**Decisões de Implementação:**

Uma thread por bus

Um GUI(Funcionario) – uma thread para atualizar a GUI a cada x segundos

GUI com java.swing

Ficheiro de configuração

Uma thread para gerar anomalias

Uma thread para detetar se o programa fecha - Uma lista de passageiros no controlador

Para detetar quando o programa acaba

Location parecido com SinglyLinkedList

Verificações do ficheiro de configuração

Simulação Tempo real x 1000

Distâncias entre locations +/- reais

Passageiros a sair do autocarro são eliminados

**Problemas de Implementação:**

Problema de remoção de passageiro (da localização) – ao remover passageiros da localização para o autocarro, caso o autocarro fosse expresso alguns passageiros não poderiam entrar, pois o seu destino era diferente, isto causou problemas de passageiros presos infinitamente num autocarro que não iria para o seu destino. Resolvemos com uma simples condição,

Problema de remoção de passageiro (do autocarro) – Ao remover passageiros do autocarro, tínhamos um problema que só a metade dos passageiros cuja paragem era a atual eram removidos, este erro era causado pela remoção do passageiro que faria a lista de passageiros ser menor, que significa que as posições podem ser diferentes, exemplo lista = 0, 1, 2 lista.get(1) dará o valor 1, mas após remover o valor 1 dará o valor 2.

Problema de sincronização de threads – se vários autocarros chegarem à mesma localização ao mesmo tempo estes podem duplicar passageiros na entrada dos passageiros no autocarro. Para resolver isto, nós utilizamos sincronização para que se um autocarro acedesse à variável que contem os passageiros bloquea-se esta mesma (dar lock) para que nenhum outro autocarro aceda ao mesmo tempo.

Problema de parar uma thread a partir de outra – em várias situações, como por exemplo as anomalias, queremos pausar uma thread a partir de outra thread. Nós não conseguimos isto com o método sleep nem o método wait, por isso tivemos de recorrer a dois métodos “deprecated” que são thread.suspend() e thread.resume(), estes métodos são propícios a problemas de deadlock.

Problema de finalizar o programa – este problema é parecido ao anterior, mas surge da thread que verifica se a simulação acabou e tenta terminar todas as threads. Neste caso, utilizamos o método .stop() que é “deprecated”, pois não encontramos outra solução.

Problema com o GUI – na criação da GUI tivemos vários problemas a demonstrar texto referente ao estado da simulação e após tentar usar “JLabel”, “JTextField” e “JTextArea”, sem conseguir demonstrar texto e o elemento desaparecia do painel principal, trocamos para “JTextPane” que funcionou como nós pretendíamos.

Problema de FileNotFoundException (ao iniciar o programa no IDE InteliJ ou na CMD) – O caminho para o ficheiro de configuração dava erro ao iniciar pela CMD ou pelo InteliJ, para resolvermos isto criamos uma variável que tem o caminho absoluto para o projeto, mesmo que este seja iniciado dentro do diretório src.

GUI.java – 80%Nuno + 10%Teixeira + 10%André

Starter.java – Teixeira

Program.java – 50%Nuno + 50%Teixeira

Controller.java – 45%Nuno + 45%Teixeira + 10%André

Controller.java (verificação do ficheiro de configuração) – Teixeira

Bus.java – 75%Nuno + 25%Teixeira

Passenger.java – 50%Nuno + 50%Teixeira

Location.java – 50%Nuno + 50%Teixeira

Diagram.puml – 20%Nuno + 80%Teixeira