UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

GABRIEL FERNANDO SEFFRIN

EXPEDISOFT - SISTEMA PARA A GESTÃO E RASTREABILIDADE NO CARREGAMENTO DE MERCADORIA

GUARAPUAVA

2026

GABRIEL FERNANDO SEFFRIN

EXPEDISOFT - SISTEMA PARA A GESTÃO E RASTREABILIDADE NO CARREGAMENTO DE MERCADORIA

ExpediSoft - System for Cargo Loading Management and Traceability

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Dr. Emerson André Fedechen

GUARAPUAVA

2026

GABRIEL FERNANDO SEFFRIN

EXPEDISOFT - SISTEMA PARA A GESTÃO E RASTREABILIDADE NO CARREGAMENTO DE MERCADORIA

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 30/06/2026

Emerson André Fedechen
Doutor (Orientador)
Universidade Tecnológica Federal do Paraná, Campus Guarapuava

---

Sediane Carmem Lunardi Hernandes
Doutora (Membro da banca avaliadora)
Universidade Tecnológica Federal do Paraná, Campus Guarapuava

---

Roni Fabio Banaszewski
Doutor (Membro da banca avaliadora)
Universidade Tecnológica Federal do Paraná, Campus Guarapuava

GUARAPUAVA

2026

AGRADECIMENTOS

Quando entrei na faculdade, me formar parecia um cenário tão distante, e hoje percebo como esse período passou rapidamente. Recordo com carinho a ansiedade e a empolgação de saber que iria estudar na UTFPR, conhecer novos colegas e construir novas amizades. Certamente foi uma experiência única que, se pudesse, viveria novamente.

Hoje, escrevendo estas linhas, percebo que estou próximo de concluir essa etapa e que definitivamente não sou mais o mesmo. A universidade transforma a vida das pessoas, proporcionando crescimento pessoal e profissional. Agradeço a Deus por todo discernimento, sabedoria e paciência durante essa caminhada. Também gostaria que minha madrinha Adriana estivesse aqui para poder acompanhar o profissional que me tornei.

Quero agradecer à minha família, meu pai Sandro, minha mãe Maura e minha irmã Camila, pelo apoio constante durante toda a minha trajetória acadêmica. Obrigado por todo incentivo, paciência e ajuda nesses anos de estudo. Sem vocês, essa caminhada teria sido muito mais difícil.

Aos meus avós Orlando e Noeli, obrigado por serem parte fundamental da minha vida. Tenho muito orgulho de ser o primeiro neto a concluir a graduação.

À Chelsea Brito, que a vida me trouxe e que se torna a cada dia mais importante na minha caminhada, obrigado por fazer parte dessa trajetória. Seu incentivo, compreensão e apoio foram essenciais durante esses anos.

Aos meus colegas e amigos de faculdade, muitos dos quais levarei para a vida, meu muito obrigado. Vocês foram parte importante dessa etapa, e certamente aprendi muito com cada um de vocês.

Também agradeço ao meu orientador, Prof. Dr. Emerson André Fedechen, por ser meu guia e conduzir este trabalho pelo melhor caminho. Sua orientação, conhecimento e apoio foram essenciais para a construção deste projeto.

Agradeço também a todos os professores da UTFPR. Sem dúvida, vocês tiveram papel fundamental na minha formação, e muitos levarei como referência profissional para minha vida. Estendo meus agradecimentos a todos os colaboradores da universidade, pois cada um contribui para que esse ambiente exista e funcione.

Estas linhas são poucas para mencionar todos que fizeram parte dessa caminhada, mas deixo aqui meu sincero agradecimento a todos que cruzaram meu caminho durante esse período.

RESUMO

O processo de expedição de mercadorias é uma etapa crítica na cadeia logística, na qual falhas operacionais, como o carregamento incorreto e a falta de rastreabilidade, geram prejuízos financeiros significativos e comprometem a credibilidade das empresas corporativas. Diante desse cenário, este trabalho apresenta o desenvolvimento do Expedisoft, um sistema integrado composto por uma plataforma administrativa *web* de gestão e um aplicativo móvel voltado ao controle operacional e à auditoria do carregamento de cargas. A solução proposta atua de forma integrada aos sistemas centrais da empresa, permitindo a importação automatizada de pedidos e o agendamento estratégico de cargas associadas a veículos, motoristas e operadores específicos. Na ponta de execução, o operador utiliza o dispositivo móvel para realizar um *checklist* digital baseado na leitura de códigos de identificação por volume, validando os itens em tempo real e bloqueando falhas humanas por duplicidade ou embarque cruzado de mercadorias. Em caso de inconformidades na carga, a ferramenta exige o preenchimento de justificativas e gerencia a captura de evidências fotográficas, que são enviadas de forma assíncrona para armazenamento remoto em nuvem para garantir a integridade dos dados e a fluidez do trabalho em campo. Como resultados, o ecossistema foi integralmente construído e validado por meio de testes automatizados de funcionalidades. O sistema demonstrou-se apto a proporcionar maior confiabilidade operacional, eliminação do uso de planilhas e relatórios em papel, aumento na velocidade de liberação de veículos e suporte abrangente à tomada de decisões gerenciais através de indicadores visuais e telemetria em tempo real do desempenho do armazém.

Palavras-chave: expedição de mercadorias; rastreabilidade logística; integração de sistemas; conferência digital; sistema web e mobile.

ABSTRACT

The goods dispatch process is a critical stage in the logistics chain, where operational failures such as incorrect loading and a lack of traceability generate significant financial losses and compromise corporate credibility. Against this backdrop, this work presents the development of Expedisoft, an integrated system comprising a web-based administrative management platform and a mobile application dedicated to operational control and cargo loading auditing. The proposed solution operates seamlessly with the company's core systems, enabling automated order importing and the strategic scheduling of loads associated with specific vehicles, drivers, and operators. At the execution end, the operator utilizes a mobile device to perform a digital checklist based on scanning unique identification codes per volume, validating items in real time and instantly blocking human errors caused by duplication or mismatched cargo loading. In the event of cargo non-conformities, the tool enforces the submission of justifications and manages the capture of photographic evidence, which is asynchronously transmitted to remote cloud storage to ensure data integrity and seamless field workflow continuity. As a result, the ecosystem was fully built and validated through automated feature testing. The system proved capable of providing greater operational reliability, eliminating the use of spreadsheets and paper reports, accelerating vehicle dispatch speed, and offering comprehensive support for managerial decision-making through visual indicators and real-time warehouse performance telemetry.

Keywords: cargo expedition; logistics management; traceability system; barcode scanning; system integration.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTAGEM DE CÓDIGOS FONTE

LISTA DE ABREVIATURAS E SIGLAS

Siglas

API	*Application Programming Interface*
CI	*Continuous Integration*
DDD	*Domain-Driven Design*
ERP	*Enterprise Resource Planning*
HMR	*Hot Module Replacement*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
MVC	*Model-View-Controller*
MVP	*Minimum Viable Product*
ORM	*Object-Relational Mapper*
QR Code	*Quick Response Code*
REST	*Representational State Transfer*
SP	*Story Points*
SQL	*Structured Query Language*
TDD	*Test-Driven Development*
UI	*User Interface*
UUID	*Universally Unique Identifier*
UX	*User Experience*

SUMÁRIO

1 INTRODUÇÃO

Longe de ser apenas a etapa final de entrega, o processo de expedição é um pilar estratégico para o sucesso e a escalabilidade de um negócio. Para empresas de todos os portes e mercados, de operações locais a grandes exportadoras, a eficiência e a confiabilidade deste processo são determinantes para a competitividade e a satisfação do cliente. Conforme a visão de Ballou (2006), o nível de serviço prestado ao cliente é o resultado direto de um bom desempenho logístico, funcionando como um dos principais diferenciais no mercado.

Operacionalmente, o processo de expedição compreende uma sequência de atividades críticas que finalizam o ciclo do pedido dentro do armazém, como a conferência de itens, a embalagem (packing), a geração da documentação fiscal e de transporte, e o correto carregamento do veículo. Essa etapa é de alta responsabilidade, pois, como defendem Bowersox, Closs e Cooper (2013), é o último ponto de controle para assegurar os atributos do "pedido perfeito": a entrega completa, no prazo, sem avarias e com a documentação correta. Desta forma, a expedição funciona como a garantia final da qualidade percebida pelo cliente.

Contudo, a criticidade dessa etapa a torna um ponto de vulnerabilidade para a operação. Falhas nesta etapa podem gerar diversos problemas, como atrasos nas entregas, inconsistências nas documentações dos pedidos, problemas em empacotamentos, avarias no carregamento dos produtos e até mesmo itens errados no pedido do cliente. Essas falhas impactam diretamente o desempenho logístico, gerando retrabalho, aumento de custos operacionais com devoluções e reenvios, e, de forma mais danosa, comprometem a satisfação e a confiança do cliente, pondo em risco a reputação da empresa no mercado. Essa cadeia de prejuízos vai na contramão do que Bowersox, Closs e Cooper (2013) defendem como um dos objetivos da logística integrada: a busca pelo menor custo total para atender às necessidades do cliente.

A concepção deste projeto foi inspirada na observação de desafios operacionais em uma empresa de grande porte do setor de exportação. Embora o problema tenha sido identificado em um cenário específico, ele pode ocorrer em qualquer organização, de modo que a solução proposta aqui visa a adaptabilidade a diferentes contextos empresariais.

Todos estes problemas são acentuados em um cenário observado como objeto de estudo: operações de expedição de grande porte, especialmente no setor de exportação de manufaturados, cujo modelo de negócio é majoritariamente voltado para mercados de alta exigência, como Estados Unidos e Europa. Em operações dessa natureza, o processo de expedição frequentemente ocorre como a última etapa antes do faturamento, tornando qualquer alteração posterior extremamente complexa e custosa.

Nesse tipo de operação é comum o desafio crítico e recorrente de falhas na etapa de carregamento: ocorre com frequência o esquecimento de caixas pertencentes ao pedido e/ou carregamento de pacotes que não pertencem ao pedido, fazendo com que a remessa seja enviada inconsistente ao cliente. Esta situação específica mobiliza diversas áreas para sua correção, impactando prazos, metas e, principalmente, a credibilidade junto aos clientes internacionais.

Nesse cenário, torna-se essencial investir em ferramentas que ofereçam maior controle e rastreabilidade para esta fase crítica da operação, auxiliando no carregamento e mitigando erros operacionais. Desta forma, a presente proposta visa o desenvolvimento de uma solução, por meio da análise, planejamento e desenvolvimento de um sistema web e mobile integrados.

Entre os principais desafios do projeto, destaca-se a definição de uma arquitetura flexível, capaz de se comunicar com sistemas legados. Um ponto crítico é o tratamento das imagens, cuja integração deve visar a estabilidade da aplicação. Neste trabalho, um sistema robusto é caracterizado por sua confiabilidade (operar de forma consistente), tolerância a erros (lidar com inconsistências de dados) e manutenibilidade (facilidade para evoluir e corrigir). Nesse contexto, a decisão de utilizar armazenamento por referência é fundamental, pois, além de garantir a escalabilidade, contribui diretamente para esses pilares da estabilidade do sistema.

Em síntese, este projeto visa o desenvolvimento de uma ferramenta voltada à confiabilidade operacional e à escalabilidade para apoiar os colaboradores no setor de expedição, mas podendo ser adaptado a outras empresas.

1.1 Objetivos

Nesta seção serão apresentados o objetivo geral e os objetivos específicos do presente trabalho.

1.1.1 Objetivo Geral

Desenvolver um sistema para o controle e rastreabilidade do processo de expedição de mercadorias.

1.1.2 Objetivos Específicos

Os objetivos específicos foram divididos em duas subseções, interface Web e interface Mobile.

Interface Mobile

- Desenvolver um módulo para identificação das mercadorias, auxiliando na verificação da caixa correta para o carregamento;
- Implementar um *checklist* de carregamento, possibilitando marcar as caixas já carregadas, auxiliando que nenhum item seja esquecido ou adicionado indevidamente;
- Adicionar a funcionalidade de inclusão de observações, possibilitando o registro de comentários e justificativas após o carregamento;
- Desenvolver a funcionalidade de *upload* de fotos via aplicativo para conferência.

Interface Web

- Desenvolver um sistema para agendamento de carregamentos e alocação do funcionário responsável, facilitando a organização das operações logísticas;
- Criar um recurso para registro de responsabilidade, associando cada carregamento ao colaborador responsável, promovendo rastreabilidade e responsabilização;
- Criar a funcionalidade de conferência de carregamentos anteriores, com acesso às justificativas, fotos e observações previamente registradas.

2 MATERIAIS E MÉTODOS

A seguir, um descritivo de como o trabalho foi conduzido e quais tecnologias foram utilizadas.

2.1 Materiais

Para o desenvolvimento do projeto, foram utilizadas as seguintes tecnologias, ferramentas e ambientes, cujas escolhas são justificadas a seguir.

- ***Backend*: *PHP* 8+ e Laravel 10+** (LARAVEL, 2025):

  O Laravel foi escolhido por ser um *framework*[1] *PHP* robusto e com um ecossistema que acelera o desenvolvimento de *Application Programming Interface* (API)s. Sua arquitetura *Model-View-Controller* (MVC), o *Object-Relational Mapper* (ORM) *Eloquent* e as ferramentas integradas permitem a construção de um *backend*[2] seguro e de manutenção descomplicada.

- ***Frontend* (Web): *React* 18+ e *Vite*** (META, 2025) (VITE, 2025):

  O *React* é consistentemente classificado como a biblioteca *JavaScript* mais utilizada pela comunidade de desenvolvedores para a criação de interfaces de usuário. Sua abordagem baseada em componentes permite a criação de *User Interface* (UI)s reativas e escaláveis. A adoção do *TypeScript* é justificada por sua capacidade de adicionar tipagem estática ao *JavaScript*, auxiliando no desenvolvimento de um código mais seguro. O *Vite* é utilizado como ferramenta de *build*[3], pois sua arquitetura serve módulos ES nativos durante o desenvolvimento, resultando em um *Hot Module Replacement* (HMR) e inicialização de servidor mais rápidos que ferramentas como o *Webpack*.

- ***Mobile*: *React Native* e *Expo*** (REACT, 2025):

  Para manter a consistência tecnológica e otimizar o tempo de desenvolvimento, o *React Native* foi selecionado. O uso do *Expo* simplifica o processo de compilação, distribuição e acesso a recursos nativos, mantendo a consistência entre as plataformas Android e iOS.

- **Componentes de UI (Web): *shadcn/ui*** (SHADCN, 2025):

[1] *Framework*: Estrutura de suporte definida na qual um outro projeto de software pode ser organizado e desenvolvido.

[2] *Backend*: Camada de acesso a dados e regras de negócio da aplicação, não visível diretamente ao usuário final.

[3] *Build*: Processo de converter código-fonte em artefatos de software executáveis ou otimizados para produção.

Para a construção da UI, optou-se pelo *shadcn/ui*. Diferente das bibliotecas de UI tradicionais, esta é uma coleção de componentes não-estilizados (baseados em *Radix UI*) que são copiados para a base de código do projeto. Conforme sua documentação oficial, essa abordagem possibilita total propriedade e controle sobre o código do componente, facilitando a personalização e a acessibilidade sem adicionar dependências externas ao projeto.

- **Banco de Dados: *PostgreSQL* 15+** (GROUP, 2025):

  O *PostgreSQL* é um sistema de gerenciamento de banco de dados relacional de código aberto (*open source*[4]), reconhecido por sua conformidade com os padrões *Structured Query Language* (SQL).

- **Ambiente de Contêineres: *Docker*** (DOCKER, 2025):

  O *Docker* foi utilizado para a criação de ambientes padronizados e isolados através de contêineres. Conforme a literatura de *DevOps*, a conteinerização resolve o problema de inconsistência entre ambientes de desenvolvimento e produção. Ela garante que a aplicação e todas as suas dependências (*PHP*, *PostgreSQL* etc.) sejam empacotadas juntas, possibilitando a portabilidade e simplificando o processo de implantação (*deploy*[5]) (VASYUKOV A. PETROV, 2018).

- **Prototipação: *Figma*** (FIGMA, 2025):

  O *Figma* é uma ferramenta de *design* de interfaces UI e experiência do usuário *User Experience* (UX), utilizada para a criação de protótipos interativos nos testes de usabilidade antes do início da codificação.

- **Controle de Versão: *Git*** (GIT, 2025):

  Justificativa: Ferramenta para controle e versionamento de código, fundamental para o gerenciamento do projeto, combinada com o *GitHub*.

- **GitHub Copilot: *Copilot*** (GitHub Inc., 2025):

  Para auxilio no desenvolvimento de código, utilizou-se a ferramenta de assistência por inteligência artificial *GitHub Copilot* para revisão de código.

- **IDE: *WebStorm*, *PHPStorm* e *DataGrip***:

  Por conta da familiaridade e recursos avançados, serão utilizadas as IDEs da *JetBrains* como ferramentas de desenvolvimento.

[4] *Open source*: Software cujo código-fonte é disponibilizado publicamente para uso, modificação e distribuição.

[5] *Deploy*: Ato de implantar ou disponibilizar uma aplicação para uso em um ambiente específico, como produção ou testes.

- **Documentação API: *Swagger***:

  A escolha pelo *Swagger* é importante, pois o projeto possui um *backend* que serve a dois *frontends* distintos (*Web e Mobile*). O *Swagger* gera uma documentação interativa que permite a qualquer desenvolvedor visualizar e testar os *endpoints*[6] diretamente pelo navegador.

2.2 Métodos

O desenvolvimento do sistema seguiu uma abordagem aplicada, fundamentada em princípios da Engenharia de Software e metodologias ágeis, com foco na entrega incremental de valor e na validação contínua das funcionalidades implementadas. As etapas metodológicas são descritas a seguir.

- **Levantamento de Requisitos**

  O levantamento de requisitos do sistema Expedisoft foi fundamentado na experiência prévia do autor no setor de expedição de uma empresa exportadora de grande porte, complementada por entrevistas com usuários-chave e observação direta das rotinas operacionais. Essa abordagem é comumente utilizada em projetos aplicados, conforme destaca Sommervile (2011), possibilitando que o pesquisador derive requisitos a partir da análise de sua própria vivência e conhecimento técnico do domínio.

