UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LEONARDO TOSIN

ZARPA: PLATAFORMA INTELIGENTE PARA INTERMEDIAÇÃO DE ENTREGAS URBANAS COM OTIMIZAÇÃO COLABORATIVA DE ROTAS E REDISTRIBUIÇÃO DINÂMICA DE FRETE

GUARAPUAVA

2026

LEONARDO TOSIN

ZARPA: PLATAFORMA INTELIGENTE PARA INTERMEDIAÇÃO DE ENTREGAS URBANAS COM OTIMIZAÇÃO COLABORATIVA DE ROTAS E REDISTRIBUIÇÃO DINÂMICA DE FRETE

ZARPA: INTELLIGENT PLATFORM FOR URBAN DELIVERY INTERMEDIATION WITH COLLABORATIVE ROUTE OPTIMIZATION AND DYNAMIC FREIGHT REDISTRIBUTION

Projeto de Trabalho de conclusão de Curso de graduação apresentado como requisito para obtenção do título de Tecnólogo em Sistemas Para Internet da Universidade Tecnológica Federal do Paraná (UTFPR).
Orientador(a): Prof. Dr. Andres Jessé Porfirio.

GUARAPUAVA

2026

RESUMO

TOSIN, Leonardo. **Plataforma inteligente para intermediação de entregas urbanas com otimização colaborativa de rotas e redistribuição dinâmica de frete**. 2026. Projeto de Trabalho de Conclusão de Curso (Tecnólogo em Sistemas Para Internet) – Universidade Tecnológica Federal do Paraná, Guarapuava, 2026.

A expansão do comércio eletrônico no cenário brasileiro impulsionou a busca por eficiência na etapa logística do *last mile* (última milha), demandando o desenvolvimento de artefatos tecnológicos capazes de equilibrar a viabilidade econômica e o desempenho operacional nos centros urbanos. Em municípios de médio porte, a exemplo de Guarapuava, a precificação de fretes executados por entregadores autônomos carece de padronização técnica, sendo regida por critérios subjetivos que resultam em trajetos subotimizados, sobreposição de rotas viárias e disparidade de valores. Paralelamente, constata-se a escassez de mecanismos estruturados para a consolidação dinâmica de encomendas de diferentes solicitantes em um único deslocamento viário com a devida repartição de custos. Visando mitigar essas lacunas, este projeto propõe o desenvolvimento da plataforma inteligente Zarpa, voltada à intermediação de entregas urbanas apoiada por algoritmos de agrupamento geográfico e rateio dinâmico. A metodologia adota princípios da engenharia de software e práticas ágeis, utilizando a técnica MoSCoW e a matriz RICE para a priorização de requisitos do Produto Mínimo Viável (MVP). O ecossistema computacional é estruturado sob uma arquitetura cliente-servidor, empregando o framework Laravel no *back-end*, React Native no *front-end*, banco de dados PostgreSQL com a extensão geoespacial PostGIS, e integração com a API OpenRouteService para o cálculo georreferenciado de distâncias e roteirização das entregas. O diferencial da solução reside no processamento matemático financeiro, que calcula e distribui proporcionalmente a economia gerada em trajetos compartilhados. Espera-se, como resultado, a validação de um modelo logístico sustentável que reduza o custo individual do frete para os clientes e incremente a rentabilidade horária dos entregadores parceiros.

Palavras-chave: Logística urbana; Otimização de rotas; Rateio dinâmico; Plataforma de entregas.

ABSTRACT

TOSIN, Leonardo. **Intelligent platform for urban delivery intermediation with collaborative route optimization and dynamic freight redistribution**. 2026. Projeto de Trabalho de Conclusão de Curso (Tecnólogo em Sistemas para Internet) – Universidade Tecnológica Federal do Paraná, Guarapuava, 2026. Título original: Plataforma inteligente para intermediação de entregas urbanas com otimização colaborativa de rotas e redistribuição dinâmica de frete.

The expansion of e-commerce in the Brazilian scenario has driven the search for efficiency in the last-mile logistics stage, demanding the development of technological artifacts capable of balancing economic viability and operational performance in urban centers. In medium-sized municipalities, such as Guarapuava, the pricing of freight executed by autonomous delivery drivers lacks technical standardization, being governed by subjective criteria that result in sub-optimized routes, overlapping road routes, and price disparity. In parallel, there is a scarcity of structured mechanisms for the dynamic consolidation of orders from different requesters into a single road trip with the proper cost sharing. Aiming to mitigate these gaps, this project proposes the development of the intelligent platform Zarpa, focused on the intermediation of urban deliveries supported by geographic clustering and dynamic apportionment algorithms. The methodology adopts software engineering principles and agile practices, using the MoSCoW technique and the RICE matrix for prioritizing the requirements of the Minimum Viable Product (MVP). The computational ecosystem is structured under a client-server architecture, employing the Laravel framework in the back-end, React Native in the front-end, a PostgreSQL database with the PostGIS geospatial extension, and integration with the OpenRouteService API for georeferenced distance calculation and delivery routing. The solution's differential lies in the financial mathematical processing, which calculates and proportionally redistributes the savings generated in shared routes. As a result, it is expected to validate a sustainable logistics model that reduces the individual freight cost for customers and increases the hourly profitability of partner delivery drivers.

Keywords: Urban logistics; Route optimization; Dynamic cost splitting; Delivery platform.

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface* (Interface de Programação de Aplicação)
GPS	*Global Positioning System* (Sistema de Posicionamento Global)
MVC	*Model-View-Controller* (Modelo-Visão-Controlador)
MVP	*Minimum Viable Product* (Produto mínimo viável)
PHP	*Hypertext Preprocessor* (pré processador de hipertexto)
REST	*Representational State Transfer* (Transferência de Estado Representacional)

SUMÁRIO

1 INTRODUÇÃO..... 6
1.1 Objetivos..... 7
1.1.1 Objetivo Geral..... 7
1.1.2 Objetivos Específicos..... 7
2 TRABALHOS RELACIONADOS..... 8
2.1 Análise do Mercado..... 8
2.2 O Diferencial Estrutural e Mercadológico da Plataforma Zarpa..... 9
3 SOLUÇÃO PROPOSTA..... 10
3.1 Visão Geral da Plataforma..... 10
3.2 O Modelo Financeiro de Rateio Dinâmico..... 11
3.3 Descrição dos Usuários..... 13
3.3.1 Cliente..... 13
3.3.2 Entregador..... 13
3.3.3 Administrador..... 14
4 MATERIAIS E MÉTODOS..... 15
4.1 Materiais..... 15
4.2 Métodos..... 16
4.2.1 O Motor de Roteamento (OpenRouteService)..... 17
4.2.2 Fluxo de Entrega Expressa (On-Demand)..... 17
4.2.3 Fluxo de Lote Econômico (Batch Noturno)..... 18
4.2.4 Estratégia de Rateio e Desconto Cooperativo..... 19
4.2.5 Controle de Concorrência e Cliques Simultâneos..... 20
5 RESULTADOS PARCIAIS..... 21
5.1 Diagnóstico de Campo e Análise Empírica da Demanda..... 21
5.2 Levantamento e Priorização de Requisitos..... 24
5.3 Modelagem do Banco de Dados..... 25
5.3.1 Módulo de Atores do Sistema..... 25
5.3.2. Fluxo de Pedidos e Carga..... 26
5.3.3. Inteligência de Agrupamento e Rateio Dinâmico..... 26
5.5 Prototipação de telas..... 28
5.5.1 Módulo do Cliente: Solicitação e Seleção de Modalidade..... 28
5.5.2 Módulo do Entregador: Radar Expresso e Agenda de Rotas..... 29
6 CONSIDERAÇÕES FINAIS..... 31
REFERÊNCIAS..... 32
APÊNDICES..... 33
APÊNDICE A – LEVANTAMENTO COMPLETO DE REQUISITOS (MOSCOW)..... 34
APÊNDICE B – MATRIZ RICE DOS REQUISITOS LEVANTADOS..... 36
APÊNDICE C – MODELAGEM BANCO DE DADOS COMPLETA..... 38

1 INTRODUÇÃO

O avanço das tecnologias de informação e comunicação e a popularização dos dispositivos móveis impulsionaram expressivamente o mercado de entregas sob demanda no Brasil. Esse cenário consolidou um padrão de consumo baseado na conveniência, exigindo agilidade, transparência e flexibilidade no transporte urbano de mercadorias. No entanto, a etapa de entrega final, conhecida na literatura como *last mile* (última milha), permanece como um dos principais gargalos logísticos e econômicos do setor. Trata-se frequentemente da fase mais custosa e ineficiente da cadeia de suprimentos, chegando a representar mais da metade dos custos totais de envio (WORLD ECONOMIC FORUM, 2020). Esse gargalo impacta diretamente as margens de lucro de pequenos negócios locais em função dos altos custos operacionais e da ociosidade das frotas.

Nesse contexto, micro comerciantes e entregadores autônomos encontram-se em uma posição de vulnerabilidade logística. Diferentemente de grandes corporações que utilizam sistemas robustos de gerenciamento de frotas e roteirização automatizada, os pequenos lojistas gerenciam suas demandas de forma fragmentada. Como consequência, as entregas são contratadas individualmente, gerando rotas sobrepostas, desperdício de combustível, aumento na emissão de poluentes e encarecimento do frete para o consumidor final. Além disso, a precificação deste serviço em municípios de médio porte, como Guarapuava (PR), costuma ser empírica e inconsistente, baseando-se em diárias fixas ou taxas arbitrárias que prejudicam a previsibilidade financeira de lojistas e entregadores.

Para mitigar esses problemas, surge a oportunidade de desenvolver uma solução computacional que atue como intermediadora inteligente. A proposta consiste em integrar técnicas de roteirização dinâmica à logística colaborativa, permitindo o agrupamento automatizado de pedidos geograficamente compatíveis e a redistribuição proporcional da economia financeira gerada entre as partes envolvidas.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver uma plataforma inteligente que utilize uma aplicação móvel integrada a uma API backend para mediar entregas urbanas, focando na otimização colaborativa de trajetos e na redistribuição dinâmica de fretes.

