

# RELATÓRIO FINAL - SISTEMA TRANSLOG

**Disciplina:** Programação Orientada a Objetos

**Instituição:** iCEV - Instituto de Ensino Superior

**Data:** 25 de Novembro de 2025

**Equipe:**

- Dhian Pablo Dias Sepedro
- Arthur Hagel Porfirio Mendes
- Gabriel Leal Batista

**Professor:** Samuel Vinicius Pereira de Oliveira

---

## 1. FUNCIONALIDADES IMPLEMENTADAS PELO GRUPO

O grupo desenvolveu um sistema completo de gestão logística com as seguintes funcionalidades:

### 1.1. Cadastro de Clientes e Motoristas

- Cadastro de dois tipos de clientes: Empresariais (10% desconto) e Prioritários (20% desconto)
- Validação matemática completa de CPF/CNPJ com dígitos verificadores
- Cadastro de motoristas com validação básica de CNH (11 dígitos, sem repetições)
- Interface gráfica com formulário duplo lado a lado

### 1.2. Sistema de Cálculo de Frete

- Cálculo automático baseado em: distância × peso × tipo de carga
- Classificação automática de carga por peso: Leve ( $\leq 150\text{kg}$ ), Média (150-500kg), Pesada ( $> 500\text{kg}$ )
- Adicional de 40% para cargas especiais (frágeis/perigosas)
- Aplicação automática de desconto por tipo de cliente

### 1.3. Agendamento com Bloqueio Inteligente

- Sistema de bloqueio de horários: 1 hora a cada 100km de distância
- Validação automática de conflitos de agenda
- Prevenção de alocação dupla de motoristas

## 1.4. Emissão de Notas Fiscais

- Geração automática de nota fiscal em formato TXT
- Discriminação completa dos valores (base, adicional, desconto)
- Salvamento organizado na pasta `notas_fiscais/`

## 1.5. Persistência de Dados

- Salvamento automático em CSV ao fechar o sistema
- Carregamento automático ao iniciar
- Relacionamentos mantidos via CPF/CNPJ e CNH como chaves

## 1.6. Interface Gráfica Completa (Swing)

- **Tela 1 - Cadastros:** Formulário duplo para clientes e motoristas
  - **Tela 2 - Nova Entrega:** Cálculo em tempo real, seleção de combobox, máscara de data
  - **Tela 3 - Relatório:** Tabela de entregas com exportação para CSV
- 

# 2. FUNCIONALIDADES NÃO IMPLEMENTADAS

## 2.1. Controle de Penalidades por Atraso

**Motivo:** O projeto focou na fase de agendamento. Para calcular atrasos, seria necessário um módulo de rastreamento pós-entrega para registrar a data real de chegada. Priorizamos a robustez do algoritmo de bloqueio de agenda e a integridade dos dados principais.

## 2.2. Validação Matemática Completa de CNH

**Motivo:** O algoritmo oficial de validação da CNH possui variações históricas complexas (CNHs antigas, diferentes formatos) que fogem do propósito acadêmico. Implementamos validação básica (11 dígitos, sem repetições) que garante integridade suficiente para o sistema.

---

# 3. CONTRIBUIÇÕES INDIVIDUAIS

## 3.1. Dhian Pablo Dias Sepedro

**Responsabilidade:** Arquitetura geral, lógica de negócio e persistência

**O que fiz:**

- Estruturação do projeto em camadas (Model, View, Service, Persistence, Util)

- Implementação das classes de serviço: `CalculadoraFrete`, `Agendamento`, `NotaFiscal`
- Desenvolvimento do sistema de persistência em CSV (`ClienteRepository`, `MotoristaRepository`, `EntregaRepository`)
- Implementação da classe `Validador` com algoritmos de CPF/CNPJ
- Integração entre todas as camadas do sistema

**O que mais gostei:** Além de ver o polimorfismo e a modularização funcionando na prática. O ponto alto foi a evolução em conjunto com o código, identificar falhas nos testes, corrigir aos poucos, e durante essa correção, ter novas ideias para melhorar ainda mais a arquitetura em geral. Aprendi que uma boa arquitetura facilita muito a manutenção, quando surgiam falhas técnicas ou a necessidade de mudar alguma variável, era simples apenas mudar factualmente, sem quebrar o restante do sistema.