  A observação das práticas cotidianas permitiu identificar as principais dificuldades enfrentadas no processo de carregamento, consolidando a necessidade de uma solução que fosse, ao mesmo tempo, descentralizada e flexível a diferentes cenários. Destacam-se como pontos críticos:

  - A fragilidade da conferência manual dos volumes destinados ao carregamento;
  - A dificuldade na coleta, organização e vinculação de evidências (registros fotográficos e justificativas de divergências);
  - A ausência de um histórico estruturado e de fácil consulta sobre carregamentos realizados anteriormente.

  A partir desse diagnóstico, os requisitos foram documentados, categorizados e priorizados de acordo com seu impacto no processo de expedição. Os requisitos funcionais descrevem as funcionalidades necessárias para apoiar a operação, como a leitura de códigos, validação de volumes e registro de divergências.

[6] *Endpoint*: Um ponto final de comunicação ou URL específica em uma API onde um serviço pode ser acessado.

Já os requisitos não funcionais estabelecem os critérios de qualidade e restrições técnicas, incluindo segurança da informação, controle de acesso, rastreabilidade, desempenho e usabilidade em dispositivos móveis. A padronização da comunicação por meio de uma API foi considerada de forma transversal, de modo que a arquitetura do sistema suporte os requisitos de integração e escalabilidade necessários para o ambiente corporativo.

- **Priorização dos Requisitos**

  Após definidos, os requisitos foram organizados segundo a técnica *MoSCoW*, que classifica as funcionalidades em "*Must Have*", "*Should Have*", "*Could Have*" e "*Won't Have*" Clegg D.; Barker (1994). Essa técnica permitiu a construção de um *Minimum Viable Product* (MVP), concentrando esforços nas funcionalidades essenciais para o controle e rastreabilidade do carregamento.

  ***Must Have*:** Requisitos críticos e indispensáveis para o funcionamento do sistema, detalhados na Tabela 1.

  ***Should Have*:** Requisitos importantes que agregam valor significativo ao produto, detalhados na Tabela 2.

  ***Could Have*:** Requisitos desejáveis, detalhados na Tabela 3.

  ***Won't Have*:** Requisitos que estão fora do escopo do sistema, detalhados na Tabela 4.

ID	Categoria	Requisito	Descrição
R1	Geral	API de Integração (*Backend*)	Desenvolvimento de *endpoints Representational State Transfer* (REST) (método *POST*) para receber dados de ordens de carregamento.
R2	Geral	Autenticação e Autorização	Sistema seguro de *login* para diferenciar perfis de acesso (Gestor *Web* e Operador).
R3	App *Mobile*	Leitura de	Funcionalidade para leitura de *Quick Response Code* (QR Code) utilizando a câmera do dispositivo *mobile*.
R4	App *Mobile*	Validação em Tempo Real	Lógica no aplicativo que verifica se o volume lido corresponde à ordem de carregamento ativa, mitigando erros.
R5	Plataforma *Web*	Gestão de Ordens	Interface para visualizar as ordens de carregamento importadas, acompanhar seu status e realizar o agendamento da ordem.
R6	App *Mobile*	Documentação Visual	Funcionalidade para capturar e/ou enviar fotos da carga como comprovante visual.
R7	App *Mobile*	Registro de Divergências	Fluxo que obriga o operador a inserir uma justificativa em texto ao finalizar uma carga com divergências.
R8	Plataforma *Web*	Detalhamento de Carga	Visualização detalhada do histórico de conferência de uma ordem, incluindo quem realizou a conferência, horário, fotos e justificativa.

Tabela 1 – Funcionalidades *Must Have*

ID	Categoria	Requisito	Descrição
R9	Plataforma *Web*	*Dashboard*	Painel inicial com *cards* e indicadores/gráficos básicos de desempenho (ex: quantidade de carregamentos programados, tempo médio de carregamento).

Tabela 2 – Funcionalidades *Should Have*

ID	Categoria	Requisito	Descrição
R10	Geral	Notificações	Alertas por *e-mail* ou *push notification* para gestores quando uma carga com divergência for finalizada.
R11	Plataforma *Web*	Integração	Página dedicada ao monitoramento de integração dos dados no sistema.

Tabela 3 – Funcionalidades *Could Have*

ID	Categoria	Requisito	Descrição
R12	Geral	Integração Ativa	Sistema buscando dados no ERP.
R13	Geral	Funcionalidade WMS	Controle de estoque e endereçamento.
R14	Geral	Documentos	Emissão de documentos fiscais (NF-e).

Tabela 4 – Funcionalidades *Won't Have*

- **Prototipação de Interfaces**

  As interfaces do sistema foram prototipadas no *Figma*, de modo a avaliar o fluxo de navegação e a experiência do usuário antes da codificação. Essa etapa permitiu antecipar problemas de usabilidade e ajustar a interface conforme necessidade.

- **Metodologia de Desenvolvimento**

  Para o acompanhamento e planejamento do desenvolvimento do sistema, foi adotada uma abordagem incremental, inspirada na metodologia *Scrum*, mesmo tratando-se de um projeto desenvolvido por um único autor. As funcionalidades foram organizadas em pequenas entregas, favorecendo a evolução contínua e validação progressiva do sistema.

  Como técnica de estimativa e controle do esforço, foram utilizados *Story Points* (SP), que representam uma medida relativa de complexidade, esforço e risco de cada funcionalidade a ser implementada. As estimativas foram realizadas com base em comparação entre tarefas, considerando critérios como volume de código, regras de negócio envolvidas e dependências técnicas.

  O registro das estimativas e do progresso foi feito por meio de planilhas, possibilitando o acompanhamento da quantidade de SP planejados e concluídos em cada período de trabalho. Essa prática permitiu analisar a evolução do desenvolvimento ao longo do tempo, fornecendo dados objetivos para avaliação do processo e apoio à reflexão sobre a execução do projeto.

- **Modelagem e Arquitetura**

  A modelagem do sistema foi orientada pelos princípios do *Domain-Driven Design* (DDD), buscando representar o domínio da expedição e manter o código coeso e de fácil manutenção. Essa abordagem favoreceu a separação de responsabilidades e a comunicação clara entre as camadas do sistema (EVANS, 2004).

- **Testes e Garantia de Qualidade**

  Foram aplicadas técnicas de *Test-Driven Development* (TDD) nos módulos principais do sistema, contribuindo para que cada funcionalidade atenda aos critérios definidos e reduzindo o risco de regressões Beck (2010).

- **Gerenciamento do Projeto**

  A implantação foi realizada por meio de contêineres *Docker*, que favorecem a padronização do ambiente de execução e a portabilidade do sistema entre desenvolvimento e produção. O controle de versão foi feito com *Git* e *GitHub*, utilizando o fluxo *Git Flow* e *Conventional Commits* aliado a práticas de *Continuous Integration* (CI), em conformidade com os princípios de *DevOps* KIM Gene; HUMBLE (2016).

- **Integração entre Sistema e ERP Cliente**

  Para que a integração entre o sistema desenvolvido e os diferentes sistemas ERP dos clientes ocorra de maneira flexível e padronizada, foram elaborados três modelos (ordens para carregamento, usuários e docas) de integração baseados em *JavaScript Object Notation* (JSON), permitindo que o sistema receba dados estruturados independente da tecnologia utilizada pelo cliente.

  Essa abordagem visa garantir compatibilidade e simplicidade na comunicação entre sistemas, utilizando o protocolo e o método *POST* em *endpoints* REST disponibilizados pela API do *backend*. Dessa forma, os dados referentes às ordens e aos usuários podem ser transmitidos diretamente pelo ERP do cliente para o sistema, possibilitando a integridade e atualização das informações em tempo real.

  O formato JSON foi escolhido por ser amplamente aceito e de fácil manipulação, além de permitir uma estrutura clara para a transmissão de dados.

  Importante salientar que todas as integrações entre os sistemas devem ter uma camada de segurança via *token* de autenticação, com o objetivo de estabelecer uma comunicação segura.

- **Banco de Dados**

  A modelagem do banco de dados foi estruturada com o objetivo de representar, de forma organizada e fiel, todos os elementos envolvidos no processo de expedição.

  O banco foi projetado em *PostgreSQL*, utilizando identificadores do tipo *Universally Unique Identifier* (UUID). As entidades principais foram agrupadas em quatro categorias funcionais: estrutura organizacional, entidades externas, entidades operacionais e entidades de auditoria e rastreabilidade.

3 ANÁLISE E PROJETO DO SISTEMA

Este capítulo apresenta as decisões de análise e projeto que nortearam a construção do Expedisoft. São descritos: a visão geral do sistema e seu fluxo operacional, a metodologia de desenvolvimento adotada, o levantamento e a priorização dos requisitos, a arquitetura definida, a modelagem do banco de dados e os contratos de integração com sistemas ERP externos.

3.1 Visão Geral do Sistema

O Expedisoft foi idealizado para funcionar como um sistema integrado que digitaliza e controla o processo de carregamento de pedidos, substituindo a conferência manual por um fluxo de trabalho digital e rastreável. A arquitetura do sistema é estruturada em duas interfaces interdependentes: uma interface *web* para gestão e uma interface *mobile* para execução operacional.

O fluxo de trabalho se inicia na interface *web*, onde um gestor logístico realiza o agendamento de um novo carregamento. Nessa etapa, ele associa um pedido (previamente importado do sistema ERP do cliente) a um operador específico e define detalhes como a doca de carregamento e o veículo. A Figura 1 exemplifica o fluxo do sistema.

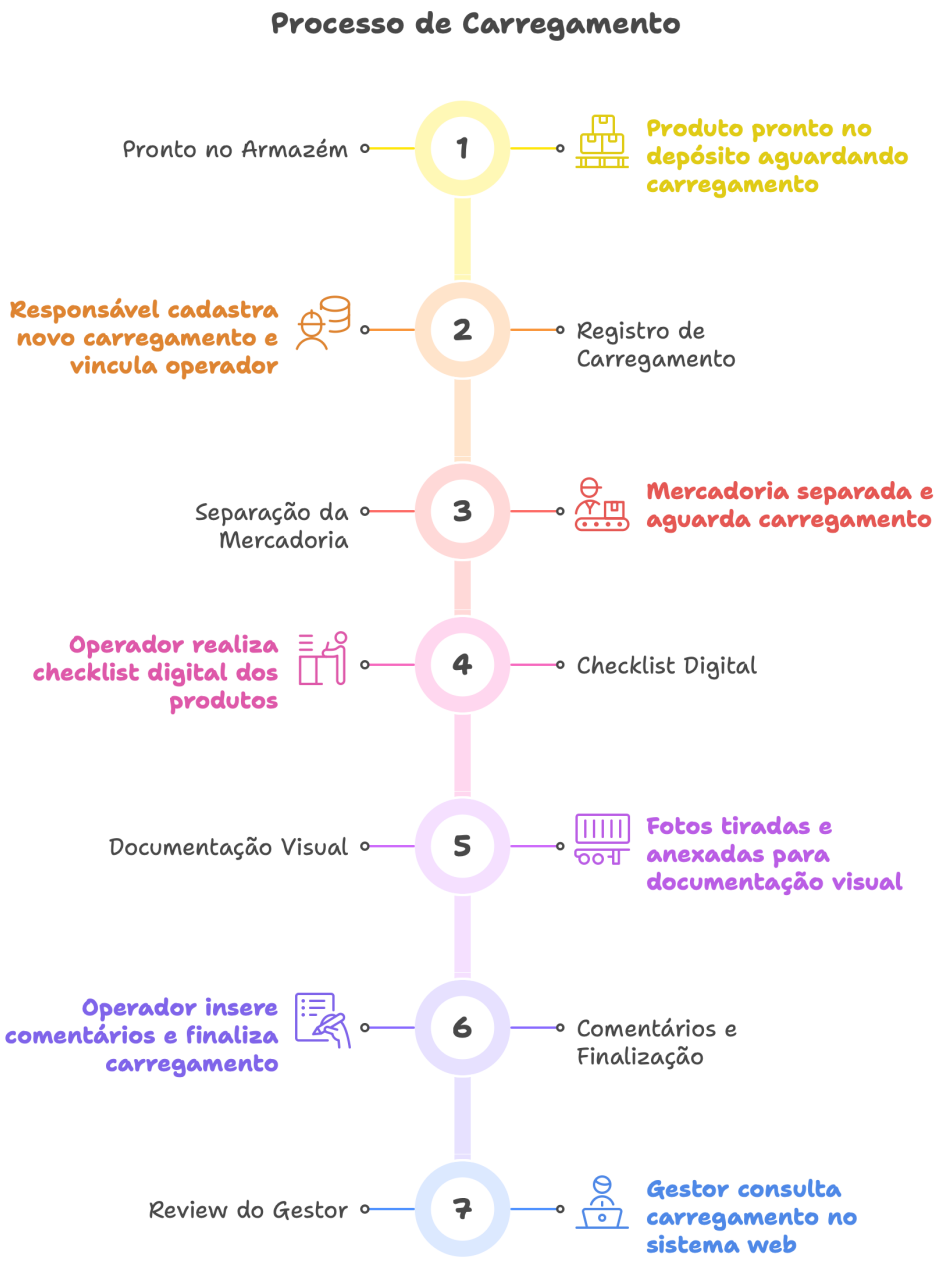

Figura 1 – Exemplo do fluxo do sistema.

Com a tarefa atribuída, o operador utiliza o aplicativo *mobile* para visualizar o carregamento em sua lista de tarefas. Ao iniciar o processo, o aplicativo exibe um *checklist* digital de todos os volumes que devem ser carregados. Para cada caixa, o operador pode utilizar a câmera do dispositivo (celular ou coletor Android) para realizar a leitura de um QR Code, ou realizar manualmente a conferência no dispositivo. O sistema valida a informação em tempo real, atualizando o *status* do item no *checklist* e emitindo *feedbacks* sonoros e visuais em caso de leitura de um item incorreto ou duplicado. O QR Code deve ser gerado pelo ERP do cliente e afixado à caixa para identificação.

Em caso de finalização com divergências (caixas faltando), o aplicativo exige uma justificativa obrigatória, fazendo com que o operador registre os motivos da divergência identificada pelo sistema; do contrário, a observação é opcional. Também é possível capturar e anexar fotos do carregamento, que servirão como evidência visual da integridade da carga.

Após a finalização, todas as informações são sincronizadas e ficam disponíveis no painel de histórico da interface *web*. O gestor pode consultar o *status* de qualquer carregamento, auditar os itens conferidos, visualizar as fotos e ler as observações do operador, tendo uma visão completa e centralizada do processo.

O sistema disponibiliza ainda uma central de monitoramento com todos os *status* dos carregamentos em andamento, indicadores como a quantidade de carregamentos realizados *versus* programados e o desempenho dos operadores. Por fim, uma página de relatórios exibe gráficos de eficiência e tempo de carregamento, com filtros por período, operador e doca.

3.2 Metodologia de Desenvolvimento

O desenvolvimento do sistema fundamentou-se na metodologia ágil *Scrum*, adaptada para a realidade de um projeto individual. A gestão do escopo e do tempo foi conduzida por meio do controle de esforço via SP, o que permitiu o mapeamento sistemático das funcionalidades, a mitigação de riscos técnicos e a organização de iterações com níveis de complexidade sustentáveis.

3.2.1 Estimativa de Esforço e *User Stories*

Por tratar-se de um projeto individual, que diverge do padrão de equipes encontrado no *Scrum*, os pesos de esforço passaram por uma revisão, conforme detalhado na Tabela 5. Para a definição e o refinamento dos requisitos, utilizou-se a técnica de Histórias de Usuário (*User Stories*), estruturadas sob o padrão:

- **Como:** [papel do usuário]
- **Quero:** [ação ou funcionalidade]

- **Para:** [valor de negócio ou objetivo]

O conjunto dessas definições compõe o *Product Backlog* dividido nas tabelas: Tabela 7, Tabela 8, Tabela 9, Tabela 10, Tabela 11, Tabela 12, Tabela 13, Tabela 14, Tabela 15, que totaliza a pontuação de esforço entregue ao longo do projeto.

3.2.2 Ciclos de Entrega e Velocidade

O cronograma de execução foi segmentado em 9 *sprints* com periodicidade quinzenal (duas semanas por ciclo). A dedicação diária ao desenvolvimento foi estabelecida em aproximadamente duas horas, com a quantidade de pontos por *sprint* concentrada em torno de 12 a 15 pontos.

Considerando que o tempo cronológico em projetos de *software* apresenta variáveis de incerteza, a métrica principal de desempenho adotada foi a velocidade, calculada a partir da soma dos pontos entregues por *sprint*, conforme a coluna “Entrega” da Tabela 6.

Estruturar o desenvolvimento em *sprints* foi fundamental para reforçar a percepção de progresso e mitigar a ansiedade. Esse modelo evitou a sensação de atraso e eliminou a incerteza sobre as etapas seguintes, fatores relevantes dado o nível de complexidade do sistema, cujo fluxo de desenvolvimento poderia se tornar confuso sem o devido direcionamento.

Story Point	**Significado Prático**
1	Muito simples, trivial.
2	Simples.
3	Médio.
5	Complexo.
8	Muito complexo / incerto.

Tabela 5 – Peso para cada Story Point

Sprint	Início	Fim	Planj.	Entreg.	Dif.	Observações
Sprint 1	05/01/26	13/01/26	13	8	-5	Fim de ano e festas atrasaram o desenvolvimento.
Sprint 2	26/01/26	13/02/26	11	6	-5	Alta demanda profissional comprometeu a continuidade.
Sprint 3	16/02/26	25/02/26	10	10	0	Atividades realizadas conforme planejado.
Sprint 4	02/03/26	13/03/26	11	8	-3	Dificuldades na refatoração do código de integração com ERP.
Sprint 5	17/03/26	25/03/26	13	10	-3	Atividades extracurriculares comprometeram os últimos dias.
Sprint 6	30/03/26	04/04/26	14	14	0	Maior tempo disponível devido ao feriado de Páscoa.
Sprint 7	04/04/26	11/04/26	16	16	0	Desenvolvidas as interfaces *web* e *mobile*, além da funcionalidade de início/finalização do carregamento e da API de conferência.
Sprint 8	16/04/26	20/04/26	15	15	0	Desempenho favorecido pela reutilização de estruturas de ciclos anteriores.
Sprint 9	22/04/26	01/05/26	13	13	0	Implementação do envio de fotos e configuração da página inicial concluídas, finalizando o escopo do TCC.

