# Documento Técnico: Simulação de Coleta de Lixo

Sistema Desenvolvido para Simulação de Gestão de Resíduos

Maio de 2025

## 1 Introdução

Este documento descreve a modelagem, as estruturas de dados e os algoritmos implementados em um sistema de simulação de coleta de lixo desenvolvido em Java. O sistema simula a coleta de lixo em cinco zonas geográficas (Sul, Sudeste, Centro, Leste, Norte), utilizando caminhões pequenos para coletar lixo nas zonas e transferilo para estações de transferência, e caminhões grandes para transportar o lixo das estações ao aterro sanitário. O objetivo é gerenciar a logística de coleta, considerando tempos de trajeto, períodos de pico de tráfego e capacidades limitadas dos veículos.

## 2 Modelagem do Sistema

### 2.1 Visão Geral

O sistema é modelado como uma simulação baseada em eventos discretos, onde as operações (coleta, transferência, descarregamento) são representadas por eventos agendados e processados em ordem cronológica. A simulação ocorre em dias, com cada dia incluindo geração de lixo, coleta, transferência e descarregamento. As principais entidades e suas interações são:

- Zonas: Áreas geográficas que geram quantidades aleatórias de lixo diariamente, com limites definidos.
- Caminhões Pequenos: Veículos com capacidades de 2t, 4t , 8t, 10t que coletam lixo nas zonas e o levam às estações de transferência. Cada caminhão tem uma rota predefinida e um número máximo de viagens diárias.
- Estações de Transferência: Locais (A e B) que recebem lixo de caminhões pequenos, armazenam em filas se necessário, e transferem para caminhões grandes.
- Caminhões Grandes: Veículos com capacidade fixa de 20t, que transportam lixo das estações ao aterro sanitário.
- Eventos: Ações agendadas (ex.: coleta, transferência) que avançam a simulação.

## 2.2 Arquitetura

O sistema é organizado em pacotes Java:

- simulador.caminhoes: Classes CaminhaoPequeno e CaminhaoGrande.
- simulador.estacoes: Classe EstacaoDeTransferencia.
- simulador.eventos: Classes para eventos (Evento, EventoColeta, EventoTransferenciaParaEs etc.) e AgendaEventos.
- simulador.zona: Classes Zona, Zonas, e GerenciadorZonas.
- simulador.util: Classes GerenciadorTempo e TempoDetalhado.
- simulador.configuracao: Classe configuracao com constantes.

A classe Simulador coordena a simulação, inicializando zonas, estações, caminhões e processando eventos via AgendaEventos.

## 2.3 Fluxo Operacional

1. Inicialização: O Simulador cria cinco zonas e duas estações (A: Leste, Norte, Centro; B: Sul, Sudeste).
2. Geração de Lixo: Cada zona gera lixo diário aleatoriamente, conforme limites em configuracao.
3. Criação de Caminhões: EventoDistribuidorDeRotas cria caminhões pequenos com rotas e agenda coletas iniciais.
4. Coleta: EventoColeta gerencia a coleta de lixo por caminhões pequenos, respeitando capacidade e viagens disponíveis.
5. Transferência: EventoTransferenciaParaEstacao desloca caminhões pequenos às estações, calculando tempos de trajeto.
6. EstacaoDeTransferencia descarrega caminhões pequenos diretamente ou os enfileira, criando caminhões grandes via EventoGerarCaminhaoGrande se necessário.
7. Descarregamento: Caminhões grandes cheios são descarregados no aterro.
8. Fim do Dia: Simulador exibe um resumo (lixo remanescente, tempos, caminhões) e reseta a agenda para o próximo dia.

# 3 Estruturas de Dados

As estruturas de dados são implementadas no pacote estruturas, com duas classes principais:

- Lista<T>:
  - Descrição: Lista simplesmente encadeada genérica, usada para armazenar zonas, caminhões pequenos e eventos.
  - Métodos Principais:
    - adicionar(int indice, T valor): Insere um elemento em um índice.
    - adicionarOrdenado(T valor, Comparator<T> comparador): Insere em ordem, usado em AgendaEventos.
    - removerHead(): Remove e retorna o primeiro elemento.
    - removerProcurado(T valor): Remove um elemento específico.
    - getValor(int indice): Acessa um elemento por índice.
    - estaVazia(): Verifica se a lista está vazia.
    - getTamanho(): Retorna o tamanho.
  - Uso:
    - AgendaEventos: Armazena eventos ordenados por tempo.
    - Simulador: Lista de zonas.
    - CaminhaoPequeno: Rota de zonas.
    - GerenciadorZonas: Listas de zonas e caminhões.
    - EventoDistribuidorDeRotas: Caminhões criados.
  - Vantagens: Inserção ordenada eficiente para eventos, acesso rápido por índice para zonas e caminhões.
  - Limitações: Remoção de elementos específicos tem complexidade $O(n)$.
- Fila<T>:
  - Descrição: Fila genérica (FIFO), usada para gerenciar caminhões pequenos em espera nas estações.
  - Métodos Principais:
    - enqueue(T valor): Adiciona ao final.
    - poll(): Remove e retorna o primeiro elemento.
    - isEmpty(): Verifica se está vazia.
    - size(): Retorna o tamanho.
  - Uso: EstacaoDeTransferencia armazena caminhões pequenos quando o caminhão grande está cheio.

- Vantagens: Implementação simples e eficiente para FIFO (inserção e remoção em $O(1)$).
- Limitações: Não suporta acesso aleatório ou ordenação.

