Faculdade de Engenharia da Universidade do Porto

Gestão de Bilhetes no Metro
AEDA – 2018/2019

Muriel Pinho - up201700132

Gustavo Speranzini Tosi Tavares - up201700129

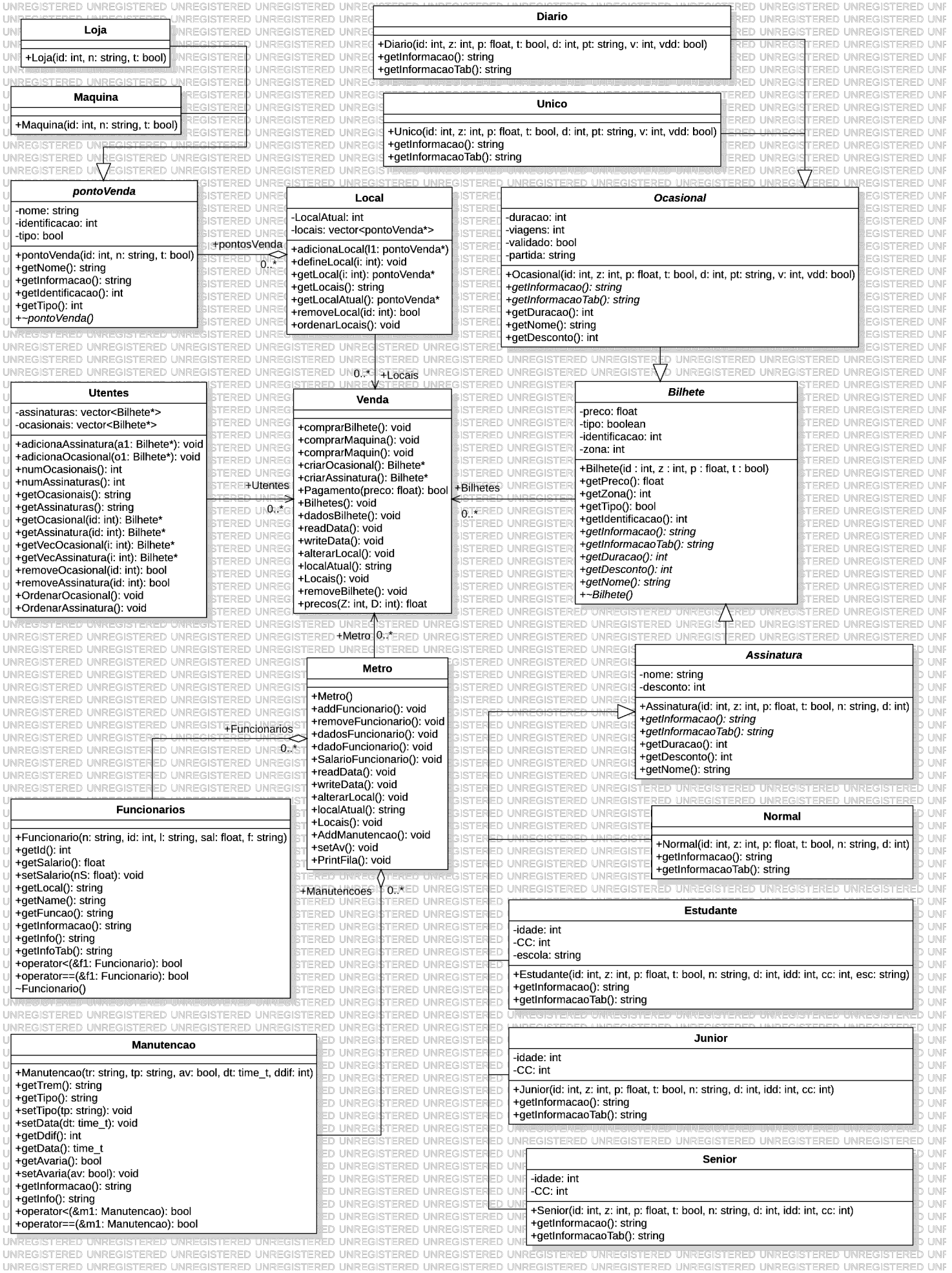
Porto 2019

Indice

Tema

O projeto implementa um sistema de gestão de informação relativo aos bilhetes e outras partes do metro na cidade do porto. Os bilhetes podem ser do tipo ocasional ou assinatura. Um bilhete ocasional pode ser único (duração de 2h) ou diário (duração de 24h). Um bilhete de tipo assinatura pode ser normal, estudante, júnior ou sénior. Uma assinatura está associada a um utente, que deve ser identificado pelo seu nome (normal), nome, idade e CC (júnior e sénior), nome, idade, CC e escola (estudante). As assinaturas do tipo estudante, sénior e júnior possuem um desconto de 25%.
É possível acessar diferentes funções relacionadas com a gestão dos bilhetes, funcionários e manutenções, essas estão indicadas na seção Casos de Utilização.

Diagrama UML



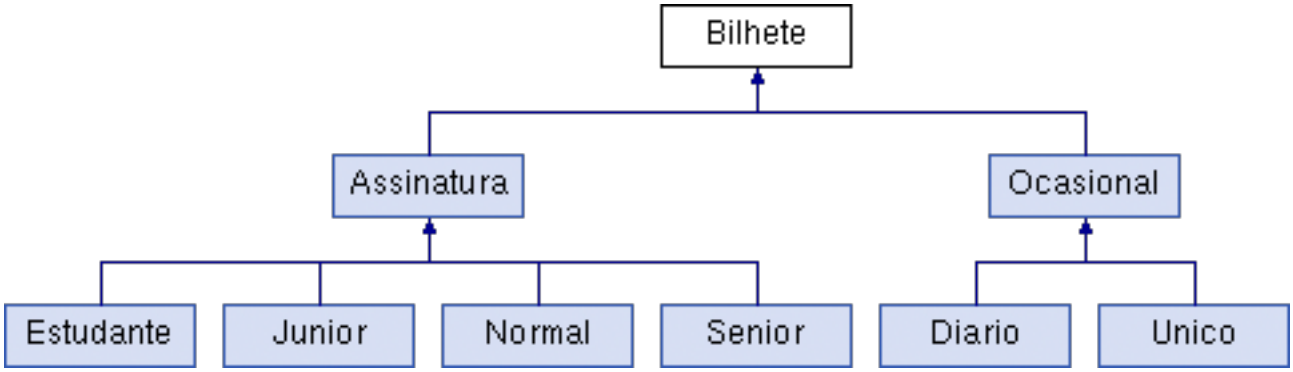
Solução Implementada

Utilizamos uma estrutura de dados com uma classe controladora (Venda) e três classes de dados (Bilhete, Local, Metro e Utentes) as quais contem subclasses que ou são usadas como tipos de dado que compõem as classes de dados, como exemplo pontoVenda é usado em Local, ou como diferentes estruturas do mesmo tipo de dado, no caso de Estudante ser um tipo de Assinatura que por sua vez é um tipo de Bilhete. Toda a estrutura de subclasses está detalhada no capitulo seguinte do relatório, assim como a utilização detalhada de todas as funções do projeto estão contidas na documentação do Doxygen mas de maneira geral a classe Venda utiliza as outras classes para gerenciar a venda de bilhetes e a gestão do metro.
