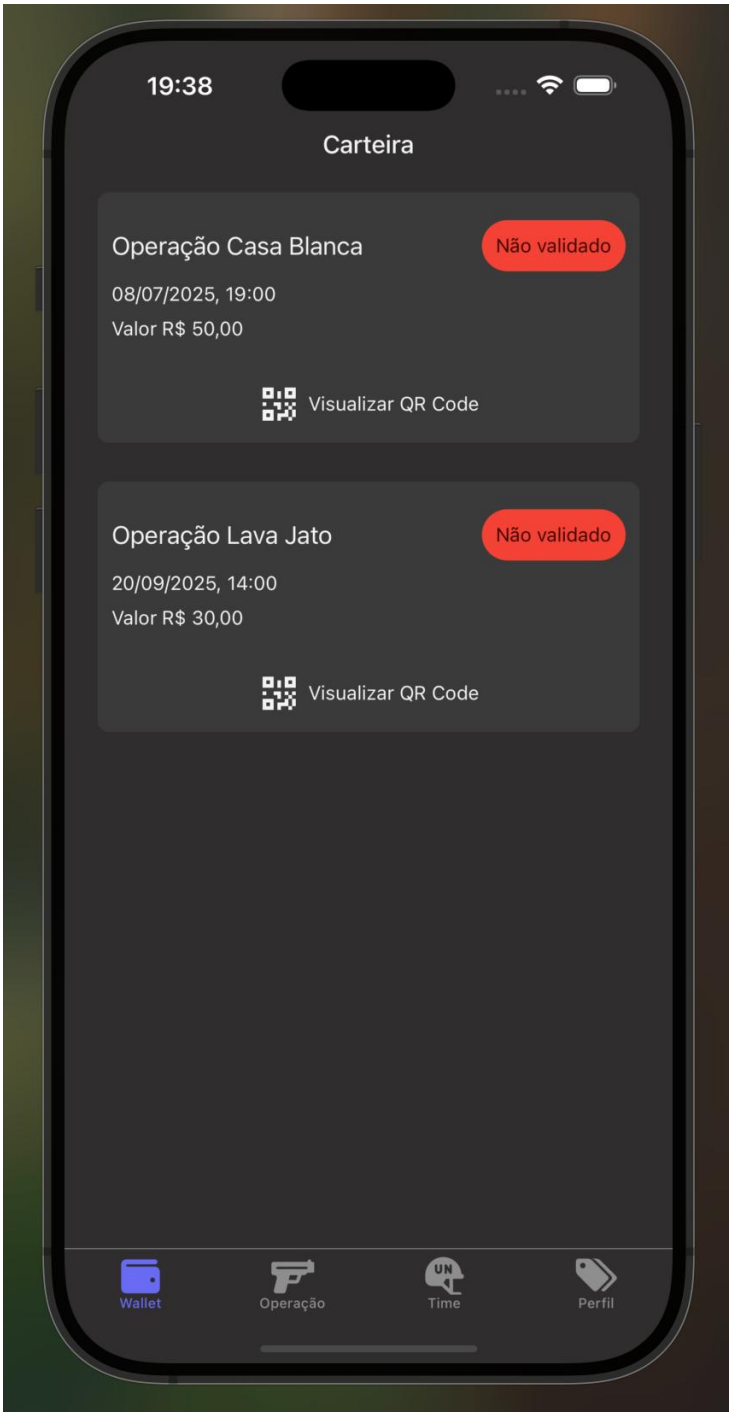


Figura 22 – Tela de Perfil de Operador

Figura 23 – Tela de Perfil de Organizador

Figura 24 – Tela de Perfil de Ranger

Figura 25 – Tela da Carteira Virtual



25. Nela o operador encontra os ingressos para as operações que ele irá participar, bem como para as que já participou.

Ao selecionar um ingresso, abre-se a tela de visualização do QRCode, como evidencia a Figura 26.

Outra aba de navegação importante do aplicativo é a tela de exibição de times. Esta é representada pela Figura 27.