Tabela 6 – Cronograma de Sprints e Entrega de Story Points (SP)

ID	Categoria	User Story	SP	Sprint
US01	Infra	Como gestor, quero autenticar-me no sistema web para acessar funcionalidades restritas.	3	1
US13	Infra	Como sistema, quero configurar o ambiente inicial (backend, frontend e mobile).	2	1
US06	Integração	Como sistema, quero armazenar ordens e itens no banco para persistência dos dados.	2	1
US11	Infra	Como gestor, quero encerrar sessão com segurança (Logout).	1	1

Tabela 7 – Product Backlog: Sprint 1

ID	Categoria	User Story	SP	Sprint
US03	Integração	Como sistema, quero expor uma API REST para receber dados do ERP.	3	2
US04	Integração	Como sistema, quero validar os dados recebidos via API para garantir integridade.	2	2
US08	API	Como sistema, quero padronizar respostas da API para facilitar consumo.	1	2

Tabela 8 – Product Backlog: Sprint 2

ID	Categoria	User Story	SP	Sprint
US09	API	Como desenvolvedor, quero documentar *endpoints* da API (Swagger/OpenAPI).	3	3
US10	Obs.	Como sistema, quero registrar *logs* básicos para rastrear erros.	2	3
US15	Integração	Como sistema, quero disponibilizar API para integração de usuários.	2	3
US16	API	Como sistema, quero disponibilizar *endpoints* para consulta de ordens.	3	3

Tabela 9 – Product Backlog: Sprint 3

ID	Categoria	User Story	SP	Sprint
US05	Web	Como gestor, quero visualizar uma lista de ordens importadas.	2	4
US14	Integração	Como sistema, quero autenticar integrações via TOKEN.	3	4
US21	Integração	Como sistema, quero gerenciar integrações em fila (processamento assíncrono).	3	4

Tabela 10 – Product Backlog: Sprint 4

ID	Categoria	User Story	SP	Sprint
US17	Web	Como gestor, quero agendar ordens de carregamento para um operador.	3	5
US18	API	Como sistema, quero disponibilizar *endpoints* para agendamento de ordens.	3	5
US19	Integração	Como sistema, quero disponibilizar API para integração de docas.	2	5
US20	Web	Como gestor, quero visualizar dados resumidos no painel inicial (Dashboard).	2	5

Tabela 11 – Product Backlog: Sprint 5

ID	Categoria	User Story	SP	Sprint
US07	Mobile	Como operador, quero autenticar-me no aplicativo mobile para acessar minhas tarefas.	3	6
US12	Mobile	Como operador, quero manter sessão ativa no aplicativo.	2	6
US22	Mobile	Como operador, quero visualizar as ordens atribuídas a mim.	3	6
US30	API	Como sistema, quero consultar os itens de uma ordem para exibição em tela.	2	6
US31	API	Como sistema, quero consultar os *packages* vinculados a um item para validação.	2	6
US32	Operação	Como operador, quero visualizar o *status* do carregamento.	2	6

Tabela 12 – Product Backlog: Sprint 6

ID	Categoria	User Story	SP	Sprint
US02	Acesso	Como sistema, quero gerenciar perfis de usuário (Gestor vs Operador).	2	7
US23	Mobile	Como operador, quero abrir uma ordem para visualizar detalhes.	3	7
US24	Mobile	Como operador, quero visualizar os itens da ordem.	2	7
US29	Web	Como gestor, quero visualizar o detalhamento completo da ordem no sistema web.	3	7
US33	Operação	Como sistema, quero registrar o início do carregamento (Timestamp).	2	7
US34	Operação	Como sistema, quero registrar a finalização do carregamento.	2	7
US40	API	Como sistema, quero registrar o início/fim do processo de conferência.	2	7

Tabela 13 – Product Backlog: Sprint 7

ID	Categoria	User Story	SP	Sprint
US25	Mobile	Como operador, quero escanear o QR Code de um *package*.	5	8
US26	Mobile	Como operador, quero confirmar a conferência do item.	3	8
US36	Operação	Como sistema, quero validar se o QR Code pertence àquela ordem.	3	8
US37	Operação	Como operador, quero visualizar o progresso da conferência.	2	8
US42	Mobile	Como sistema, quero exigir justificativa para carregamentos incompletos.	2	8

Tabela 14 – Product Backlog: Sprint 8

ID	Categoria	User Story	SP	Sprint
US27	Mobile	Como operador, quero registrar fotos do carregamento.	3	9
US28	API	Como sistema, quero realizar *upload* das fotos para o *backend*.	3	9
US38	Obs.	Como sistema, quero registrar *logs* detalhados de operação.	2	9
US39	Web	Como gestor, quero visualizar métricas de carregamento no Dashboard.	2	9
US35	Operação	Como gestor, quero visualizar o histórico de *status* da ordem.	3	9

Tabela 15 – Product Backlog: Sprint 9

3.3 Arquitetura do Sistema

A arquitetura do Expedisoft foi projetada sob o modelo desacoplado (*client-server*), no qual o *backend* funciona como um provedor central de serviços (API) para múltiplas interfaces de consumo. Essa escolha permite que a lógica de negócio seja centralizada enquanto as interfaces *Web* e *Mobile* evoluem de forma independente, comunicando-se por meio de contratos no protocolo *Hypertext Transfer Protocol* (HTTP) e no formato JSON.

3.3.1 Camada de Serviços (API REST)

O ponto central do sistema reside no *backend* desenvolvido em Laravel, que atua como uma API. Conforme observado na estrutura de rotas e controladores, o sistema utiliza o padrão de Camada de Serviços (*Service Layer Pattern*) para isolar a lógica de negócio complexa dos controladores HTTP.

A comunicação entre os componentes ocorre da seguinte forma:

- **Contrato de Dados:** As informações trafegam exclusivamente no formato JSON, favorecendo a conectividade com sistemas ERP externos e garantindo respostas padronizadas para as interfaces *Web* e *Mobile*.
- **Processamento Assíncrono:** Para operações de maior custo computacional — como o recebimento de grandes volumes de dados de integração ou o *upload* de imagens para a nuvem —, o sistema utiliza Filas e *Jobs* (processamento em segundo plano), evitando o bloqueio da interface do usuário e contribuindo para a estabilidade operacional da aplicação.

3.3.2 Persistência e Integridade dos Dados

A modelagem do banco de dados PostgreSQL reflete uma preocupação com a normalização e a segurança. Uma decisão arquitetural crítica foi a adoção de UUIDs como chaves primárias, prática que reduz a previsibilidade dos identificadores e facilita a futura escalabilidade horizontal do banco de dados.

As tabelas principais foram estruturadas de modo a permitir a rastreabilidade completa do carregamento:

- ***Loading Orders***: Tabela central do sistema; contém os principais relacionamentos.
- ***Order Items***: Relacionamento entre produtos e ordens de carregamento específicas.
- ***Packages***: Identificação individual de cada volume físico por meio de códigos únicos, permitindo a conferência item a item via dispositivo *mobile*.

3.3.3 Segurança e Autenticação

Para proteger o acesso às informações do sistema, a arquitetura implementa o Laravel Sanctum, que provê autenticação baseada em *tokens* para as APIs. O estado de autenticação é gerenciado de forma reativa em cada interface:

- No *Frontend Web*, utiliza-se a *Context API* do React para manter a sessão do gestor e proteger rotas administrativas.
- No Aplicativo *Mobile*, a persistência do *token* permite que o operador mantenha a sessão ativa durante o turno de trabalho, garantindo agilidade no acesso às ordens atribuídas.

3.4 Modelagem do Banco de Dados

O banco de dados do Expedisoft foi projetado com foco na confiabilidade da integridade das informações e na viabilização de processos rigorosos de auditoria. A modelagem final divergiu do planejamento inicial para acomodar a necessidade de maior rastreabilidade operacional, resultando na inclusão de tabelas dedicadas a *logs* de operação e colunas que identificam a origem e a integridade de cada registro.

O banco conta com 25 tabelas, organizadas em quatro categorias funcionais: estrutura organizacional, entidades externas, entidades operacionais e entidades de auditoria e rastreabilidade. Dentre elas, destacam-se:

- ***loading_order:*** Tabela central que consolida o *status* do carregamento e vincula os operadores às docas.
- ***packages:*** Responsável pelo vínculo entre o item e os pacotes da ordem de carregamento, armazenando o código único do pacote para leitura.
- ***scan_logs:*** Contém todas as leituras realizadas pelo operador independentemente de sucesso ou falha, armazenando o código da caixa, o conteúdo da requisição, o *status* e a mensagem de erro.
- ***photos:*** Armazena as informações das fotos coletadas pelo operador durante o carregamento, incluindo colunas como *store_path*, ID do *Google Drive* e tipo do arquivo.
- ***integration_logs:*** Contém as informações dos dados recebidos do ERP do cliente, como *endpoint* requisitado, *payload* e *status* da integração.
- ***order_status_history:*** Registra todas as mudanças de *status* das ordens de carregamento, contendo o *status* novo e antigo e o usuário que realizou a alteração.

O diagrama completo do banco de dados pode ser conferido no Apêndice B.

3.5 Integração com Sistemas Externos ERP

Com o objetivo de favorecer a flexibilidade do Expedisoft, a comunicação com sistemas corporativos externos (ERP) baseia-se em contratos de dados padronizados via API REST. O *backend* disponibiliza rotas específicas de integração baseadas em JSON, estruturadas de maneira a favorecer a integridade e a validação dos dados recebidos em tempo de integração.

O ecossistema de integração é composto por três fluxos distintos: ordens de carregamento, usuários e docas. Cada fluxo possui seu respectivo modelo de *payload* padrão. Essa abordagem promove o desacoplamento de responsabilidades, exigindo que o sistema parceiro

se adeque ao formato esperado pelo Expedisoft e reduzindo a necessidade de adaptações específicas para diferentes formatos de integração oriundos de sistemas legados.

Ao receber uma requisição de integração, a camada de controle valida as propriedades e delega o processamento para uma fila de execução assíncrona. Durante a persistência, o serviço adota a estratégia de busca e criação automática de registros: caso entidades de infraestrutura como cliente, destino, transportadora ou produto não sejam encontradas na base de dados, o sistema efetua o cadastro ou a atualização automaticamente.

Os fluxos e suas especificações técnicas são detalhados a seguir:

- **Ordem de Carregamento:** Representa o fluxo principal de dados do sistema, consolidando o cabeçalho do pedido, os dados cadastrais de transporte, os itens comerciais e os agrupamentos de volumes físicos para conferência. O modelo estrutural é apresentado na Listagem 3.
- **Usuário:** Fluxo responsável pela sincronização de colaboradores da empresa cliente, populando a base interna com perfis de acesso restritos e credenciais iniciais, conforme exemplificado na Listagem 1.
- **Doca:** Módulo encarregado do mapeamento físico das áreas de carregamento da empresa, alimentando o cadastro de docas para agendamento logístico na interface administrativa, cujo modelo consta na Listagem 2.

3.5.1 Modelo de Dados: Ordens de Carregamento

O padrão definido para a integração de ordens de carregamento encapsula toda a complexidade operacional da expedição logística. A estrutura organiza hierarquicamente a rastreabilidade do processo, vinculando uma ordem comercial a seus respectivos volumes físicos mapeados em campo.

Detalhes do modelo:

- ***external_id*:** Atua como chave estrangeira lógica, preservando o vínculo direto com o identificador nativo do ERP de origem.
- ***unique_package_code*:** Código atribuído a cada volume físico, consumido diretamente via *scanner* de QR Code pelo aplicativo móvel para a validação do *checklist*.
- **Processamento de Itens:** A tabela de itens (*order_items*) é vinculada a um catálogo normalizado de produtos (*products*), minimizando a redundância de dados cadastrais.
- **Interface de Acesso:** Disponibilizado sob o método HTTP *POST* no caminho */api/integration/order*.

3.5.2 Modelo de Dados: Usuários

A integração de usuários centraliza e otimiza a gestão de acessos e a governança de credenciais no sistema. O modelo mapeia os identificadores necessários para o controle interno de privilégios, possibilitando que a sincronização popule a base de dados mantendo os vínculos corporativos intactos.

Detalhes do modelo:

- ***external_id*:** Identificador único do funcionário no banco de dados do sistema central do cliente.
- ***email*:** Atributo obrigatório utilizado como credencial exclusiva e persistente para a autenticação (*login*).
- **Segurança:** Os usuários importados são associados a perfis padrão de permissões (Operador ou Gestor), com senhas geradas por algoritmos de criptografia *hash* ativadas no primeiro acesso.
- **Interface de Acesso:** Disponibilizado sob o método HTTP *POST* no caminho */api/integration/user*.

3.5.3 Modelo de Dados: Docas

A infraestrutura logística do armazém é alimentada por meio do contrato de docas, estruturando os locais físicos disponíveis para a alocação de carregamentos no painel administrativo do gestor logístico.

Detalhes do modelo:

- ***external_id*:** Código de identificação exclusivo da doca originário do ERP.
- ***dock_code*:** Código abreviado ou sigla utilizada nas listagens visuais do painel administrativo.
- ***location*:** Delimitação de espaço (unidade, filial ou pátio) onde a estrutura operacional está instalada.
- **Interface de Acesso:** Disponibilizado sob o método HTTP *POST* no caminho */api/integration/dock*.

4 DESENVOLVIMENTO E RESULTADOS

Este capítulo descreve as decisões de implementação tomadas durante o desenvolvimento do Expedisoft e apresenta os resultados obtidos. São abordados: o processamento e armazenamento assíncrono de registros fotográficos, a estratégia de testes e qualidade adotada, com base no TDD, e as interfaces construídas, tanto na plataforma *web* quanto no aplicativo *mobile*.

4.1 Processamento e Armazenamento de Fotos

Conforme planejado nos requisitos operacionais do sistema, o Expedisoft contempla a captura e o armazenamento de registros fotográficos durante o encerramento do carregamento logístico para fins de auditoria e comprovação de integridade da carga. Alinhado às boas práticas de modelagem de dados, optou-se por não persistir os arquivos de imagem em formato bruto (*Binary Large Objects* — BLOB) diretamente no banco de dados, evitando o crescimento desordenado da base e a degradação da performance das consultas.

Para solucionar esse desafio, foi arquitetada uma solução de armazenamento distribuído por referência, integrada à API do *Google Drive*. Visando a alta disponibilidade do sistema, o fluxo operacional foi estruturado por meio de um desacoplamento síncrono e assíncrono, composto pelas seguintes etapas:

- **Interface Móvel e Envio:** A interface *mobile* gerencia a captura ou a seleção da imagem na galeria do dispositivo. Ao finalizar a conferência, os arquivos são transmitidos para o *backend* por meio de uma requisição HTTP do tipo *POST* no formato *multipart/form-data*.

- **Recepção e Resposta Ágil:** O processamento síncrono é intermediado pela classe *OrderPhotoController*. O sistema valida a requisição, salva temporariamente os arquivos binários no disco do servidor local por meio do *OrderPhotoService* e cria um registro na tabela *photos* com o estado inicial definido como pendente (*STATUS_PENDING*). Imediatamente, a API retorna uma resposta JSON com o código de estado HTTP *202 Accepted*, liberando a interface do operador móvel para novos carregamentos sem a necessidade de aguardar o término da transferência de mídia.

- **Fila Assíncrona e *Upload* Remoto:** O serviço dispara o *Job* de processamento em segundo plano, denominado *UploadPhotoToDriveJob*, gerenciado pelas filas do Laravel. Esse componente lê o binário do disco local, estabelece conexão segura com o *Google Drive* por meio do *Flysystem Driver* e realiza a persistência na pasta de destino correspondente ao identificador externo da ordem logística.

- **Atualização Cadastral e Coleta de Metadados:** Concluída a transferência com sucesso, o algoritmo extrai os metadados do adaptador de rede da nuvem externa, capturando o identificador global único do arquivo (*drive_id*). O registro é atualizado para o estado enviado (*STATUS_UPLOADED*), salvando o caminho lógico para renderizações futuras no painel *web* administrativo. Por fim, o arquivo binário temporário é removido do servidor, otimizando o espaço em disco.

4.2 Estratégia de Testes e Qualidade

Para assegurar a confiabilidade, a tolerância a falhas e a manutenibilidade do Expedisoft, o processo de codificação dos módulos principais do *backend* fundamentou-se na técnica de TDD. Essa abordagem consiste em um ciclo iterativo no qual os cenários de teste são escritos antes mesmo da implementação da funcionalidade de produção (*Red*), observando-se sua falha controlada; em seguida, desenvolve-se o código mínimo necessário para a aprovação do teste (*Green*) e, por fim, realiza-se a melhoria do código sob uma rede de segurança estável (*Refactor*).

A adoção do desenvolvimento guiado por testes mostrou-se um elemento estratégico mitigador de riscos, especialmente no contexto de um projeto de autoria individual, no qual a ausência de uma equipe de garantia de qualidade (*QA*) exige mecanismos automatizados de verificação de regressões.

A suíte de testes foi estruturada com a ferramenta nativa do Laravel (PHPUnit), concentrando esforços em testes de integração de funcionalidades (*Feature Tests*). Esses testes simulam requisições HTTP reais contra os *endpoints* da API, avaliando desde o comportamento das regras de negócio até a persistência correta no banco de dados.

4.2.1 Impacto do TDD no Ciclo de Desenvolvimento

O impacto prático da aplicação do TDD no Expedisoft manifestou-se em quatro vertentes principais:

- **Design de Código Orientado ao Domínio:** Escrever o teste antes do código de produção forçou o isolamento de responsabilidades em camadas de serviço puras (*Service Layer*). Classes complexas, como o validador de bipagem de pacotes (*CheckListEntryService*), foram criadas de forma modular para permitir o isolamento de cenários de sucesso e falha nas leituras operacionais.