Utilizamos os conceitos de herança e polimorfismo nas classes como forma de alterar a função que era invocada em tempo real, tendo assim no caso da função “getInformacao()” da classe Bilhete um retorno diferente caso essa seja chamada para o tipo Estudante ou para o tipo Diário, dessa maneira aumentando a eficiência do programa.
O programa oferece 2 modos de uso para o usuário, gerenciamento dos bilhetes ou gerenciamento do metro, cada um conta com tipos de utilização diferentes para o usuário os quais estão detalhados no capitulo de casos de utilização, mas resumidamente no modo de gerenciamento de bilhetes todos os objetos relevantes criados são guardados de maneira ordenada em seus respectivos vetores, já no modo de gerenciamento do metro, foram utilizados diferentes tipos de estruturas de dados, nomeadamente Arvore binaria de pesquisa, tabela de dispersão e fila de prioridade, e quando a execução do programa é encerrada ambos são armazenados em um ficheiro de texto, para assim serem utilizados ao realizar execuções posteriores. O programa possui ainda mecanismos de verificação dos dados inseridos, os quais rejeitam os dados caso esses não se adequem à estrutura idealizada.
A junção de todas essas implementações e conceitos utilizados no projeto tornam essa uma solução eficiente e muito util para gerir os bilhetes e outras partes de um sistema de Metro.