Figura 26 – Tela de Visualização de um Ingresso

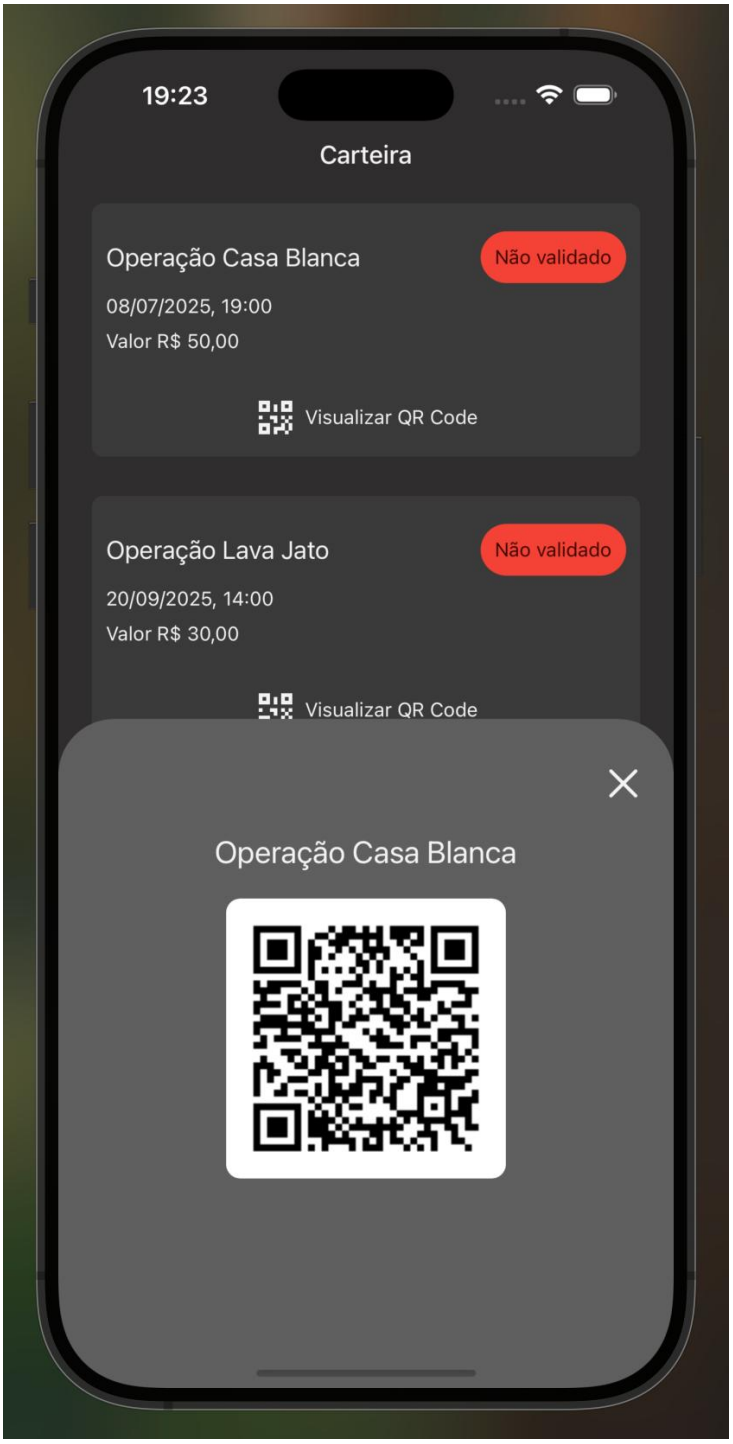


Figura 27 – Tela de Times

3.6 Validando o MVP

Para validar o funcionamento, a viabilidade técnica e a necessidade do TROPA, foram feitas várias *builds* do aplicativo durante o curso de aproximadamente um mês, as quais foram disponibilizadas para alguns usuários interessados na solução. Estes frequentemente relatavam sobre o uso do aplicativo. Concomitantemente, o *backend* já havia sido implantado na Azure e a API estava disponível em:

https://**api.tropairsoft.com**

Além disso, durante uma rodada de testes de uso da aplicação, algumas tarefas específicas foram passadas aos usuários para que fosse possível identificar pontos positivos e negativos no aplicativo. Isso também foi feito para validar algumas principais funcionalidades, bem como para avaliar tanto a interface (*UI*) quanto a experiência de usuário (*UX*). As tarefas variaram conforme o perfil de usuário, mas eram complementares.