1.1.2 Objetivos Específicos

- **Investigar** as dinâmicas operacionais e os critérios de precificação de fretes urbanos praticados por micro comerciantes e entregadores autônomos no cenário local;
- **Levantar** e delimitar os requisitos funcionais e não funcionais da plataforma, definindo o escopo do Produto Mínimo Viável (MVP);
- **Modelar** a estrutura de persistência de dados com suporte a informações geoespaciais para o armazenamento de coordenadas, rotas e perfis de usuários;
- **Desenvolver** a camada de serviços de servidor (*back-end*) para o processamento das regras de negócio e consumo de serviços de geolocalização e roteirização;
- **Implementar** o módulo computacional responsável pelo cálculo matemático do rateio financeiro dinâmico e pela redistribuição proporcional dos custos logísticos;
- **Desenvolver** a aplicação móvel (*front-end*) com interfaces dedicadas tanto para os estabelecimentos comerciais (solicitantes) quanto para os entregadores autônomos (prestadores), priorizando a usabilidade e o acompanhamento das rotas.

2 TRABALHOS RELACIONADOS

Para posicionar a plataforma Zarpa em relação ao estado da arte da logística de última milha (*last mile*), esta seção analisa de forma crítica três soluções consolidadas no mercado de intermediação de entregas: Uber Flash, Lalamove e Borzo (antiga Dostavista). O intuito é contrastar os modelos operacionais vigentes com a abordagem colaborativa e o mecanismo de rateio proposto neste trabalho.

2.1 Análise do Mercado

A modalidade Uber Flash (UBER, 2026) atende demandas estritamente imediatas (*on-demand*) em um regime ponto a ponto exclusivo. Seu modelo conecta um único cliente a um entregador para a execução de um trajeto direto e individualizado. Embora ofereça alta conveniência e ampla cobertura de motoristas, a solução não otimiza o uso do espaço urbano coletivo, pois calcula a tarifa de forma isolada por corrida. Não há mecanismos algorítmicos para unificar pedidos de clientes diferentes que compartilhem rotas semelhantes, o que mantém elevado o custo individual do frete.

Por sua vez, a Lalamove (LALAMOVE, 2026) implementa o conceito de rotas com múltiplas paradas, permitindo que um mesmo itinerário inclua diversos pontos de coleta e entrega. No entanto, essa consolidação logística ocorre de forma manual e centralizada. A otimização depende de que um único lojista possua um volume alto de encomendas e cadastre todas as paradas em uma mesma ordem de serviço. A plataforma carece de inteligência automatizada para interceptar e agrupar pedidos fracionados de pequenos comércios distintos localizados em uma mesma região geoespacial.

A Borzo (BORZO, 2026) se aproxima da proposta deste projeto ao utilizar o processamento em lote (*batch processing*) assíncrono para agrupar pedidos automaticamente, reduzindo o deslocamento sem carga (*deadhead*) dos entregadores. Contudo, a divergência reside na distribuição do valor econômico gerado. Os algoritmos da Borzo visam maximizar o lucro da própria plataforma e a produtividade do condutor. A eficiência da rota combinada não é revertida em transparência tarifária ou descontos para os clientes: dois lojistas com pacotes transportados simultaneamente pelo mesmo motoboy continuam pagando o valor integral de corridas isoladas.

2.2 O Diferencial Estrutural e Mercadológico da Plataforma Zarpa

O diferencial da plataforma Zarpa reside no preenchimento de uma lacuna comercial e tecnológica ignorada pelas grandes corporações: a democratização da eficiência logística por meio do paradigma colaborativo focado em micro comércios locais e entregadores autônomos. Enquanto os concorrentes utilizam a tecnologia de maneira centralizadora, seja mantendo a exclusividade de rotas com custos elevados (Uber Flash), exigindo alta volumetria de um único cliente (Lalamove) ou retendo os ganhos da otimização (Borzo), a Zarpa propõe uma arquitetura com comportamento dual e pós-processamento financeiro transparente. O diferencial competitivo da solução estrutura-se em dois pilares principais:

- **Modelo Operacional Dual Automatizado:** O sistema atende nativamente tanto a demandas urgentes, utilizando filtros de proximidade espacial em tempo real (modalidade expressa), quanto ao planejamento preventivo de custos reduzidos, por meio do agrupamento algorítmico de pedidos em lote para distribuição programada (modalidade econômica);

- **Modelo de Rateio Dinâmico (50/50):** O sistema não se limita a sequenciar paradas viárias. Ao unificar as trajetórias de clientes independentes por meio de rotinas de agrupamento espacial (*clustering*), a plataforma calcula a quilometragem total economizada em relação aos trajetos individuais. Essa economia é dividida igualmente: 50% retorna aos lojistas como desconto retroativo no frete e 50% é repassada ao entregador como bônus de produtividade.

3 SOLUÇÃO PROPOSTA

Este capítulo delimita a estrutura conceitual e o modelo operacional da plataforma Zarpa. A solução foi desenhada para conectar microcomerciantes locais e entregadores autônomos no município de Guarapuava - PR, transformando a logística tradicional de última milha por meio de uma abordagem colaborativa e sustentável.

3.1 Visão Geral da Plataforma

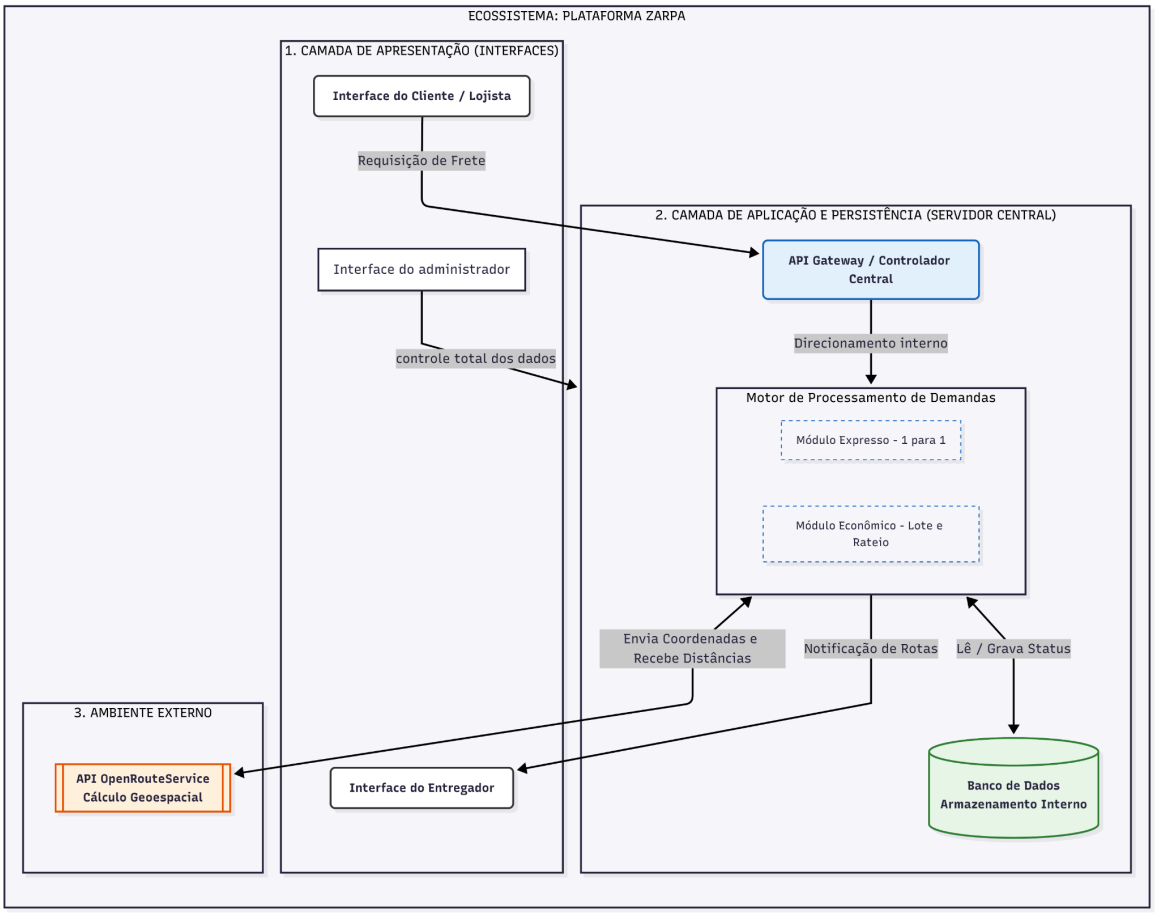
A plataforma Zarpa é estruturada sob uma arquitetura cliente-servidor centralizada, projetada para a intermediação e a roteirização de serviços de logística urbana de última milha (*last mile*). Diferenciando-se de sistemas convencionais que operam meramente como quadros de anúncios ou canais de comunicação direta e linear, a solução proposta integra um módulo de processamento focado na melhoria da eficiência operacional de cada deslocamento viário.

O funcionamento do sistema baseia-se na análise geoespacial contínua de oportunidades de consolidação de fretes na malha urbana. Ao receber uma nova requisição de transporte, o *backend* não a processa de maneira isolada; o sistema executa uma varredura para cruzar as coordenadas de origem, destino e os limites temporais das demandas correntes. Quando o algoritmo identifica compatibilidade geográfica entre duas ou mais requisições, a plataforma realiza o agrupamento automático desses pedidos em uma rota unificada. Esta abordagem viabiliza que um único prestador de serviço atenda a múltiplos solicitantes por meio de um trajeto compartilhado, reduzindo o tempo ocioso, a sobreposição de trajetos viários e a emissão de poluentes no perímetro urbano. Diante do exposto, o sistema opera sob duas modalidades distintas de atendimento para os usuários:

- **Modalidade Expressa:** Voltada ao atendimento imediato, onde a requisição é direcionada com exclusividade e em tempo real ao entregador mais próximo, priorizando o menor tempo de resposta.
- **Modalidade Econômica:** Configura o núcleo colaborativo da plataforma. Nesta opção, o pedido do cliente é retido temporariamente em uma janela de tempo operacional, permitindo que o sistema intercepte e agrupe demandas de outros usuários com trajetos convergentes antes da consolidação e do disparo da rota final.