Hierarquia das Subclasses

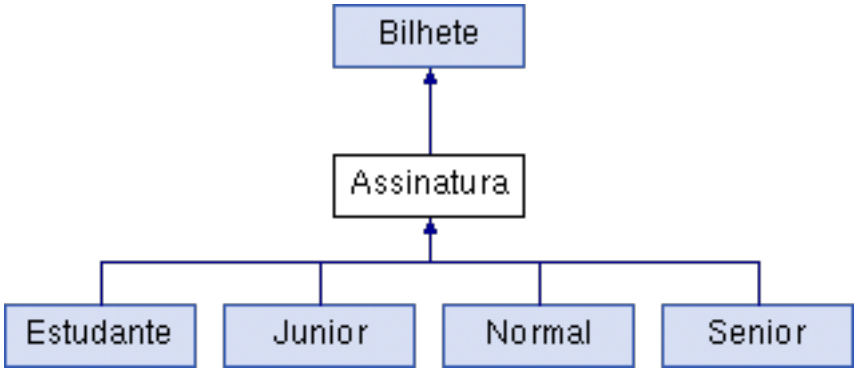
Classe Bilhete

Classe origem relacionada aos bilhetes, gera qualquer tipo de Bilhete e como elementos, apresenta número de identificação, preço, tipo e zona. Contém funções de retornar os dados (identificação, preço, tipo e zona) e funções virtuais para retornar os dados (desconto, duração, nome, informação de todos os dados).



Classe Assinatura

Classe responsável pelos Bilhetes de tipo Assinatura, nos quais podem ser dos tipos Estudantes, Junior, Normal ou Senior. Apresenta o diferencial de desconto e nome em relação a Classe Bilhete.



Classe Estudante

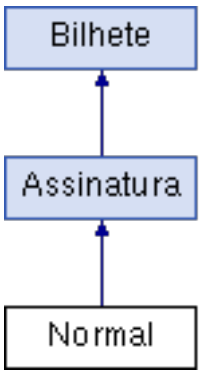
Classe responsável pelos Bilhetes de Assinatura do tipo Estudante. Apresenta o diferencial de idade, CC e escola em relação a Classe Assinatura.

Classe Junior

Classe responsável pelos Bilhetes de Assinatura do tipo Junior. Apresenta o diferencial de idade e CC em relação a Classe Assinatura.

Classe Normal

Classe responsável pelos Bilhetes de Assinatura do tipo Normal. Apresenta os mesmos dados da Classe Assinatura.

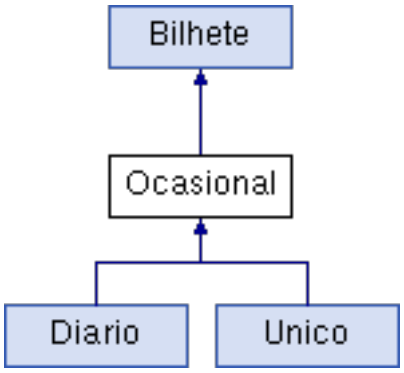


Classe Senior

Classe responsável pelos Bilhetes de Assinatura do tipo Senior. Apresenta o diferencial de idade e CC em relação a Classe Assinatura.