Usuário 1 - perfil de organizador

Tarefas:

- Baixe e instale o aplicativo a partir do link: https://expo.dev/accounts/clever-ltda/projects/tropa-app/builds/<id da build>
- Crie sua conta de organizador;
- Faça login;
- Crie uma operação chamada "Valida MVP", começando no dia em que receber a tarefa, e terminando 24h depois, com o valor de R$1,00 e com localização no DF;
- Quando houver operadores que compraram ingressos, divida-os em exércitos conforme achar melhor;
- Use a função de leitura de QR Code para validar ingressos;
- Use a função de disparar comandos para a operação: pelo menos 1 vez cada comando;
- Na data e horário de final de operação, dispare o comando de "fim de game";

Usuários 2, 3 e 4 - perfis de operador

Tarefas:

- Baixe e instale o aplicativo a partir do link: https://expo.dev/accounts/clever-ltda/projects/tropa-app/builds/<id da build>

- Crie sua conta de operador;
- Faça login;
- Compre um ingresso na Operação Valida MVP, do DF;
- Feito isso, navegue para sua carteira e verifique que seu ingresso está presente;
- (Opcional) Compareça ao local da operação na data e no horário especificados. Apresente seu QR Code para que o organizador valide seu ingresso;
- Fique atento(a) aos comandos disparados e verifique em seu e-mail que os recebeu. Ex.: pista fria; pista quente; fim de game;
- Ao final da operação, caso a funcionalidade de digitar código de conclusão esteja disponível, utilize-a. Depois, verifique no seu perfil que ganhou XP;

Durante a realização dessas tarefas, alguns usuários já começaram a relatar pontos bons e ruins no aplicativo, bem como dificuldades no uso. O autor deste trabalho absteve-se ao máximo de auxiliá-los para que suas experiências fossem o mais realistas possíveis, de forma a não haver interferência nos testes.

3.6.1 *Feedbacks* coletados

Após a realização das tarefas, os usuários que participaram foram indagados a relatar sobre a experiência de uso do aplicativo, a qualidade da interface e a efetividade das funcionalidades disponíveis. Os seguintes *feedbacks* foram coletados, na íntegra:

Usuário 1 - perfil de organizador

> "durante o uso do aplicativo tropa senti uma facilidade para criar e gerir minhas operações, no geral o aplicativo funciona muito bem e é bem didático para nos organizadores, facilitando a criação e a gerência dos jogos, me permitindo gerenciar os rangers/juízes e criar apenas dois times, o'que poderia ser aumentado, pois dependendo da operação existe a necessidade de mais equipes, os comandos são fáceis e permite a qualquer momento parar o jogo e continuar de forma fácil e pratica, no entanto percebi alguns pontos de ajuste, na tela de login o campo para escolher o tipo da conta fica oculto pelo botão login quando se esta colocando login e senha, em algumas telas o contraste da escrita preta com o fundo cinza atrapalha a visualização, sugestão trocar a cor da escrita, seria interessante o organizador poder alterar informações como a data, horário e local da operação, mas no geral o aplicativo é bem prático, irei usá lo em outras operações."