- **Segurança em Ciclos de Refatoração:** Conforme registrado na Tabela 6, a Sprint 4 apresentou dificuldades técnicas na refatoração do código de integração com o ERP. A existência de uma base sólida de testes automatizados permitiu reestruturar com-

pletamente os algoritmos de processamento de carga sem o risco de introduzir efeitos colaterais nas APIs já consumidas pelas interfaces *web* e *mobile*.

- **Consistência e Concorrência:** Os testes guiaram a implementação de mecanismos para lidar com cenários críticos de produção, como tentativas de duplicação de leitura de pacotes e requisições concorrentes idênticas enviadas pelo ERP, favorecendo respostas HTTP consistentes e estabilidade transacional.
- **Compreensão do Domínio e Refinamento de Escopo:** A antecipação lógica exigida pelo método propiciou um entendimento analítico profundo do comportamento esperado para cada funcionalidade. Ao mapear o que o sistema deveria ou não permitir, materializaram-se cenários práticos que reduziram o nível de abstração da modelagem conceitual estática. Esse exercício revelou inconsistências no planejamento, motivando ajustes estruturais em tabelas, otimização de relacionamentos e a especificação de novas Histórias de Usuário.

4.2.2 Mapeamento dos Testes de Funcionalidades (*Feature Tests*)

A cobertura de testes automatizados concentrou-se nos fluxos críticos do ecossistema, mapeando os seguintes componentes de produção:

- ***AuthApiTest*:** Valida os mecanismos de autenticação e proteção de rotas via Laravel Sanctum, garantindo que requisições sem *token* válido sejam rejeitadas com o código HTTP 401.
- ***OrderIntegrationTest* e *DockIntegrationTest*:** Testam o comportamento dos *endpoints* de ingestão de dados, simulando o envio de estruturas JSON válidas e inválidas para verificar o correto funcionamento dos validadores de requisição (*FormRequests*) e o disparo das filas assíncronas (*Jobs*).
- ***CheckListEntryTest*:** Blinda a regra de negócio mais sensível do aplicativo operacional, verificando se o sistema valida corretamente os identificadores (*unique_package_code*), incrementa o progresso do *checklist* e rejeita volumes pertencentes a outras ordens de carregamento.
- ***PhotoUploadTest* e *UploadPhotoToDriveJobTest*:** Asseguram a integridade do fluxo de documentação visual, avaliando o recebimento do arquivo de imagem binária pela API e o encadeamento correto do *Job* assíncrono responsável por persistir o arquivo no serviço de armazenamento remoto.

4.3 Interfaces e Funcionamento do Sistema

Esta seção apresenta as interfaces desenvolvidas no ecossistema Expedisoft, relacionando-as às respectivas regras de negócio, fluxos operacionais e componentes implementados nas arquiteturas *web* e *mobile*.

4.3.1 Interface Web (Módulo Administrativo)

A interface *web* concentra as funcionalidades de gestão, agendamento de cargas, auditoria e acompanhamento logístico por parte dos gestores.

- **Tela de Autenticação:** Apresentada na Figura 2, a tela de autenticação restringe o acesso às funcionalidades administrativas do sistema. Após a validação das credenciais do usuário, a sessão é mantida de forma segura, permitindo a navegação entre os módulos da plataforma sem a necessidade de novas autenticações durante o período de uso.

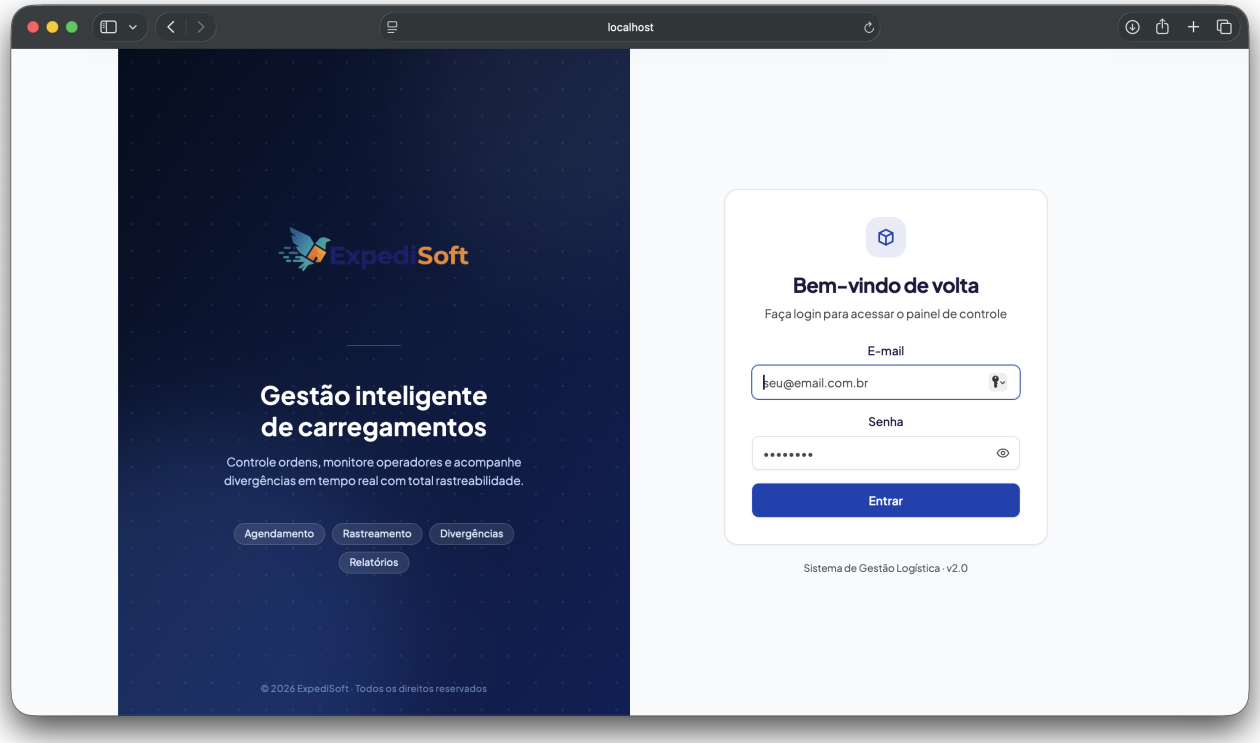

Figura 2 – Web: Tela de Login.

- **Monitoramento (*Dashboard*):** Conforme ilustrado na Figura 3, a página inicial disponibiliza indicadores operacionais que permitem acompanhar o desempenho da expedição em tempo real. O painel apresenta a distribuição das ordens de carregamento por *status*, contemplando carregamentos agendados, em andamento, concluídos e fi-

nalizados com divergência. Além disso, são exibidos gráficos de acompanhamento das operações realizadas nos últimos quatorze dias e uma tabela com as ordens mais recentemente atualizadas no sistema.

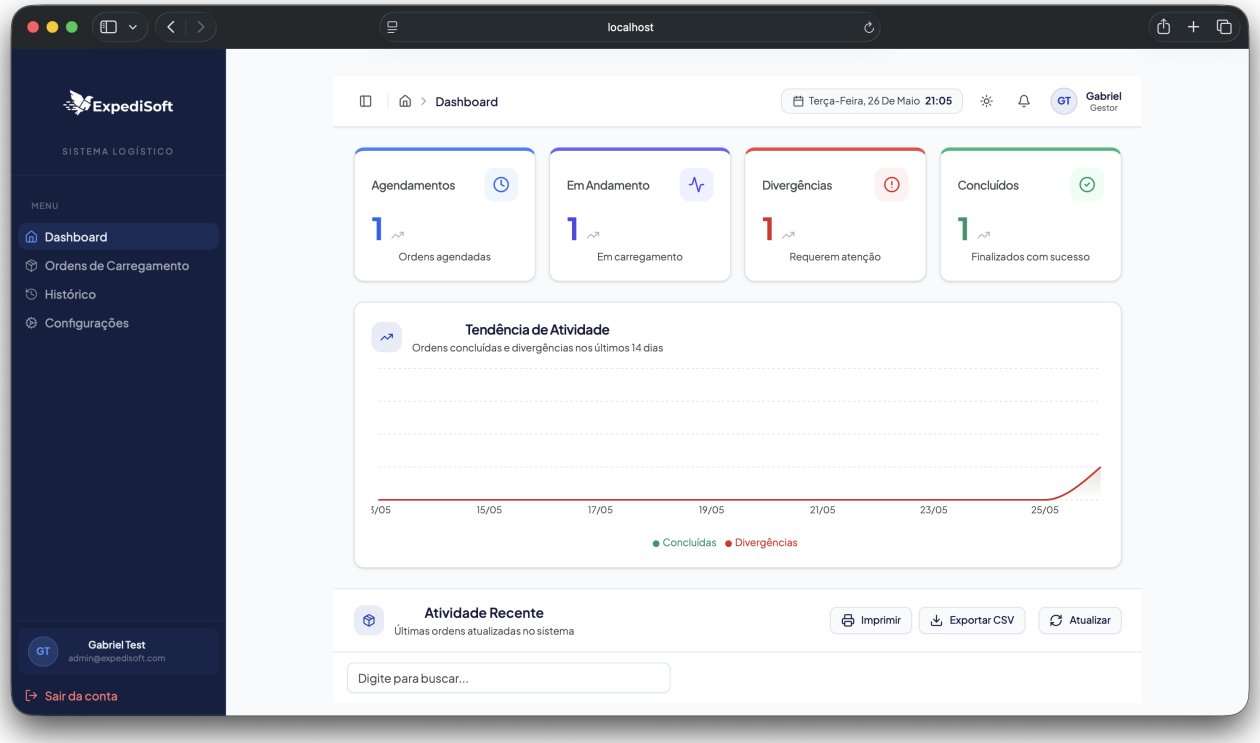

Figura 3 – Web: Página Principal.

- **Listagem e Agendamento de Ordens:** A tela de gerenciamento logístico, apresentada na Figura 4, reúne todas as ordens de carregamento integradas ao sistema. Para programar uma operação, o gestor seleciona a opção “Agendar Carregamento” na coluna de ações da ordem desejada. Em seguida, é exibido um formulário modal para o preenchimento das informações necessárias ao planejamento da atividade, incluindo data, horário, operador responsável e doca de carregamento, conforme demonstrado na Figura 5.

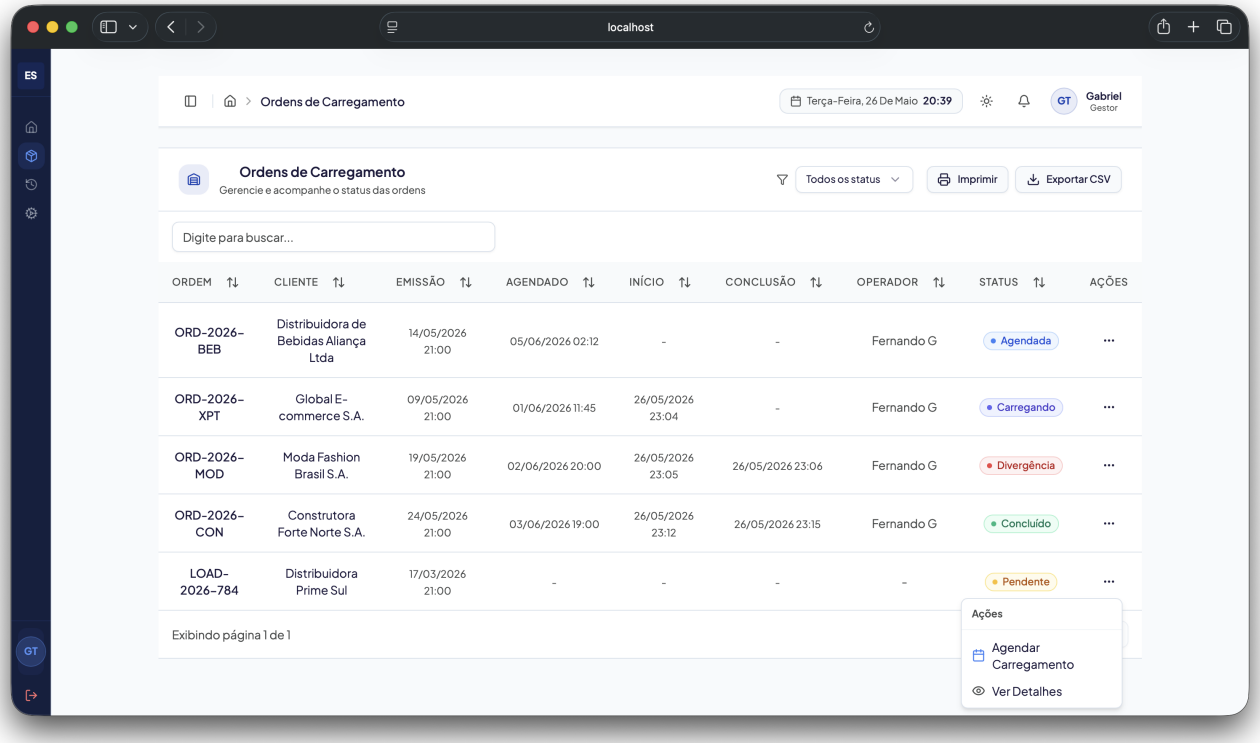

Figura 4 – Web: Ordens de Carregamento.

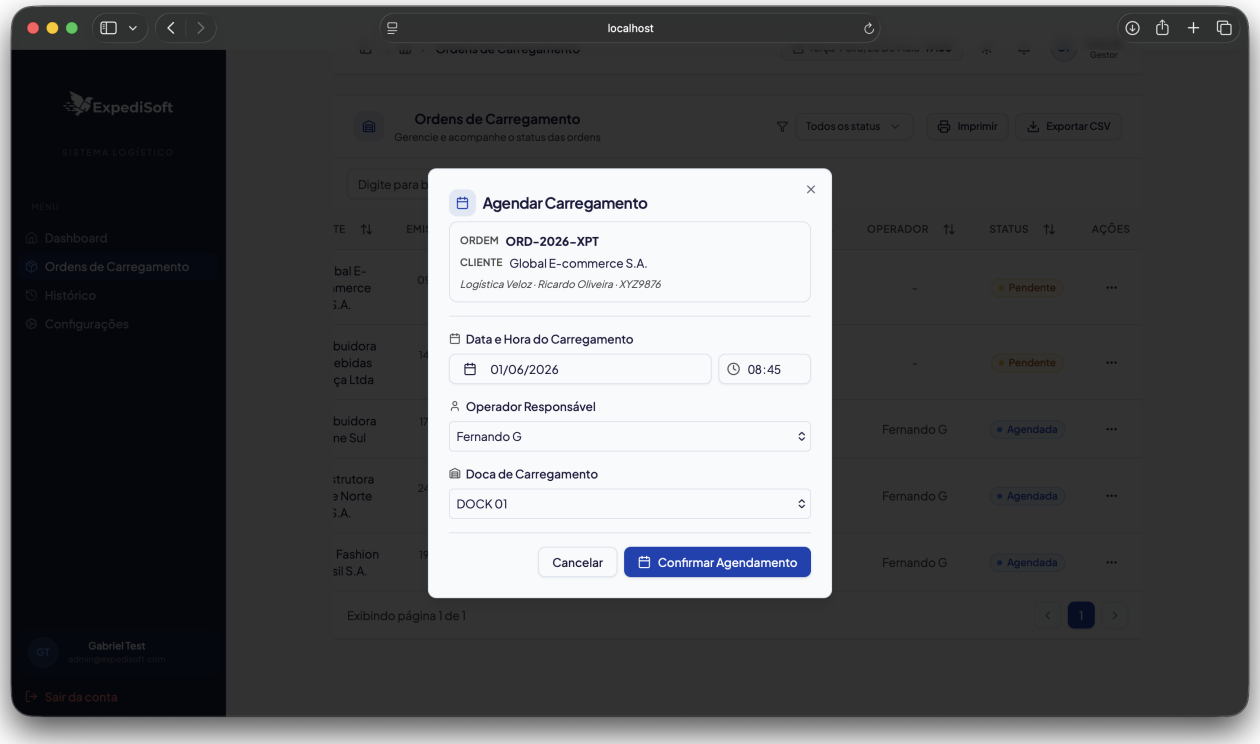

Figura 5 – Web: Modal de Agendamento.

- **Consulta e Auditoria de Ordens:** Para acompanhar ou auditar uma operação, o gestor seleciona a opção “Ver Detalhes” na coluna de ações da ordem desejada, conforme ilustrado na Figura 6. A partir desse acesso, o sistema exibe informações detalhadas sobre a operação logística, incluindo dados da carga, cliente, transportadora, operador responsável, cronologia de execução e situação atual do carregamento.

  As telas apresentadas nas Figura 7 e Figura 8 demonstram a visualização completa da ordem, permitindo a consulta dos itens vinculados ao carregamento e seus respectivos *status* de conferência, além das evidências fotográficas registradas pelo operador durante a execução da atividade.

  Ao selecionar uma imagem anexada, o sistema apresenta uma visualização ampliada da fotografia, conforme exemplificado na Figura 9, favorecendo a análise detalhada das evidências coletadas durante o carregamento.