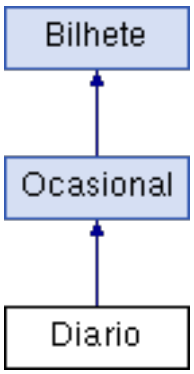
Ocasional

Classe responsável pelos Bilhetes de tipo Ocasional, nos quais podem dos tipos Diário ou Único. Apresenta o diferencial de
duração, local de partida, número de viagens e valor booleano de validado em relação a Classe Bilhete.



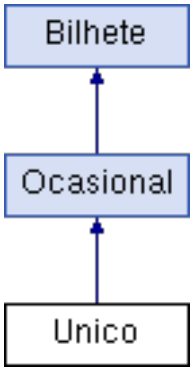
Classe Diario

Classe responsável pelos Bilhetes Ocasionais do tipo Diário. Apresenta os mesmos dados da classe Ocasional, mas coloca a duração em 24 horas.



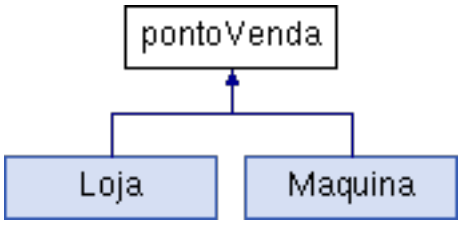
Classe Unico

Classe responsável pelos Bilhetes Ocasionais do tipo Único. Apresenta os mesmos dados da classe Ocasional, mas coloca a duração em 2 horas.



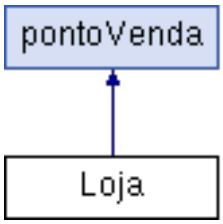
Classe PontoVenda

Classe origem relacionada aos pontos de venda, gera qualquer tipo de ponto de Venda e como elementos, apresenta número de identificação, nome e tipo.



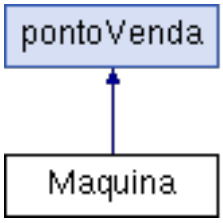
Classe Loja

Classe responsável pelos Pontos de Venda do tipo Loja, apresenta os mesmos dados da Classe PontoVenda.



Classe Maquina

Classe responsável pelos Pontos de Venda do tipo Máquina, apresenta os mesmos dados da Classe PontoVenda.

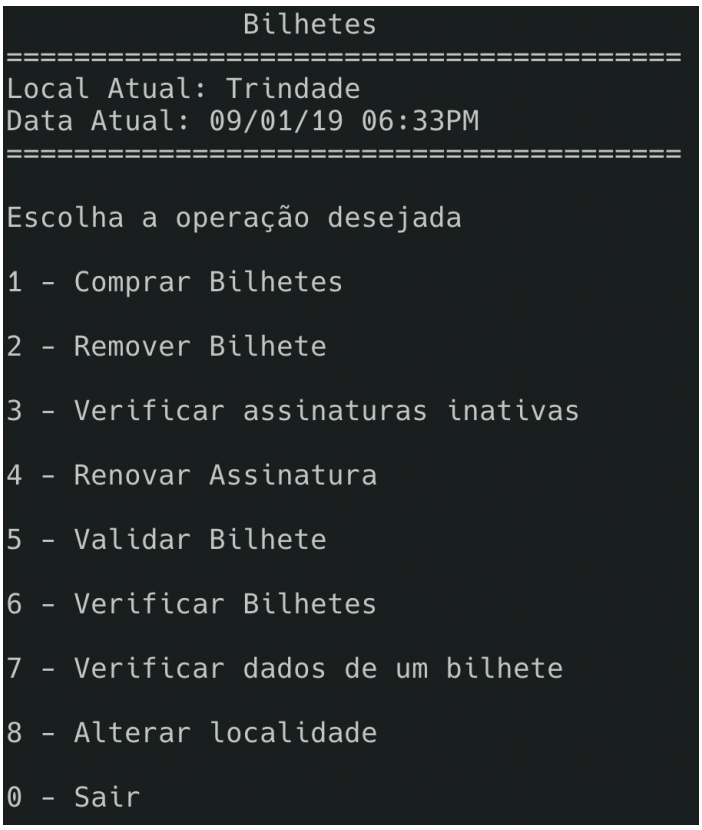


Casos de Utilização

Existem 2 modos de utilização disponíveis para o usuário:

Gerenciamento de Bilhetes

Ao escolher esse modo o usuário tem a escolha entre diferentes opções disponíveis para o usuário no terminal na interface a seguir:



- Compra de Bilhetes

Questiona ao usuário qual tipo de bilhete (assinatura ou ocasional) ele deseja comprar, depois oferece as opções pertinentes caso o bilhete escolhido seja assinatura ou ocasional. Após inserir as informações necessárias corretamente o preço do bilhete é exibido ao usuário que tem a opção de realizar o pagamento, caso o pagamento seja efetuado o bilhete é adicionado ao vetor contendo os bilhetes do seu tipo (assinatura ou ocasional) com o próximo número de identificação disponível, caso contrário o bilhete é descartado.

- Remoção de Bilhete

De acordo com as informações inseridas pelo usuário sobre o tipo de Bilhete (Assinatura ou Ocasional) e o número de identificação deste, remove o bilhete escolhido através de uma pesquisa deste número no vetor que contem os bilhetes do tipo inserido.

- Verificar Assinaturas Inativas

Imprime as informações essenciais das assinaturas que estão a mais de 2 meses sem serem renovadas.

- Renovar Assinatura

Imprime as informações essenciais de todas as assinaturas e renova a assinatura escolhida pelo usuário, estando ela vencida ou não.

- Validar Bilhete

Verifica se o bilhete tem viagens para serem usadas e caso tenha, altera o estado de validade do bilhete e retira uma viagem do bilhete.

- Verificar Bilhetes

Imprime as informações de todos os bilhetes criados. Percorrendo os vetores de Assinaturas e Ocasionais sequencialmente e escrevendo as informações de cada bilhete conforme.

- Verificar dados de um Bilhete

Imprime a informação de um único bilhete de acordo com o tipo de Bilhete (Assinatura ou Ocasional) e seu número de identificação. Pesquisando o número de identificação no vetor e escrevendo sua informação.

- Alterar Localidade

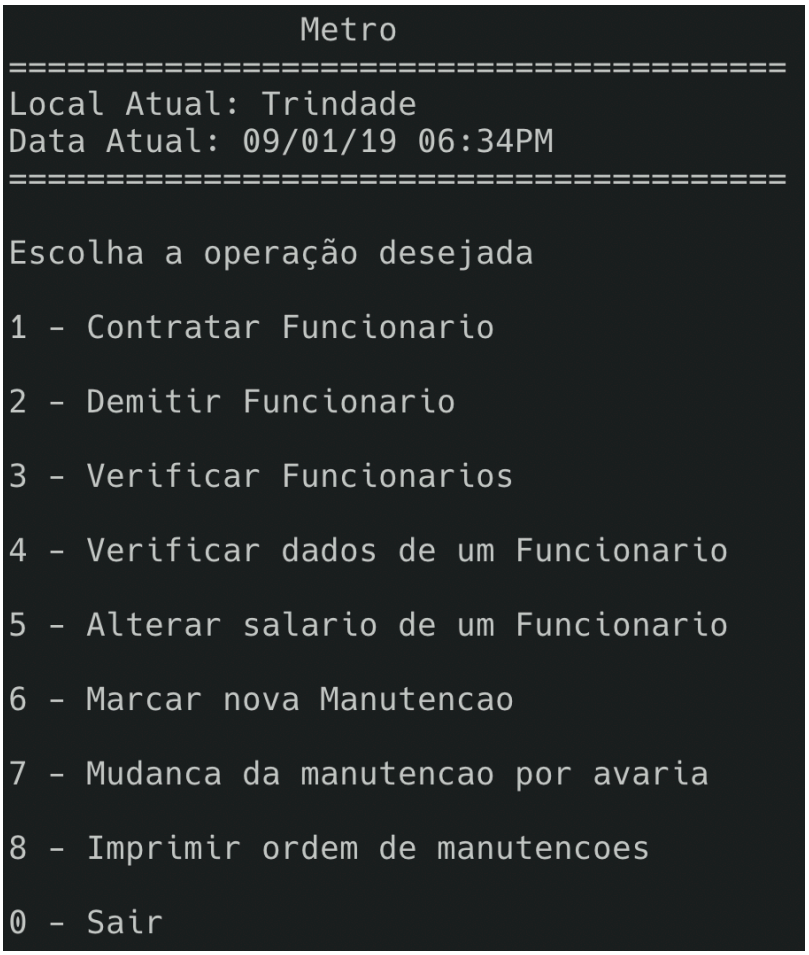
Alterar o local desejado, escolhendo um número de identificação válido para assim poder efetuar compras de bilhetes de outros locais.