Usuário 2 - perfil de operador

> "o aplicativo é bem prático e intuitivo, so precisa alterar alguns icones, eu demorei para achar onde comprar o ingresso pois o ícone confunde, seria mais interessante um icone de dinheiro ou de um ingresso, mas e bem pratico, recebi todos os comandos no meu celular, so acho que seria melhor se o celular conseguisse emitir uma voz indicando as informações sobre o jogo, como por exemplo o inicio e o fim do jogo"

Usuário 3 - perfil de operador

> "O App é tem uma funcionalidade interessante. A interface é fácil navegar e relevantes para o público alvo. Algumas áreas que podem melhorar; Primeiro, o direcionamento após pagamento, levando para outra página, dessa maneira ficaria mais fácil a compressão que realmente deu certo a compra do card. Segundo, outra coisa que acho que seria interessante de ser acrescentada, notificações que mandem informações sobre os eventos e torneios de airsoft, assim os jogadores ficariam por dentro. Terceiro, a possibilidade de conectar com as redes sociais, dessa maneira os usuários compartilha suas experiências, fotos com outros jogadores, interagindo e trazendo engajamento para o APP. Por último, além do perfil do usuário poder ter fotos, descrição detalhada, e avaliações de outros jogadores (pontuação).
>
> O app sem dúvidas é muito valioso e útil para a comunidade de Airsoft e com algumas melhorias vai se tornar ainda mais eficaz."

Usuário 4 - perfil de operador

> "Gostaria de parabenizar pelo excelente trabalho no desenvolvimento do aplicativo. A interface é limpa, intuitiva e de fácil navegação, o que facilita bastante o uso diário. As funcionalidades atendem bem às necessidades. No entanto, acredito que alguns pontos ainda podem ser aprimorados, como:
>
> Desempenho: Em certos momentos o app apresenta lentidão ao carregar algumas telas.
>
> Estabilidade: Notei que o app fecha sozinho."

4 Conclusão

O objetivo desse trabalho era, como apresentado na Introdução (1), realizar o projeto do sistema por completo e desenvolver o MVP. Para o projeto, foram feitos os documentos de Visão (Seção 3.1), de Requisitos (Seção 3.2) e de Arquitetura (Seção 3.3). O primeiro trouxe uma visão ampla, de alto nível, sobre o sistema e serviu de base para os seguintes. O segundo desceu para o nível técnico e abordou os resultados do processo de Engenharia de Requisitos para o TROPA. Já o terceiro, a partir das informações coletadas pelos dois anteriores, fez o projeto da arquitetura do sistema e especificou detalhes por meio de diferentes visões arquitetônicas.

Para o desenvolvimento, as seções 3.4 e 3.6 discorreram os pormenores executados ao longo dos meses em que o sistema fora implementado. Apresentou-se o que fazia parte do MVP TROPA, bem como alguns detalhes sobre as tecnologias utilizadas e a implementação. Após isso, as atualizações e diferenças entre o projetado e o executado foram expostas, evidenciando que o processo de Engenharia de Software realmente passa por mudanças e é flexível. Também foi possível relatar os resultados dos testes com usuários e a coleta dos *feedbacks* fornecidos.

Além disso, na seção 3.5 foi apresentada a interface de usuário referente ao MVP por meio de capturas de telas do aplicativo. De modo geral, o trabalho cumpriu com o que havia se proposto a fazer, visto que o TROPA foi inteiramente projetado, teve seu MVP entregue e foi bem aceito pelo público que o testou.

À seguir, o autor apresenta, dentre os requisitos do sistema, quais foram atendidos em totalidade ou parcialidade; reflete sobre os pontos positivos e as melhorias necessárias para o aplicativo, com base nos relatos dos usuários que o testaram e; discute sobre os trabalhos futuros, por fim.

Requisitos atendidos

- Em totalidade: R4; R6; R7; R9; R10; R13; RNFQ1; RNFQ9; RNFQ10; RNFQ11.
- Em parcialidade: R1 (falta Lojista); R2 (falta Lojista); R22 (falta Pix); RNFQ2 (falta *web*); RNFQ3 (depende da falha); RNFQ5 e RNFQ6 (alguns usuários relataram um pouco de dificuldade); RNFSec1 (falta criptografar outros dados além de senhas); RNFSec2 (falta resistir a DDoS).

Casos de Uso atendidos

Usuário Operador

- UC1; UC2; UC3; UC4; UC10; UC17.

Usuário Organizador

- Em totalidade: UC1; UC11; UC14; UC15; UC16; UC17;
- Em parcialidade: UC7 (foto de perfil e de operação atendidas)

Usuário Visitante

- UC17.