Toda a inteligência do sistema opera em segundo plano. Os estabelecimentos comerciais interagem para cadastrar suas ordens de serviço e escolher o nível de urgência, enquanto os entregadores utilizam o sistema de forma receptiva para visualizar e aceitar as rotas oferecidas pela plataforma. Essa dinâmica é concentrada em uma aplicação móvel unificada (*front-end*), desenvolvida com controle estrito de níveis de acesso e renderização condicional de interfaces para cada perfil de usuário. Paralelamente, os administradores supervisionam o ecossistema por meio de uma interface de controle *web* para garantir a moderação e o equilíbrio da comunidade. A arquitetura de alto nível da solução e as interações entre esses componentes são apresentadas na **Figura 1**.

Figura 1 – Ecossistema de funcionamento e arquitetura de alto nível da Plataforma Zarpa.



Fonte: Autoria própria (2026).

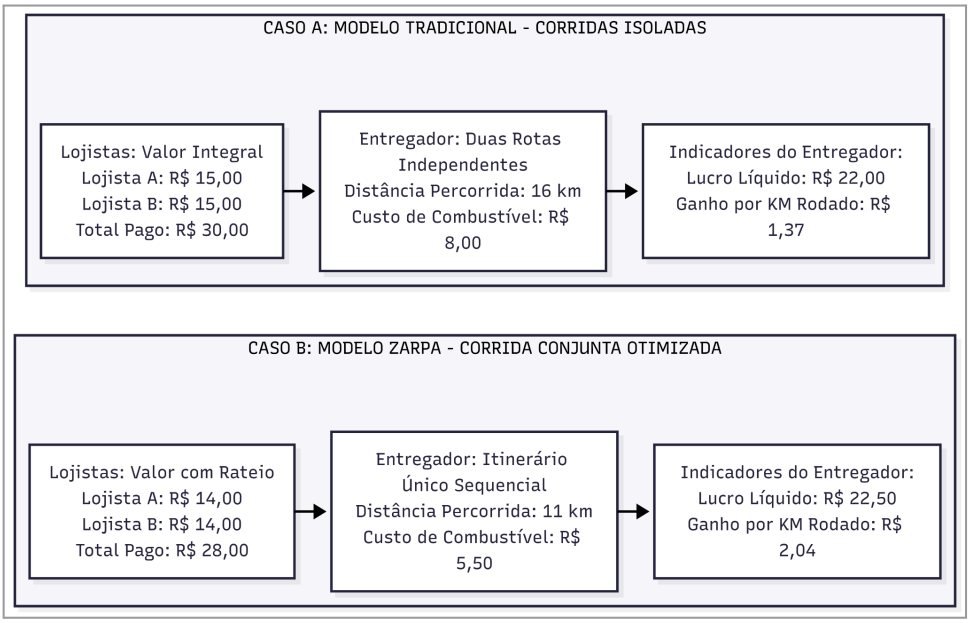
3.2 O Modelo Financeiro de Rateio Dinâmico

A viabilidade comercial do ecossistema Zarpa baseia-se em um modelo de distribuição de valor guiado pela premissa ganha-ganha. Nas entregas agrupadas (modalidade econômica), a plataforma calcula a distância que seria percorrida caso os pedidos fossem feitos separadamente e a compara com o trajeto unificado real.

A diferença em quilometragem que representa o desperdício viário evitado gera uma economia financeira imediata. Esse montante economizado é dividido equitativamente pelo sistema: metade do valor retorna para os lojistas participantes na forma de descontos retroativos em suas tarifas, e a outra metade é repassada diretamente ao entregador como um bônus de produtividade por ter executado a jornada conjunta. Dessa forma, a plataforma estimula a cooperação espontânea de mercado.

Para fins de ilustração desse comportamento econômico e de sua respectiva lógica de precificação, a **Figura 2** apresenta uma simulação de cenário hipotético comparando dois casos de atendimento para uma mesma região geográfica: o modelo tradicional de corridas isoladas (Caso B) e o modelo proposto de corrida conjunta (Caso A).

Figura 2 – Simulação comparativa de eficiência e rateio financeiro entre os cenários logística tradicional e otimizada.



Fonte: Autoria própria (2026).

No **Caso A** (Corridas Isoladas), os estabelecimentos demandam fretes de forma independente. O Lojista A e o Lojista B despacham um produto cada, arcando com o valor integral de R$ 15,00 por corrida, totalizando um custo conjunto de R$ 30,00. Para cumprir essas ordens, o entregador realiza trajetos desconexos que somam 16 km. Descontando o custo estimado de R$ 8,00 com combustível, o profissional obtém um lucro líquido de R$ 22,00, o que equivale a um rendimento de R$ 1,37 por quilômetro rodado.

No **Caso B** (Corrida Conjunta), a plataforma Zarpa consolida as demandas em uma janela de tempo, agrupando os produtos em um itinerário único e sequencial de 11 km. A reorganização do percurso reduz o custo total do frete para R$ 28,00, permitindo que ambos os lojistas recebam um desconto por rateio, passando a pagar R$ 14,00 cada.

Sob a perspectiva do entregador, a redução do trajeto diminui o custo estimado com combustível para R$ 5,50. Como consequência, seu lucro líquido absoluto na rota é preservado em R$ 22,50, enquanto sua eficiência financeira salta para R$ 2,04 por quilômetro rodado. A simulação demonstra conceitualmente a viabilidade da proposta: os estabelecimentos reduzem seus custos de frete através do compartilhamento, enquanto o entregador minimiza o desgaste do veículo e eleva a rentabilidade do seu tempo trabalhado.

3.3 Descrição dos Usuários

O ecossistema da plataforma Zarpa é coordenado pela interação de três atores principais, cujos papéis e responsabilidades são rigidamente delimitados para garantir o equilíbrio entre a demanda comercial, a execução logística e a governança do serviço.

3.3.1 Cliente

O primeiro ator é o Cliente, composto majoritariamente por micro comerciantes locais que precisam enviar mercadorias fracionadas dentro do perímetro urbano. Sua interação com o sistema consiste em cadastrar a ordem de serviço informando os endereços de coleta e destino, além do volume do pacote. A principal decisão desse usuário é a escolha da modalidade de frete: a expressa, para envios urgentes com custo individualizado, ou a econômica, na qual aceita

aguardar uma janela de tempo para compartilhar a rota com outros lojistas em troca de descontos tarifários.

3.3.2 Entregador

O segundo agente é o Entregador, profissional autônomo responsável pelo transporte físico das encomendas. Para esse usuário, a plataforma funciona como uma ferramenta para reduzir o tempo ocioso e aumentar o faturamento por trajeto rodado. Sua participação é receptiva, limitando-se a manter o aplicativo ativo para receber propostas de rotas já otimizadas, decidir pelo aceite ou recusa das jornadas e atualizar manualmente o status da entrega no dispositivo móvel, sinalizando os momentos exatos da coleta e da entrega final ao consumidor.

3.3.3 Administrador

Por fim, o Administrador atua na governança e auditoria do ecossistema por meio de uma interface de controle, sem participar dos fluxos de transporte diários. Suas funções concentram-se em assegurar a confiabilidade e a segurança da comunidade, o que inclui a análise e aprovação de cadastros de novos entregadores, o monitoramento macro dos repasses e rateios financeiros e a intervenção humana para resolver eventuais disputas ou problemas operacionais reportados pelos clientes e motoboys.

4 MATERIAIS E MÉTODOS

Este capítulo apresenta os recursos tecnológicos e os procedimentos estratégicos utilizados para a concepção e o desenvolvimento da plataforma Zarpa. A abordagem une ferramentas de desenvolvimento de software e práticas ágeis para estruturar uma solução baseada na arquitetura cliente-servidor.

4.1 Materiais

A seleção dos materiais obedeceu a critérios de escalabilidade, desempenho , documentação e compatibilidade com o ecossistema de desenvolvimento web e móvel. Os recursos utilizados compreendem:

- **Figma:** Plataforma em nuvem utilizada para o design visual das interfaces (UI) e definição dos fluxos de experiência do usuário (UX), permitindo validar as telas de lojistas e entregadores antes da fase de programação.
- **Git e GitHub:** Ferramentas utilizadas para o versionamento e controle do código-fonte, operando com ramificações (*branches*) e registros de modificações (*commits*) para garantir a organização e o histórico do projeto.
- **dbdiagram.io e DBML:** Ferramentas de modelagem utilizadas na criação dos diagramas de entidade-relacionamento (DER). O uso da linguagem descritiva DBML traz clareza e agilidade na definição e manutenção das tabelas.
- **React Native e React Native Maps:** *Framework* multiplataforma adotado para o desenvolvimento do aplicativo móvel unificado, permitindo o acesso nativo ao GPS dos dispositivos e a exibição dinâmica de mapas e rotas tanto em sistemas Android quanto iOS.
- **Laravel:** *Framework* PHP utilizado para o desenvolvimento do *back-end*, responsável pelas regras de negócio, processamento dos agrupamentos logísticos e fornecimento da API RESTful de forma transparente para as interfaces.
- **PostgreSQL e PostGIS:** Banco de dados relacional integrado à extensão geoespacial PostGIS, o que permite armazenar as coordenadas geográficas

do ecossistema e executar consultas espaciais complexas, como filtros de distância para alocação de pedidos.

- **OpenRouteService (ORS):** API geoespacial baseada em dados abertos utilizada para o roteamento de veículos, responsável por receber coordenadas e retornar as distâncias exatas, tempos estimados de viagem e o traçado real das vias urbanas.

