UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LEONARDO TOSIN

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

GUARAPUAVA

2026

LEONARDO TOSIN

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

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.

Com o recente aumento do comércio eletrônico no Brasil, a etapa logística conhecida como *last mile* (última milha) sofreu forte pressão para redução de custos, o que tem intensificado a necessidade de soluções tecnológicas capazes de tornar a logística urbana mais eficiente e economicamente equilibrada. Em cidades de médio porte, como Guarapuava, observa-se que a precificação das entregas realizadas por motoboys autônomos ocorre, frequentemente, de maneira subjetiva, variando entre cobrança por corrida, por hora ou por diária, o que pode gerar inconsistências nos valores praticados e baixa otimização das rotas. Além disso, mesmo quando múltiplas solicitações apresentam trajetórias semelhantes, raramente há um mecanismo estruturado que permita o encaixe dinâmico de pedidos em um percurso já iniciado com redistribuição proporcional dos custos entre os clientes. Diante desse contexto, este trabalho propõe o desenvolvimento de uma plataforma inteligente para intermediação de entregas urbanas com otimização colaborativa de rotas e redistribuição dinâmica de frete. A proposta fundamenta-se na utilização de algoritmos para a resolução do Problema do Caixeiro Viajante (TSP), visando a ordenação otimizada de múltiplas paradas em uma única jornada de entrega. A solução permite que o entregador selecione pedidos compatíveis geograficamente, gerando uma rota sequencial que minimiza o deslocamento total e possibilita a redistribuição proporcional dos custos de frete entre os usuários. A metodologia adotada envolve levantamento de requisitos funcionais e não funcionais, modelagem da solução, definição das regras de negócio e estruturação conceitual do algoritmo de agrupamento e redistribuição de custos. Como resultado esperado, pretende-se demonstrar a viabilidade técnica e conceitual da proposta, evidenciando seu potencial para promover eficiência e agilidade nas entregas e economia de gastos dos clientes, melhor aproveitamento das rotas e incentivo à colaboração logística. Espera-se que a aplicação possa contribuir significativamente para a modernização dos serviços de entrega urbana, tornando-os mais eficientes, transparentes e sustentáveis.

Palavras-chave: logística urbana; otimização de rotas; redistribuição de custos; entregas sob demanda; plataforma inteligente.

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.

The growth of on-demand delivery services has intensified the need for technological solutions capable of making urban logistics more efficient and economically balanced. In medium-sized cities, it is observed that the pricing of deliveries performed by independent motorcycle couriers is often defined in a subjective manner, varying between per-trip, hourly, or daily charges, which can generate inconsistencies in the prices charged and lead to poor route optimization. Furthermore, even when multiple delivery requests present similar trajectories, there is rarely a structured mechanism that allows the dynamic insertion of new orders into a route already in progress with proportional redistribution of costs among the clients. In this context, this work proposes the development of an intelligent platform for urban delivery intermediation with collaborative route optimization and dynamic freight redistribution. The main objective is to structure a computational model capable of identifying geographic compatibility between delivery requests, recalculating routes in real time, and proportionally redistributing the total operational cost, aiming to reduce the individual cost for customers while increasing the time efficiency of the courier. The methodology adopted involves the identification of functional and non-functional requirements, system modeling, definition of business rules, and the conceptual structuring of an algorithm for order grouping and cost redistribution. As an expected result, the study seeks to demonstrate the technical and conceptual feasibility of the proposed solution, highlighting its potential to promote efficiency and agility in deliveries, reduce customer costs, improve route utilization, and encourage collaborative logistics. It is expected that the application may significantly contribute to the modernization of urban delivery services, making them more efficient, transparent, and sustainable.

Keywords: urban logistics; route optimization; cost redistribution; on-demand delivery; intelligent systems.

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface* (Interface de Programação de Aplicação)
GPS	Global Positioning System
MVC	*Model-View-Controller*
GPS	*Global Positioning System* (Sistema de Posicionamento Global)
MVP	Minimum Viable Product
PHP	*Hypertext Preprocessor*
REST	*Representational State Transfer* (Transferência de Estado Representacional)
TSP	*Traveling Salesman Problem* (Problema do Caixeiro Viajante)

LISTA DE ILUSTRAÇÕES

SUMÁRIO

APÊNDICE B – MODELAGEM DO BANCO DE DADOS COMPLETA 37

1 INTRODUÇÃO

O crescimento dos serviços de entrega sob demanda no Brasil está diretamente relacionado ao avanço das tecnologias digitais e à ampla disseminação de smartphones e da internet de alta velocidade. Esse cenário consolidou um modelo de consumo baseado na conveniência, no qual usuários demandam rapidez e flexibilidade na entrega de produtos como alimentos, medicamentos e itens do cotidiano. Apesar dessas inovações tecnológicas, os desafios logísticos associados à distribuição de mercadorias, especialmente na etapa conhecida como *last mile* (última milha), permanecem sendo um dos maiores gargalos para a otimização de rotas e a redução dos custos operacionais.

Nesse contexto, destaca-se um micro-segmento ainda pouco estruturado: o de pequenas entregas realizadas entre pessoas físicas. Diferentemente das entregas vinculadas a estabelecimentos comerciais ou plataformas consolidadas, esse tipo de serviço envolve demandas pontuais, como o envio de documentos, objetos pessoais ou pequenas encomendas. Em geral, essas entregas são organizadas de maneira informal, sem padronização de processos ou suporte tecnológico adequado, o que dificulta a consolidação de múltiplas entregas em uma mesma rota e compromete a eficiência do serviço. Atualmente, a falta de sistemas automatizados para agrupar pedidos, planejar rotas e calcular preços torna as entregas urbanas independentes pouco eficientes. Fazer uma entrega de cada vez, sem planejamento geográfico, custa mais caro para o cliente e desperdiça o tempo e o combustível do entregador. Na área da computação, isso é classificado como um problema de roteirização e alocação de recursos, em que o desafio é encontrar a melhor sequência de paradas para encurtar caminhos e aproveitar o trajeto ao máximo.

No cenário local de Guarapuava, observa-se que a definição de preços para serviços de entrega não segue um padrão uniforme, sendo baseada predominantemente em critérios empíricos. Em alguns casos, adota-se a cobrança por frete individual; em outros, utilizam-se modelos baseados em tempo, como cobrança por hora, ou ainda valores fixos por diária. Essa variabilidade na precificação, associada à ausência de estratégias sistematizadas de planejamento de rotas, pode resultar em ineficiência tanto para o cliente que arca com custos

elevados em entregas isoladas quanto para o prestador do serviço, que não maximiza seu ganho em função do tempo e da distância percorrida.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolvimento de uma plataforma inteligente (aplicativo móvel e API) para intermediação de entregas urbanas sob demanda em Guarapuava. A solução usará algoritmos de TSP (Traveling Salesman Problem) e geolocalização para otimização colaborativa de rotas e redistribuição dinâmica de frete, visando reduzir custos para clientes e maximizar a eficiência de entregadores autônomos.

1.1.2 Objetivos Específicos

- Realizar o levantamento e a priorização dos requisitos funcionais e não funcionais necessários para estruturar a plataforma de logística colaborativa;
- Modelar a arquitetura lógica e visual do sistema, englobando a estrutura do banco de dados relacional e a prototipação das interfaces de usuário;
- Desenvolver o *backend* (API RESTful) para centralizar o gerenciamento de usuários, pedidos e o processamento matemático do rateio de fretes;
- Implementar a inteligência de roteirização no *back-end*, resolvendo instâncias do TSP para a ordenação otimizada da lista de pedidos selecionados pelo entregador;
- Construir a interface móvel multiplataforma consumindo os dados da API para viabilizar a interação fluida entre clientes e entregadores;
- Validar a eficácia da solução por meio de testes de rotas em tempo real e simulações de economia financeira, embasados nos dados coletados na pesquisa de campo.