Serviço externo para pagamentos

- Em totalidade: UC10;
- Em parcialidade: UC8 (falta UC6);

Serviço externo para envio de e-mails

- UC20.

4.1 Pontos positivos e melhorias

Com base nos *feedbacks* fornecidos durante a fase de validação do sistema (Seção 3.6), foi possível verificar os aspectos em que o aplicativo acertou bem, além daqueles em que há necessidade de melhoria. A Tabela 7 a seguir reúne os principais pontos positivos e de melhoria para o aplicativo.

Tabela 7 – Comparação de feedbacks dos usuários do aplicativo TROPA

Funcionalidade	**Pontos Positivos**	**Pontos de Melhoria**
Interface de Usuário	• Interface limpa e intuitiva (Usuário 4) • Fácil navegação (Usuários 3 e 4) • Interface prática e relevante para o público-alvo (Usuário 3)	• Contraste inadequado entre texto preto e fundo cinza (Usuário 1) • Ícones confusos, especialmente para compra de ingressos (Usuário 2) • Campo de seleção de tipo de conta fica oculto pelo botão de login (Usuário 1)
Gestão de Operações	• Facilidade para criar e gerir operações (Usuário 1) • Interface didática para organizadores (Usuário 1) • Gerenciamento eficiente de rangers/juízes (Usuário 1)	• Limitação de apenas dois times por operação (Usuário 1) • Impossibilidade de alterar informações como data, horário e local após criação (Usuário 1)
Controle de Jogo	• Comandos simples e práticos (Usuário 1) • Possibilidade de pausar e retomar partidas com facilidade (Usuário 1) • Recebimento eficaz de comandos via celular (Usuário 2)	• Ausência de notificações sonoras para início e fim de partidas (Usuário 2)

Funcionalidade	**Pontos Positivos**	**Pontos de Melhoria**
Sistema de Pagamento	• Compra de ingressos funcional (Usuários 2 e 3)	• Ícone de compra pouco intuitivo (Usuário 2) • Falta de redirecionamento após pagamento para confirmação clara (Usuário 3)
Desempenho e Estabilidade	• Aplicativo funciona bem no geral (Usuário 1)	• Lentidão ao carregar certas telas (Usuário 4) • Fechamentos inesperados do aplicativo (Usuário 4)
Recursos Sociais e Perfil	• N/A	• Falta de integração com redes sociais (Usuário 3) • Perfil poderia conter fotos, descrição e avaliações (Usuário 3) • Ausência de mecanismos para compartilhamento entre usuários (Usuário 3)
Notificações e Engajamento	• N/A	• Ausência de notificações sobre eventos e torneios (Usuário 3) • Falta de alertas automáticos e personalizáveis (Usuário 3)

Vale esclarecer que alguns dos aspectos de melhoria já haviam sido feitos enquanto

os usuários testavam o aplicativo, porém em versões em que não foram feitas *builds*. Isso porque a plataforma Expo, escolhida como a responsável pela compilação de código para Android e iOS, está sendo utilizada em plano gratuito, o qual limita o número de *builds* no mês. Outros pontos para aperfeiçoamento da aplicação já estão em andamento, visto que o desenvolvimento de *software* é um processo contínuo.

4.2 Trabalhos futuros

Como mencionado anteriormente, o desenvolvimento de software é um processo iterativo e contínuo. Embora o MVP do sistema TROPA tenha alcançado seu objetivo e sido bem recebido pelos usuários, diversos aspectos ainda podem ser aprimorados e ampliados em versões futuras. Abaixo são descritas as principais frentes que devem nortear os próximos ciclos de desenvolvimento.

Atender aos pontos de melhoria identificados

Os *feedbacks* coletados durante a fase de testes revelaram oportunidades claras de melhoria, especialmente no que diz respeito à usabilidade, estabilidade e clareza visual da interface. Aspectos como o contraste de textos, desempenho em telas específicas, ícones pouco intuitivos e ausência de redirecionamento pós-pagamento devem ser tratados com prioridade. Além disso, melhorias na personalização do perfil e notificações sonoras podem ampliar a imersão do operador no jogo.