**Maiores dificuldades:** Implementar a persistência relacional em CSV. Especificamente, salvar uma `Entrega` e ao carregar reconectar esse objeto aos objetos `Cliente` e `Motorista` corretos na memória usando CPF/CNPJ e CNH como chaves.

### 3.2. Arthur

**Responsabilidade:** Tela de Cadastros

**O que fiz:**

- Desenvolvimento do `PanelCadastro` com formulário duplo (Cliente e Motorista)
- Implementação da validação em tempo real de CPF/CNPJ
- Integração com a classe `Validador` para verificação matemática dos documentos
- Layout responsivo mantendo clareza lado a lado

**O que mais gostei:** A dinâmica de trabalho em equipe, especialmente nas sessões de code review, que permitiram resolver problemas técnicos complexos de forma rápida e colaborativa.

**Maiores dificuldades:**

- Otimizar a lógica de validação dupla (CPF/CNPJ) para garantir que fosse rápida, em tempo real, sem causar lentidão na interface
- Garantir que o layout do formulário duplo fosse totalmente responsivo em diferentes resoluções

**Aprendizados:** Aprofundei conhecimento em algoritmos de validação de documentos fiscais e a importância de verificações robustas no frontend. Pratiquei integração com outras camadas do sistema, reforçando a importância do versionamento e de lógicas bem definidas.

### 3.3. Gabriel

**Responsabilidade:** Telas de Nova Entrega e Relatório

### O que fiz:

- Desenvolvimento do **PanelEntrega** com cálculo em tempo real
- Implementação da classificação automática de carga por peso
- Criação do **PanelListagem** com tabela de monitoramento
- Funcionalidade de exportação para CSV

**O que mais gostei:** Desenvolver funcionalidades que trazem eficiência, controle e confiabilidade à operação. Ver o sistema calculando automaticamente e impedindo conflitos de agenda em tempo real foi muito satisfatório.

**Maiores dificuldades:** Implementar a máscara de data/hora (dd/MM/yyyy HH:mm) e garantir que a validação de conflitos funcionasse corretamente, considerando a duração baseada na distância.

**Aprendizados:** Entendi na prática como componentes Swing interagem entre si e como eventos de interface podem disparar lógica de negócio de forma eficiente. A experiência com JTable e exportação de dados foi especialmente valiosa.

---

## 4. ARQUITETURA E PRINCÍPIOS DE POO

### 4.1. Organização em Camadas

```
br.edu.icev.translog/
└── model/      # Entidades (Cliente, Motorista, Entrega, Carga)
└── view/       # Interface Swing (JanelaPrincipal, Panels)
└── service/    # Lógica de negócios (CalculadoraFrete, Agendamento)
└── persistencia/ # Acesso a dados (Repositories CSV)
└── util/        # Validador de documentos
└── main/        # Ponto de entrada
```

### 4.2. Princípios POO Aplicados

**Abstração:** Classe **Cliente** abstrata define modelo base, impedindo instanciação genérica.

**Herança:** **ClienteEmpresarial** e **ClientePrioritario** herdam de **Cliente**, reutilizando código e especializando apenas o comportamento de desconto.

**Polimorfismo:** Método **obterDesconto()** sobrescrito nas subclasses. A **CalculadoraFrete** chama esse método sem saber o tipo específico do cliente, facilitando adição de novos tipos futuros (Princípio Aberto/Fechado).

**Encapsulamento:** Todos os atributos são **private** e acessados via métodos, protegendo a integridade dos dados.

---

## 5. BIBLIOTECAS UTILIZADAS

Utilizamos apenas bibliotecas padrão do Java JDK 8+:

- **java.util** (List, ArrayList, Scanner): Estruturas de dados
- **java.io** (BufferedReader, BufferedWriter, FileReader, FileWriter): Manipulação de arquivos CSV
- **javax.swing** (JFrame, JPanel, JTable, JComboBox, etc.): Interface gráfica
- **java.time** (LocalDateTime, DateTimeFormatter): Manipulação de datas e cálculo de intervalos

**Motivo da escolha:** Evitar dependências externas, facilitando compilação e execução em qualquer ambiente Java. Bibliotecas nativas são estáveis, bem documentadas e suficientes para o escopo acadêmico do projeto.

