**ESPECIFICAÇÃO E DOCUMENTAÇÃO DO SOFTWARE PARA VISUALIZAÇÃO E RESERVA DE ASSENTOS**

1. **INTRODUÇÃO**

Um sistema de reserva de assentos consiste em um programa que mantém informações sobre a alocação de assentos em um espaço físico ou em um meio de transporte, permitindo que usuários remotos possam ver o estado das alocações, solicitar nova alocação ou cancelar uma alocação realizada previamente pelo próprio usuário. Normalmente esse tipo de sistema é executado em um ambiente distribuído com duas aplicações distintas: uma que executa o código que gerencia as estruturas de dados usadas para manter o estado de alocação dos assentos e outra que é executada em máquinas remotas pelos usuários em essência, trata-se de uma aplicação concorrente com vários fluxos de execução independentes.

1. **ESPECIFICAÇÕES**

O processo de desenvolvimento adotado foi a programação paralela, não foi utilizado nenhum tipo de metodologia ágil para o mesmo. O processo durou em torno de 30 dias e as tarefas foram divididas entre os membros de acordo com suas especialidades dividido entre os membros.

1. **TECNOLOGIA**

Foram utilizadas algumas tecnologias para o auxílio do desenvolvimento:

* Java JDK 7
* Eclipse
* GitHub

Quanto ao armazenamento, foi utilizado, como pedido, um arquivo de texto, onde é colocado o LOG do programa.

1. **ESTRATÉGIA CONCORRENTE E IDEIA DO PROJETO**

Foi utilizado HashMap para a implementação

1. **VISÃO E ESCOPO DO PROJETO**

ConcurrentHashMap é um HashMap com acessos concorrentes gerenciados pela própria estrutura. Usando o método replace, atualizamos um valor no hashmap caso o valor corrente seja o valor especificado como parâmetro. Como a estrutura já tem concorrência nativamente, funciona corretamente mesmo que duas threads tentem trocar o valor para uma dada chave do hashmap ao mesmo tempo.

O método replace recebe como parâmetros a chave (key, ou index) a ser alterada, o valor condicional para ser substituído, e o novo valor. A substituição do valor no mapa só ocorrerá se for igual ao valor passado no parâmetro.  
**Exemplo:** mConcurrentMap.replace(1, 0, 9)  
Altera a posição 1 do hashmap para 9 somente se o valornaquela posição for 0. Retorna true se conseguir, retorna false se não conseguir alterar.

**Thread1**

Gerencia 3 assentos. Repete 3 vezes os passos:

Gera um primeiro assento aleatório.

Aloca três assentos consecutivos a partir do primeiro gerado a partir de um primeiro assento aleatório.

Desaloca os três assentos alocados no passo anterior.

Pausa por 1 segundo e em seguida desaloca os 3 assentos.

**Thread2**

Thread que aloca um assento aleatório livre, e em seguida desaloca o mesmo.

**Thread3**

Thread 3 aloca dois assentos: um livre e um fixo, que é passado como parâmetro no construtor.

**T\_Log**

Cria o Log do Sistema

**VisualizaAssentos** ()

Visualiza os assentos ocupados e os assentos não ocupados, listando-os com seus respectivos status.

**alocaAssentosDado** (**int** assento, **int** id)

Passados dois parâmetros, alocam-se o número dado pelo thread id

**int** assento é o número do assento a ser alocado

**int** id é o id do thread que está tentando alocar o assento.

**alocaAssentoLivre** (**int** id)

**int** id é o id do thread que está tentando alocar o assento.

O método aloca um assento aleatório disponível seguindo os seguintes passos:

1 - Faz um loop por no máximo 5 vezes onde ele gera um número aleatório e tenta alocar esse lugar.

2 - Caso não consiga, faz um loop por todos os assentos, do primeiro até o último, buscando um assento vazio.

3 - Caso encontre um assento vazio, o aloca. Se não, retorna.

@param id ID da Thread que chamou a função.

@return 0, se não conseguir reservar um assento, ou o número do assento alocado caso consiga.

**liberaAssento** (**int** assento, **int** id)

Libera um assento dado se o id for igual ao id do thread que o alocou.

@param assento Número do assento a ser desalocado

@param ID do thread que alocou o assento

@return 0 - se o assento não for desalocado. return 1 - se o assento for desalocado.

**fazString** ()

Transforma o mapa de assentos em uma String para ser usado no log.

**inicializaBuffer** ()

abre o arquivo de texto para escrita

**buffer** (**int** codigo, **int** id\_thread, **int** assento)

Escreve no Log.

**Exemplo:** (1, id\_thread, t\_Assentos)

**finalizaBuffer** ()

fecha o arquivo no final da execução, ou seja, salva o log