Desenvolver funcionalidades pendentes

Durante a fase de implementação do MVP, algumas funcionalidades previstas originalmente foram adiadas. Entre elas destacam-se:

- o sistema de **ranking** entre jogadores, que permitirá avaliações baseadas em desempenho e engajamento;
- a **galeria de fotos e vídeos** das operações, integrando aspectos sociais e de compartilhamento;
- suporte completo ao papel de **Lojista**, incluindo a edição e o gerenciamento de produtos;
- a adição de métodos de pagamento como **Pix**, ampliando a acessibilidade e conveniência.

Evoluir para arquitetura de microsserviços

Atualmente, a aplicação foi estruturada com base em módulos internos comunicando-se por injeção de dependência, aproveitando a estrutura modular do NestJS. No entanto, o projeto arquitetônico original previa uma arquitetura de *microsserviços* distribuídos. Uma futura reestruturação nesse sentido permitirá:

- maior escalabilidade e tolerância a falhas;
- possibilidade de *deploys* independentes por serviço;
- distribuição de carga entre múltiplos nós.

Disponibilizar a versão web

Embora o sistema já tenha sido desenvolvido com React Native e Expo, visando compatibilidade com múltiplas plataformas, o *frontend* ainda não foi adaptado completamente para navegação via *web*. Esta versão permitirá acesso facilitado por navegadores e poderá ampliar o público-alvo, especialmente para usuários organizadores e lojistas que preferem operar a partir de computadores.

Desenvolver o sistema de gerenciamento administrativo (Admin Service)

Por fim, a criação de um serviço administrativo (Admin Service) para gestão global da plataforma se apresenta como uma etapa essencial para a escalabilidade e manutenção do sistema. Esse painel permitiria:

- visualizar e moderar operações ativas;
- controlar permissões de usuários;
- analisar estatísticas de uso, vendas e engajamento;
- aplicar medidas de segurança e banimento, caso necessário.

Esse sistema administrativo poderá ser implementado como uma aplicação independente, com sua própria interface e permissões restritas, comunicando-se com os demais serviços pela API TROPA ou por mensageria assíncrona.

Implementar o gateway e balanceador de carga do sistema

A arquitetura originalmente proposta para o sistema TROPA incluía um **API Gateway** ou **load balancer** dedicado como ponto único de entrada para todas as requisições externas. Este componente ainda não foi implementado no MVP, mas representa

uma etapa importante para garantir maior segurança, controle e escalabilidade em versões futuras do sistema.

O gateway poderá assumir responsabilidades como:

- **Roteamento inteligente** das requisições para os serviços internos do sistema, evitando exposição direta dos microsserviços à Internet;
- **Autenticação e verificação de origem**, por exemplo, distinguindo se uma requisição foi feita por um aplicativo móvel oficial, pela versão web ou por fontes não autorizadas;
- **Controle de acesso**, com filtragem de IPs, limitação de taxa (*rate limiting*) e proteção contra ataques de negação de serviço (*DDoS*);
- **Monitoramento centralizado**, possibilitando auditoria, *logging* e geração de métricas de uso por serviço ou origem.

Esse componente poderá ser implementado com soluções presentes no mercado ou serviços gerenciados na própria nuvem, como o *Azure Application Gateway*. Ele também poderá ser integrado a sistemas de autenticação existentes e prover suporte a rotas dinâmicas, versionamento de APIs e políticas de segurança personalizadas.

Ao assumir esse papel de orquestração e segurança, o *gateway* fortalecerá a robustez arquitetônica do TROPA e permitirá sua escalabilidade segura em ambientes de produção.

Referências

ACADEMY, A. *AWS Academy Cloud Security Foundations*. [S.l.]: AWS, 2023. Disponível em: <https://aws.amazon.com/training/awsacademy/>. Acesso em: 26 out. 2023. Citado na página 52.