4.2 Métodos

O desenvolvimento da plataforma Zarpa divide-se em sete etapas interdependentes, fundamentadas nos princípios da engenharia de software e em práticas ágeis. Essa organização visa transformar os problemas da logística urbana em uma solução computacional eficiente:

- **Etapa 1 - Investigação de Campo e Delimitação de Escopo:** Diagnóstico inicial sobre o funcionamento dos fretes locais. A análise dos problemas enfrentados por lojistas e entregadores define as fronteiras do projeto, transformando as demandas coletadas em requisitos técnicos.

- **Etapa 2 - Priorização de Funcionalidades (Técnica MoSCoW):** Os critérios de priorização de requisitos objetivam atender o escopo do MVP, garantindo a implementação completa do algoritmo de agrupamento de rotas e do modelo de rateio financeiro. Fatores secundários são então classificados com menor prioridade, podendo ser contemplados em fases futuras. A justificativa de corte e inclusão de cada requisito é validada analiticamente pela aplicação preliminar da matriz RICE (Alcance, Impacto, Confiança e Esforço).

- **Etapa 3 - Mapeamento de Requisitos e Regras de Negócio:** Detalhamento das interações e dos fluxos operacionais para cada perfil de usuário estruturado no sistema (Lojista, Entregador e Administrador). Essa abordagem define os critérios de aceitação que guiam o desenvolvimento.

- **Etapa 4 - Prototipação e Design de Interfaces:** Criação das telas no Figma para validar a usabilidade e o apelo visual do sistema. O objetivo é simular os fluxos do software — como a criação de pedidos e a tela de ofertas — antes do início da codificação.

- **Etapa 5 - Arquitetura Lógica e Persistência de Dados:** Envolve a modelagem do banco de dados relacional com suporte geoespacial, utilizando diagramas baseados na linguagem descritiva do DBML. Esta fase estrutura as tabelas principais, como Usuários, Pedidos e Rotas.

- **Etapa 6 - Gestão Ágil (Ciclos de Sprint):** As tarefas mapeadas são segmentadas por nível de complexidade e organizadas em ciclos curtos de desenvolvimento (*Sprints*). O monitoramento do progresso técnico de cada artefato é centralizado na ferramenta GitHub Projects.

- **Etapa 7 - Codificação e Versionamento (Git):** A implementação utiliza o fluxo de ramificações isoladas do Git (*Feature Branch Workflow*). Cada funcionalidade é desenvolvida separadamente e passa por uma revisão de código (*Pull Request*) antes de ser integrada à versão estável de produção (*main*).

4.2.1 O Motor de Roteamento (OpenRouteService)

O sequenciamento viário e o traçado cartográfico da plataforma Zarpa são delegados à API externa do **OpenRouteService** (ORS), baseada na infraestrutura do **OpenStreetMap** (LUXEN; VETTER, 2011). Essa escolha arquitetural evita a sobrecarga computacional desnecessária do servidor *backend*. Para calcular o caminho mais curto entre os pontos de coleta e entrega, o ORS utiliza internamente o Algoritmo de **Dijkstra** (DIJKSTRA, 1959), um padrão consolidado na literatura de ciência da computação para encontrar trajetórias de menor custo em grafos.

Para que a API consiga devolver respostas em milissegundos mesmo operando sobre a complexa malha viária continental, o ORS aprimora o cálculo clássico utilizando a técnica de Hierarquias de Contração (GEISSBERGER *et al.*, 2008). Esse método pré-processa os dados geográficos criando atalhos matemáticos entre nós de tráfego intenso. Como a plataforma utiliza uma solução já estabelecida e amplamente otimizada para o roteamento, o esforço de desenvolvimento do projeto concentra-se exclusivamente na lógica de negócio e na distribuição inteligente dos lotes.

4.2.2 Fluxo de Entrega Expressa (On-Demand)

A entrega expressa foi desenhada para lidar com pedidos urgentes em tempo real. O fluxo acompanha o ciclo de vida do pacote desde o registro pelo cliente até o aceite do condutor. A lógica de negócio deste serviço pode ser representada pelo seguinte pseudocódigo:

```
INICIO Fluxo_Entrega_Expressa
  RECEBER novo_pedido_urgente DO lojista
  lista_entregadores <- BUSCAR entregadores com status "ONLINE"

  PARA CADA entregador NA lista_entregadores FACA
    distancia <- CALCULAR_DISTANCIA_RETA(lojista.coordenadas,
entregador.coordenadas)
    SE distancia <= RAIO_MAXIMO_PERMITIDO ENTAO
      ENVIAR notificacao_push(entregador)
    FIM SE
  FIM PARA
  ENQUANTO pedido.status == "Pendente" FACA
    SE entregador_clicou_aceitar ENTAO
      TENTAR_BLOQUEIO_PESSIMISTA_ORM(pedido)
      SE bloqueio_bem_sucedido ENTAO
        pedido.status <- "Aceito"
        rota_tracada <- ORS.calcular_rota(lojista, cliente_final)
        valor_frete <- CALCULAR_TARIFA_BASE(rota_tracada.distancia)
        VINCULAR rota_tracada E valor_frete AO entregador
        RETORNAR "Corrida confirmada com sucesso"
      SENAO
        RETORNAR "Erro: Corrida já aceita por outro condutor"
      FIM SE
    FIM SE
  FIM ENQUANTO
FIM Fluxo_Entrega_Expressa
```

4.2.3 Fluxo de Lote Econômico (Batch Noturno)

O Lote Econômico resolve o problema de fragmentação das entregas da modalidade "Sem Pressa". Este fluxo opera de forma centralizada e assíncrona, sendo disparado automaticamente em horários pré-programados no servidor para consolidar e preparar as rotas. O algoritmo foca em agrupar pedidos próximos geograficamente de forma equitativa entre os condutores escalados:

```
INICIO Fluxo_Lote_Economico
  pedidos_sem_pressa <- BUSCAR todos pedidos pendentes do dia
  lista_entregadores <- BUSCAR entregadores escalados para o
proximo turno
  limite_pacotes <- TOTAL(pedidos_sem_pressa) /
TOTAL(lista_entregadores)

  PARA CADA entregador NA lista_entregadores FACA
    rota_atual <-
INICIAR_PLANEJAMENTO_ROTA(entregador.endereco_partida)
    ENQUANTO TOTAL_ITENS(rota_atual) < limite_pacotes E
pedidos_sem_pressa NAO VAZIO FACA
      pedido_proximo <-
BUSCAR_PEDIDO_MAIS_PROXIMO(rota_atual.ultimo_ponto,
pedidos_sem_pressa)
      ADICIONAR pedido_proximo A rota_atual
      REMOVER pedido_proximo DA lista pedidos_sem_pressa
    FIM ENQUANTO
    rota_otimizada <- ORS.otimizar_multiplos_pontos(rota_atual)
    rateio_financeiro <-
CALCULAR_RATEIO_COOPERATIVO(rota_otimizada, rota_atual)
    VINCULAR rota_otimizada AO entregador
    SALVAR_NO_BANCO(rota_otimizada, rateio_financeiro)
  FIM PARA
FIM Fluxo_Lote_Economico
```

4.2.4 Estratégia de Rateio e Desconto Cooperativo

Concluída a etapa de agrupamento do Lote Econômico, o sistema calcula a economia total gerada na malha urbana. Esse cálculo é realizado subtraindo o custo da trajetória unificada real e otimizada do somatório das tarifas individuais e isoladas que os lojistas pagariam caso as demandas fossem despachadas separadamente.

Desse montante economizado, deduz-se a taxa de administração da plataforma (a ser definida futuramente, quando o MVP for implantado e entrar em fase de validação comercial) e divide-se o saldo restante de forma equitativa (50/50). A metade destinada ao condutor é incorporada de maneira direta como um bônus financeiro pela execução da jornada conjunta. A outra metade retorna para os estabelecimentos na forma de descontos retroativos em suas respectivas tabelas de frete, distribuídos de maneira proporcional à distância original de cada requisição. Esta arquitetura de rateio compensa a redução do preço unitário através da indução do volume de mercado, gerando benefícios financeiros mútuos para lojistas e entregadores.

4.2.5 Controle de Concorrência e Cliques Simultâneos

Como demonstrado no fluxo analítico da entrega expressa, a estabilidade transacional do ecossistema depende criticamente de impedir falhas de dupla alocação quando múltiplos entregadores tentam aceitar o mesmo pedido em frações de segundo idênticas. Para mitigar esse problema de concorrência (*race conditions*), a plataforma implementará travas pessimistas (*Pessimistic Locking*) gerenciadas de forma nativa pelas camadas de abstração do *framework* Laravel através do Eloquent ORM (utilizando o método estruturado `lockForUpdate()`).

Essa rotina intercepta a requisição e bloqueia temporariamente o registro do pedido no banco de dados durante o processamento da validação. Se a consulta isolada confirmar que o *status* do pedido permanece como "Pendente", a transação é finalizada, atualizando o pedido para "Aceito" e vinculando a rota ao primeiro entregador. Caso outro condutor tenha consolidado o clique momentos antes, o mecanismo invalida a segunda tentativa de alteração e retorna um erro controlado de concorrência.