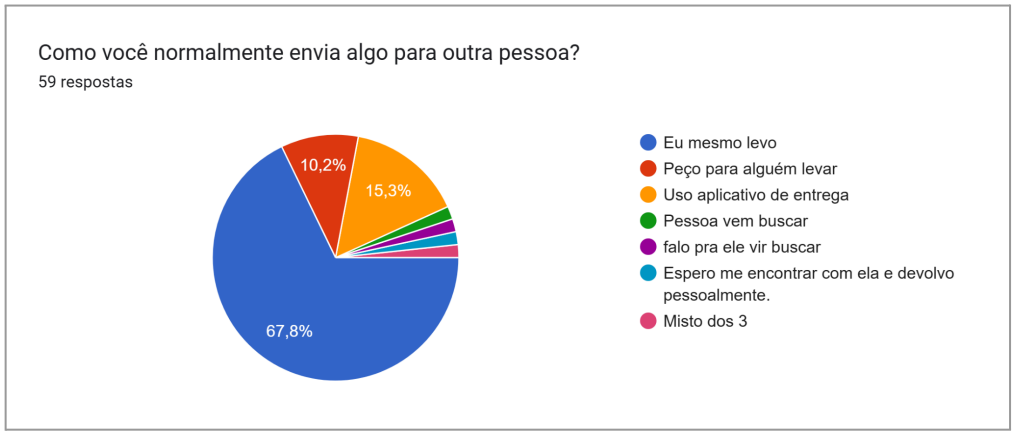
1.2 Justificativa

A necessidade desta plataforma justifica-se pela informalidade e ineficiência observadas no setor logístico local, tanto na perspectiva dos estabelecimentos comerciais quanto dos usuários finais. Para fundamentar o desenvolvimento do sistema, foram realizadas duas pesquisas de campo distintas: uma abordagem qualitativa com comerciantes locais e um levantamento quantitativo com 59 potenciais usuários, cujos dados detalhados encontram-se nos Apêndices deste trabalho.

Durante as entrevistas com os estabelecimentos, ao serem questionados sobre os seus sistemas de pedidos, a maioria relatou o uso de métodos manuais, como o aplicativo WhatsApp e anotações em papel, o que impossibilita a criação de um histórico rastreável. Ademais, identificou-se que a negociação financeira com os entregadores é subjetiva e altamente dependente da disponibilidade imediata e da relação interpessoal, gerando inconsistência e incerteza nos custos de frete.

Simultaneamente, a pesquisa quantitativa focada no usuário final corroborou a oportunidade de inovação no modelo de cobrança. Conforme ilustram os resultados da pesquisa, embora a maioria dos respondentes ainda realize suas entregas de forma autônoma (Gráfico 1), existe uma ampla aceitação para soluções otimizadas. Exatos 96,6% dos 59 entrevistados afirmaram que aceitariam pagar um valor menor pela entrega caso o entregador realizasse outra entrega no mesmo trajeto (Gráfico 2).

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



Fonte: Autoria própria (2026).

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

Fonte: Autoria própria (2026).

Em suma, a convergência entre a carência tecnológica dos comerciantes e a expressiva aceitação pública (96,6%) por um modelo de rotas compartilhadas justifica e viabiliza a construção desta plataforma. O sistema proposto atuará não apenas como um intermediador digital, mas como uma ferramenta de otimização logística, utilizando algoritmos de agrupamento geográfico para redistribuir proporcionalmente os custos do frete, profissionalizando o serviço autônomo e gerando economia para ambas as pontas da cadeia.

2 SOLUÇÃO PROPOSTA

Esta seção descreve a arquitetura técnica e a lógica algorítmica proposta para a plataforma de intermediação de entregas, a partir de agora, nomeada como “plataforma Zarpa”. O sistema visa mitigar a ineficiência logística em Guarapuava através da automação do roteamento e transparência no cálculo de custos.

2.1 Visão Geral da Plataforma

A plataforma Zarpa foi concebida para atuar como um ecossistema digital inteligente que redefine a forma como as entregas urbanas são realizadas e precificadas. Ao contrário das soluções convencionais, que funcionam apenas como um mural de anúncios ou um canal de comunicação direta, a Zarpa introduz uma camada de inteligência computacional que busca constantemente a eficiência máxima em cada deslocamento. O sistema opera através de uma estrutura distribuída, onde um núcleo de processamento centralizado (o *backend*) organiza e cruza as informações geradas pelos aplicativos dos clientes e dos entregadores em tempo real.

O funcionamento da solução baseia-se na identificação de oportunidades de colaboração dentro do mapa urbano. Quando um novo pedido de entrega é registrado, o sistema não o trata de forma isolada; em vez disso, ele analisa a origem e o destino daquela encomenda em relação a outros pedidos já existentes ou em andamento. Caso o algoritmo detecte que duas ou mais entregas possuem trajetos compatíveis, a plataforma propõe automaticamente o agrupamento desses pedidos. Essa abordagem permite que o entregador realize uma única rota otimizada para atender múltiplos clientes, o que potencialmente reduzirá o tempo ocioso do profissional e a emissão de poluentes no trânsito urbano.

Além da otimização física das rotas, a plataforma implementa uma lógica de redistribuição financeira que garante a sustentabilidade do modelo. O diferencial desta solução está na capacidade de transformar a eficiência logística em benefício econômico compartilhado. Quando o sistema consolida uma rota compartilhada, ele recalcula o custo do frete para os clientes envolvidos, oferecendo um desconto que incentiva o uso da plataforma. Simultaneamente, o entregador recebe uma remuneração superior à que obteria em uma entrega individual, pois sua produtividade por quilômetro rodado aumenta. Dessa forma, a Zarpa deixa de ser apenas uma ferramenta de transporte para se tornar um motor de economia colaborativa, onde a tecnologia assegura ganhos reais para todas as pontas da operação.

2.2 Descrição dos Usuários

A plataforma Zarpa foi idealizada para conectar três perfis distintos de usuários, equilibrando necessidades que, à primeira vista, podem parecer opostas dentro do ecossistema de logística urbana. A seguir, detalham-se as características e os objetivos de cada ator no sistema.

2.2.1 Cliente

O cliente abrange tanto pessoas físicas quanto pequenos comerciantes locais que possuem a necessidade de enviar ou receber objetos dentro da malha urbana. O grande atrativo da plataforma para este público é a flexibilidade financeira aliada à conveniência. Ao sinalizar que sua entrega não possui urgência máxima, o cliente permite que o sistema tente encaixar seu pacote na rota de outro usuário. Ao abrir mão da exclusividade da viagem, ele é recompensado com uma redução direta no valor final do frete, mantendo a segurança e a rastreabilidade do envio.

2.2.2 Entregador

O entregador, geralmente um motociclista autônomo, é o parceiro logístico responsável pelo transporte físico das mercadorias. Enquanto o cliente busca reduzir custos, o objetivo central deste usuário é maximizar sua rentabilidade diária e otimizar seu tempo. A plataforma atende a essa necessidade mitigando o tempo ocioso e as viagens de retorno com o baú vazio. Através do agrupamento de pedidos, o entregador consegue coletar e entregar múltiplas encomendas em um único trajeto contínuo, fazendo com que seu ganho acumulado por quilômetro rodado seja significativamente maior do que em corridas exclusivas.

2.2.3 Administrador

O administrador atua nos bastidores e representa a equipe de gestão da plataforma, sendo o responsável pela manutenção técnica e operacional de todo o ecossistema. Suas atividades envolvem o monitoramento do fluxo logístico, a aprovação e validação de novos cadastros de entregadores para garantir a segurança da operação, e a mediação de eventuais disputas. Além do suporte básico, este perfil tem a função crítica de parametrizar e auditar as taxas de rateio geradas pelo algoritmo, assegurando que o modelo financeiro do aplicativo permaneça viável e sustentável.

3 MATERIAIS E MÉTODOS

A construção da plataforma Zarpa fundamenta-se num conjunto de ferramentas tecnológicas e modelos matemáticos de otimização que, integrados, permitem a resolução de problemas complexos de logística urbana. Este capítulo descreve os recursos computacionais selecionados e a metodologia adotada para transformar os requisitos do projeto em uma solução de software funcional.

3.1 Materiais

Nesta seção, descrevem-se os materiais e as tecnologias selecionadas para o desenvolvimento da plataforma proposta, abrangendo as etapas de prototipação, controle de versão, desenvolvimento de software e processamento de dados geoespaciais.

- **Figma:** Ferramenta de prototipação de alta fidelidade e colaboração em nuvem. É utilizada para a concepção das interfaces de usuário (UI) e definição da experiência do usuário (UX), permitindo a validação de fluxos operacionais antes da implementação técnica (FIGMA, 2024).