# 4 Algoritmos Implementados

## 4.1 Gestão de Eventos (AgendaEventos)

- Descrição: Processa eventos em ordem cronológica, mantendo a simulação temporalmente consistente.
- Algoritmo (processarEventos):
  1. Enquanto eventos não estiver vazia:
     - Remover o primeiro evento (removerHead).
     - Atualizar tempoUltimoEvento e ultimoEventoExecutado. – Executar o evento (evento.executar()).
- Complexidade:
  - Inserção ordenada: $O(n)$ devido à busca na lista encadeada.
  - Remoção do início: $O(1)$.
  - Processamento: $O(n \cdot E)$, onde $n$ é o número de eventos e $E$ é o custo de executar.
- Exemplo de Uso: Processa EventoColeta, EventoTransferenciaParaEstacao, etc., em ordem de tempo.

## 4.2 Cálculo de Tempos (GerenciadorTempo.calcularTempoDetalhado)

- Descrição: Calcula tempos de coleta, trajeto e extra carga, considerando horários de pico e carga.
- Algoritmo:
  1. Verificar período de pico (isPeriodoCongestionado).
  2. Selecionar tempoMinimo e tempoMaximo com base no horário.
  3. Gerar tempoBaseViagem aleatoriamente (ThreadLocalRandom).
  4. Calcular tempoViagem com estimarTempoViagem, aplicando MULTIPLICADOR$_T$*EMPO*$_P$*ICO*(1.5)*o*
- 4. Complexidade: $O(d)$, onde $d$ é a duração base da viagem (devido ao loop em estimarTempoViagem).

- Exemplo de Uso: Calcula tempos para EventoColeta (ex.: 80 minutos para 8t) e EventoTransferenciaParaEstacao (ex.: 110 minutos de trajeto).

## 4.3 Coleta de Lixo (EventoColeta.executar)

- Descrição: Gerencia a coleta de lixo por um caminhão pequeno em uma zona.
- Algoritmo:
  1. Se a zona está limpa (lixoAcumulado == 0):
     - Registrar viagem.
     - Se há viagens disponíveis, tentar atualizarProximaZonaAlvo ou ir à estação.
  2. Caso contrário:
     - Enquanto houver viagens, capacidade e lixo:
       * Calcular quantidade a coletar (min(lixoZona, espacoCaminhao)).
       * Executar caminhao.coletar e zona.coletarLixo.
       * Acumular totalColetado.
     - Calcular tempos com GerenciadorTempo.
     - Se coletou e há viagens, agendar nova EventoColeta. – Senão, agendar EventoTransferenciaParaEstacao.
- Complexidade: $O(c)$, onde $c$ é o número de iterações de coleta (proporcional à capacidade do caminhão).
- Exemplo de Uso: Caminhão C1 coleta 2t na zona Sul, agenda nova coleta ou transferência.

## 4.4 Distribuição de Rotas (EventoDistribuidorDeRotas.distribuir)

- Descrição: Cria caminhões pequenos e atribui rotas balanceadas.
- Algoritmo:
  1. Para cada caminhão (1 a quantidadeCaminhoes):
     - Criar lista de zonas (rotaCaminhao).
     - Para cada viagem (1 a viagensPorCaminhao):
       * Atribuir zona zonas.getValor((i+j) % quantidadeZonas).
     - Criar CaminhaoPequeno com id, capacidade, viagens e rota. – Agendar EventoColeta inicial.
  2. Retornar lista de caminhões.

- Complexidade: $O(n \cdot v)$, onde $n$ é o número de caminhões e $v$ é o número de viagens por caminhão.
- Exemplo de Uso: Cria 8 caminhões de 2t com 5 viagens cada, distribuindo zonas ciclicamente.

### 4.5 Gestão de Filas (EstacaoDeTransferencia.receberCaminhaoPequeno)

- Descrição: Gerencia a chegada de caminhões pequenos, descarregando ou enfileirando.
- Algoritmo:
  1. Se não há caminhão grande ou está cheio:
     - Enfileirar caminhão (filaCaminhoes.enqueue).
     - Se não agendado, criar EventoGerarCaminhaoGrande para tempoAtual + 100.
  2. Senão:
     - Cancelar evento agendado, se existir.
     - Descarregar carga (caminhaoGrandeAtual.receberCarga, caminhao.descarregar).
     - Calcular tempoDescarga (carga × TEMPO$_{D}$*ESCARGA*$_{P}$*OR*$_{T}$*ONELADA*).*Registrar*$v$ – Se há viagens, agendar EventoColeta.
     - Se caminhão grande está cheio, descarregar no aterro e processar fila (descarregarFilaEspera).
- Complexidade: $O(f)$ para processar a fila, onde $f$ é o tamanho da fila.
- **Exemplo de Uso**: Caminhão C1 chega à estação A, descarrega 2t ou entra na fila.

# 5 Tecnologias usadas

- IntelliJ
- JAVA
- GIT
- Programação Orientada a Objetos (POO)
- Estruturas de Dados Personalizadas
- Simulação Baseada em Eventos

# 6 Conclusão

O sistema é uma simulação robusta baseada em eventos, com estruturas de dados (Lista, Fila) adequadas e algoritmos eficientes para gestão de eventos, tempos e filas. Apesar de pequenos problemas, como lixo remanescente e erros de tempo, a arquitetura modular facilita ajustes. As recomendações propostas podem eliminar os erros e melhorar a eficiência.