5 RESULTADOS PARCIAIS

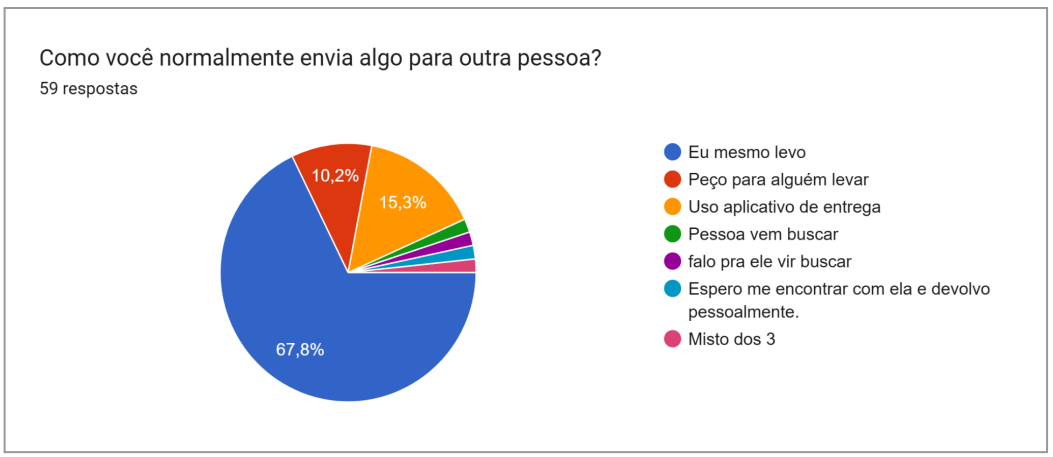
Este capítulo apresenta os resultados técnicos, empíricos e de modelagem obtidos durante as fases de concepção, análise e design da plataforma Zarpa. Os dados e artefatos aqui expostos consolidam o diagnóstico de mercado e a engenharia de requisitos que fundamentam e direcionam o desenvolvimento do Produto Mínimo Viável (*Minimum Viable Product* — MVP).

5.1 Diagnóstico de Campo e Análise Empírica da Demanda

Para compreender os gargalos logísticos da etapa de última milha (*last-mile*) em Guarapuava (PR) e subsidiar a especificação dos requisitos do sistema, realizou-se uma pesquisa de campo estruturada em duas abordagens: uma análise qualitativa com comerciantes locais e um levantamento quantitativo aplicado a uma amostragem de 59 potenciais usuários. A etapa qualitativa revelou que a maioria dos lojistas gerencia suas entregas de forma manual, recorrendo a registros em papel ou a aplicativos de mensagens instantâneas (como o WhatsApp). Essa ausência de padronização inviabiliza o registro histórico e o planejamento estruturado de rotas. Adicionalmente, os comerciantes relataram que a precificação do frete com os condutores ocorre de forma casuística, variando conforme a urgência do pedido ou a proximidade do entregador, o que resulta em custos operacionais instáveis.

O questionário quantitativo mapeou o comportamento dos consumidores e o nível de aceitação de um modelo tarifário compartilhado. O perfil de envio atual dos respondentes está consolidado no Gráfico 1.

Gráfico 1 – Frequência de necessidade de envio de objetos no perímetro urbano



Fonte: Autoria própria (2026).

Os dados do Gráfico 1 evidenciam que a maior parte dos participantes (67,8%) realiza os próprios deslocamentos para a entrega de objetos ("Eu mesmo levo"), enquanto 15,3% utilizam aplicações de logística existentes no mercado e 10,2% recorrem a redes de apoio informal ("Peço para alguém levar"). O percentual restante adota um comportamento misto, alternando a escolha com base na urgência ou no custo do frete. Esses indicadores sinalizam que a escassez de serviços acessíveis e com preços previsíveis induz os usuários a despender tempo e recursos próprios em trajetos redundantes.

Para avaliar a viabilidade do modelo econômico proposto, investigou-se a disposição dos usuários em aceitar um tempo de espera estendido em troca de uma redução tarifária (Gráfico 2).

Gráfico 2 – Aceitação do modelo de frete reduzido via rotas compartilhadas



Fonte: Autoria própria (2026).

A taxa de aceitação atingiu 96,6%, confirmando a disposição dos participantes em adotar rotas compartilhadas desde que houvesse uma redução proporcional no valor do frete. A convergência entre a carência tecnológica dos lojistas e a elevada receptividade do público valida as premissas de negócio da plataforma Zarpa, justificando a divisão entre as modalidades expressa e econômica, bem como o desenvolvimento do algoritmo de rateio financeiro.

5.2 Levantamento e Priorização de Requisitos

Para definir o escopo do Produto Mínimo Viável (MVP) da plataforma Zarpa e reduzir a subjetividade na escolha das funcionalidades, o projeto combinou a matriz

RICE (*Reach, Impact, Confidence, Effort*) com o framework MoSCoW. Enquanto o MoSCoW organiza a relevância dos requisitos para o negócio, a matriz RICE traz critérios numéricos para essa escolha, avaliando quatro fatores: o alcance de usuários, o impacto na experiência, a confiança da equipe na implementação e o esforço técnico exigido.

O cálculo da pontuação RICE funcionou como o critério objetivo do projeto. Ao cruzar o valor gerado para a logística urbana com o custo de desenvolvimento, o método evitou o foco em funções complexas de baixo retorno econômico ou operacional. Essa pontuação permitiu ordenar os 17 requisitos funcionais (RF) identificados de forma estritamente analítica.

A transição dos dados da matriz RICE para as categorias do MoSCoW seguiu as seguintes regras de corte:

- **Must Have (Obrigatórios):** Requisitos com pontuações altas na matriz RICE e necessidade para o MVP. Possuem grande alcance e impacto direto na proposta de valor (essenciais para o algoritmo de roteirização e para a divisão proporcional dos fretes). Sem estas funções, o MVP não opera.

- **Should Have (Importantes):** Funcionalidades com pontuação média-alta. São importantes para o modelo de negócios e para reter usuários, mas sua ausência não impede o início das operações da plataforma.

- **Could Have (Desejáveis):** Recursos com pontuação moderada ou baixa. São funções que melhoram a experiência do usuário (como customizações de interface), mas possuem baixo impacto imediato na validação do sistema.

- **Won't Have (Adiados):** Itens com as menores pontuações na matriz RICE. Representam ideias complexas, com alto esforço de desenvolvimento ou incerteza técnica nesta fase inicial, sendo adiadas para versões futuras.

O detalhamento completo da matriz de priorização e as notas de cada requisito estão documentados nos Apêndices A e B. A seguir, são apresentados os Requisitos Funcionais classificados como *Must Have*, que servem de base para a construção da arquitetura do sistema:

- **RF 01 – Cadastro e Autenticação de Usuários:** Permitir o registro e o login seguro de lojistas e entregadores, segmentando suas permissões e interfaces de forma isolada na aplicação;
- **RF 02 – Interface de Postagem de Pedidos:** Disponibilizar ao lojista um formulário para inserção dos dados de entrega (origem, destino, cubagem e peso) e seleção da modalidade de frete;
- **RF 03 – Geocodificação de Endereços:** Converter os endereços textuais digitados pelos usuários em coordenadas geográficas válidas (latitude e longitude) por meio de integração com APIs de mapas;
- **RF 04 – Agrupamento Geográfico de Pedidos (Batch):** Executar rotinas assíncronas em segundo plano para agrupar pedidos compatíveis por proximidade espacial e raio de atuação, consolidando as rotas da modalidade econômica;
- **RF 05 – Integração de Roteamento Viário (ORS):** Consumir a API externa do OpenRouteService para calcular as distâncias reais sobre a malha de ruas, estimar tempos de viagem e desenhar o traçado cartográfico no mapa;
- **RF 06 – Módulo de Rateio Financeiro:** Processar o algoritmo de divisão cooperativa de frete, calculando a economia gerada pela rota unificada e distribuindo-a de forma equitativa (50/50) entre o bônus do entregador e o desconto do lojista;
- **RF 07 – Controle de Concorrência Transacional:** Evitar anomalias de dupla alocação de frete através da aplicação de travas pessimistas (*Pessimistic Locking*) nativas na camada do Laravel Eloquent ORM via método `lockForUpdate()`, garantindo o isolamento de registros na camada do PostgreSQL;
- **RF 09 – Atualização Manual de Status:** Permitir que o entregador altere o ciclo de vida do pedido em tempo real (ex: "Coletado", "A caminho", "Entregue") para acompanhamento do lojista;

- **RF 10 – Notificações Push (Expressa):** Disparar alertas imediatos para os entregadores ativos localizados dentro do raio geográfico de um novo pedido urgente da modalidade sob demanda;

- **RF 12 – Transbordamento para Navegação Externa:** Permitir que o entregador exporte as coordenadas da rota otimizada diretamente para softwares de navegação GPS nativos do dispositivo móvel.

Para cumprir integralmente as diretrizes metodológicas estabelecidas nos objetivos específicos do projeto, foram mapeados e formalizados os Requisitos Não Funcionais (RNF) essenciais. Tais requisitos atuam como premissas arquiteturais inegociáveis para garantir o desempenho, a segurança e a robustez da infraestrutura cliente-servidor:

- **RNF 01 – Persistência Geoespacial:** O banco de dados deve utilizar a extensão PostGIS sobre o PostgreSQL para indexar coordenadas geográficas, garantindo que consultas de proximidade espacial operem em tempo inferior a 1,5 segundos;

- **RNF 02 – Segurança Transacional na Concorrência:** O mecanismo de trava pessimista gerenciado pelo Laravel deve garantir o isolamento completo de registros concorrentes, rejeitando requisições simultâneas com margem de colisão de milissegundos;

- **RNF 03 – Compatibilidade Multiplataforma:** O aplicativo móvel deve ser construído sobre o ecossistema Expo e React Native, assegurando uma experiência de interface fluida e compartilhamento de código-fonte nativo para sistemas Android e iOS;

- **RNF 04 – Desempenho do Motor de Roteamento:** O tempo total de processamento para receber as coordenadas das paradas, enviá-las ao OpenRouteService e renderizar o traçado geométrico na tela não deve ultrapassar 2,5 segundos sob condições estáveis de conectividade.