- **Git e GitHub:** O Git é empregado como sistema de controle de versões distribuído, assegurando a integridade do código-fonte e o histórico de alterações. O GitHub atua como repositório remoto e ferramenta de gestão de projeto, utilizando o método Kanban para organização de tarefas (GIT, 2024; GITHUB, 2024).
- **Mermaid:** Ferramenta baseada em JavaScript que utiliza definições de texto inspiradas em Markdown para a geração dinâmica de diagramas e fluxogramas. Sua aplicação no projeto visa documentar a arquitetura do sistema e os processos de decisão do algoritmo de rateio, permitindo que diagramas técnicos evoluam em conjunto com o código-fonte (MERMAID, 2024).
- **React Native:** Framework para desenvolvimento de aplicações móveis multiplataforma baseado em JavaScript e React. A escolha fundamenta-se na necessidade de prover uma aplicação nativa para Android e iOS com uma base de código unificada, otimizando o tempo de desenvolvimento sem comprometer o acesso a APIs de hardware, como GPS (META PLATFORMS, 2024).
- **Laravel:** Framework PHP orientado a objetos que segue o padrão MVC (Model-View-Controller). É responsável pela camada de backend e construção da API RESTful, gerenciando a lógica de negócio, autenticação e a implementação do algoritmo de redistribuição proporcional de fretes (LARAVEL, 2024).
- **Laravel Reverb:** Servidor de *WebSockets* nativo, utilizado para viabilizar a comunicação bidirecional e em tempo real. Esta tecnologia é fundamental para a sincronização de estado da plataforma, garantindo que a atualização de disponibilidade de pedidos e o rastreamento dos entregadores ocorram com baixa latência (LARAVEL, 2024).
- **PostgreSQL e PostGIS:** O PostgreSQL atua como sistema de gerenciamento de banco de dados relacional. A extensão PostGIS é incorporada para adicionar suporte a objetos geográficos, permitindo a execução de consultas espaciais complexas, essenciais para o agrupamento geográfico de pedidos (POSTGIS CONTRIBUTORS, 2024).
- **OpenRouteService (ORS):** Serviço de roteamento baseado em dados do *OpenStreetMap*. Fornece matrizes de tempo e distância, além de algoritmos

de otimização de rotas, servindo como motor principal para o cálculo de trajetos eficientes para os condutores (HEIDELBERG INSTITUTE FOR GEOINFORMATION TECHNOLOGY, 2024).

- **React Native Maps:** Componente de integração para visualização de mapas em dispositivos móveis. Diferente de soluções de renderização via *web*, esta biblioteca utiliza os motores nativos de renderização cartográfica do sistema operacional (Google Maps ou Apple Maps), garantindo maior fluidez na interação tátil e no rastreamento em tempo real (REACT NATIVE COMMUNITY, 2024).

3.2 Métodos

O desenvolvimento da plataforma Zarpa é conduzido através de uma sequência lógica de etapas técnicas, fundamentadas em metodologias ágeis, que visam transformar a problemática da logística urbana em um artefato computacional escalável. O processo está estruturado em sete etapas interdependentes:

Etapa 1 : O ponto de partida consiste na definição do público-alvo (lojistas e entregadores de Guarapuava) e na identificação das dores latentes no processo de frete atual. Esta etapa baseia-se em dados coletados via pesquisa de campo (onde identificou-se, por exemplo, a aceitação de 96,6% para o modelo colaborativo), permitindo a tradução de necessidades de negócio em requisitos técnicos funcionais e não funcionais.

Etapa 2: Para garantir a viabilidade do MVP (do inglês, Minimum Viable Product - Produto Mínimo Viável) no cronograma acadêmico, utiliza-se a técnica **MoSCoW (CLEGG; BARKER, 1994)**. Esta etapa é crucial para separar as funcionalidades vitais (agrupamento de rotas e cálculo de rateio) das funcionalidades desejáveis, evitando o "inchaço" do escopo e focando no núcleo inovador da plataforma.

Etapa 3: A partir dos requisitos priorizados, elaboram-se as *User Stories* para detalhar a interação sob a perspectiva do usuário. Para elevar o rigor técnico e alinhar o comportamento do sistema ao desenvolvimento, adota-se a sintaxe

Gherkin (NORTH, 2006), baseada na metodologia *Behavior-Driven Development* (BDD). Essa abordagem facilita a criação de testes automatizados e garante que o critério de aceitação seja binário e claro.

Etapa 4: Com o fluxo de comportamento definido, utiliza-se a ferramenta Figma para a criação de protótipos de alta fidelidade. O objetivo aqui não é apenas a estética, mas a validação do fluxo de navegação e da experiência do usuário (UX), simulando a jornada do lojista ao solicitar uma entrega e do entregador ao aceitar um agrupamento.

Etapa 5: Paralelamente ao design, realiza-se a modelagem lógica do banco de dados. Utilizando a ferramenta Mermaid, definem-se as entidades (Usuários, Pedidos, Rotas, Transações) e seus relacionamentos. Esta etapa é vital para suportar a lógica de rateio, garantindo que o banco de dados seja capaz de persistir a economia gerada por cada rota de forma auditável.

Etapa 6: Governança Ágil e Planejamento de Sprints Em seguida, as tarefas de desenvolvimento serão estruturadas com base nos resultados das etapas anteriores. Cada tarefa será estimada conforme sua complexidade e organizada em *Sprints*, de acordo com a metodologia *Scrum Sprints* (ATLASSIAN, 2025b). Essa abordagem permite que o projeto seja desenvolvido em pequenas entregas, facilitando o controle do progresso e a adaptação a mudanças. Para a criação e organização das *Sprints*, será utilizada a ferramenta *Projects* do GitHub.

Etapa 7: Fluxo de Desenvolvimento e Padronização de Código Por fim, inicia-se o desenvolvimento técnico utilizando Git e GitHub para controle de versão. Adota-se o padrão *Git Feature Branch Workflow* (ATLASSIAN, 2025a), onde cada funcionalidade é isolada em uma ramificação específica. Para garantir a rastreabilidade entre o código e os requisitos, as ramificações seguirão a nomenclatura padronizada:

```
@git-user/local-mobile|backend/issue-number/feature-name
```

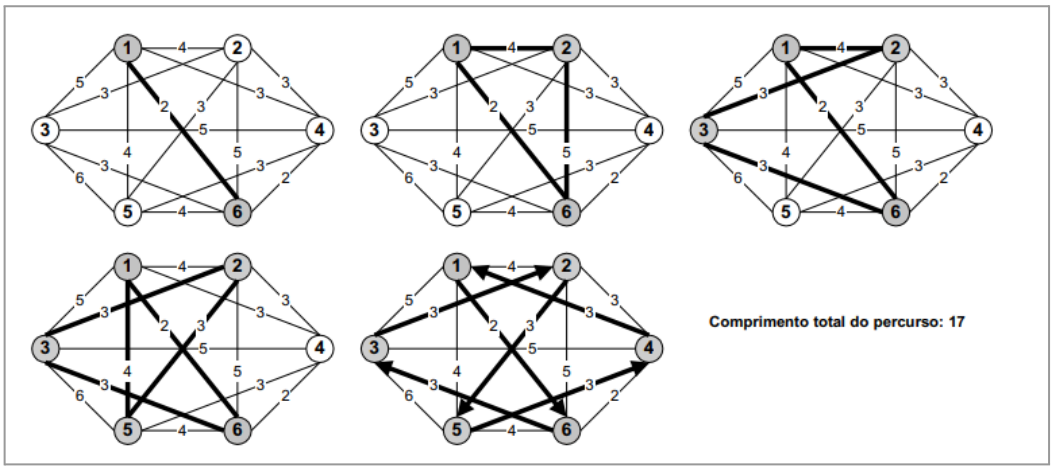
Após a finalização de cada tarefa, será criada uma *Pull Request* para revisão e posterior integração à branch principal.

3.2.1 Modelagem Lógica e Algoritmos de Roteamento

Diferente das ferramentas físicas, a inteligência da plataforma Zarpa baseia-se em um modelo híbrido de processamento geoespacial e otimização combinatória, estruturado para ser executado em três etapas consecutivas que compõem o motor de decisão do sistema:

- **Definição do Pedido Âncora:** A lógica inicia-se quando o entregador aceita uma ordem de serviço individual. Este pedido torna-se a "âncora" do trajeto, definindo o vetor principal de deslocamento do profissional.
- **Filtragem Espacial de Oportunidades (Clustering):** Com a rota ativa, o sistema executa uma consulta espacial no banco de dados para identificar outros pedidos pendentes em um raio de proximidade (ex: 2 km) de toda a extensão da rota atual. O objetivo é apresentar ao entregador apenas "pacotes extras" que não demandem desvios significativos do caminho original.
- **Otimização via Problema do TSP:** Uma vez que o entregador seleciona os pedidos adicionais da lista sugerida, o sistema consolida todas as coordenadas de coleta e entrega. Para definir a sequência mais eficiente, utiliza-se a lógica do Problema do Caixeiro Viajante (TSP) através da integração com uma API de roteirização externa (como o *OpenRouteService*). O algoritmo organiza as paradas de forma a minimizar a distância total e o tempo, garantindo que a jornada múltipla seja mais curta que a soma das jornadas individuais, representados na Figura 1 abaixo:

Figura 1 – Exemplo de solução por Inserção Mais Próxima do TSP



Fonte: Adaptado de UP (2024).

3.2.2 Estratégia de Redistribuição e Rateio

O modelo financeiro da plataforma baseia-se na divisão de custos de rota entre múltiplos clientes. A lógica de rateio funciona em dois pilares:

1. **Rateio de Custos:** Pedidos agrupados geram uma rota com custo total inferior à soma de fretes individuais. Essa economia é repassada automaticamente aos clientes como desconto.
2. **Divisão de Pagamento (*Split*):** O montante arrecadado é dividido entre a taxa de intermediação da plataforma e a remuneração do entregador. O algoritmo garante que o ganho do profissional em rotas agrupadas seja sempre superior ao de uma entrega individual.