- Sair

O programa salva os dados dos bilhetes existentes em um arquivo txt para uso em uma próxima execução do programa e após concluir esse processo termina o a execução do programa.

Gerenciamento do Metro

Ao escolher esse modo o usuário tem a escolha entre diferentes opções disponíveis para o usuário no terminal na interface a seguir:



- Contratar Funcionario

Questiona ao usuário o nome, salário e função do novo funcionário e o adiciona na arvore binaria de pesquisa que por sua vez ordena em ordem crescente do salário e só aceita a inserção caso não encontre nenhum funcionário igual na arvore, através do comparador que inserimos na classe.

- Demitir Funcionario

Imprime os funcionários que estão na arvore e após o usuário inserir qual será removido o programa imprime os dados do funcionário novamente e pede por um confirmação, caso seja confirmado o funcionário é deletado.

- Verificar Funcionarios

Imprime as informações essenciais de todos os funcionários que estão na árvore binaria de pesquisa.

- Verificar Funcionario

Imprime as informações essenciais de todos os funcionários e depois imprime todas as informações do funcionário escolhido pelo usuário.

- Alterar Salario

Imprime as informações essenciais de todos os funcionários e depois altera o salário do funcionário escolhido pelo usuário.

- Marcar nova Manutenção

Realiza a marcação de um manutenção com os dados fornecidos pelo usuário caso esse trem não possua uma manutenção marcada, e após ter sido marcada relembra ao usuário qual a data e horário escolhidos.

- Adiantar manutenção por avaria

Adianta a manutenção de um trem, devido a uma avaria, caso esse trem não possua uma manutenção agendada para a próxima semana.

- Imprimir ordens de manutenção

Imprime as informações essenciais de todas as ordens de manutenção, ordenadas em ordem crescente de dias restantes para a manutenção.

- Sair

O programa salva os dados dos funcionários e manutenções existentes em um arquivo txt para uso em uma próxima execução do programa e após concluir esse processo termina o a execução do programa.

Dificuldades

- Adicionar funções nas classes já estruturadas forçou mudanças nas classes que já estavam implementadas, exigindo em certos casos alteração na classe base e consequentemente alterar as funções seguintes na hierarquia.
- Por falta de tempo hábil algumas das funções que estavam planejadas não foram implementadas, deixando alguns membros-dados sem utilidade.
- Por não ter documentado o projeto enquanto implementávamos o código, ao ter que retornar a programar demorávamos para relembrar todas as funções e estruturas de classe.

Participação

- Muriel de Araújo Pinho (up20170132):

55%

Fez a maior parte das funções, estrutra e interface do programa.

- Gustavo Speranzini Tosi Tavares (up201700129):

45%

Auxiliou nas funções e estrutura de classes do programa, fez o relatório e a diagramação do programa através do Doxygen.