Estrutura de Gerenciamento de Transporte Público

Gustavo Silva Ribeiro

[1]Bacharelado de Engenharia da Computação

Universidade Estadual de Feira de Santana (UEFS)
CEP 44.036.900 – Feira de Santana– BA – Brasil

tavoribeiro007@gmail.com

Resumo. *A cidade de TechVille está buscando modernizar o transporte público. Para isso, buscou alunos da Universidade Estadual de Feira de Santana (UEFS), no intuito de buscar uma solução para o problema. A operação compõe-se no dever de desenvolver através da linguagem Python, um código piloto simples, no qual, o usuário poderá registrar recargas, dar desconto na compra da passagem por categoria e verificar se o saldo é suficiente para embarcar no ônibus.*

1. Introdução

A tecnologia tem um papel essencial na vida cotidiana, transformando a maneira como as pessoas vivem, trabalham e interagem. No contexto do transporte público de TechVille, a modernização através da tecnologia pode trazer benefícios como eficiência, acessibilidade e sustentabilidade. O transporte público em TechVille, por meio do seu projeto em Python, é um exemplo prático de como a tecnologia melhora o dia a dia, tornando os serviços mais acessíveis e eficientes para toda a população.

Antes da tecnologia, pegar um ônibus era um verdadeiro desafio. As pessoas precisavam carregar dinheiro trocado o tempo todo, já que os motoristas e cobradores nem sempre tinham troco. Isso fazia com que muita gente perdesse tempo ou até deixasse de embarcar. Além disso, os descontos para estudantes e idosos eram conferidos manualmente, o que podia gerar confusão e atrasos na hora de entrar no ônibus.

Com a chegada da tecnologia, tudo mudou. Os cartões eletrônicos facilitam a vida de todo mundo: basta aproximar do validador e pronto, sem precisar mexer em dinheiro ou se preocupar com troco. As recargas ficaram mais práticas, podendo ser feitas em totens, pontos de venda ou até pelo celular. Os descontos agora são aplicados automaticamente, garantindo mais agilidade e menos dor de cabeça.

Além de tornar tudo mais rápido e organizado, a tecnologia também trouxe mais segurança e transparência, tanto para os passageiros quanto para o sistema de transporte. Hoje, o embarque é mais simples, os dados são melhor gerenciados e as pessoas têm mais controle sobre seus gastos com transporte. No fim das contas, a modernização tornou a experiência de andar de ônibus muito mais prática e acessível.

2. Metodologia

O processo da estrutura do código foi organizado e debatido em equipe. Realizamos sessões tutoriais em que cada participante teve um papel importante, contribuindo com ideias, metas, questionamentos e soluções para o projeto. Desde o início, fomos informados sobre algumas restrições, como a proibição do uso de funções como def, try e except e interrupções de laço break, limitando-nos ao uso de estruturas de repetição, como o while. Essas limitações nos desafiaram a encontrar abordagens criativas e eficientes para a construção do sistema.

2.1. Requisitos do Programa

- Permitir a recarga de saldo para os usuários.
- Aplicar descontos na compra de passagens, de acordo com a categoria do usuário (padrão, estudante/idoso, social).
- Verificar se o saldo é suficiente para o embarque e, caso positivo, debitar o valor correspondente.
- Exibir informações sobre o saldo disponível e permitir que o usuário consulte seus gastos.
- Gerar relatórios com informações detalhadas sobre o uso do sistema.
- Utilizar apenas estruturas condicionais e laços permitidos.

2.2. Questões e Soluções

Durante o desenvolvimento, surgiram diversas questões. Algumas das principais questões e suas soluções estão descritas abaixo.

Como registrar variáveis? Desde o início, utilizamos variáveis para armazenar e manipular informações ao longo do programa. Isso foi essencial para garantir que os dados fossem mantidos e atualizados corretamente durante a execução

Como usar uma variável para finalizar while? Para finalizar o while, fiz uma variável de controle que seja modificada dentro do loop. A variável valor_passagem já está sendo usada para essa finalidade. Quando valor_passagem for maior que 0, o loop será encerrado (Imagem 1).

```
while valor_passagem <= 0:
    entrada = input('DIGITE O VALOR DA PASSAGEM EM R$: ').strip()
    if entrada.replace('.', '', 1).isdigit(): # VERIFIÇÃO DA QUANTIDADE DE DECIMAIS
        valor_passagem = float(entrada)
        if valor_passagem > 0:
            print(f'VALOR DA PASSAGEM CONFIGURADO PARA R${valor_passagem:.2f}.')
        else:
            print('ERRO! O VALOR DA PASSAGEM NÃO PODE SER NEGATIVO.')
    else:
        print('ERRO! APENAS VALORES NUMÉRICOS.')
```

Imagem 1

Como usar .isdigit() e .replace()? A string ".replace('.', '', 1) aceita apenas números com um ponto decimal. Enquanto a string "isdigit" verifica se existem apenas números entre 0 e 9 (Imagem 2).

Imagem 2

Como foi feito a disposição dos comentários no código? De início, é apresentado um comentário de não plágio, em seguida, a inicialização de variáveis e a lógica do código, além de explicar algumas verificações de entrada.