AIRSOFT, Q. *Entendendo a Legislação do Airsoft no Brasil*. 2023. <https://qgairsoft.com.br/legislacao>. Acesso em: 30 jan. 2025. Citado na página 21.

AIRSOFTPEDIA. *Legislação Airsoft Brasil*. 2023. <https://airsoftpedia.com.br/legislacao-airsoft-brasil/>. Acesso em: 21 jan. 2025. Citado 2 vezes nas páginas 18 e 20.

AIRSOFTZONE. 2024. "Disponível em: <https://airsoftzone.com.br">. Citado na página 35.

AVENTURA, L. e. *Legislação sobre Airsoft no Brasil: Entenda como Funciona*. 2023. <https://blog.lazereaventura.com.br/legislacao-sobre-airsoft-no-brasil-entenda-como-funciona/>. Acesso em: 30 jan. 2025. Citado na página 19.

AWS. *AWS Pricing Calculator*. 2024. "Disponível em: <https://calculator.aws/>". Citado na página 37.

BURGUéS, X.; FRANCH, X. A language for stating component quality. In: *SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE (SBES), 14. , 2000, João Pessoa/PB. Anais [...]*. Porto Alegre: Sociedade Brasileira de Computação, 2000. P. 69-84. DOI: https://doi.org/10.5753/sbes.2000.25921. Citado na página 51.

CAMPOS, V. F. *Gerenciamento da rotina do trabalho do dia a dia*. 9. ed. [S.l.]: Falconi, 2004. Citado na página 31.

CANDIDO, W. *A Verdadeira Origem do Airsoft: Uma Jornada Histórica*. 2024. <https://combatgame.com.br/blog/2024/01/17/a-verdadeira-origem-do-airsoft-uma-jornada-historica/>. Acesso em: 21 jan. 2025. Citado na página 18.

CATEDRAL, L. 2024. "Disponível em: <https://lacatedralairsoft.com>". Citado 2 vezes nas páginas 34 e 38.

CHUNG, L. *Non-Functional Requirements*. 2006. The Universisty of Texas at Dallas. Department of Computer Science. Disponível em: <https://personal.utdallas.edu/~chung/RE/NFR-18.pdf>. Citado na página 53.

COMBAT, A. A. of S. *A Ascensão do Airsoft: História, Disseminação Global e Chegada ao Brasil*. 2023. <https://ascairsoft.com.br/a-ascensao-do-airsoft-historia-disseminacao-global-e-chegada-ao-brasil/>. Acesso em: 21 jan. 2025. Citado na página 18.

COMBAT, A. A. of S. *A Ascensão do Airsoft: História, Disseminação Global e Chegada ao Brasil*. 2023. <https://ascairsoft.com.br/

a-ascensao-do-airsoft-historia-disseminacao-global-e-chegada-ao-brasil/>. Acesso em: 30 jan. 2025. Citado na página 21.

DIGITAL, B. 2024. "Disponível em: <https://www.bilheteriadigital.com/">. Citado na página 35.

EVENTBRITE. 2024. "Disponível em: <https://www.eventbrite.com/">. Citado na página 35.

EVENTIZA. 2024. "Disponível em: <https://eventiza.com.br">. Citado na página 35.

FABE, F. de Airsoft de Brasília e E. 2024. Disponível em: <https://www.instagram.com/fabebsb/>. Citado 3 vezes nas páginas 33, 34 e 38.

GROUP, T. P. G. D. *PostgreSQL: Documentation*. 2023. <https://www.postgresql.org/docs/>. Acesso em: 20 jan. 2025. Citado na página 25.

INC., D. *Docker Documentation*. 2023. <https://docs.docker.com/>. Acesso em: 20 jan. 2025. Citado na página 30.

INC., S. *Stripe Company History*. 2023. <https://stripe.com/about>. Acesso em: 20 jan. 2025. Citado na página 28.

INC., S. *Stripe Documentation*. 2023. <https://stripe.com/docs>. Acesso em: 20 jan. 2025. Citado na página 28.