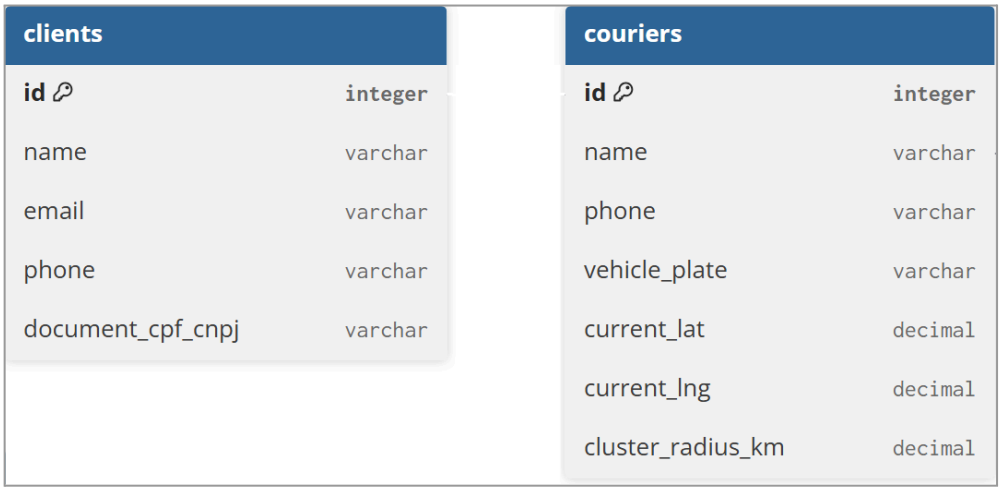
5.3 Modelagem do Banco de Dados

A modelagem do banco de dados foi desenvolvida sob os princípios de normalização de dados, visando garantir a integridade das transações financeiras e a eficiência na execução das consultas de agrupamento viário. Para viabilizar a operação estável e a escalabilidade do MVP, o modelo concentra-se em cinco entidades principais, mitigando redundâncias de armazenamento. Essa estrutura está detalhada a seguir, e o esquema completo encontra-se mapeado no Apêndice C.

5.3.1 Módulo de Atores do Sistema

O gerenciamento de usuários da plataforma Zarpa divide-se nos dois principais perfis operacionais do ecossistema, descritos na **Figura 3**. A tabela `clients` (Figura 3a) armazena os dados cadastrais e de identificação dos lojistas. Em contrapartida, a tabela `couriers` (Figura 3b) gerencia as informações dinâmicas de localização por meio dos campos `current_lat` e `current_lng`, responsáveis por registrar a última posição geográfica conhecida do entregador. O atributo `cluster_radius_km` delimita o raio de operação do condutor, assegurando o direcionamento exclusivo de pedidos compatíveis com sua área de atuação.

Figura 3 – Módulo de atores: (a) tabela clients; (b) tabela couriers.

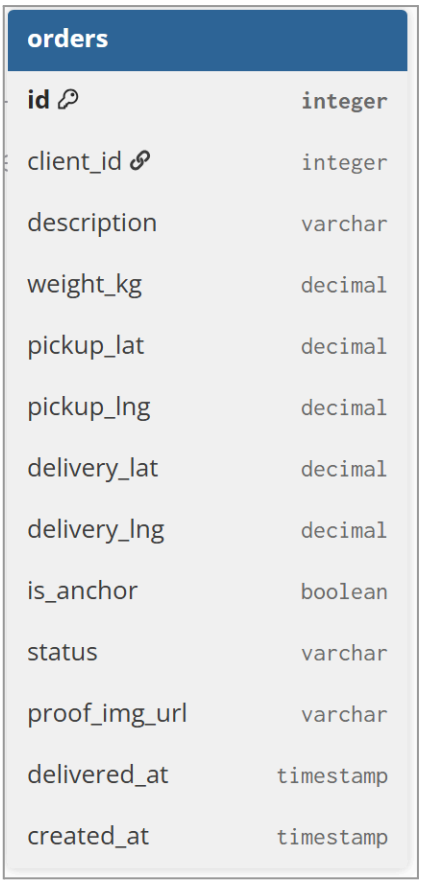


Fonte: Autoria própria (2026)

5.3.2. Fluxo de Pedidos e Carga

O fluxo de processamento de pedidos é estruturado conforme ilustrado na **Figura 4**. A tabela `orders` centraliza as coordenadas geográficas de coleta e entrega, utilizadas pela API de roteamento para o cálculo do trajeto ideal. Para otimizar a arquitetura inicial do MVP, atributos relativos à descrição e ao peso da carga foram incorporados diretamente nesta entidade, assim como os registros de confirmação de entrega (comprovante fotográfico e carimbo de data/hora). O campo `is_anchor` desempenha papel estratégico, sinalizando se o pedido deve atuar como ponto de referência geométrico para a criação de um novo lote de entregas.

Figura 4 – Fluxo de Pedidos e Carga: Tabela orders

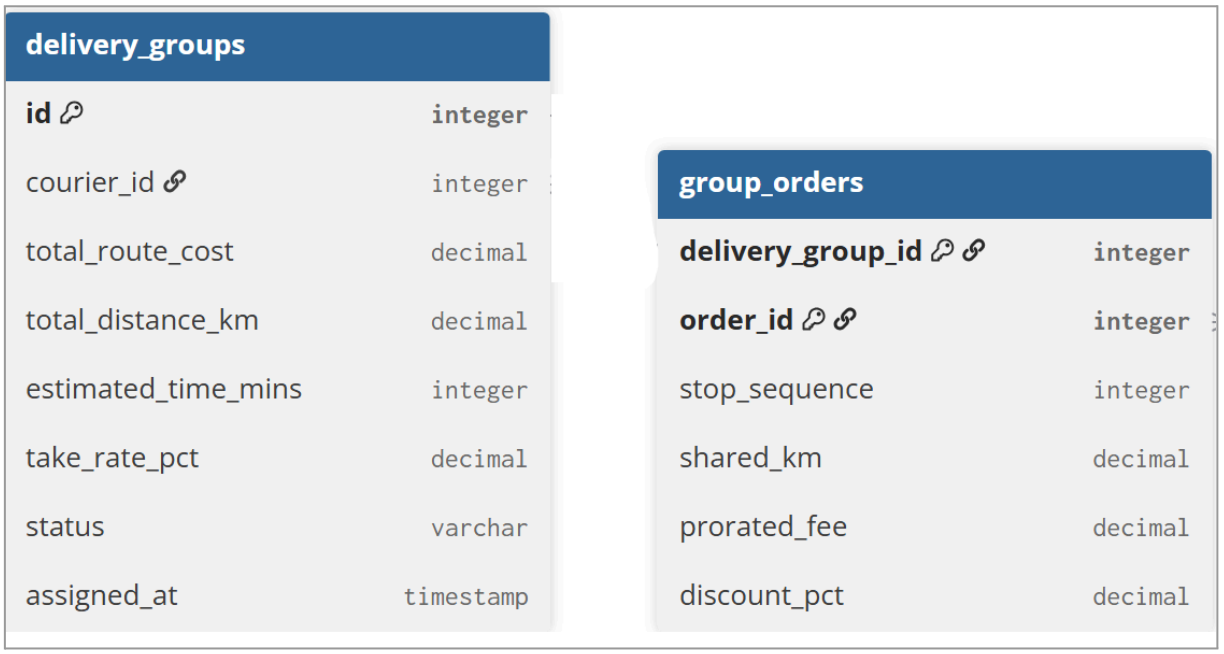


Fonte: Autoria própria (2026)

5.3.3. Inteligência de Agrupamento e Rateio Dinâmico

O núcleo de inteligência logística da plataforma é detalhado na **Figura 5**. A tabela `delivery_groups` (Figura 5a) organiza os lotes de entrega atribuídos a cada condutor, armazenando métricas consolidadas como distância total da rota, tempo estimado de deslocamento e taxas de serviço aplicadas. A tabela intermediária `group_orders` (Figura 5b) vincula os pedidos aos seus respectivos grupos, determinando a sequência cronológica das paradas. Essa entidade também registra os indicadores da economia colaborativa, incluindo a quilometragem compartilhada, a fração do custo operacional distribuída e o desconto concedido ao lojista em decorrência da eficiência do agrupamento.

Figura 5 – Motor de rateio dinâmico: (a) tabela delivery_groups; (b) tabela group_orders.



Fonte: Autoria própria (2026)

5.4 Prototipação de telas

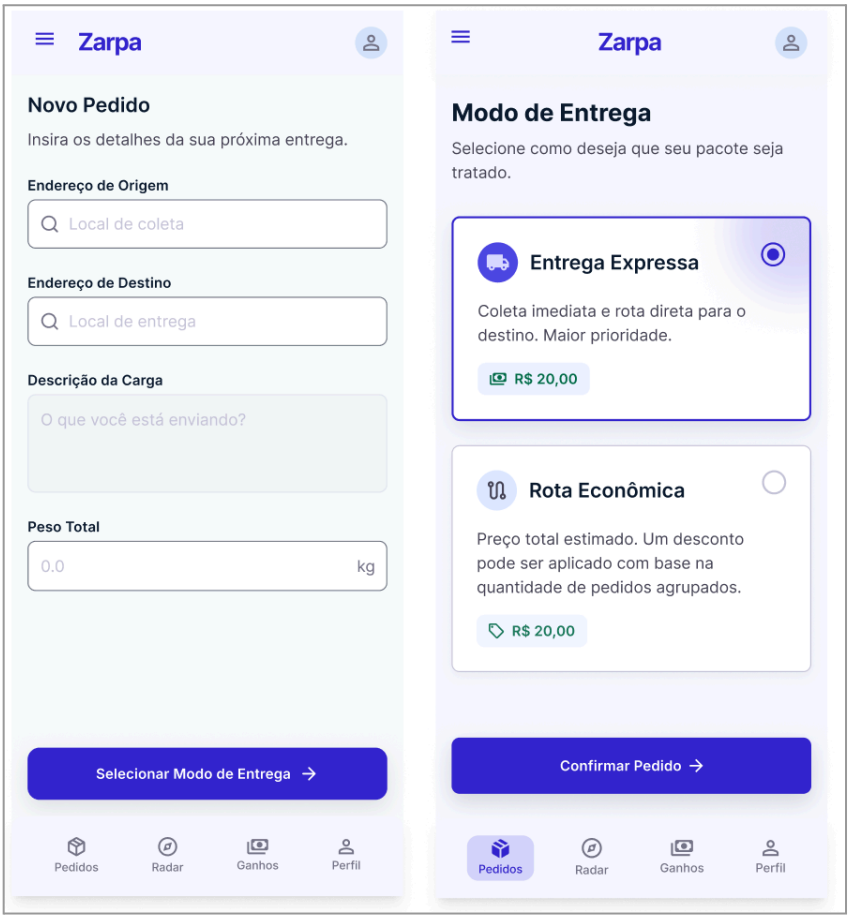
Para evidenciar o funcionamento prático da plataforma Zarpa e validar as regras de negócio modeladas, esta seção detalha as interfaces gráficas e as jornadas planejadas para os usuários. O design foi concebido sob o princípio do minimalismo operacional, priorizando a usabilidade tanto para os contratantes de frete quanto para os entregadores terceirizados. O fluxo visual divide-se de forma a suportar as duas modalidades centrais do sistema: o atendimento expresso (*On-Demand*) e o agrupamento econômico em lote (*Batch*).