---

## 6. REFERÊNCIAS CONSULTADAS

### 1. Oracle Java Documentation

- <https://docs.oracle.com/javase/8/docs/api/>
- Consultada para: LocalDateTime, Swing components, File I/O

### 2. Stack Overflow

- <https://stackoverflow.com/questions/tagged/java>
- Consultada para: Validação de documentos, máscaras de data, eventos Swing

### 3. CadCobol - Validação de Documentos

- [https://www.cadcobol.com.br/calcula\\_cpf\\_cnpj\\_caepf.htm](https://www.cadcobol.com.br/calcula_cpf_cnpj_caepf.htm)
- Consultada para: Algoritmos matemáticos de CPF/CNPJ

### 4. DevMedia - Tutoriais Java

- <https://www.devmedia.com.br/>
- Consultada para: Padrões de projeto e boas práticas

### 5. YouTube - Tutoriais Swing

- Diversos vídeos sobre JTable, Layout Managers e Event Handling
- 

## 7. CLASSE IMPORTANTE DO PROJETO

## Agendamento.java

Esta é a classe mais importante do sistema por conter o coração das regras de negócio.

```
//agenda nova entrega
public void agendarEntrega(Entrega nova) throws Exception {

    //calcula a "duracao" da nova entrega, a cada 100km bloqueia 1h, arredonda pra cima
    long duracaoNovaHoras = (long) Math.ceil(nova.getDistanciaKm() / 100.0);
    if (duracaoNovaHoras < 1) duracaoNovaHoras = 1;

    //bloqueia o horario
    LocalDateTime inicioNova = nova.getDataHorario();
    LocalDateTime fimNova = inicioNova.plusHours(duracaoNovaHoras);

    //verifica os possiveis conflitos
    for (Entrega agendada : entregasAgendadas) {

        //verifica se é o mesmo motorista
        if (agendada.getMotorista().getCnh().equals(nova.getMotorista().getCnh())) {

            //confere que horas termina a entrega agendada
            long duracaoAgendadaHoras = (long) Math.ceil(agendada.getDistanciaKm() / 100.0);
            if (duracaoAgendadaHoras < 1) duracaoAgendadaHoras = 1;

            LocalDateTime inicioAgendada = agendada.getDataHorario();
            LocalDateTime fimAgendada = inicioAgendada.plusHours(duracaoAgendadaHoras);

            //verifica se os horarios se sobreponem
            if (inicioNova.isBefore(fimAgendada) && inicioAgendada.isBefore(fimNova)) {

                throw new Exception("CONFLITO DE ROTA:\n" +
                    "O motorista " + nova.getMotorista().getNome() + " já possui entrega neste horário.\n" +
                    "Entrega Existente: " + inicioAgendada + " até " + fimAgendada + "\n" +
                    "(Bloqueio de 1h a cada 100km)");
            }
        }
    }
}
```

Por que esta classe é importante:

1. **Algoritmo de Detecção de Conflitos:** Implementa lógica não-trivial de verificação de sobreposição de intervalos temporais.
2. **Validação Crítica de Integridade:** Previne estado inconsistente do sistema (motorista em dois lugares simultaneamente). Esta classe é uma linha de defesa contra dados corrompidos, demonstrando pensamento sobre robustez e confiabilidade.
3. **Lógica Temporal Dinâmica:** Calcula janelas de tempo variáveis baseadas em regra (1h/100km).
4. **Escalabilidade através de Iteração:** Verifica todos os agendamentos existentes de forma eficiente. Embora seja O(n), a implementação é clara e poderia ser otimizada para O(log n) com estruturas de árvores em versões futuras.

---

## 8. CONCLUSÃO

O Sistema Translog atende completamente aos requisitos obrigatórios e vai além com interface gráfica, validações robustas e sistema inteligente de bloqueio. A experiência foi

fundamental para consolidar conhecimentos de POO e preparar a equipe para projetos profissionais.

Como melhorias futuras, poderíamos implementar: banco de dados relacional, relatórios em PDF, sistema de autenticação, monitoramento da entrega e testes unitários automatizados