Nesta etapa do projeto, o foco reside na validação dos cálculos matemáticos de rateio, postergando integração com gateways de pagamento[1] para versões futuras.

[1] Interface tecnológica que processa transações financeiras em ambientes virtuais, atuando como ponte de comunicação entre a plataforma, as instituições bancárias e as operadoras de cartão de crédito para autorização e liquidação de pagamentos.

3.2.3 Controle de Concorrência e Sincronização

Para evitar que múltiplos entregadores aceitem o mesmo pedido simultaneamente, aplicam-se duas camadas de controle:

1. **Bloqueio de Dados:** O sistema utiliza o estado do banco de dados para validar o aceite. A transação só é concluída se o pedido permanecer com o status "Pendente" no exato instante da gravação, impedindo duplicidade.
2. **Sincronização via WebSockets:** Utilizando o protocolo WebSocket, o servidor remove instantaneamente a oferta do mapa de todos os condutores assim que o primeiro aceite é validado, garantindo que a interface reflita o estado real do sistema sem necessidade de atualização manual.

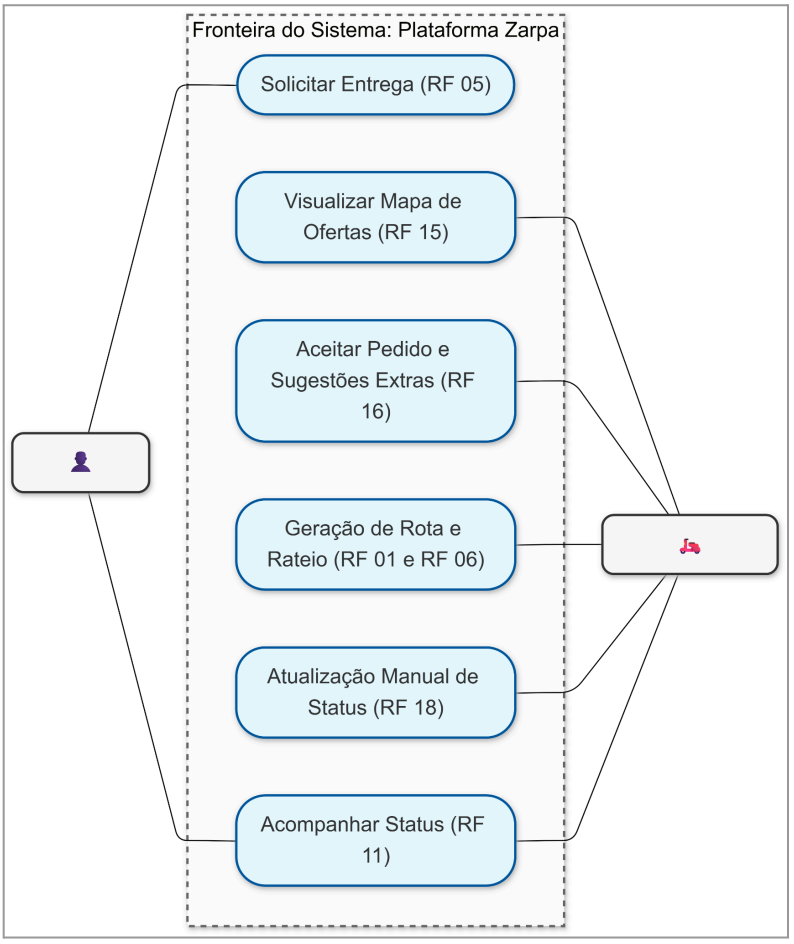
4 RESULTADOS PARCIAIS

Este capítulo apresenta os resultados técnicos obtidos durante a fase de análise e design da plataforma Zarpa. Os produtos aqui descritos consolidam a base estrutural necessária para a implementação do MVP, transpondo as necessidades dos usuários para modelos formais de software.

4.1 Modelagem Funcional e de Processos

O primeiro resultado da modelagem é o **Diagrama de Caso de Uso**, exposto na Figura 2, que define as fronteiras do sistema e as interações entre os atores. Este modelo foca na autonomia do entregador e no papel do sistema como agente monitor de prazos e otimizador de trajetos.

Figura 2 – Diagrama de Caso de Uso da Plataforma Zarpa.



Fonte: Autoria própria (2026)

A modelagem evidencia que a plataforma não apenas intermedeia pedidos, mas gerencia ativamente o ciclo de vida da entrega, desde a solicitação inicial até o processamento da rota final.

4.2 Levantamento e Priorização de Requisitos

A definição das funcionalidades da plataforma Zarpa fundamenta-se na metodologia MoSCoW, visando a entrega de um Produto Mínimo Viável (MVP) focado na validação do motor de roteirização e no algoritmo de rateio. O foco reside no núcleo da solução, postergando recursos de conveniência ou alta complexidade operacional, como o monitoramento automático de prazos.

Os principais requisitos identificados são:

- **RF 01 – Geração de Rota e Sequenciamento Otimizado (Problema TSP):** O sistema deve permitir que o entregador solicite o cálculo do melhor trajeto, processando as coordenadas via integração com a API *OpenRouteService* para retornar a sequência de paradas com menor distância ou tempo.
- **RF 02 – Sincronização em Tempo Real (Broadcast):** O sistema deve utilizar *WebSockets* (via Laravel Reverb) para notificar os dispositivos conectados sobre mudanças de estado, removendo pacotes aceitos da tela dos demais usuários.
- **RF 04 – Agrupamento Dinâmico por Raio:** O sistema deve monitorar a posição do condutor e buscar pedidos pendentes em um raio definido em relação à sua rota atual.
- **RF 06 – Cálculo de Rateio Financeiro:** Algoritmo que calcula o desconto do cliente e o pagamento do entregador baseado na quilometragem economizada.
- **RF 11 – Acompanhamento de Fluxo (Status):** Notificações de mudança de estado e interface para o cliente e entregador acompanharem o ciclo de vida do pedido.

A listagem completa e original de todos os requisitos funcionais encontra-se detalhada no **Apêndice A**.

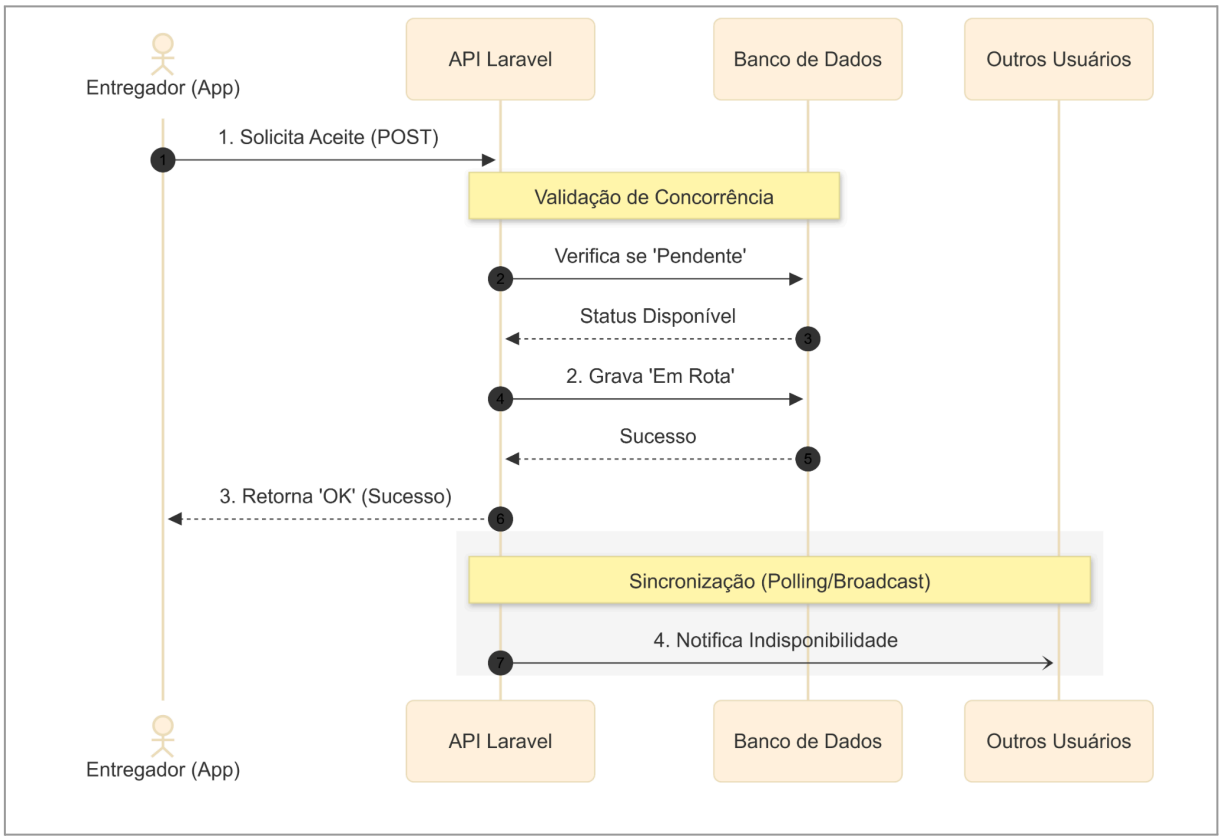
4.3 Detalhamento da Lógica de Negócio

Esta seção materializa as definições metodológicas e os requisitos funcionais em fluxos operacionais, detalhando a dinâmica de execução do sistema em ambiente real. O processo de intermediação e otimização da plataforma Zarpa é regido por dois fluxos críticos: o ciclo de aceite de pedidos e a execução da rota com o cálculo de rateio.