5.4.1 Módulo do Cliente: Solicitação e Seleção de Modalidade

A jornada inicia-se na aplicação móvel do cliente, conforme ilustrado na **Figura 6**. O formulário de solicitação de entrega (Figura 6a) requer o preenchimento de quatro parâmetros fundamentais: endereço de origem, endereço de destino, categoria do produto e peso estimado, dados que subsidiam o cálculo inicial da rota.

Na etapa seguinte (Figura 6b), o usuário realiza a seleção da modalidade de frete. A interface contrapõe a Entrega Expressa (envio imediato e exclusivo) à Entrega Econômica (tarifa reduzida baseada em logística compartilhada). Na opção econômica, o pedido é retido para processamento noturno, permitindo que o algoritmo agrupe demandas de múltiplos estabelecimentos em um mesmo itinerário viário, viabilizando a concessão de descontos por meio da otimização do trajeto.

Figura 6 – Jornada do cliente: (a) formulário de solicitação; (b) seleção de modalidade de frete.



Fonte: Autoria própria (2026)

5.4.2 Módulo do Entregador: Radar Expresso e Agenda de Rotas

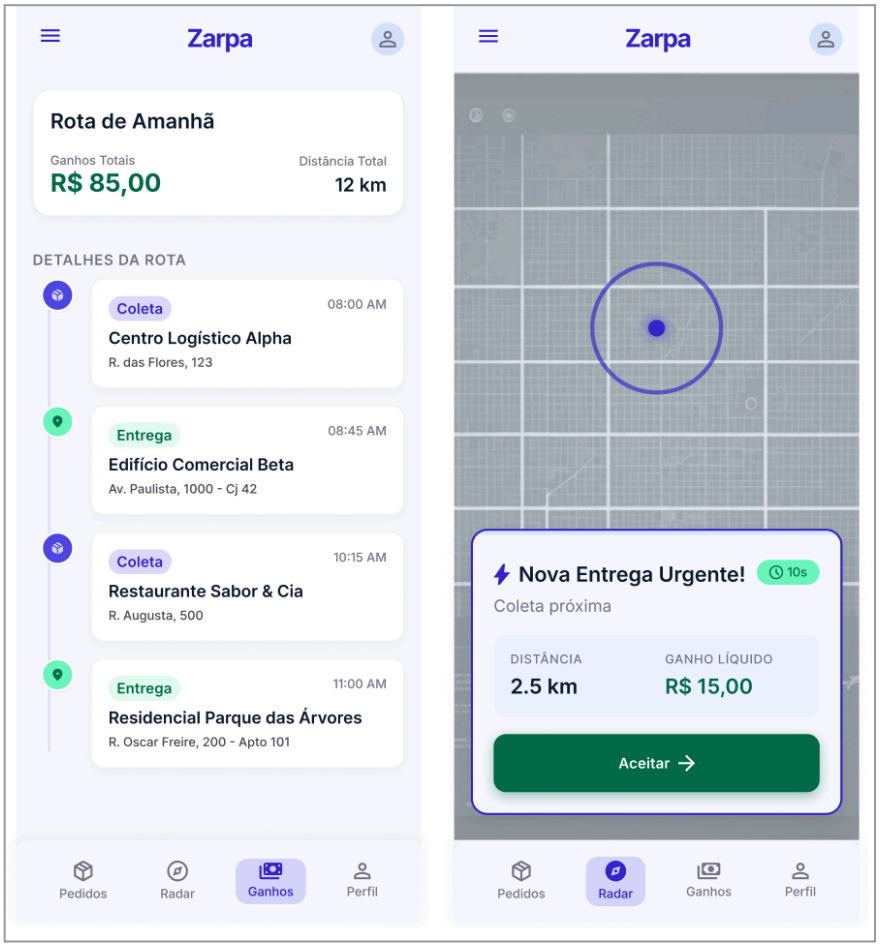
A interface do entregador opera sob duas dinâmicas distintas, mapeadas na **Figura 7**, cujas capturas de tela foram organizadas de forma a refletir os fluxos assíncronos e sob demanda.

Para o fluxo de rotas compartilhadas geradas pelo lote econômico (Figura 7a), o profissional dispõe de uma agenda com o itinerário previamente otimizado pelo processamento assíncrono noturno. Ao iniciar a jornada, o condutor executa o percurso sequencial guiado pelo mapa interativo, que indica a ordem exata de coletas e entregas, eliminando a necessidade de disputa por corridas e exibindo, no topo da interface, a quilometragem total e a remuneração consolidada do lote.

Por outro lado, para as demandas urgentes da modalidade expressa (Figura 7b), o aplicativo atua como um radar de ofertas ativas por proximidade espacial. A tela exibe o valor líquido do frete, a distância até o ponto de coleta e uma contagem regressiva de dez segundos para o aceite. Devido à natureza altamente

concorrencial deste modelo, a primeira aceitação válida disparará o bloqueio transacional pessimista gerenciado pelo Laravel ORM no banco de dados, inviabilizando imediatamente a oferta e atualizando a interface dos demais condutores na região.

Figura 7 – Visão do entregador: (a) agenda de rotas sequenciadas para o fluxo econômico; (b) radar de ofertas para o fluxo expresso imediato.



Fonte: Autoria própria (2026)

6 CONSIDERAÇÕES FINAIS

A logística de distribuição urbana na etapa de última milha (*last-mile*) enfrenta gargalos históricos associados à ociosidade de deslocamento e à volatilidade de custos, fatores que comprimem as margens de lucro de condutores autônomos e de pequenos estabelecimentos comerciais. A plataforma Zarpa mitiga essas problemáticas por meio de uma abordagem arquitetural baseada em dois modelos operacionais complementares: o atendimento expresso sob demanda, focado em alocação síncrona por proximidade, e o lote econômico, baseado no agrupamento assíncrono de demandas e no rateio colaborativo de tarifas.

Com a conclusão da etapa correspondente ao TCC 1, todos os objetivos iniciais propostos foram alcançados. Os marcos atingidos compreendem a fundamentação teórica sobre algoritmos de roteamento, a validação empírica de mercado via pesquisa de campo com lojistas e consumidores locais, o levantamento e a priorização da engenharia de requisitos através de matrizes analíticas, a modelagem conceitual e lógica do banco de dados relacional geoespacial e a prototipação de alta fidelidade das interfaces computacionais da aplicação.

Contudo, a transição da modelagem teórica para a implementação prática impõe desafios técnicos significativos, tais como a precisão e a latência na coleta de coordenadas de posicionamento global (GPS) em dispositivos móveis e a necessidade de uma massa crítica de usuários ativos para garantir a eficiência estatística do algoritmo de agrupamento noturno.

Para a etapa subsequente do trabalho (TCC 2), o escopo concentrar-se-á no desenvolvimento integral e na codificação da solução tecnológica. O cronograma de atividades priorizará a construção da infraestrutura de *back-end* (regras de negócio e motor de rateio), a implementação das aplicações móveis multiplataforma (*front-end*) e a integração síncrona com as APIs geoespaciais do *OpenRouteService*. Por fim, serão conduzidos testes experimentais sob cenários de tráfego real no perímetro urbano de Guarapuava (PR), visando homologar a precisão do algoritmo de distribuição financeira e a otimização dos itinerários viários calculados.

REFERÊNCIAS

BORZO. **Serviço de entrega expressa**. [S. l.], 2026. Disponível em: https://borzodelivery.com/br. Acesso em: 30 jun. 2026.

DANTZIG, George B.; RAMSER, John H. **The Truck Dispatching Problem. Management Science**, v. 6, n. 1, p. 80-91, 1959.

DIJKSTRA, E. W. **A note on two problems in connexion with graphs**. Numerische Mathematik, v. 1, n. 1, p. 269-271, 1959.

FETTE, I.; MELNIKOV, A. **The WebSocket Protocol**: RFC 6455. [S. l.]: Internet Engineering Task Force (IETF), 2011. Disponível em: https://datatracker.ietf.org/doc/html/rfc6455. Acesso em: 11 maio 2026.

FIGMA. **Figma**: the collaborative interface design tool. São Francisco, 2024. Disponível em: https://www.figma.com. Acesso em: 27 abr. 2026.

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

GEISBERGER, R.; SANDERS, P.; SCHULTES, D.; DELLING, D. **Contraction hierarchies**: faster and simpler hierarchical routing in road networks. In: INTERNATIONAL WORKSHOP ON EXPERIMENTAL AND EFFICIENT ALGORITHMS, 7., 2008, Provincetown. Proceedings [...]. Berlin: Springer, 2008. p. 319-333.

GIT. **Git**: --fast-version-control. [S. l.], 2024. Disponível em: https://git-scm.com. Acesso em: 27 abr. 2026.

GITHUB. **GitHub**: Where the world builds software. [S. l.], 2024. Disponível em: https://github.com. Acesso em: 27 abr. 2026.

