PUC Minas

Trabalho Interdisciplinar

Fundamentos de Engenharia de Software

Algoritmos e Dados I

Documentação e teste de aplicação de console em C que simula uma loja revendedora de veículos fictícia

Matheus Caetano Rocha

Pedro Henrique Caetano Soares

Rafael Caetano da Silva

Belo Horizonte, 2023

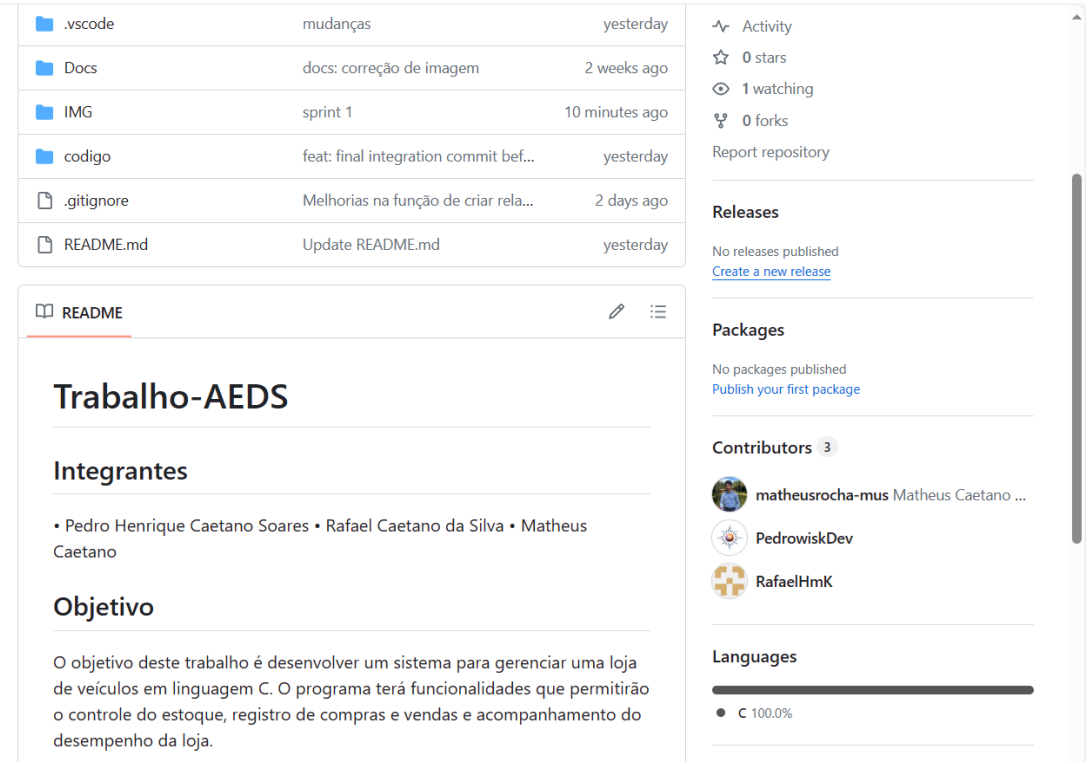
Sumário

Apresentação:

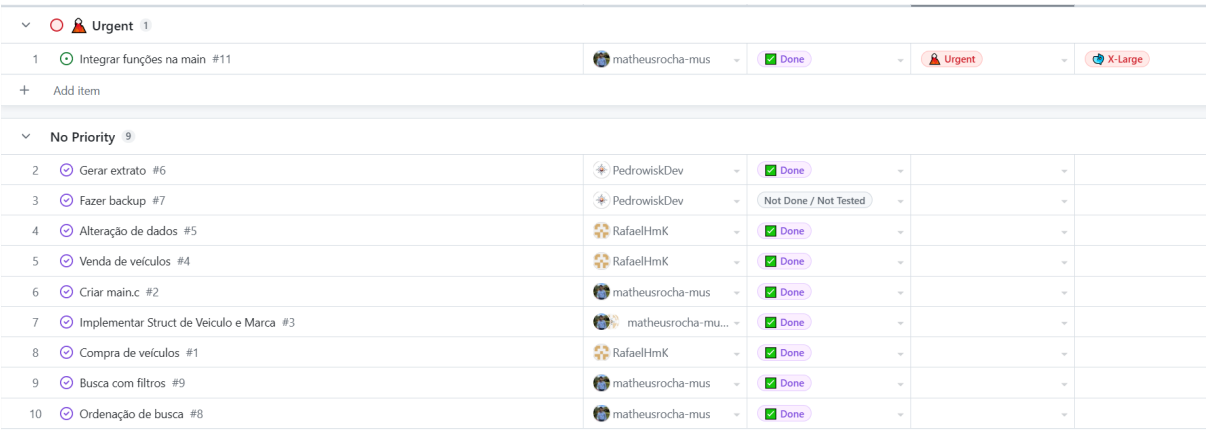
O objetivo deste trabalho é desenvolver um sistema para gerenciar uma loja de veículos em linguagem C. O programa terá funcionalidades que permitirão o controle do estoque, registro de compras e vendas e acompanhamento do desempenho da loja.

Backlog do produto:

Primeiro print apresenta o projeto criado no github, mostrando o nome de cada integrante, seus arquivos e objetivo.