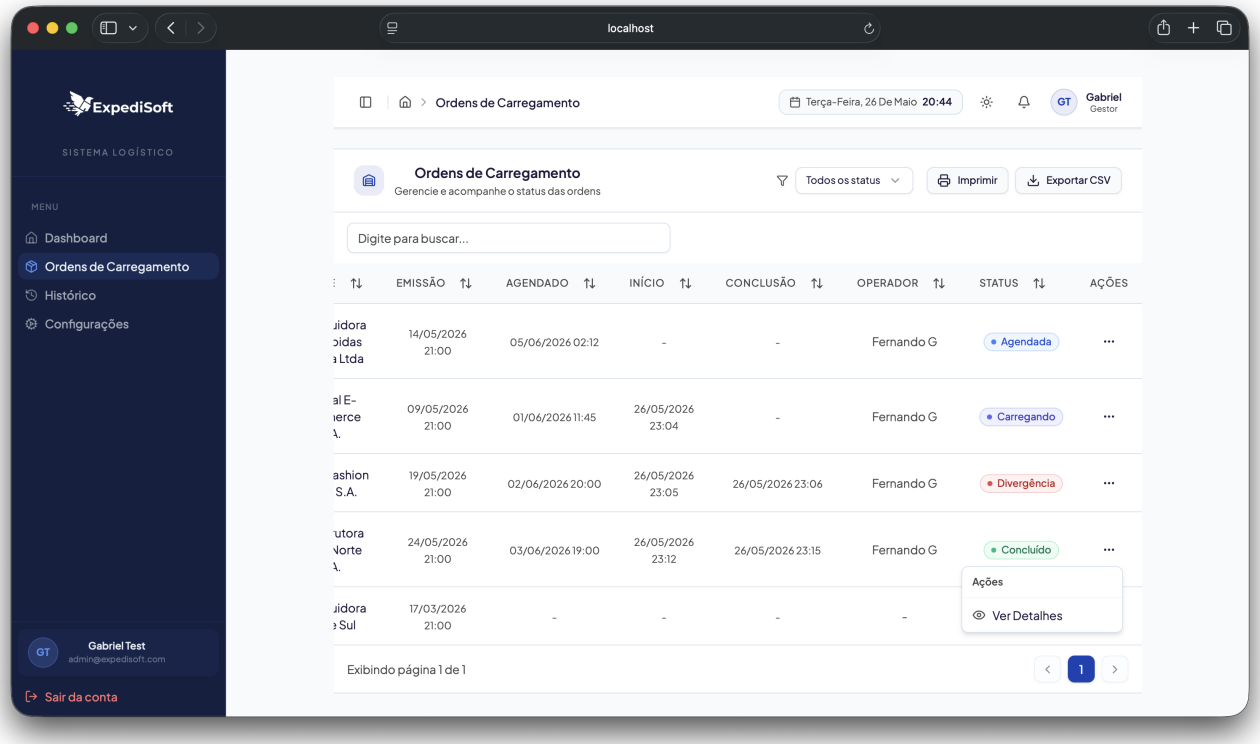

Figura 6 – Web: Selecionar Ordem para Detalhes.

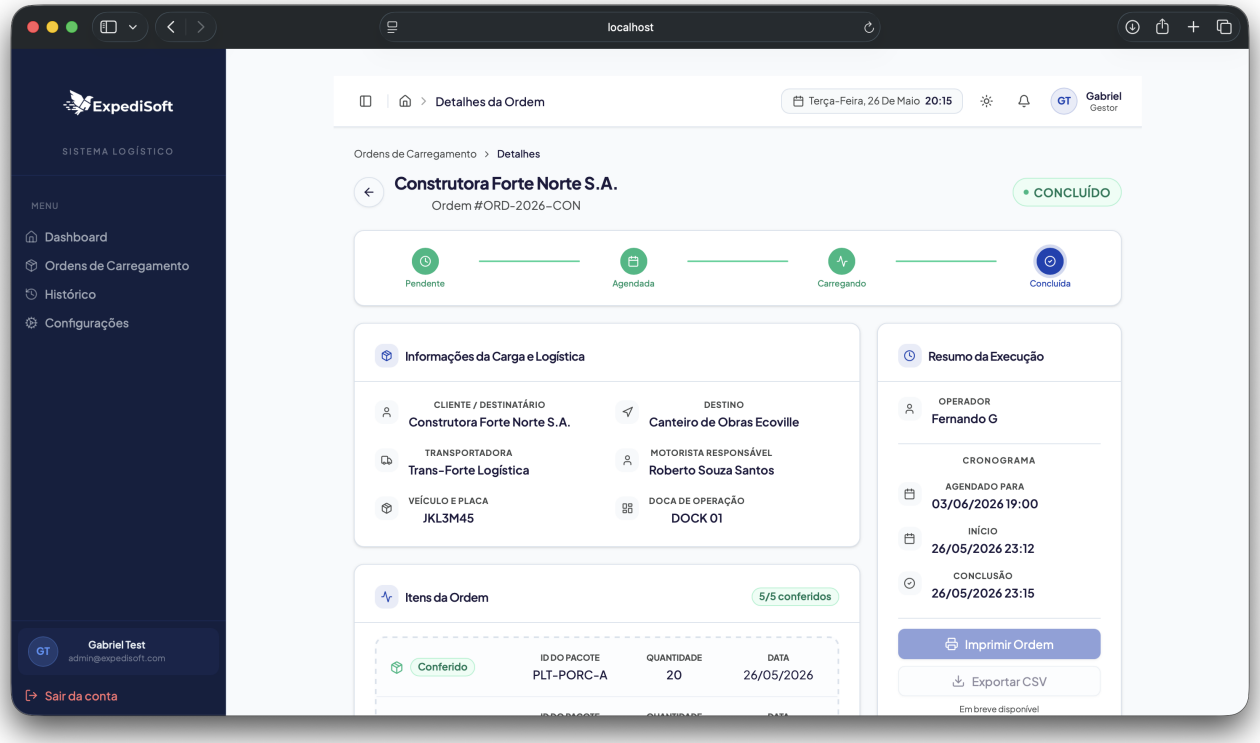

Figura 7 – Web: Detalhes da Ordem.

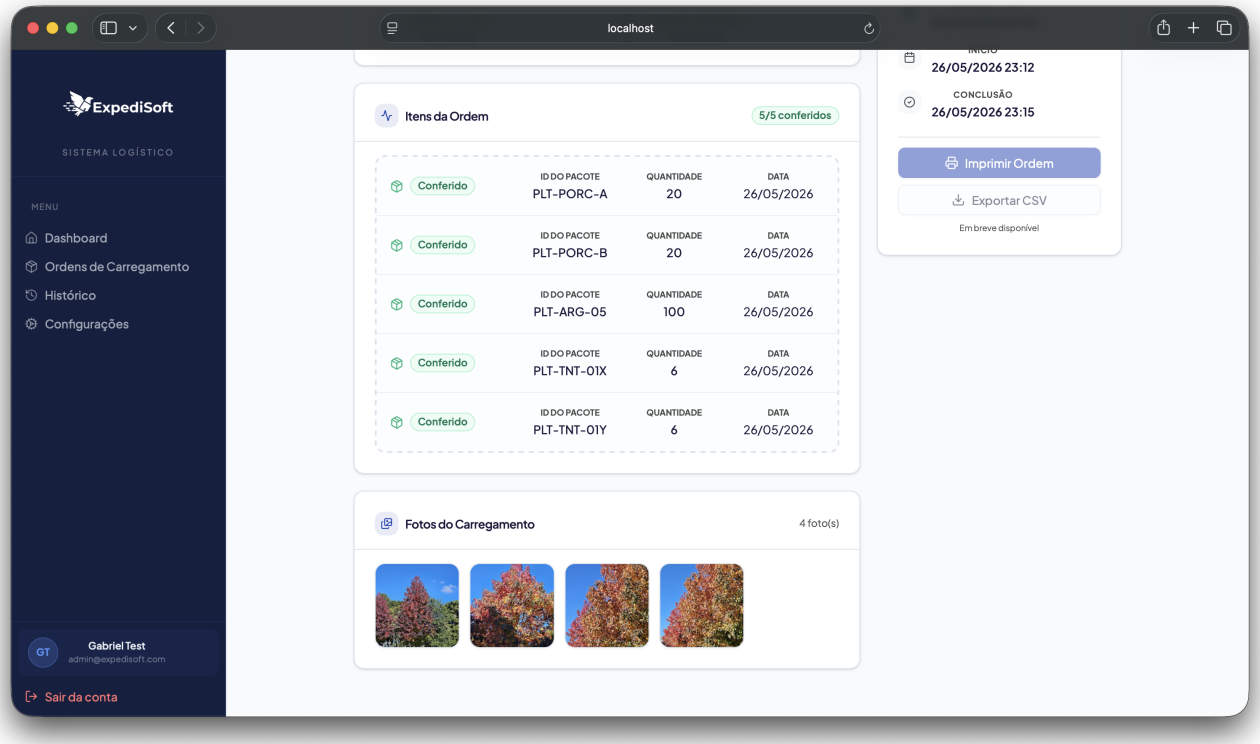

Figura 8 – Web: Detalhes da Ordem 2.

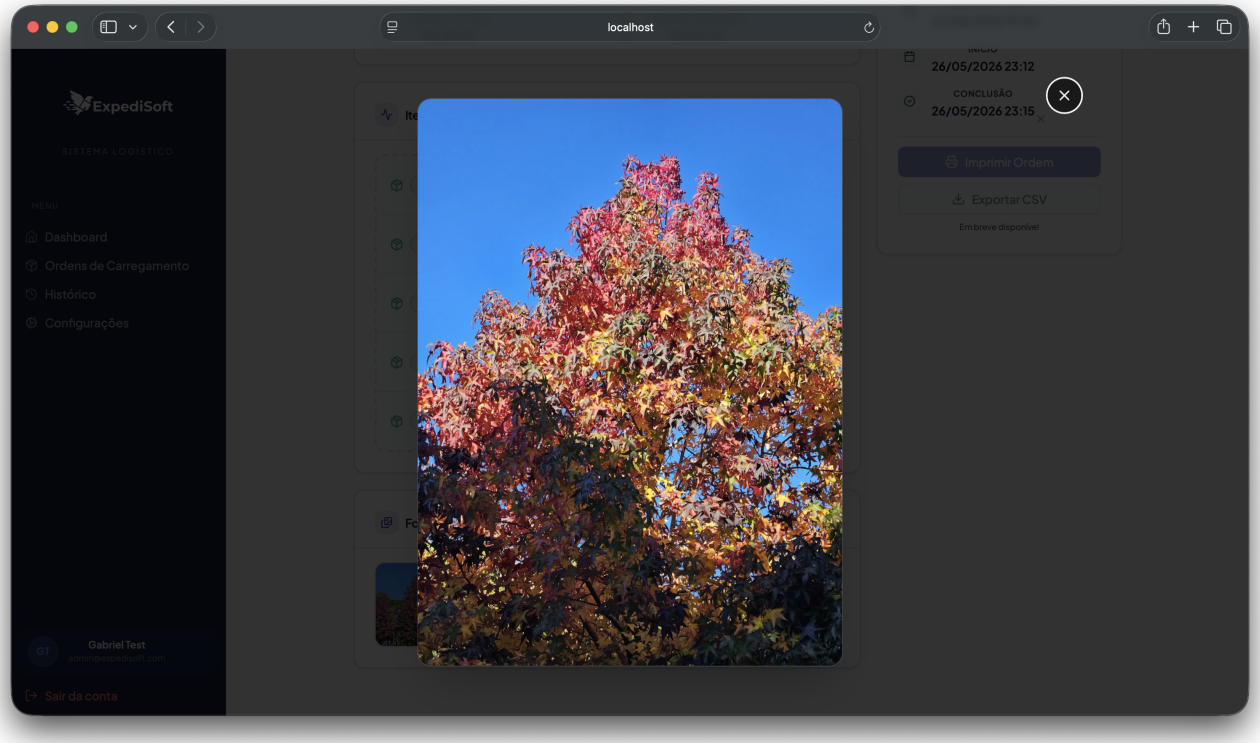

Figura 9 – Web: Detalhes da Foto da Carga.

- **Ordens com Divergência:** Quando uma ordem é finalizada com divergência, a interface exibe a justificativa informada pelo operador durante a execução da atividade. Essa informação permite aos gestores compreender os motivos da inconsistência registrada, conforme apresentado na Figura 10.

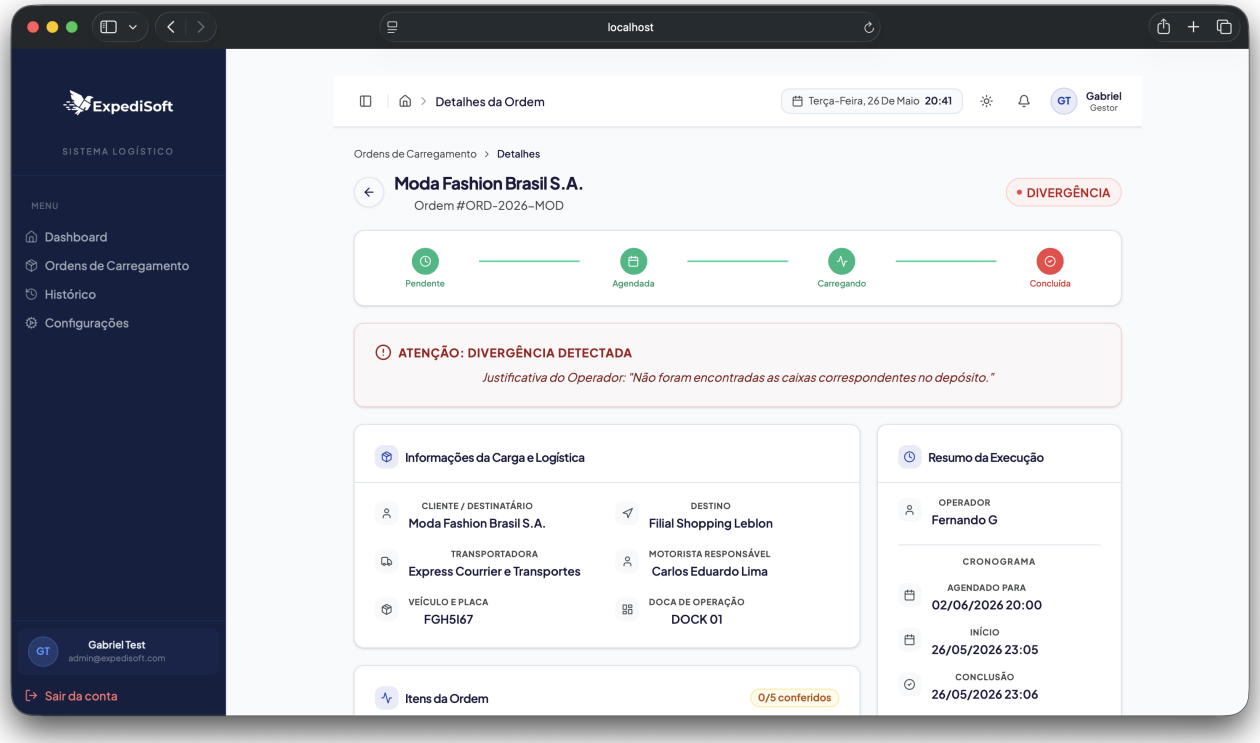

Figura 10 – Web: Ordem Apresentando Divergência.

- **Conferindo uma Ordem em Andamento:** Ao acessar uma ordem em execução, o gestor pode acompanhar seu progresso por meio da tela de detalhes. Conforme exemplificado na Figura 11, é apresentado o *status* atual da operação, permitindo identificar a etapa em que o carregamento se encontra.

  Complementarmente, a Figura 12 apresenta os itens já conferidos e aqueles ainda pendentes, possibilitando o acompanhamento da execução em tempo real.

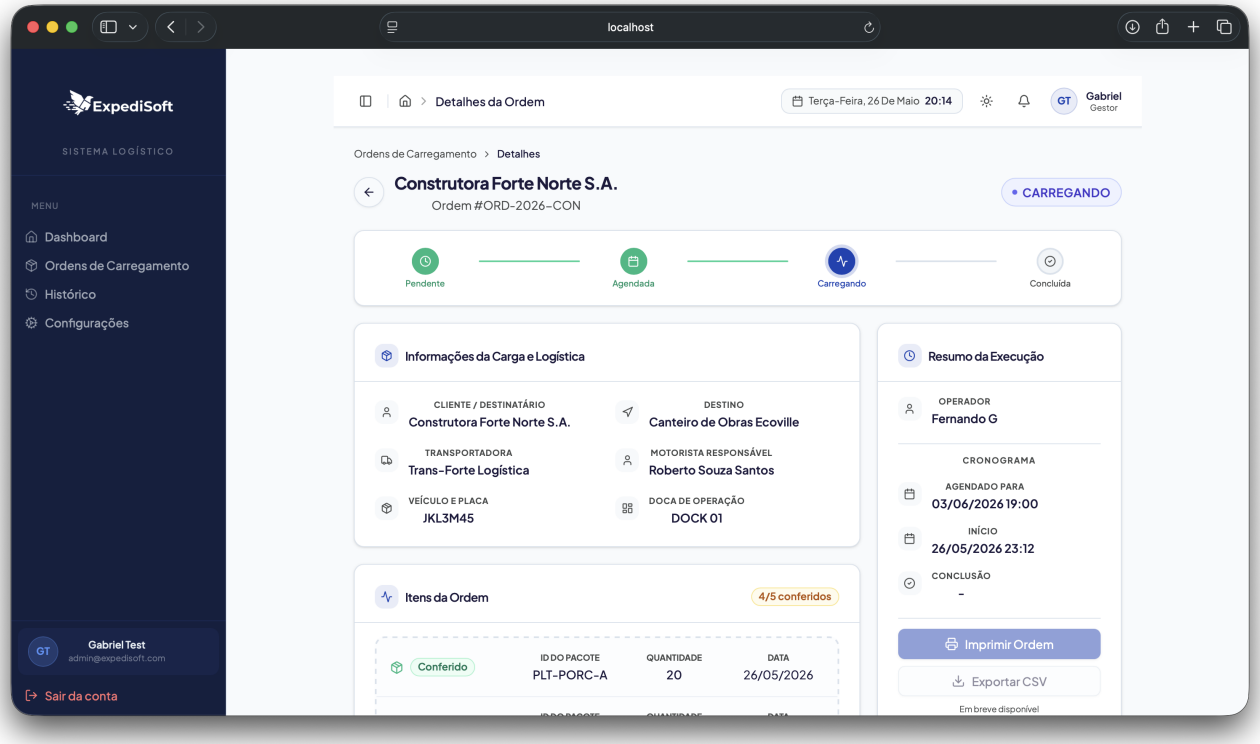

Figura 11 – Web: Ordem em Andamento.