O primeiro fluxo, ilustrado no **Diagrama de Sequência da Figura 3**, detalha a interação entre a aplicação móvel, o backend em Laravel e a persistência de dados.

A lógica aqui prioriza a integridade do sistema através da prevenção de condições de corrida. Quando um entregador solicita o aceite de um pedido (passo 1), o sistema executa uma validação atômica no banco de dados para confirmar se o status ainda é "Pendente". Caso o pedido já tenha sido processado por outro condutor no mesmo milissegundo, a API retorna um erro de conflito (HTTP 409), impedindo a duplicidade de atribuição. Confirmada a disponibilidade, o status é atualizado para "Em Rota" e um evento de sincronização é disparado para os demais usuários (via *WebSockets* ou *Polling*, conforme o RF 02), removendo a oferta das interfaces vizinhas e garantindo a consistência da oferta na rede.

Figura 3 – Diagrama de Sequência: Aceite de Pedidos



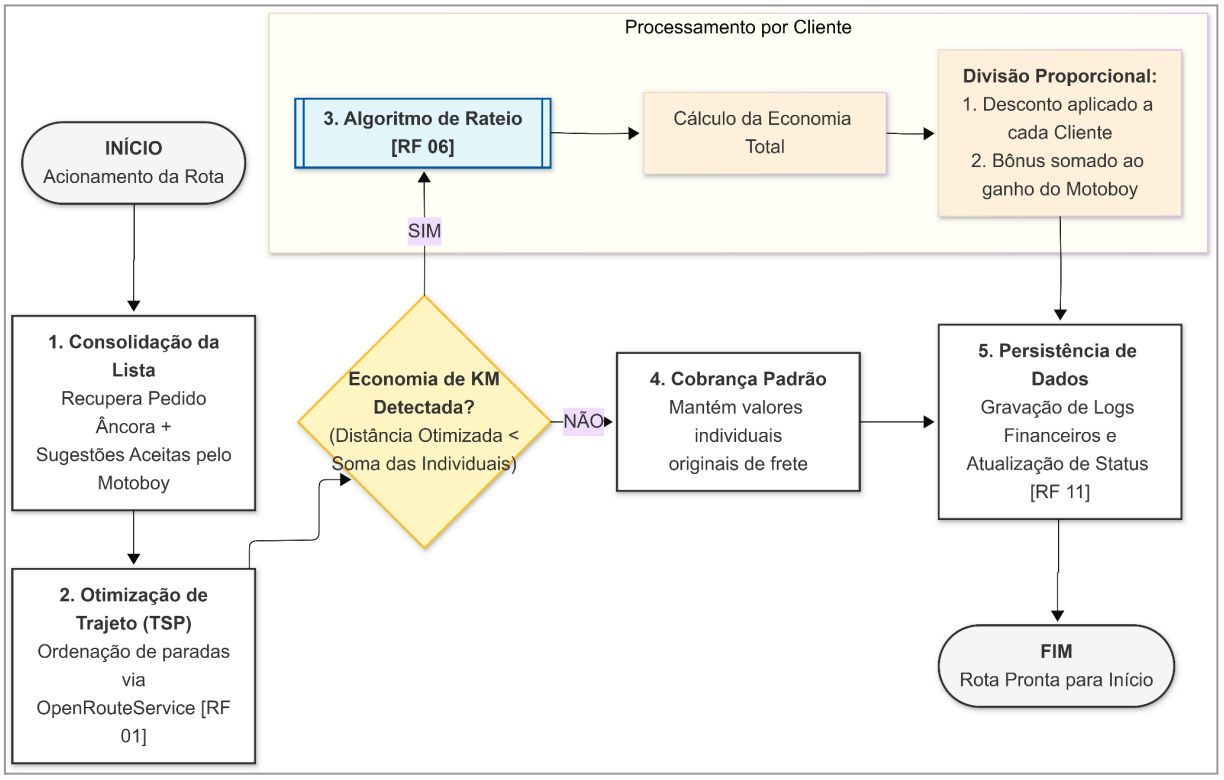
Fonte: Autoria própria (2026)

Dando continuidade à jornada, após a consolidação de uma lista de entregas — composta por um "Pedido Âncora" e sugestões adicionais aceitas pelo condutor dentro de um raio de alcance (RF 04) — o sistema processa a lógica de roteirização e viabilidade financeira, conforme apresentado no **Fluxograma da Figura 4**. O

diferencial desta lógica é a automação do **Algoritmo de Rateio (RF 06)** baseada na eficiência logística.

Ao acionar a execução da rota, o sistema utiliza a API *OpenRouteService* para calcular o melhor trajeto (Problema do Caixeiro Viajante - TSP, conforme o RF 01). O gatilho para a redistribuição do frete é a comparação entre a distância total da rota otimizada e a somatória das distâncias que seriam percorridas em trajetos individuais. Caso seja detectada uma economia real de quilometragem, o sistema aplica o rateio: uma parcela do valor economizado é convertida em desconto para cada cliente da rota, enquanto a outra parcela é revertida como bônus de produtividade ao entregador. Se a rota não gerar ganho de eficiência, o sistema mantém as cobranças individuais padrão, protegendo a margem de ganho do entregador e garantindo a transparência do processo (RF 11 e RF 12). Esta lógica assegura que a colaboração seja sempre vantajosa para todos os atores envolvidos, sem onerar o sistema em rotas geograficamente inviáveis.

Figura 4 – Fluxo de Rota do Entregador



Fonte: Autoria própria (2026)

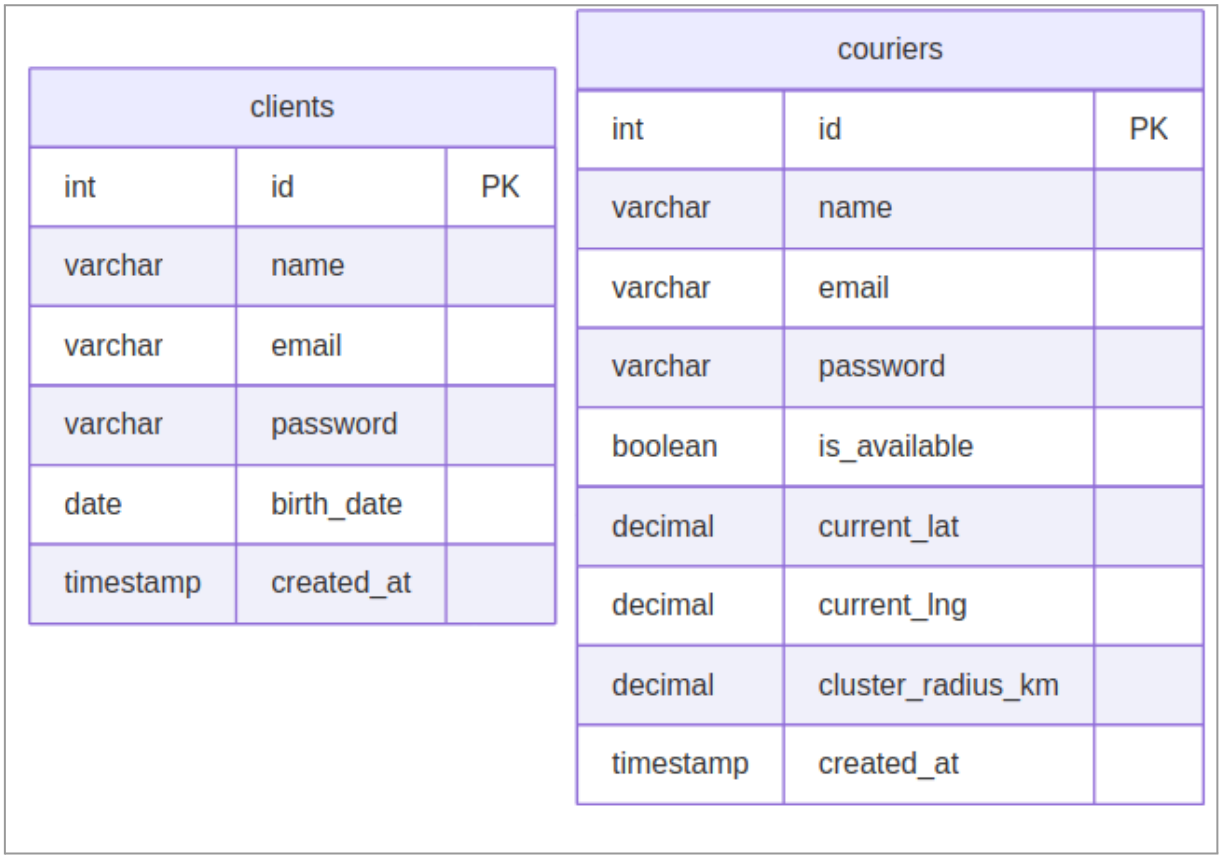
4.4 Modelagem do Banco de Dados

A estrutura do banco de dados foi idealizada e normalizada para garantir a integridade das transações financeiras e a eficiência dos algoritmos de roteirização. A seguir, o modelo é detalhado em quatro recortes principais. A modelagem completa do banco de dados pode ser visualizada no Apêndice B.