GUTIÉRREZ, Antonio; DE VOS, Jonas. **Last mile logistics and the e-commerce boom**: challenges and solutions. Transport Reviews, v. 40, n. 6, p. 734-754, 2020.

HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY. **OpenRouteService API Documentation**. Heidelberg, Alemanha, 2024. Disponível em: https://openrouteservice.org/. Acesso em: 27 abr. 2026.

LALAMOVE. **Entregas rápidas e seguras para o seu negócio**. [S. l.], 2026. Disponível em: https://www.lalamove.com/pt-br/. Acesso em: 30 jun. 2026.

LUXEN, D.; VETTER, C. **Real-time routing with OpenStreetMap data**. In: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '11). New York, NY, USA: Association for Computing Machinery, 2011. p. 513–516. DOI: https://doi.org/10.1145/2093973.2094062.

UBER. **Uber Flash**: entregas no mesmo dia. [S. l.], 2026. Disponível em: https://www.uber.com/br/pt-br/drive/services/uber-flash/. Acesso em: 30 jun. 2026.

APÊNDICES

APÊNDICE A – LEVANTAMENTO COMPLETO DE REQUISITOS (MOSCOW)

Este apêndice apresenta os requisitos funcionais da plataforma Zarpa classificados pela metodologia MoSCoW (*Must*, *Should*, *Could*, *Won't Have*). Essa divisão define o escopo do projeto de forma direta, separando o que é obrigatório para o Produto Mínimo Viável (MVP) das funcionalidades planejadas para versões futuras.

ID	Categoria	Requisito	Descrição	Prioridade
RF 01	Funcional	Cadastro e Autenticação de Usuários	Criação de perfis e login isolado para Lojistas e Entregadores.	Must Have
RF 02	Funcional	Interface de Postagem de Pedidos	Painel para envio de objeto e marcação da modalidade de frete.	Must Have
RF 03	Funcional	Geocodificação de Endereços	Conversão de texto do endereço em coordenadas geográficas nativas.	Must Have
RF 04	Funcional	Algoritmo de Lote Econômico (Batch)	Rotina assíncrona para agrupamento automático de pedidos.	Must Have
RF 05	Funcional	Integração ORS (Roteamento Viário)	Consumo da API externa para cálculo de distâncias e traçado cartográfico.	Must Have
RF 06	Funcional	Módulo de Rateio Financeiro (50/50)	Cálculo automático da economia e distribuição equitativa do valor.	Must Have
RF 07	Funcional	Controle de Concorrência (Lock no BD)	Uso de travas pessimistas (Pessimistic Locking) via Laravel Eloquent.	Must Have
RF 08	Funcional	Painel de Tarefas do Entregador	Visualização estática da rota e lista de entregas agendadas.	Must Have
RF 09	Funcional	Atualização Manual de Status	Botões de alteração de ciclo de vida (Coletado e Entregue).	Must Have
RF 10	Funcional	Notificações Push (Expressa)	Disparo de aviso imediato para condutores num raio geográfico.	Should Have
RF 11	Funcional	Extrato e Histórico Simplificado	Exibição do valor final ganho/pago após o fecho da rota.	Should Have
RF 12	Funcional	Transbordamento para Navegação	Exportação de rotas para Waze ou Google Maps.	Could Have
RF 13	Funcional	Parametrização de Regras de Negócio	Painel de administração para alterar raio de busca e taxas.	Could Have
RF 14	Funcional	Sincronização via WebSockets	Atualização bidirecional de ecrãs em tempo real sem	Won't Have

			reload.	
RF 15	Funcional	Rastreamento Contínuo de Mapa	Acompanhamento dinâmico do ícone do motoboy em movimento.	Won't Have
RNF 01	Não Funcional	Persistência Geoespacial	Uso da extensão PostGIS no PostgreSQL para indexação espacial.	Must Have
RNF 02	Não Funcional	Segurança Transacional	Garantia de isolamento de registos concorrentes em milissegundos.	Must Have
RNF 03	Não Funcional	Compatibilidade Multiplataforma	Aplicação construída em React Native/Expo para Android e iOS.	Must Have
RNF 04	Não Funcional	Desempenho de Roteamento	Processamento e renderização de rotas do ORS em menos de 2.5s.	Must Have

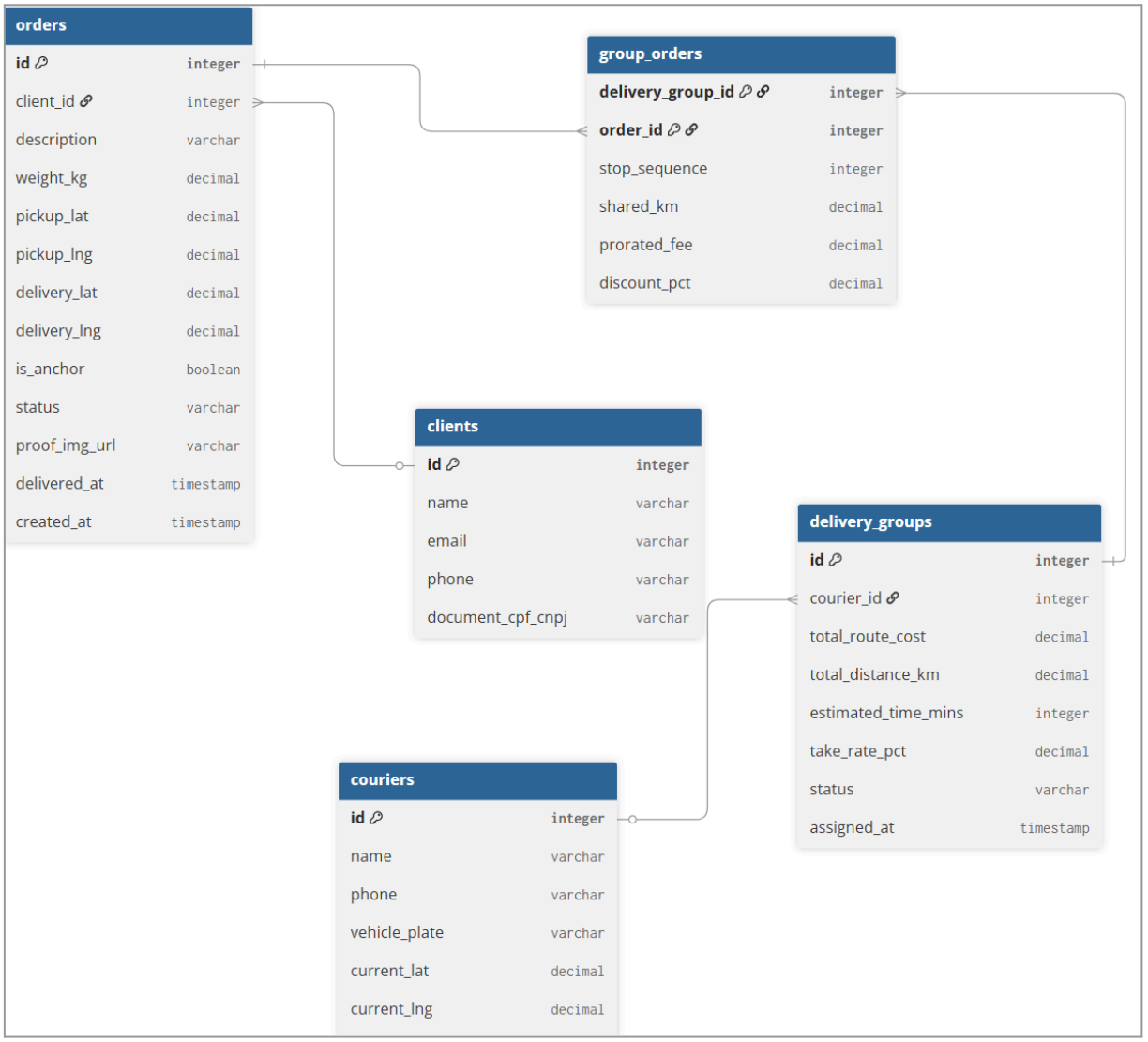
APÊNDICE B – MATRIZ RICE DOS REQUISITOS LEVANTADOS

Este apêndice detalha a priorização dos requisitos funcionais do sistema utilizando a metodologia RICE. A partir do cálculo matemático estabeleceu-se uma ordenação técnica e analítica para justificar a sequência de desenvolvimento.

ID	Requisito / Funcionalidade	Alcance (R)	Impacto (I)	Confiança (C)	Esforço (E)	Score RICE
RF 01	Cadastro e Autenticação de Usuários	1000	3	1.00	0.5	6000
RF 09	Atualização Manual de Status	1000	3	1.00	0.5	6000
RF 03	Geocodificação de Endereços (APIs)	1000	3	0.90	1.0	2700
RF 05	Integração ORS (Roteamento Viário)	1000	3	0.80	1.0	2400
RF 08	Painel de Tarefas do Entregador	1000	2	1.00	1.0	2000
RF 12	Transbordamento para Navegação Externa	500	2	1.00	0.5	2000
RF 02	Interface de Postagem de Pedidos	1000	3	0.90	1.5	1800
RF 07	Controle de Concorrência (Lock no BD)	1000	2	0.90	1.0	1800
RF 04	Algoritmo de Lote Econômico (Batch)	1000	3	0.80	1.5	1600
RF 11	Extrato e Histórico Simplificado	700	2	1.00	1.0	1400
RF 06	Módulo de Rateio Financeiro (50/50)	1000	3	0.80	2.0	1200
RF 10	Notificações Push (Expressa)	1000	2	0.80	1.5	1067
RF 13	Parametrização de Regras de Negócio	100	2	1.00	1.0	200
RF 14	Sincronização via WebSockets (Reverb)	1000	3	0.50	3.0	500
RF 15	Rastreamento Contínuo de Mapa (GPS)	1000	3	0.50	3.0	500

APÊNDICE C – MODELAGEM BANCO DE DADOS COMPLETA

Figura 8 – Modelagem Do Banco De Dados Completa



Fonte: Autoria própria (2026)