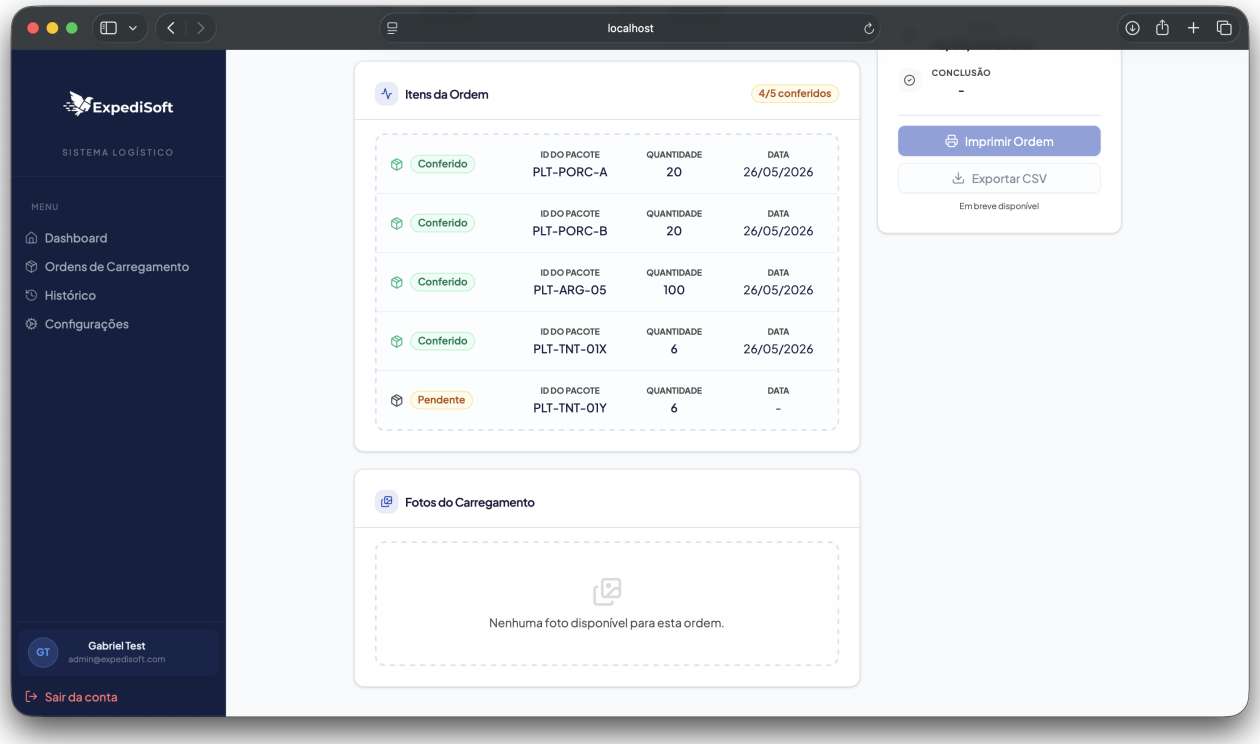

Figura 12 – Web: Ordem em Andamento 2.

4.3.2 Aplicativo Móvel do Operador

A interface *mobile* contém as funcionalidades de listagem e detalhamento dos carregamentos, captura de fotos e o processo de conferência que o operador realiza em campo.

- **Autenticação do Aplicativo:** A tela de acesso (*login*), apresentada na Figura 13, restringe o uso do aplicativo a funcionários autorizados. Após a validação das credenciais, o operador é direcionado para a tela inicial do sistema (Figura 14), que funciona como menu principal para a seleção das atividades. A sessão permanece ativa no dispositivo de forma contínua, evitando a necessidade de novas autenticações a cada abertura do aplicativo.

Figura 13 – Mobile: Tela de Autenticação.

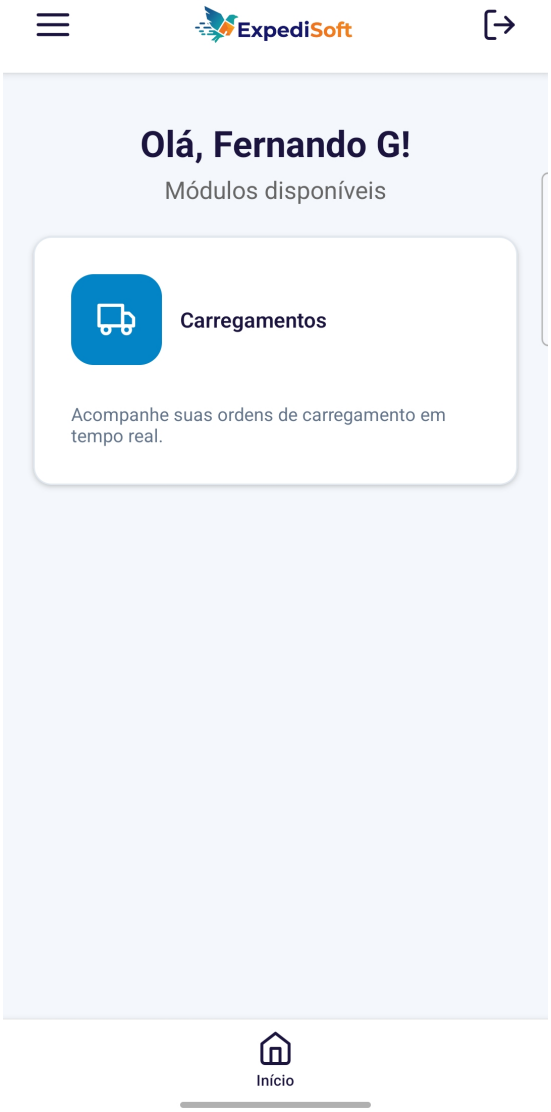

Figura 14 – Mobile: Tela Inicial.

- **Lista de Carregamentos:** Nessa página são exibidas de maneira organizada as ordens de carregamento atribuídas ao funcionário que opera o dispositivo, conforme ilustrado na Figura 15. Por meio dela, o operador confere rapidamente a situação atual de cada tarefa, o número de identificação da carga e a data e hora programada para a expedição.

Figura 15 – Mobile: Lista de Cargas Atribuídas.

- **Detalhamento e Início do *Checklist*:** Ao selecionar uma atividade na lista, a tela de detalhes (Figura 16) apresenta as informações operacionais do carregamento: número do pedido de origem, cliente de destino, placa do veículo transportador e doca de alocação física. Abaixo dessas informações, o aplicativo lista a porcentagem de conferência concluída e detalha as caixas que devem ser embarcadas, exibindo o código do produto, nome comercial, quantidade total e unidade de medida, além de disponibilizar botões para iniciar o fluxo, acionar a câmera ou anexar imagens.

Figura 16 – Mobile: Detalhamento da Ordem Logística.

- **Conferência Automatizada via QR Code:** Durante a conferência física, o operador aciona a câmera nativa do aparelho para realizar a leitura contínua das etiquetas coladas nos volumes. O sistema valida as informações em tempo real: caso o pacote pertença à carga correta, a lista é atualizada e exibida a mensagem de sucesso (Figura 17). Se houver leitura duplicada, volume pertencente a outro pedido, erro ou inconsistência, a interface dispara uma mensagem de alerta (Figura 18), prevenindo falhas humanas.

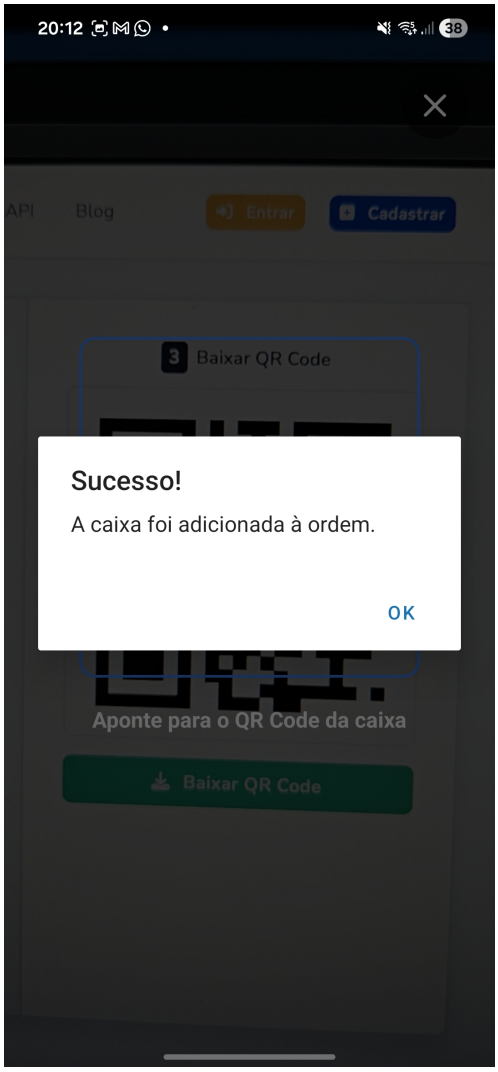

Figura 17 – Mobile: Validação com Sucesso.

Figura 18 – Mobile: Alerta de Inconsistência.

- **Tratamento de Divergências e Coleta de Fotos:** Se a conferência for finalizada com alguma inconformidade (por exemplo, caixas faltando), o aplicativo bloqueia a conclusão direta. O operador torna-se obrigado a registrar uma justificativa textual detalhando o motivo da divergência (Figura 19). Também é disponibilizada a funcionalidade de capturar fotos comprobatórias da carga (Figura 20). Essas imagens e relatos são enviados, após a finalização, para a central administrativa, onde a gerência pode acompanhá-los em tempo real.

Figura 19 – Mobile: Registro de Justificativa.

Figura 20 – Mobile: Captura de Evidências.

5 CONCLUSÃO

O presente trabalho teve como objetivo desenvolver uma solução para auxiliar o processo de expedição de mercadorias, buscando solucionar limitações identificadas em processos realizados de forma manual, especialmente relacionadas ao acompanhamento das operações, registro de evidências e rastreabilidade das atividades executadas durante o carregamento.

A partir da análise do processo operacional, foi identificada a necessidade de uma ferramenta que permitisse integrar as etapas de planejamento, execução e acompanhamento das ordens de carregamento. Nesse contexto, o sistema Expedisoft foi desenvolvido contemplando uma plataforma *web* destinada à gestão administrativa e um aplicativo móvel direcionado aos operadores responsáveis pela conferência em campo.

A solução desenvolvida atende ao problema proposto ao possibilitar que os gestores acompanhem o andamento das operações, consultem informações detalhadas dos carregamentos e tenham acesso aos registros realizados durante a execução das atividades. Além disso, o aplicativo móvel permite que os operadores realizem a conferência dos volumes por meio da leitura dos códigos identificadores, registrem divergências encontradas e anexem evidências fotográficas, proporcionando maior confiabilidade às informações geradas durante o processo.

A integração entre as interfaces desenvolvidas possibilita centralizar os dados relacionados à expedição, permitindo maior visibilidade sobre o fluxo operacional e facilitando a identificação de inconsistências. Dessa forma, o sistema contribui para reduzir a dependência de controles manuais, melhorar a organização das informações e fornecer suporte à tomada de decisão por parte dos responsáveis pela operação logística.

Durante o desenvolvimento, foram aplicados conceitos relacionados à engenharia de software, arquitetura de sistemas, desenvolvimento de aplicações *web* e móveis, integração entre sistemas e boas práticas de desenvolvimento. A utilização dessas abordagens possibilitou a criação de uma solução estruturada, com separação de responsabilidades e preparada para futuras evoluções conforme as necessidades do processo operacional.

Dessa forma, o Expedisoft estabelece uma base tecnológica para apoiar o processo de expedição de mercadorias, atendendo aos requisitos levantados durante a análise do problema e oferecendo uma alternativa integrada para o acompanhamento, execução e rastreabilidade das operações logísticas.

5.1 Trabalhos Futuros

Embora a solução desenvolvida atenda aos objetivos definidos para este trabalho, durante o desenvolvimento do Expedisoft foram identificadas oportunidades de evolução relacionadas principalmente à governança da plataforma, segurança das integrações e ampliação das funcionalidades de apoio ao processo logístico. Em razão das limitações de tempo e do

escopo estabelecido para este projeto, essas funcionalidades não foram implementadas, mas representam possibilidades de continuidade e expansão da solução.

Entre as funcionalidades identificadas destaca-se a implementação de um módulo de auditoria, responsável por disponibilizar aos gestores um histórico detalhado das alterações realizadas no sistema. Esse recurso permitiria consultar modificações relacionadas às principais entidades da aplicação, como ordens de carregamento, processos executados em segundo plano e demais registros operacionais. Sua implementação ampliaria a rastreabilidade das ações realizadas pelos usuários, além de facilitar processos de análise, auditoria e conferência das informações armazenadas.

Outra funcionalidade prevista consiste na implementação de um módulo de configurações administrativas, permitindo centralizar o gerenciamento de usuários, perfis de acesso, redefinição de credenciais e parâmetros gerais da plataforma. Esse módulo também possibilitaria configurar integrações externas e demais informações necessárias para adaptação do sistema às necessidades específicas de diferentes organizações.

Em relação às integrações com sistemas externos, atualmente a comunicação utiliza um mecanismo de autenticação baseado em *token* fixo. Como evolução, pretende-se implementar um processo de renovação automática das credenciais de integração, incluindo controle de expiração e atualização dos *tokens* utilizados na comunicação. Essa abordagem proporcionará maior segurança ao processo de integração e facilitará sua utilização em ambientes com múltiplos clientes.

Considerando uma possível evolução da solução para utilização por diferentes organizações, uma melhoria arquitetural relevante consiste na adaptação do sistema para uma abordagem *multi-tenant*. Esse modelo permitiria que diferentes empresas utilizassem a mesma plataforma mantendo o isolamento lógico das informações, configurações específicas e controle individualizado de acesso, tornando a solução mais preparada para cenários de expansão.

Além das melhorias estruturais, novos módulos relacionados ao processo logístico poderão ser incorporados ao sistema. Um exemplo consiste na implementação de um módulo de rastreamento de cargas, utilizando informações provenientes de dispositivos de rastreamento instalados nos veículos de transporte. Essa funcionalidade permitiria acompanhar a movimentação das cargas durante o deslocamento, disponibilizando informações relacionadas à localização e ao status do transporte.

Para viabilizar essa integração de forma escalável, uma possível abordagem consiste na utilização do Apache Kafka como plataforma de processamento de eventos. Essa tecnologia permitiria receber, distribuir e processar dados provenientes de diferentes fontes externas, como sistemas de rastreamento, reduzindo o acoplamento entre os componentes da aplicação e permitindo que novas integrações sejam adicionadas futuramente sem comprometer a arquitetura existente.

Outro módulo de expansão consiste no planejamento inteligente de carregamentos. Essa funcionalidade teria como objetivo auxiliar na organização dos volumes dentro do veículo

ou contêiner, considerando as dimensões disponíveis e as características dos itens transportados. A partir dessas informações, o sistema poderia gerar uma simulação do carregamento, indicando o posicionamento recomendado dos volumes para melhor aproveitamento do espaço físico, além de disponibilizar uma representação tridimensional da distribuição dos itens.

Soluções existentes no mercado, voltadas ao planejamento e otimização logística, demonstram a viabilidade desse tipo de recurso ao utilizar informações relacionadas às dimensões dos volumes e dos espaços de transporte para gerar simulações de carregamento. Como evolução do Expedisoft, uma funcionalidade semelhante poderia ser integrada ao processo de expedição, utilizando os dados já disponíveis nas ordens de carregamento para auxiliar o planejamento operacional.

Como possibilidade adicional de evolução, destaca-se a integração do sistema com agentes baseados em modelos de linguagem de grande escala (*Large Language Models – LLMs*). Essa integração poderá auxiliar os usuários na consulta de informações operacionais, geração de relatórios, análise de ocorrências e apoio à tomada de decisão, utilizando os dados registrados pela própria plataforma.

Dessa forma, as funcionalidades apresentadas representam possibilidades de continuidade do Expedisoft, permitindo ampliar sua abrangência funcional, fortalecer aspectos relacionados à governança, segurança e escalabilidade da plataforma e incorporar novos recursos voltados à otimização dos processos logísticos. Essas evoluções poderão ser exploradas em trabalhos futuros, contribuindo para o amadurecimento da solução e sua aplicação em diferentes contextos organizacionais.

REFERÊNCIAS

BALLOU, R. H. **Gerenciamento da cadeia de suprimentos/logística empresarial. 5. ed. Porto Alegre: Bookman**. 2006. Disponível em: https://www.google.com.br/books/edition/Gerenciamento_da_Cadeia_de_Suprimentos_5/QAHrq0r6E7cC?hl=pt-BR&gbpv=1&pg=PA3&printsec=frontcover. Acesso em: 13 ago. 2025.

BECK, K. **TDD desenvolvimento guiado por testes**. [*S.l.*]: Bookman, 2010.

BOWERSOX, D. J.; CLOSS, D. J.; COOPER, M. B. **Gestão da cadeia de suprimentos e logística. 4. ed. Rio de Janeiro: Elsevier**. 2013. Disponível em: https://www.google.com.br/books/edition/Gerenciamento_da_Cadeia_de_Suprimentos_5/QAHrq0r6E7cC.

CLEGG D.; BARKER, R. **CASE Method Fast-track: A RAD Approach**. [*S.l.*]: Addison-Wesley Publishing Company, 1994.

DOCKER. **Docker Documentation. [S.l.]: Docker, Inc.** 2025. Disponível em: https://docs.docker.com/.

EVANS, E. **Domain-Driven Design: Tackling Complexity in the Heart of Software.** [*S.l.*]: Boston: Addison-Wesley, 2004.

FIGMA. **Figma: The collaborative interface design tool. [S.l.]: Figma, Inc.** 2025. Disponível em: https://www.figma.com/.

GIT. **Git Reference. [S.l.]: Git.** 2025. Disponível em: https://git-scm.com/doc.

GitHub Inc. **GitHub Copilot: Your AI pair programmer**. 2025. Acesso em: 03 nov. 2025. Disponível em: https://github.com/features/copilot.

GROUP, P. G. D. **PostgreSQL: Documentation (Version 15). [S.l.]: The PostgreSQL Global Development Group**. 2025. Disponível em: https://www.postgresql.org/docs/15/.

KIM GENE; HUMBLE, J. D. P. W. J. **The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations.** [*S.l.*]: IT Revolution Press, 2016.

LARAVEL. **Laravel: The PHP Framework for Web Artisans**. 2025. Disponível em: https://laravel.com/docs/10.x.

META. **React: The library for web and native user interfaces. [S.l.]: Meta Platforms, Inc.** 2025. Disponível em: https://react.dev/.