4.4.1 Módulo de Atores do Sistema

Este módulo (representado na Figura 5) gerencia as duas entidades fundamentais da plataforma. A tabela `clients` armazena os dados cadastrais dos solicitantes. Já a tabela `couriers` (entregadores) possui campos críticos para a inteligência de geolocalização, como `current_lat`, `current_lng` e `cluster_radius_km`. Este último campo define o raio de atuação preferencial do entregador, permitindo que o sistema filtre ofertas de pedidos que estejam dentro de sua zona de operação, otimizando o tempo de deslocamento inicial.

Figura 5 – Tabelas clients e courier da modelagem do banco



clients		
int	id	PK
varchar	name	
varchar	email	
varchar	password	
date	birth_date	
timestamp	created_at	

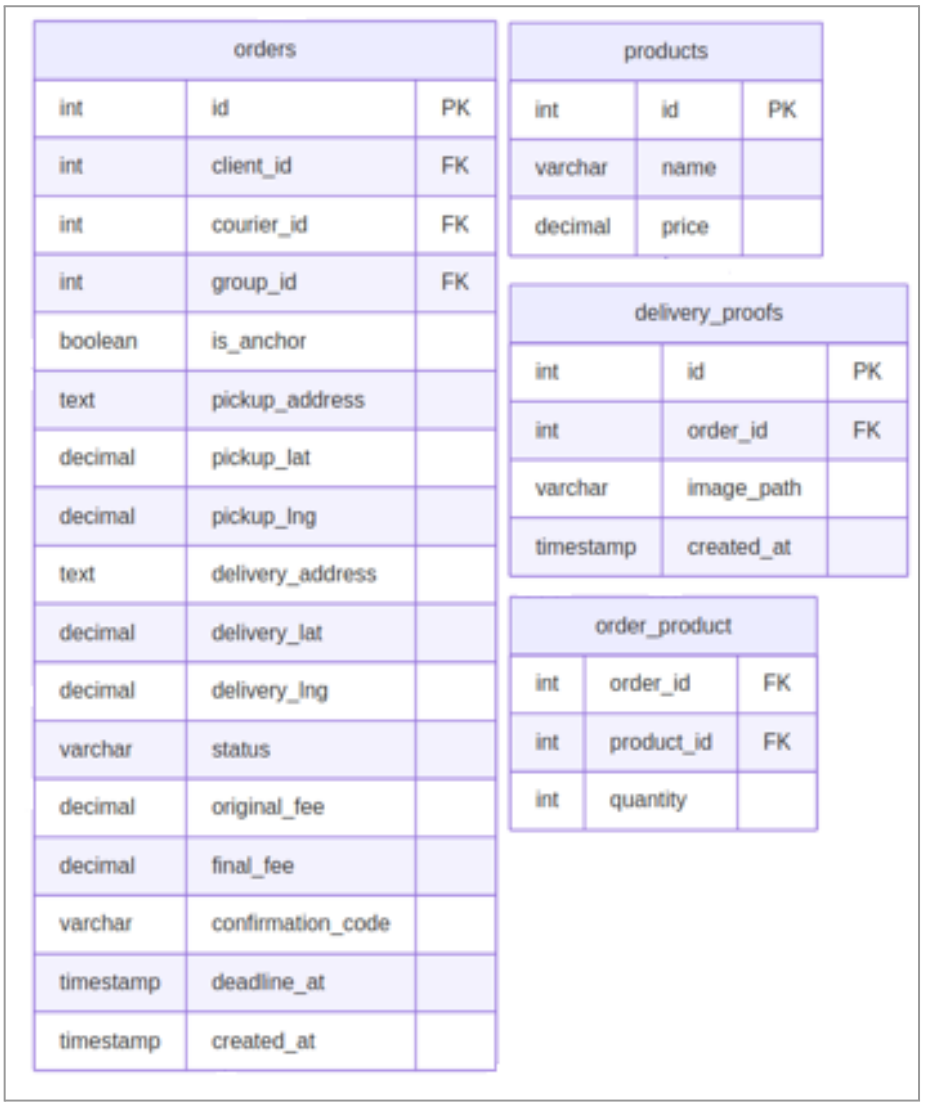
couriers		
int	id	PK
varchar	name	
varchar	email	
varchar	password	
boolean	is_available	
decimal	current_lat	
decimal	current_lng	
decimal	cluster_radius_km	
timestamp	created_at	

Fonte: Autoria própria (2026)

4.4.2. Pedidos e Composição de Carga

O fluxo de demanda (representado na Figura 6) é centralizado na tabela `orders`, que utiliza o conceito de pedido âncora (`is_anchor`). Um pedido âncora é aquele que inicia uma nova rota, enquanto pedidos subsequentes podem ser acoplados a ela para reduzir custos. A relação entre pedidos e itens é resolvida pela tabela associativa `order_product`, permitindo o detalhamento da carga via tabela `products`. Os campos `pickup_lat/lng` e `delivery_lat/lng` são os insumos fundamentais para o cálculo das matrizes de distância.

Figura 6 – Tabelas orders, delivery_proofs, products e order_products

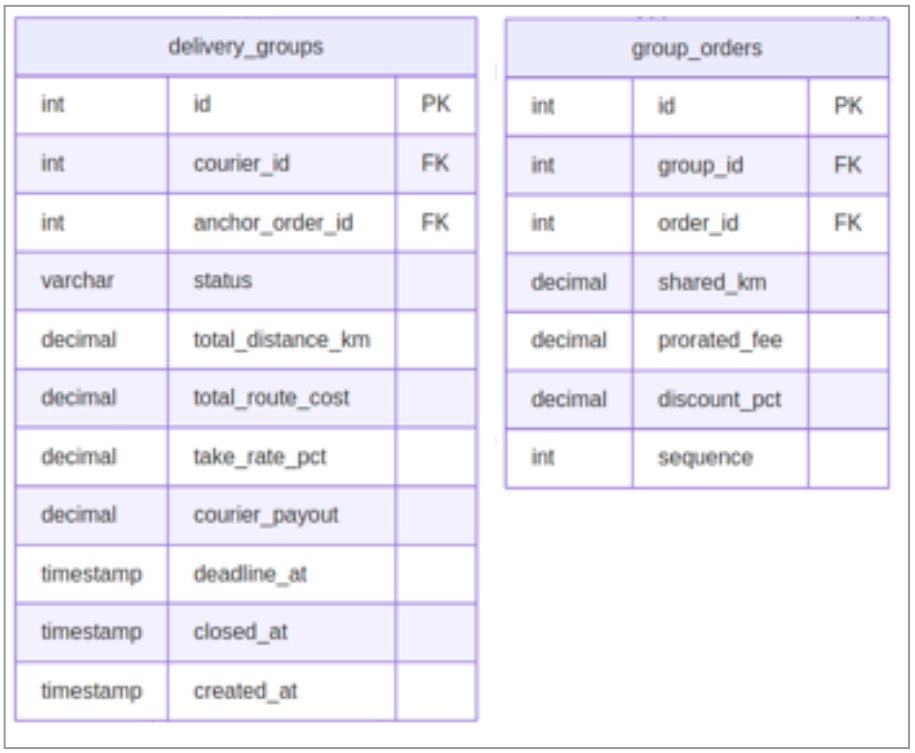


Fonte: Autoria própria (2026)

4.4.3. Inteligência de Agrupamento e Rateio Dinâmico

Este é o núcleo da inovação proposta (representado na Figura 7): o motor de rateio. A tabela `delivery_groups` consolida o grupo de entregas sob responsabilidade de um `courier`, gerindo o custo total (`total_route_cost`) e a comissão da plataforma (`take_rate_pct`). A tabela `group_orders` gerencia a economia colaborativa: o campo `shared_km` registra o trecho percorrido em comum com outros pedidos, servindo de base para o cálculo da `prorated_fee` (taxa rateada). O campo `discount_pct` armazena o benefício financeiro gerado ao cliente final pelo compartilhamento da rota.