2.3. Descrição do Programa

O programa foi estruturado em diferentes etapas para organizar suas funcionalidades. Abaixo, descreverei a lógica geral do sistema.

- **Inicialização**: O programa solicita o valor da passagem;
- **Menu Opção**: O usuário pode escolher entre recarregar saldo, comprar passagem, verificar embarque, consultar saldo, gerar relatório ou sair;
- **Recarregar Saldo**: O usuário escolhe o valor da recarga e sua categoria;
- **Comprar Passagem**: O sistema solicita a escolha da categoria do usuário e aplica os descontos correspondentes antes de debitar o valor da passagem;
- **Verificar Embarque**: A passagem é conferida antes do embarque;
- **Consultar saldo**: O usuário pode consultar o saldo a qualquer momento;
- **Geração de relatórios**: O sistema apresenta informações sobre informações das categorias;
- **Sair**: O programa é finalizado.

2.4. Ordem de Codificação e Justificativa

O desenvolvimento do código foi feito em etapas, priorizando funcionalidades essenciais antes de adicionar detalhes. A ordem seguida foi:

- Definição das variáveis principais;
- Inserção do valor da passagem;
- Implementação da recarga de saldo;
- Desenvolvimento da lógica de compra de passagens;
- Implementação da verificação de saldo e exibição de mensagens;

- Aprimoramento da interface e validação de entradas;
- Geração de relatórios;
- Validação de todas as entradas do código.

2.5. Ferramentas Utilizadas

As ferramentas utilizadas para a elaboração do software foram a plataforma IDE (Integrated Development Environment ou Ambiente de Desenvolvimento Integrado) “Visual Studio Code”, o sistema operacional Windows – edição 11 –, e a versão 3.13.2 de 64 bits do Python.

3. Resultados e Discussões

3.1. Manual de Uso

O programa opera por meio de um menu interativo onde o usuário pode realizar a recarga de saldo, comprar passagens, verificar saldo, consultar possibilidade de embarque e gerar relatórios:

1. **Iniciar o programa:** O sistema solicita inicialmente a configuração do valor da passagem (Apenas valores numéricos maiores que 0).
2. **Menu principal:** O usuário escolhe entre as seguintes opções:
    - Recarregar saldo;
    - Comprar passagem;
    - Embarcar;
    - Consultar saldo;
    - Gerar relatório;
    - Sair do programa.
3. **Interação:** O sistema solicita entradas do usuário conforme a opção escolhida.

3.2. Entrada de Dados

Os dados de entrada do programa:

- Valor inicial da passagem;
- Valor da recarga desejada;
- Categoria do usuário (padrão, estudante/idoso, social);
- Escolha das opções no menu.

3.3. Saída de Dados

As saídas do programa incluem:

- Confirmação de recarga e saldo atualizado;
- Aplicação de descontos automática na compra de passagens;
- Mensagens indicando saldo insuficiente para embarque;
- Mensagens de erro caso a opção escolhida for incorreta;
- Relatórios com dados detalhados sobre informações de cada categoria.

3.4. Testes e Resultados Obtidos

Durante o desenvolvimento, foram realizados diversos testes para validar a funcionalidade do sistema. Alguns dos principais testes incluem:

- **Teste de recarga**: Verifica se a recarga de saldo é registrada corretamente;
- **Teste de compra de passagem**: Avalia se o desconto por categoria está sendo aplicado corretamente;
- **Teste de verificação de embarque**: Garante que apenas usuários com passagem suficiente consigam embarcar;
- **Teste de relatório**: Confirma a precisão das informações geradas.

Os testes mostraram que o programa opera conforme esperado dentro das condições previstas.

3.5. Erros Encontrados

Durante os testes, alguns erros foram identificados e corrigidos:

- **Entrada de valores inválidos**: O programa foi ajustado para validar as entradas do usuário e evitar valores negativo;
- **Erro na aplicação de descontos**: Inicialmente, os descontos não estavam sendo aplicados corretamente, sendo ajustados para considerar a categoria escolhida no momento da compra;
- **Saldos incorretos**: Ajustes na lógica de cálculo evitaram casos em que o saldo não era atualizado corretamente após a compra da passagem.

4. Conclusão

O desenvolvimento do sistema de gerenciamento do transporte público de TechVille atingiu seus principais objetivos, além de garantir que o usuário não quebre o código e que seja de fácil manutenção. Outros pontos a quais foram conquistados são os usuários realizassem recargas, comprem passagens com desconto por categoria, consultar o saldo e que o administrador possa gerar relatório.

4.1. Melhorias

Apesar de conseguir concluir os objetivos, há algumas melhorias a serem feitas, como a validação de entradas. O método ".replace('.', '', 1)" só aceita números com apenas um ponto decimal. Assim, caso o usuário digite '1.000.50', o código lê como uma entrada inválida.

5. Bibliografia Consultada

1. *PYTHON SOFTWARE FOUNDATION.* **Python Documentation.** Disponível em: https://docs.python.org/pt-br/3.13/. Acesso em: 28 mar. 2025.