REACT. **React Native: Learn once, write anywhere. [S.l.]: Meta Platforms, Inc.** 2025. Disponível em: https://reactnative.dev/.

SHADCN. **shadcn/ui. [S.l.]: shadcn**. 2025. Disponível em: https://ui.shadcn.com/.

SOMMERVILE, I. **Software Engineering. 9ª ed.** [*S.l.*]: Pearson, 2011.

VASYUKOV A. PETROV, I. **Using Computing Containers and Continuous Integration to Improve Numerical Research Reproducibility**. 2018. Disponível em: https://ijcjournal.org/InternationalJournalOfComputer/article/view/1249/503.

VITE. **Vite: Next Generation Frontend Tooling. [S.l.]: Vite.** 2025. Disponível em: https://vitejs.dev/.

APÊNDICES

APÊNDICE A – Contrato de Dados

Este apêndice apresenta os *payloads* (contratos) da API de integração do sistema, sendo eles: Ordem de Carregamento, Usuários e Docas.

Listagem 1 – Modelo JSON de Usuários

```
{
  "source_system": "SISTEMA_A",
  "user": {
    "external_id": "123463",
    "name": "teste user",
    "email": "teste@email.com"
  }
}
```

Fonte: Autoria própria (2026).

Listagem 2 – Modelo JSON de Doca e Localização

```
{
  "source_system": "SAP",
  "dock": {
    "external_id": "ERP-DOCK-123",
    "dock_code": "DOCK 01",
    "description": "Pátio A",
    "location": "Matriz"
  }
}
```

Fonte: Autoria própria (2026).

Listagem 3 – Modelo JSON das Ordens de Carregamento

```
{
  "source_system": "SAP",
  "loadingOrder": {
    "external_id": "TESTE-2025",
    "issue_date": "2025-11-05",
    "status": "pending",
    "customer": {
      "external_id": "999888777",
      "name": "Tech Store Brasil",
      "document": "9998887770001",
      "email": "logistica@techstore.com.br",
      "phone": "1133224455"
    },
    "destination": {
      "external_id": "WH-CTBA-01",
      "name": "Centro de Distribuição Curitiba",
      "address": "Rua das Indústrias, 500",
      "postal_code": "80010000",
      "city": "Curitiba",
      "state": "PR"
    },
    "carrier": {
      "external_id": "202",
      "name": "Rapidão Trans",
      "document": "202202"
    },
    "vehicle": {
      "external_id": "ABC1234",
      "vehiclePlate": "ABC1234",
      "model": "Mercedes-Benz Accelo"
    },
    "driver": {
      "external_id": "455",
      "name": "João Silva",
      "document": "455",
      "phone": "41999887766"
    },
    "items": [
      {
        "product_sku": "EL-001",
        "product_description": "Smart TV 55 pol",
        "quantity": 20,
        "unit": "un",
        "packages": [
          { "unique_package_code": "PKG-TV-101-A",
   "quantity_in_package": 10 },
          { "unique_package_code": "PKG-TV-101-B",
   "quantity_in_package": 10 }
        ]
      }
    ]
  }
}
```

Fonte: Autoria própria (2026).

APÊNDICE B – Diagrama Entidade-Relacionamento e Dicionário de Dados

Este apêndice documenta a estrutura completa do banco de dados do Expedisoft. A modelagem foi concebida em PostgreSQL 15 e organiza suas 25 tabelas em quatro grupos funcionais: **estrutura do sistema**, **entidades externas**, **entidades operacionais** e **entidades de auditoria e rastreabilidade**. Todas as chaves primárias utilizam o tipo UUID, decisão que reduz a previsibilidade dos identificadores e favorece a escalabilidade horizontal do banco.

As seções a seguir apresentam os diagramas recortados por grupo, a descrição das cardinalidades dos relacionamentos e o dicionário de dados de cada tabela. O diagrama completo, contendo todas as tabelas e seus relacionamentos, está disponível em: https://drive.google.com/file/d/1tXgLdh_Bqvkpw8cuLZTGaimXL2p3ZeQa/view?usp=share_link.

B.1 Organização e Cardinalidades

O esquema é estruturado conforme os grupos descritos abaixo.

Estrutura do sistema. As tabelas `users`, `sessions` e `personal_access_tokens` gerenciam autenticação e controle de sessão. Um usuário pode possuir múltiplos tokens de acesso (1:N entre `users` e `personal_access_tokens`) e múltiplas sessões simultâneas (1:N entre `users` e `sessions`).

Entidades externas. Representam os agentes e a infraestrutura logística importados do ERP: `customers`, `destinations`, `carriers`, `vehicles`, `drivers` e `docks`. As cardinalidades relevantes são:

- Uma transportadora (`carriers`) pode possuir múltiplos veículos (`vehicles`) e motoristas (`drivers`): 1:N em ambos os casos.
- Um motorista está vinculado a exatamente uma transportadora (N:1 de `drivers` para `carriers`).
- Uma doca (`docks`) pode ser alocada a múltiplas ordens de carregamento ao longo do tempo, mas cada ordem aponta para no máximo uma doca (N:1 de `loading_orders` para `docks`).

Entidades operacionais. Constituem o núcleo funcional do sistema: `loading_orders`, `order_items`, `packages` e `products`.

- Uma ordem de carregamento (`loading_orders`) contém um ou mais itens (`order_items`): 1:N.
- Cada item de ordem referencia um produto do catálogo (`products`): N:1.
- Cada item de ordem pode ser subdividido em um ou mais volumes físicos (`packages`): 1:N. Cada pacote possui um `unique_package_code` globalmente único, utilizado na leitura de QR Code.

Entidades de auditoria e rastreabilidade. Registram toda atividade operacional: `checklist_entries`, `scan_logs`, `photos`, `order_status_history` e `integration_logs`.

- Para cada leitura de QR Code bem-sucedida é criada uma entrada em `checklist_entries` (N:1 para `packages` e para `loading_orders`).
- `scan_logs` registra *todas* as tentativas de leitura — incluindo falhas —, com chaves estrangeiras opcionais definidas como `SET NULL` para preservar o histórico mesmo quando pacotes ou ordens forem removidos.
- Cada mudança de *status* de uma ordem gera um registro em `order_status_history` (1:N de `loading_orders`).
- Fotos são vinculadas a uma ordem (N:1 para `loading_orders`) e a um usuário que fez o *upload* (N:1 para `users`).

B.2 Diagramas por Grupo

A Figura 21 apresenta as tabelas de estrutura do sistema (`users`, `sessions`, `personal_access_tokens` e `password_reset_tokens`).

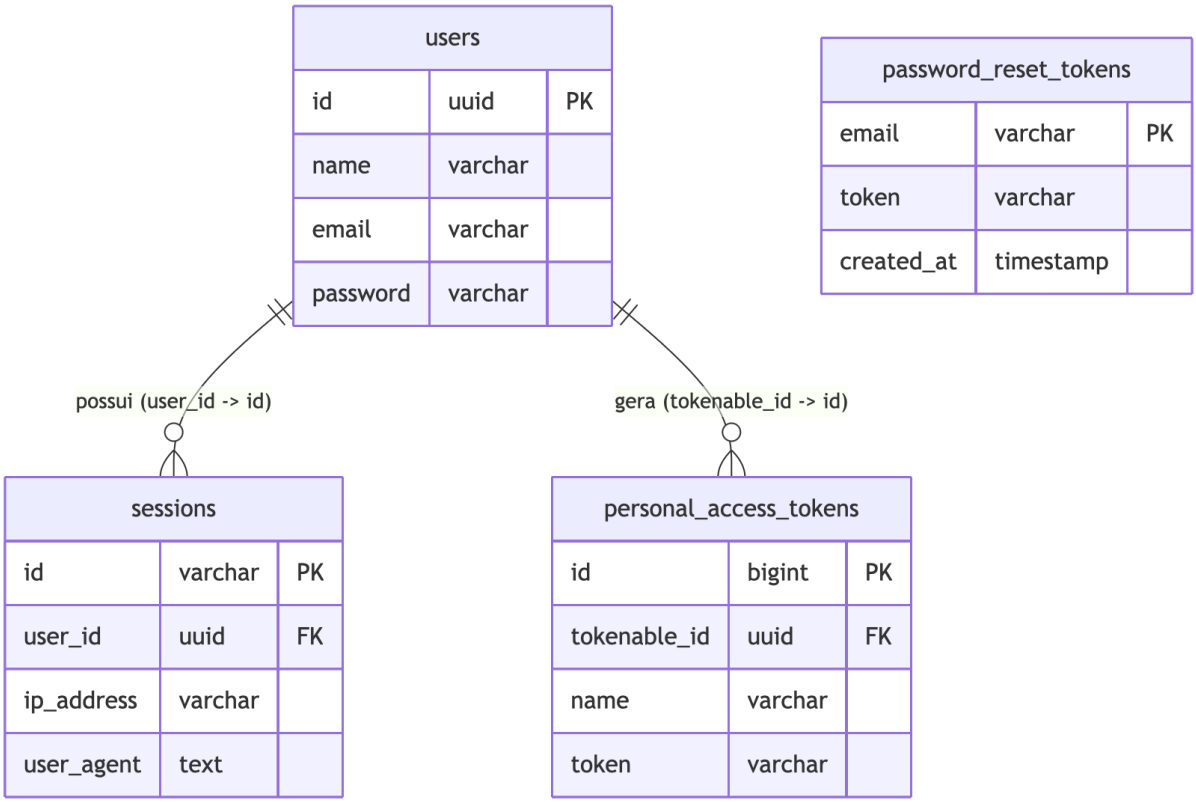

Figura 21 – Diagrama ER — Tabelas de Estrutura do Sistema.

A Figura 22 apresenta as entidades externas importadas do ERP: `carriers`, `vehicles`, `drivers`, `customers`, `destinations` e `docks`.

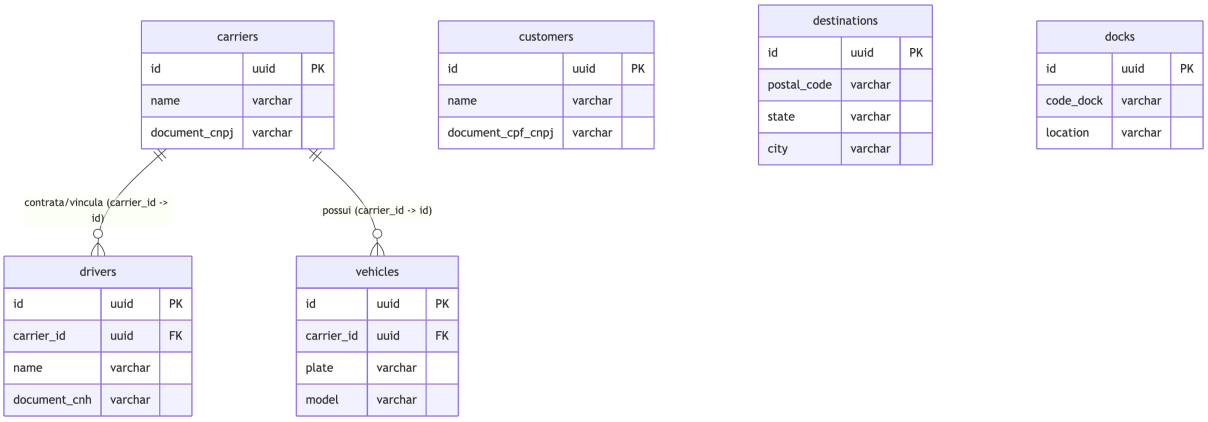

Figura 22 – Diagrama ER — Entidades Externas (ERP).

A Figura 23 apresenta as entidades operacionais: `loading_orders`, `order_items`, `packages` e `products`.

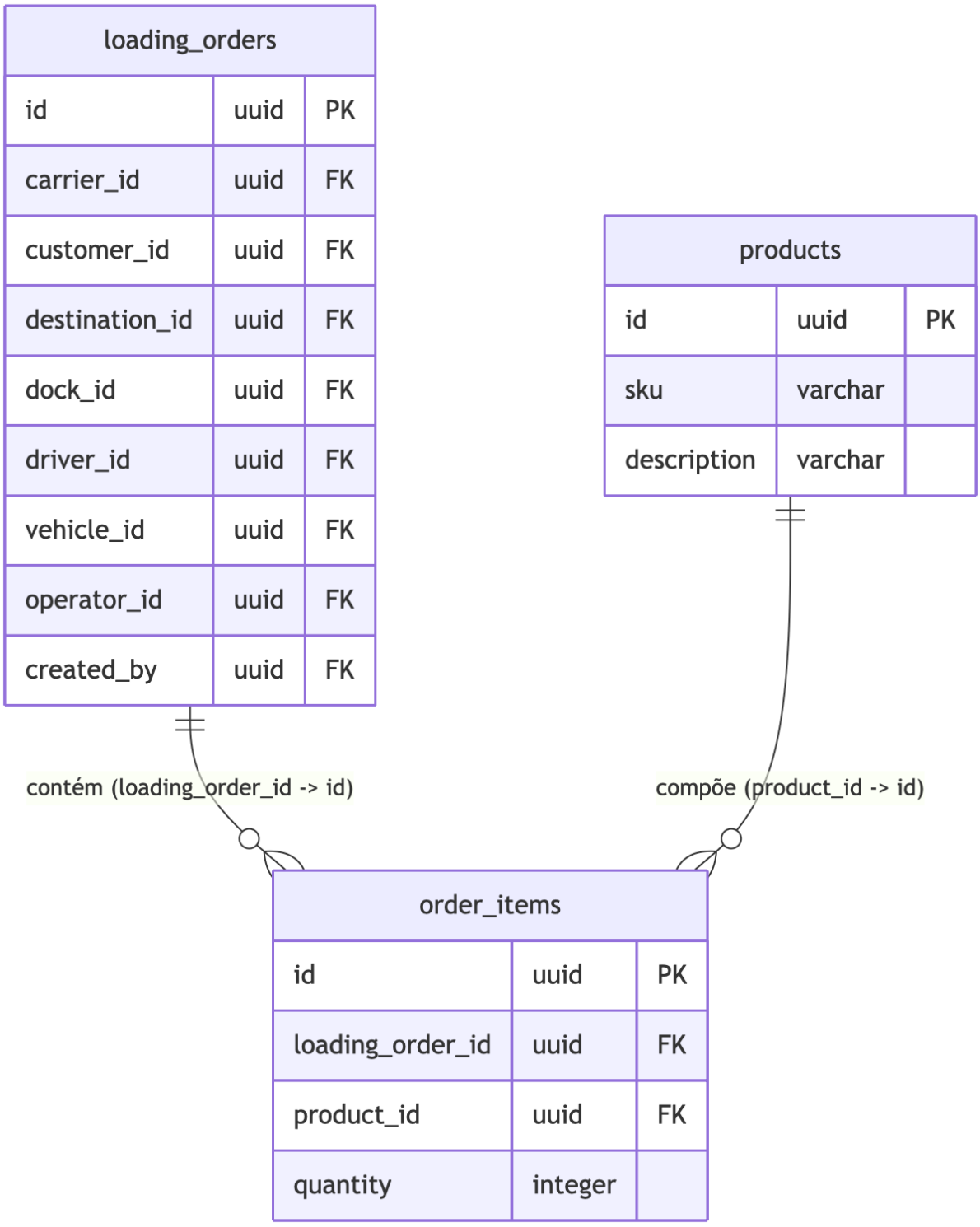

Figura 23 – Diagrama ER — Entidades Operacionais.

A Figura 24 apresenta as entidades de auditoria e rastreabilidade: `checklist_entries`, `scan_logs`, `photos`, `order_status_history` e `integration_logs`.

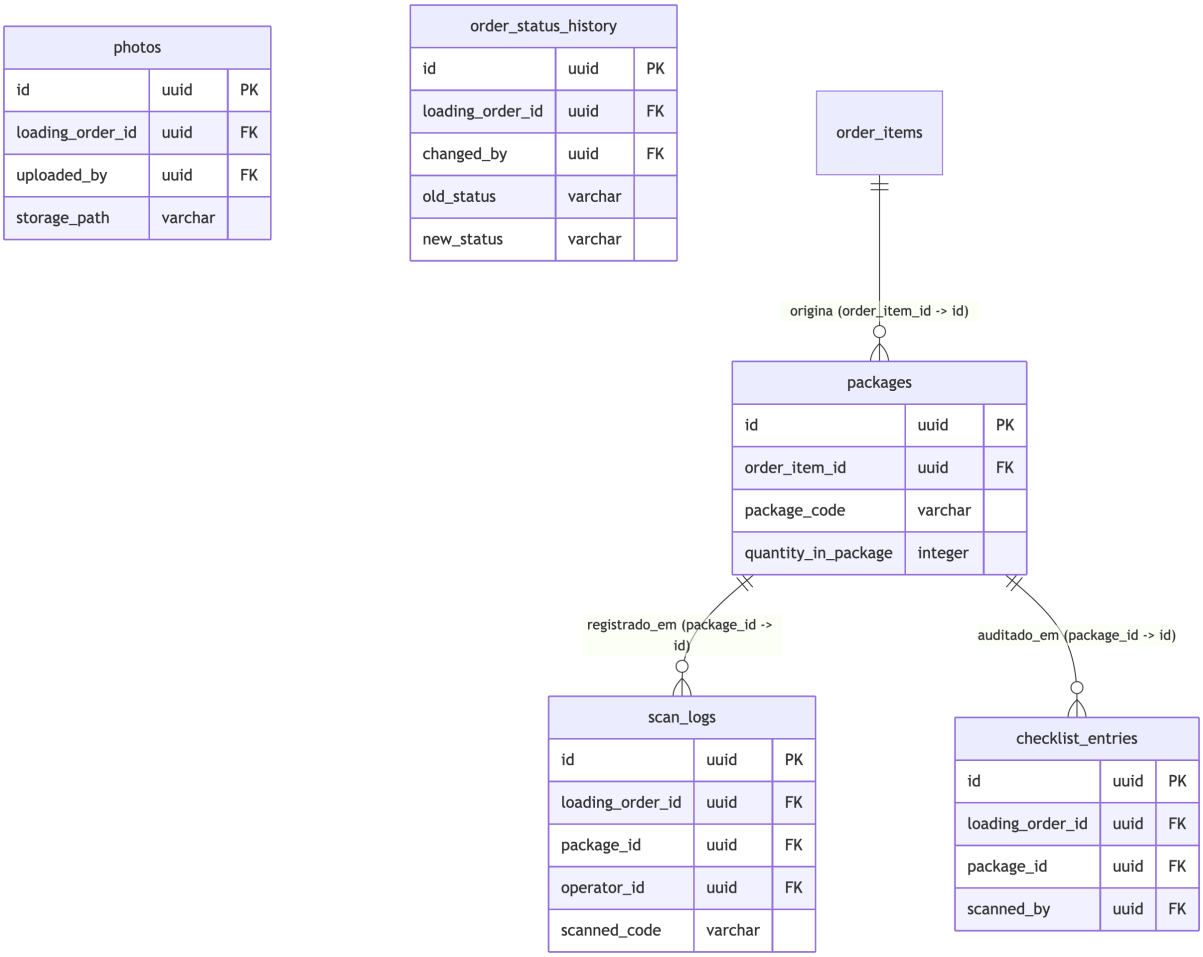

Figura 24 – Diagrama ER — Entidades de Auditoria e Rastreabilidade.