Figura 7 – Tabelas delivery_groups e group_orders



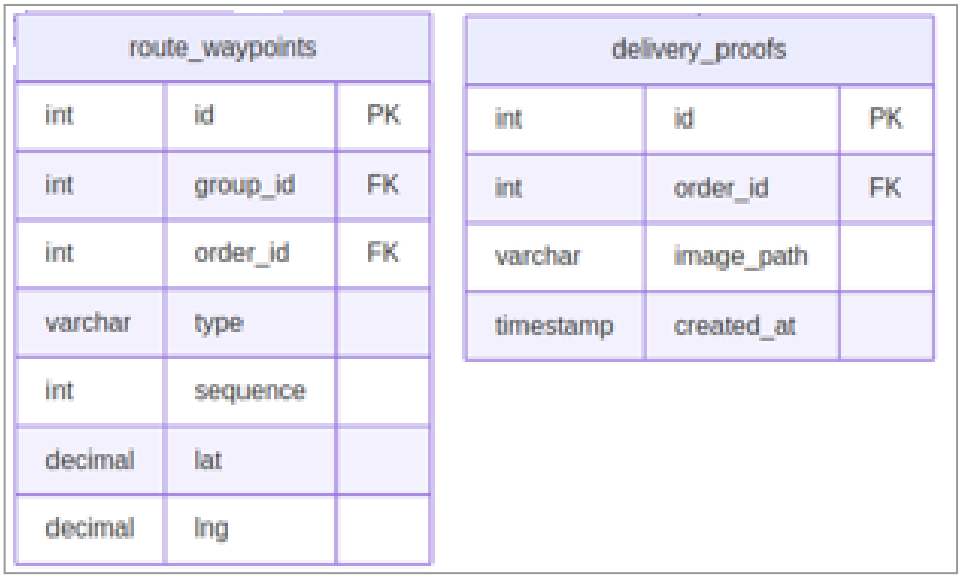
Fonte: Autoria própria (2026)

4.4.4. Execução Logística e Comprovação

A última etapa compreende o suporte à navegação e a segurança jurídica da entrega (representado na Figura 8). A tabela `route_waypoints` decompõe a

jornada em pontos geográficos específicos (`type` identifica se é coleta ou entrega), orientando o entregador através de uma `sequence` lógica de coordenadas. Para garantir a confiabilidade do serviço, a tabela `delivery_proofs` vincula cada pedido finalizado a uma prova digital (`image_path`), permitindo auditorias e auditoria de conformidade conforme o Requisito Funcional RF 18.

Figura 8 – Tabelas route_waypoints e delivery_proofs



Fonte: Autoria própria (2026)

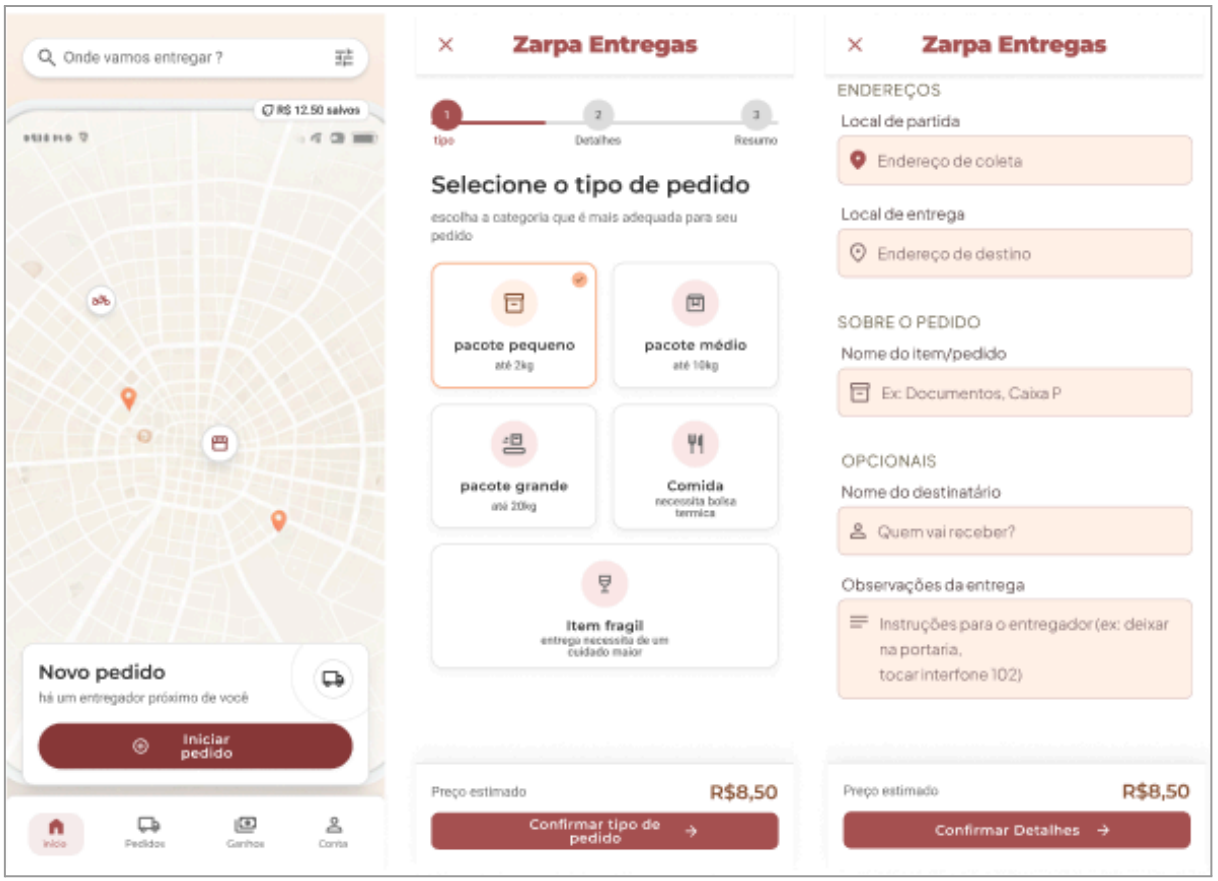
4.5 Prototipação de telas

Para compreender o funcionamento prático da plataforma Zarpa, esta seção apresenta as telas principais estruturadas como um passo a passo sequencial. O objetivo é descrever, de forma simples e direta, o caminho que um pedido percorre dentro do sistema desde a sua criação pelo cliente, até o encerramento financeiro, funcionando como um roteiro de execução das regras de negócio do software.

4.5.1 Criação e Abertura do Pedido (Visão do Cliente)

O fluxo do sistema tem início nesta interface (Figura 9), que serve para o cliente cadastrar a sua necessidade de transporte. A função principal desta tela é coletar o tipo de pedido, endereços de partida (onde o motoboy irá coletar o objeto) e de destino (onde será feita a entrega). Assim que o usuário insere essas localizações, o sistema realiza um cálculo básico de distância para sugerir um preço inicial de frete junto com um custo fixo por tipo de pedido. Após a confirmação, o pedido é registrado no banco de dados com o status de "Aguardando Entregador" e passa a ficar visível na lista geral de pendências da plataforma.