KREMER, S. et al. *Cybersecurity: Current Challenges and Inria's research directions*. 3. ed. França, 2019. Citado 2 vezes nas páginas 51 e 52.

MAILCHIMP, I. *Mailchimp Documentation*. 2023. <https://mailchimp.com/help/>. Acesso em: 20 jan. 2025. Citado na página 27.

MICROSOFT. *TypeScript: Documentation*. 2023. <https://www.typescriptlang.org/docs/>. Acesso em: 20 jan. 2025. Citado na página 22.

MICROSOFT. *Visual Studio Code Documentation*. 2023. <https://code.visualstudio.com/docs>. Acesso em: 20 jan. 2025. Citado na página 30.

MYSLIWIEC, K. *NestJS Framework: Documentation*. 2023. <https://docs.nestjs.com/>. Acesso em: 20 jan. 2025. Citado 4 vezes nas páginas 23, 24, 70 e 71.

NAIK, K.; TRIPATHY, P. Mccall's quality factors and criteria. In: *Software Testing and Quality Assurance: Theory and Practise*. New Jersey: John Wiley & Sons, Inc., 1959, (ISBN 978-0-471-78911-6). cap. 17, p. 523–527. Citado 2 vezes nas páginas 51 e 52.

OVERFLOW, S. *Developer Survey 2022*. 2022. <https://survey.stackoverflow.co/2022/>. Acesso em: 20 jan. 2025. Citado na página 22.

PLATFORMS, M. *React Native: Documentation*. 2023. <https://reactnative.dev/docs/getting-started>. Acesso em: 20 jan. 2025. Citado na página 24.

PLUS, A. *O que é Airsoft? Das Origens no Japão ao Crescimento no Brasil*. 2023. <https://www.aegplus.com/conteudo/21-artigo-o-que-e-airsoft>. Acesso em: 30 jan. 2025. Citado na página 18.

PRESSMAN, R. S. *Engenharia de software: uma abordagem profissional*. 7. ed. Porto Alegre: AMGH, 2011. Citado 2 vezes nas páginas 15 e 51.

PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software*. [S.l.]: Grupo A, 2021. ISBN 9786558040118. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9786558040118/>. Acesso em: 09 abr. 2024. Ebook. Citado 2 vezes nas páginas 17 e 44.

PROJECT, T. G. *Git Documentation*. 2023. <https://git-scm.com/doc>. Acesso em: 20 jan. 2025. Citado na página 29.

REINEHR, S. *Engenharia de Requisitos*. [S.l.]: Grupo A, 2020. ISBN 9786556900674. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9786556900674/>. Acesso em: 09 abr. 2024. Ebook. Citado 3 vezes nas páginas 22, 40 e 44.

ROSSOW, C.; JHA, S. The cyber security body of knowledge v1.0, 2019. In: _____. University of Bristol, 2019. cap. Network Security. KA Version 1.0. Disponível em: <https://www.cybok.org/>. Citado na página 52.

SENDGRID, T. *SendGrid Documentation*. 2023. <https://docs.sendgrid.com/>. Acesso em: 20 jan. 2025. Citado na página 27.

SILVA, F. R. da C. *Requisitos Não-Funcionais: NFR Framework*. 2004. Disponível em: <https://www.inf.unioeste.br/~olguin/4446-2009/RequisitosNaoFuncionais_aula.pdf>. Citado na página 53.

SOFTWARE, P. *RabbitMQ Documentation*. 2023. <https://www.rabbitmq.com/documentation.html>. Acesso em: 20 jan. 2025. Citado 2 vezes nas páginas 25 e 26.

SOFTWARE, S. *Swagger Documentation*. 2023. <https://swagger.io/docs/>. Acesso em: 20 jan. 2025. Citado na página 26.

SYMPLA. 2024. "Disponível em: <https://www.sympla.com.br">. Citado na página 35.

WARCAMP. 2024. "Disponível em: <https://warcamp.app/blog/">. Citado na página 35.

WHIMSICAL, I. *Whimsical Documentation*. 2023. <https://whimsical.com/>. Acesso em: 20 jan. 2025. Citado na página 30.