B.3 Dicionário de Dados

As tabelas a seguir descrevem cada coluna do esquema, indicando tipo de dado, restrições e propósito. A notação adota as seguintes convenções: **PK** = chave primária; **FK** = chave estrangeira; **AK** = chave alternativa (índice único); **NN** = não nulo; **D** = valor padrão (*default*).

B.3.1 Tabela `users`

Armazena os usuários do sistema, tanto importados do ERP quanto criados internamente. O campo `rule` diferencia gestores de operadores.

Coluna	Tipo	Restrições	Descrição
id	uuid	PK, NN	Identificador único do usuário.
external_id	varchar(255)	AK	Vínculo com o identificador nativo do ERP.
source_system	varchar(255)	—	Origem do usuário (ex.: "ERP", "CRM").
name	varchar(255)	NN	Nome completo do usuário.
email	varchar(255)	NN, AK	E-mail utilizado como credencial de acesso.
email_verified_at	timestamp	—	Data de verificação do e-mail.
password	varchar(255)	NN	Hash da senha (bcrypt).
rule	varchar(255)	NN, D=operador	Perfil de acesso: gestor ou operador.
remember_token	varchar(100)	—	Token de sessão persistente.
created_at	timestamp	—	Data de criação do registro.
updated_at	timestamp	—	Data da última atualização.

Tabela 16 – Dicionário de dados — `users`

B.3.2 Tabela `sessions`

Gerencia as sessões ativas dos usuários autenticados.

Coluna	Tipo	Restrições	Descrição
id	varchar(255)	PK, NN	Identificador da sessão.
user_id	bigint	—	Referência ao usuário (pode ser nulo para sessões anônimas).
ip_address	varchar(45)	—	Endereço IP da origem da sessão.
user_agent	text	—	Agente de usuário (navegador/dispositivo).
payload	text	NN	Dados serializados da sessão.
last_activity	integer	NN	Timestamp Unix da última atividade.

Tabela 17 – Dicionário de dados — `sessions`

B.3.3 Tabela `personal_access_tokens`

Armazena os tokens de autenticação emitidos pelo Laravel Sanctum para acesso às APIs.

Coluna	Tipo	Restrições	Descrição
id	bigint	PK, NN	Identificador sequencial do token.
tokenable_type	varchar(255)	NN	Tipo do modelo ao qual o token pertence.
tokenable_id	uuid	NN	ID do modelo ao qual o token pertence.
name	text	NN	Nome descritivo do token.
token	varchar(64)	NN, AK	Hash do token de acesso.
abilities	text	—	Permissões associadas ao token (JSON).
last_used_at	timestamp	—	Data do último uso do token.
expires_at	timestamp	—	Data de expiração do token.
created_at	timestamp	—	Data de criação.
updated_at	timestamp	—	Data da última atualização.

Tabela 18 – Dicionário de dados — `personal_access_tokens`

B.3.4 Tabela `password_reset_tokens`

Controla os tokens temporários emitidos para redefinição de senha.

Coluna	Tipo	Restrições	Descrição
email	varchar(255)	PK, NN	E-mail do usuário solicitante.
token	varchar(255)	NN	Token de redefinição (hash).
created_at	timestamp	—	Data de emissão do token.

Tabela 19 – Dicionário de dados — `password_reset_tokens`

B.3.5 Tabela `customers`

Representa os clientes destinatários das ordens de carregamento, sincronizados a partir do ERP.

Coluna	Tipo	Restrições	Descrição
id	uuid	PK, NN	Identificador interno.
external_id	varchar(255)	AK	ID do cliente no ERP de origem.
source_system	varchar(255)	—	Sistema de origem (ex.: "SAP").
name	varchar(255)	NN	Razão social do cliente.
document	varchar(255)	AK	CNPJ/CPF do cliente.
email	varchar(255)	—	E-mail de contato.
phone	varchar(255)	—	Telefone de contato.
created_at	timestamp	—	Data de criação.
updated_at	timestamp	—	Data da última atualização.

Tabela 20 – Dicionário de dados — `customers`

B.3.6 Tabela `destinations`

Armazena os endereços de destino das cargas exportadas.

Coluna	Tipo	Restrições	Descrição
`id`	uuid	PK, NN	Identificador interno.
`external_id`	varchar(255)	AK	ID do destino no ERP.
`source_system`	varchar(255)	—	Sistema de origem.
`name`	varchar(255)	NN	Nome do local de destino.
`address`	text	—	Endereço completo.
`postal_code`	varchar(10)	—	CEP/Código postal.
`city`	varchar(255)	—	Cidade de destino.
`state`	varchar(2)	—	UF/Estado de destino.
`created_at`	timestamp	—	Data de criação.
`updated_at`	timestamp	—	Data da última atualização.

Tabela 21 – Dicionário de dados — `destinations`

B.3.7 Tabela `carriers`

Cadastro de transportadoras responsáveis pelo transporte das cargas.

Coluna	Tipo	Restrições	Descrição
`id`	uuid	PK, NN	Identificador interno.
`external_id`	varchar(255)	AK	ID da transportadora no ERP.
`source_system`	varchar(255)	—	Sistema de origem.
`name`	varchar(255)	NN	Razão social da transportadora.
`document`	varchar(255)	—	CNPJ da transportadora.
`contact_phone`	varchar(255)	—	Telefone de contato.
`created_at`	timestamp	—	Data de criação.
`updated_at`	timestamp	—	Data da última atualização.

Tabela 22 – Dicionário de dados — `carriers`

B.3.8 Tabela `vehicles`

Veículos vinculados às transportadoras utilizados nos carregamentos.

Coluna	Tipo	Restrições	Descrição
id	uuid	PK, NN	Identificador interno.
external_id	varchar(255)	AK	ID do veículo no ERP.
source_system	varchar(255)	—	Sistema de origem.
vehiclePlate	varchar(255)	NN, AK	Placa do veículo.
model	varchar(255)	—	Modelo do veículo.
carrier_id	uuid	FK, NN	Transportadora responsável pelo veículo.
created_at	timestamp	—	Data de criação.
updated_at	timestamp	—	Data da última atualização.

Tabela 23 – Dicionário de dados — `vehicles`

B.3.9 Tabela `drivers`

Motoristas vinculados às transportadoras, associados às ordens de carregamento.

Coluna	Tipo	Restrições	Descrição
id	uuid	PK, NN	Identificador interno.
external_id	varchar(255)	AK	ID do motorista no ERP.
source_system	varchar(255)	—	Sistema de origem.
name	varchar(255)	NN	Nome completo do motorista.
document	varchar(255)	AK	CPF do motorista.
phone	varchar(255)	—	Telefone de contato.
carrier_id	uuid	FK, NN	Transportadora à qual o motorista pertence.
created_at	timestamp	—	Data de criação.
updated_at	timestamp	—	Data da última atualização.

Tabela 24 – Dicionário de dados — `drivers`

B.3.10 Tabela `docks`

Docas físicas do armazém disponíveis para alocação nos carregamentos.

Coluna	Tipo	Restrições	Descrição
id	uuid	PK, NN	Identificador interno.
external_id	varchar(255)	AK	ID da doca no ERP.
source_system	varchar(255)	—	Sistema de origem.
dock_code	varchar(255)	NN, AK	Código abreviado da doca (ex.: DOCK-01).
description	varchar(255)	—	Descrição textual da doca.
location	varchar(255)	—	Unidade, filial ou pátio onde a doca está instalada.
created_at	timestamp	—	Data de criação.
updated_at	timestamp	—	Data da última atualização.

Tabela 25 – Dicionário de dados — docks

B.3.11 Tabela products

Catálogo normalizado de produtos, eliminando redundâncias de cadastro entre ordens.

Coluna	Tipo	Restrições	Descrição
id	uuid	PK, NN	Identificador interno.
sku	varchar(255)	NN, AK	Código único de identificação do produto.
description	varchar(255)	NN	Descrição comercial do produto.
weight	numeric(10,3)	—	Peso unitário do produto (kg).
unit	varchar(10)	NN, D=un	Unidade de medida (ex.: un, cx).
barcode	varchar(255)	—	Código de barras do produto (EAN).
created_at	timestamp	—	Data de criação.
updated_at	timestamp	—	Data da última atualização.

Tabela 26 – Dicionário de dados — products

B.3.12 Tabela loading_orders

Tabela central do sistema. Consolida todos os dados de uma ordem de carregamento, vinculando clientes, transportadoras, motoristas, veículos, docas, operadores e gestores.

Coluna	Tipo	Restrições	Descrição
id	uuid	PK, NN	Identificador interno da ordem.
external_id	varchar(255)	NN, AK	ID da ordem no ERP de origem.
source_system	varchar(255)	—	Sistema que originou a ordem.
issue_date	date	NN	Data de emissão da ordem.
status	varchar(255)	NN, D=pendente	Situação atual: pendente, em_andamento, concluido, divergencia.
customer_id	uuid	FK, NN	Cliente destinatário da carga.
destination_id	uuid	FK, NN	Endereço de destino da carga.
carrier_id	uuid	FK, NN	Transportadora responsável.
vehicle_id	uuid	FK, NN	Veículo designado para o transporte.
driver_id	uuid	FK, NN	Motorista responsável pela entrega.
dock_id	uuid	FK	Doca de carregamento alocada (opcional).
operator_id	uuid	FK	Operador responsável pela conferência.
created_by	uuid	FK	Gestor que criou/agendou a ordem.
justification	text	—	Justificativa obrigatória quando há divergência.
observations	text	—	Observações livres do operador.
scheduled_at	timestamp	—	Data e hora agendadas para o carregamento.
started_at	timestamp	—	Data e hora de início efetivo da conferência.
completed_at	timestamp	—	Data e hora de conclusão do carregamento.
created_at	timestamp	—	Data de criação do registro.
updated_at	timestamp	—	Data da última atualização.

Tabela 27 – Dicionário de dados — `loading_orders`

B.3.13 Tabela `order_items`

Itens que compõem uma ordem de carregamento, vinculados ao catálogo de produtos.

Coluna	Tipo	Restrições	Descrição
id	uuid	PK, NN	Identificador interno.
loading_order_id	uuid	FK, NN	Ordem de carregamento à qual o item pertence.
product_id	uuid	FK, NN	Produto referenciado no catálogo.
quantity	integer	NN	Quantidade total do produto nesta ordem.
note	text	—	Observação específica sobre o item.
created_at	timestamp	—	Data de criação.
updated_at	timestamp	—	Data da última atualização.

Tabela 28 – Dicionário de dados — `order_items`

B.3.14 Tabela `packages`

Representa cada volume físico individual de um item de ordem. O campo `unique_package_code` é o código lido pelo QR Code durante a conferência.

Coluna	Tipo	Restrições	Descrição
id	uuid	PK, NN	Identificador interno.
order_item_id	uuid	FK, NN	Item de ordem ao qual o pacote pertence.
unique_package_code	varchar(255)	NN, AK	Código único do volume — escaneado via QR Code.
quantity_in_package	integer	NN, D=1	Quantidade de itens dentro deste pacote.
created_at	timestamp	—	Data de criação.
updated_at	timestamp	—	Data da última atualização.

Tabela 29 – Dicionário de dados — `packages`

B.3.15 Tabela `checklist_entries`

Registra cada volume confirmado com sucesso durante a conferência. Uma entrada nesta tabela significa que o pacote foi validado e aceito para a ordem em questão. O campo `scanned_code` possui restrição de unicidade global, impedindo que o mesmo volume seja confirmado mais de uma vez em qualquer ordem.

Coluna	Tipo	Restrições	Descrição
id	uuid	PK, NN	Identificador interno.
loading_order_id	uuid	FK, NN	Ordem de carregamento em que a leitura ocorreu.
package_id	uuid	FK, NN	Pacote que foi confirmado.
scanned_by	uuid	FK, NN	Usuário (operador) que realizou a leitura.
scanned_code	varchar(255)	NN, AK	Código exato lido pelo dispositivo.
scanned_at	timestamp	NN, D=now	Data e hora da leitura.
created_at	timestamp	—	Data de criação.
updated_at	timestamp	—	Data da última atualização.

Tabela 30 – Dicionário de dados — `checklist_entries`

B.3.16 Tabela `scan_logs`

Registra *todas* as tentativas de leitura de QR Code, incluindo falhas. As chaves estrangeiras são definidas como `SET NULL` para que o histórico de leituras seja preservado mesmo após a remoção de ordens ou pacotes. A coluna `status` aceita apenas os valores `success` ou `error`, controlados por uma restrição *CHECK*.

Coluna	Tipo	Restrições	Descrição
id	uuid	PK, NN	Identificador interno.
loading_order_id	uuid	FK (SET NULL)	Ordem de carregamento associada (pode ser nulo).
operator_id	uuid	FK (SET NULL)	Operador que realizou a tentativa (pode ser nulo).
package_id	uuid	FK (SET NULL)	Pacote lido (pode ser nulo em caso de código inválido).
scanned_code	varchar(255)	NN	Código exato lido pelo dispositivo.
payload	json	—	Corpo completo da requisição de bipagem.
status	varchar(255)	NN, CHECK	Resultado: `success` ou `error`.
error_message	varchar(255)	—	Mensagem de erro quando `status = error`.
scanned_at	timestamp	NN, D=now	Data e hora da tentativa de leitura.
created_at	timestamp	—	Data de criação.
updated_at	timestamp	—	Data da última atualização.

Tabela 31 – Dicionário de dados — `scan_logs`

B.3.17 Tabela `photos`

Armazena os metadados das fotografias capturadas pelo operador durante o encerramento do carregamento. O arquivo binário é armazenado externamente no *Google Drive*; esta tabela mantém apenas a referência lógica (`drive_id`) e o caminho temporário no servidor (`storage_path`).

Coluna	Tipo	Restrições	Descrição
`id`	uuid	PK, NN	Identificador interno.
`loading_order_id`	uuid	FK, NN	Ordem à qual a foto está vinculada.
`storage_path`	varchar(255)	—	Caminho temporário no disco do servidor.
`drive_id`	varchar(255)	—	ID do arquivo no *Google Drive* (preenchido após upload).
`mime`	varchar(255)	NN	Tipo MIME do arquivo (ex.: `image/jpeg`).
`status`	varchar(255)	NN, D=`pending`	Situação do upload: `pending` ou `uploaded`.
`uploaded_by`	uuid	FK, NN	Usuário que enviou a foto.
`uploaded_at`	timestamp	NN, D=now	Data e hora do envio da foto.
`created_at`	timestamp	—	Data de criação.
`updated_at`	timestamp	—	Data da última atualização.

Tabela 32 – Dicionário de dados — `photos`

B.3.18 Tabela `order_status_history`

Registra cronologicamente cada alteração de *status* de uma ordem de carregamento, identificando o usuário responsável pela mudança. Permite a reconstrução completa do ciclo de vida de qualquer ordem.

Coluna	Tipo	Restrições	Descrição
`id`	uuid	PK, NN	Identificador interno.
`loading_order_id`	uuid	FK, NN	Ordem cujo status foi alterado.
`old_status`	varchar(255)	—	Status anterior à alteração (nulo na criação).
`new_status`	varchar(255)	NN	Novo status assumido pela ordem.
`changed_by`	uuid	FK, NN	Usuário que realizou a alteração.
`changed_at`	timestamp	NN, D=now	Data e hora da alteração de status.
`note`	text	—	Observação contextual sobre a mudança.

Tabela 33 – Dicionário de dados — `order_status_history`

B.3.19 Tabela `integration_logs`

Registra cada requisição recebida pelos *endpoints* de integração com ERPs externos, armazenando o *payload* completo e o resultado do processamento. Fundamental para diagnosticar falhas de integração.

Coluna	Tipo	Restrições	Descrição
`id`	uuid	PK, NN	Identificador interno.
`endpoint`	varchar(255)	NN	Rota da API que recebeu a requisição.
`payload`	jsonb	NN	Corpo completo da requisição recebida.
`http_status`	integer	NN	Código HTTP retornado pelo sistema.
`error_message`	text	—	Mensagem de erro, quando aplicável.
`received_at`	timestamp	NN, D=now	Data e hora do recebimento da requisição.

Tabela 34 – Dicionário de dados — `integration_logs`

B.3.20 Tabelas de Infraestrutura do Framework

As tabelas a seguir são geradas automaticamente pelo Laravel e suportam funcionalidades internas do *framework*. Não fazem parte do domínio de negócio, mas são necessárias para o funcionamento do sistema.

Tabela	Finalidade
`jobs`	Fila de *jobs* assíncronos pendentes de execução (processamento em segundo plano).
`job_batches`	Agrupamento de lotes de *jobs* para execução coordenada.
`failed_jobs`	Registro de *jobs* que falharam durante a execução, incluindo exceção e *payload*.
`cache`	Cache de dados com controle de expiração por chave.
`cache_locks`	Travas distribuídas para operações de cache concorrentes.
`migrations`	Histórico de migrações aplicadas ao banco de dados.

Tabela 35 – Tabelas de infraestrutura do Laravel