Figura 9 – telas de: início do cliente, seleção do tipo de pedido e detalhes

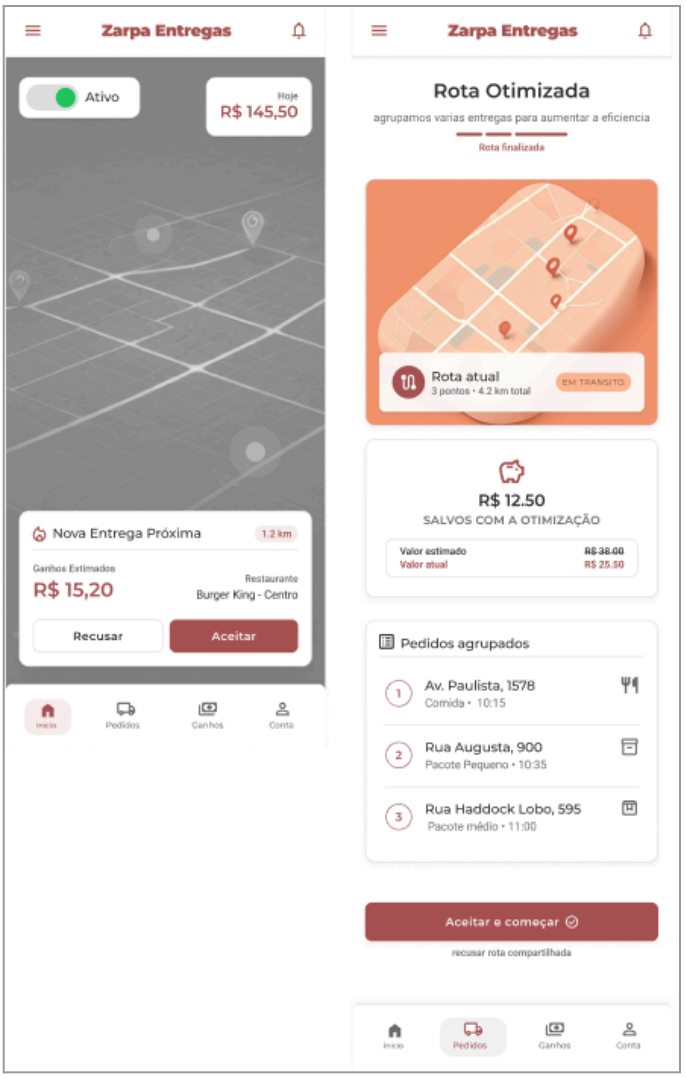


Fonte: Autoria própria (2026)

4.5.2 Lista de Pedidos e Montagem da Rota (Visão do Entregador)

Esta interface (Figura 10) é utilizada exclusivamente pelo entregador autônomo e funciona como um “radar” de oportunidades que exibe um pop-up de entregas próximas da sua localização, após aceitar a primeira entrega ela vai ser exibida na tela de pedidos que vai montar uma lista de pedidos pendentes compatíveis com a rota já selecionada do primeiro pedido, Assim que ele escolhe os pacotes que deseja levar juntos, o sistema aciona o algoritmo do Caixeiro Viajante (TSP) para organizar esses endereços na sequência mais curta possível, gerando uma rota otimizada e calculando o ganho financeiro total que o profissional terá ao realizar a viagem combinada.

Figura 10 – Tela de Seleção de Pedidos Disponíveis e Listagem



Fonte: Autoria própria (2026)

5 CONSIDERAÇÕES FINAIS

Em conclusão, as atividades de intermediação de entregas urbanas de última milha (*last mile*) enfrentam desafios claros de eficiência e custos elevados, impactando diretamente o ganho de entregadores autônomos e o orçamento de pequenos comércios. A proposta da plataforma Zarpa buscou solucionar esse cenário ao permitir que o próprio entregador selecione manualmente pedidos individuais e tenha sua rota sequenciada de forma otimizada pelo algoritmo do Caixeiro Viajante (TSP). Acredita-se que esse modelo devolve a autonomia ao profissional e gera economia para os clientes por meio do rateio dinâmico do frete colaborativo. Como projeto de TCC 1, os objetivos planejados para esta etapa foram integralmente atingidos com a fundamentação teórica, o mapeamento de requisitos e a criação dos protótipos de interface.

Contudo, a continuidade do projeto apresentará desafios práticos na próxima fase, como a dependência da precisão do GPS, a estabilidade de conexões via *WebSockets* para ofertas extras em tempo real e a necessidade de uma base crítica inicial de usuários. Como desdobramento natural, as atividades do TCC 2 compreenderão a codificação completa do software. O cronograma técnico focará no desenvolvimento do *back-end*, na construção das interfaces responsivas para clientes e entregadores, e na integração definitiva com serviços de mapas. Por fim, serão executados testes funcionais com cenários reais para homologar a consistência matemática do cálculo de rateio e validar a eficiência da ordenação de paradas sob condições operacionais urbanas.

REFERÊNCIAS

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

FETTE, I.; MELNIKOV, A. **The WebSocket Protocol**. RFC 6455. 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.

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

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

GOOGLE. **Orientação acadêmica e suporte ao desenvolvimento de sistema para logística urbana**. Ferramenta: Gemini. Versão: 1.5 Pro. Mountain View: Google, 2026. Disponível em: https://gemini.google.com. Acesso em: 18 maio 2026. Prompt: " Você deve agir como um 'Orientador de TCC' experiente, especializado na área de Desenvolvimento de Sistemas para Internet. Sua missão é guiar o aluno na criação de uma plataforma inteligente para intermediação de entregas urbanas, focando na otimização de rotas para motoboys autônomos, agrupamento geográfico de pedidos e redistribuição proporcional de custos de frete."

GOOGLE. **Vehicle Routing Problem**: OR-Tools. [2024]. Disponível em: https://developers.google.com/optimization/routing/vrp. Acesso em: 22 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.

LARAVEL. **Broadcasting**: Laravel Echo and WebSockets. 2024. Disponível em: https://laravel.com/docs/broadcasting. Acesso em: 11 maio 2026.

LARAVEL. **Laravel Reverb**: real-time communication for your Laravel application. 2024. Disponível em: https://reverb.laravel.com/. Acesso em: 11 maio 2026.

LARAVEL. **Laravel**: The PHP Framework for Web Artisans. 2024. Disponível em: https://laravel.com/docs. Acesso em: 27 abr. 2026.

META PLATFORMS. **React Native**: A framework for building native apps using React. 2024. Disponível em: https://reactnative.dev/. Acesso em: 27 abr. 2026.

POSTGIS CONTRIBUTORS. **PostGIS Spatial Database**. 2024. Disponível em: https://postgis.net/. Acesso em: 27 abr. 2026.

REACT NATIVE COMMUNITY. **React Native Maps**. 2024. Disponível em: https://github.com/react-native-maps/react-native-maps. Acesso em: 27 abr. 2026.

SAVELSBERGH, Martin; VAN WOENSEL, Tom. City Logistics: Challenges and Opportunities. **Transportation Science**, v. 50, n. 2, p. 579-590, 2016.

TOTH, Paolo; VIGO, Daniele. **Vehicle Routing: Problems, Methods, and Applications**. 2. ed. Philadelphia: SIAM, 2014.

UNIVERSIDADE FEDERAL DO PARANÁ. Departamento de Engenharia de Produção. **Problemas de Caixeiro-Viajante e Roteirização**. Curitiba: UFPR, [2024]. Material didático.

APÊNDICES

APÊNDICE A – LEVANTAMENTO COMPLETO DE REQUISITOS (MOSCOW)

Esta seção apresenta a totalidade dos requisitos funcionais mapeados para a plataforma Zarpa, categorizados por prioridade para a execução do MVP.

ID	Requisito	Descrição	Prioridade
RF 08	Parametrização de Regras de Negócio	Telas para alterar o raio máximo e taxas.	Could Have
RF 17	Transbordamento para Navegação Externa	Enviar a rota otimizada para Waze ou Google Maps.	Could Have
RF 01	Geração de Rota e Sequenciamento (TSP)	Cálculo do melhor trajeto via API OpenRouteService.	Must Have
RF 04	Agrupamento Dinâmico por Raio	Busca de pedidos pendentes em raio definido da rota.	Must Have
RF 05	Solicitação de Entrega	Interface para o cliente enviar objeto	Must Have
RF 06	Cálculo de Rateio Financeiro	Algoritmo de desconto e pagamento por economia de KM.	Must Have
RF 10	Registro de Ordem de Entrega	Formulário com endereço de coleta e destino.	Must Have
RF 11	Acompanhamento de Fluxo (Status)	Notificações de mudança de estado (Aguardando/Finalizado).	Must Have
RF 15	Mapa de Ofertas de Pedidos Âncora	Visualização dos pedidos que podem iniciar uma jornada.	Must Have
RF 16	Interface de Aceite de Sugestões Extras	Lista para aceitar pacotes adicionais dentro da rota.	Must Have
RF 18	Atualização Manual de Status	Botões de "Coletado" e "Entregue" para o entregador.	Must Have
RF 07	Gestão Básica de Usuários	Interface para aprova, bloquear e listar usuários.	Should Have
RF 02	Sincronização em Tempo Real (Broadcast)	Uso de WebSockets para notificar mudanças de estado.	Should Have
RF 12	Extrato de Frete Colaborativo	Exibição do valor final pago após o desconto.	Should Have
RF 03	Monitoramento de Prazo	Fechamento automático da lista por tempo.	Won't Have
RF 09	Dashboard e Relatórios Logísticos	Gráficos complexos e exportação de dados de economia.	Won't Have
RF 13	Rastreamento Visual Animado	Ver o motoboy se movendo no mapa em tempo real.	Won't Have
RF 14	Opção de Entrega Exclusiva	Impedir o agrupamento do próprio pedido.	Won't Have
RF 19	Histórico Financeiro e de Produtividade	Tela de extrato de ganhos diários do entregador.	Won't Have

APÊNDICE B – MODELAGEM DO BANCO DE DADOS COMPLETA

Figura 11 – Modelagem Do Banco De Dados Completa