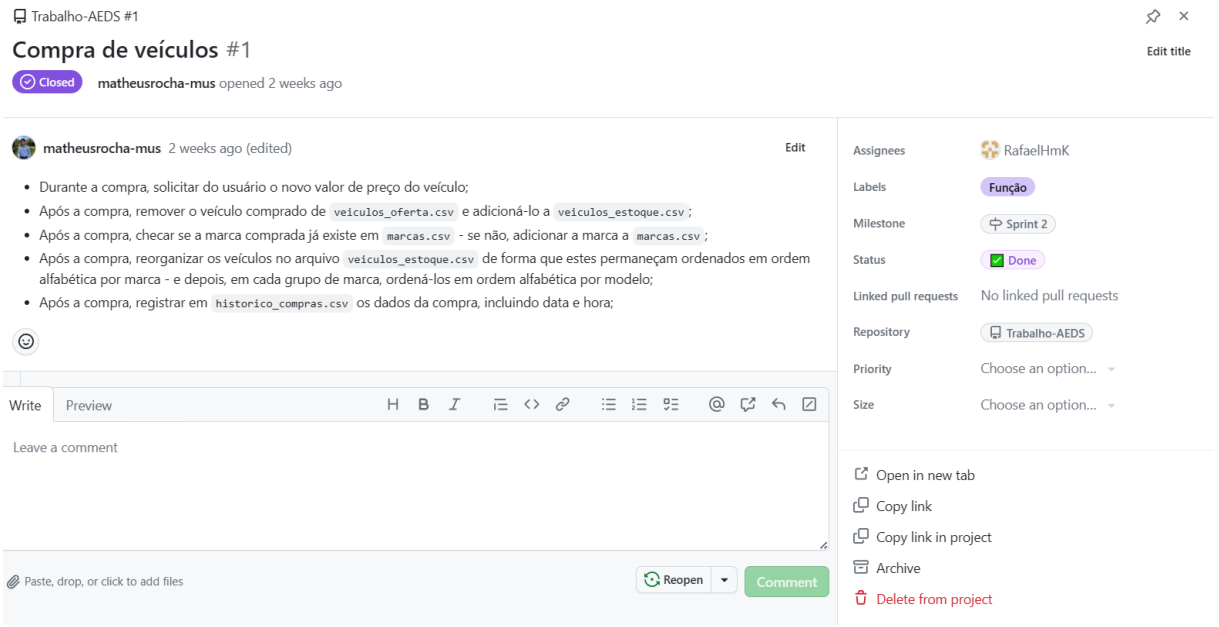


Segundo print apresenta o resultado final do backlog mostrando a divisão de tarefas.

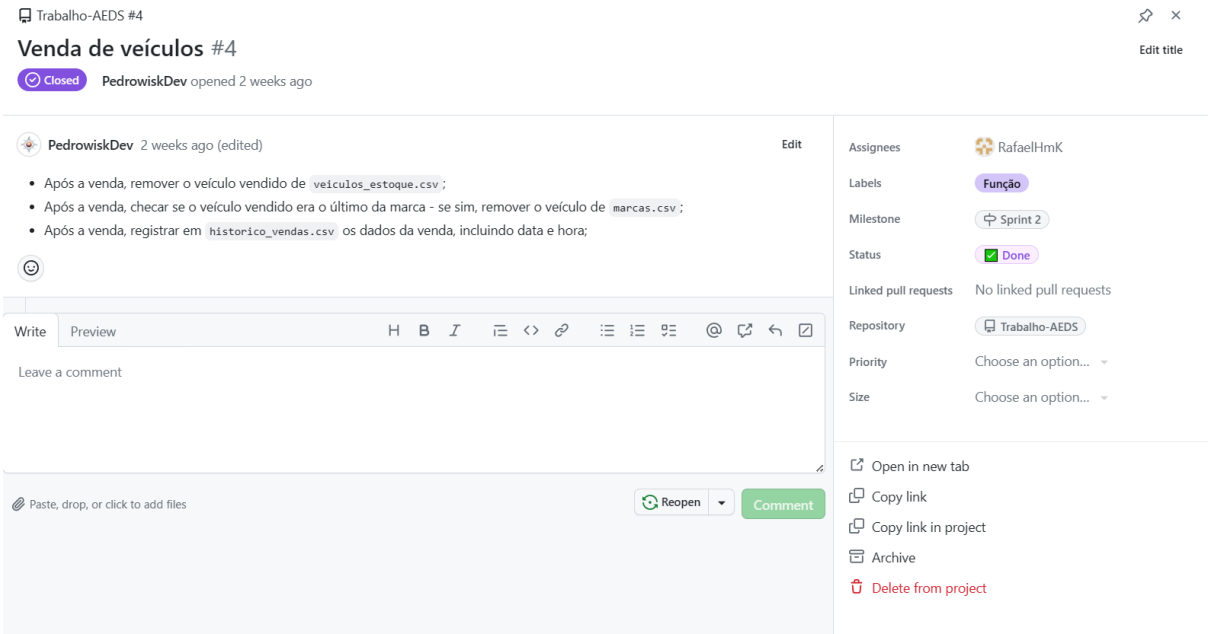


Prints das tarefas e suas descrições

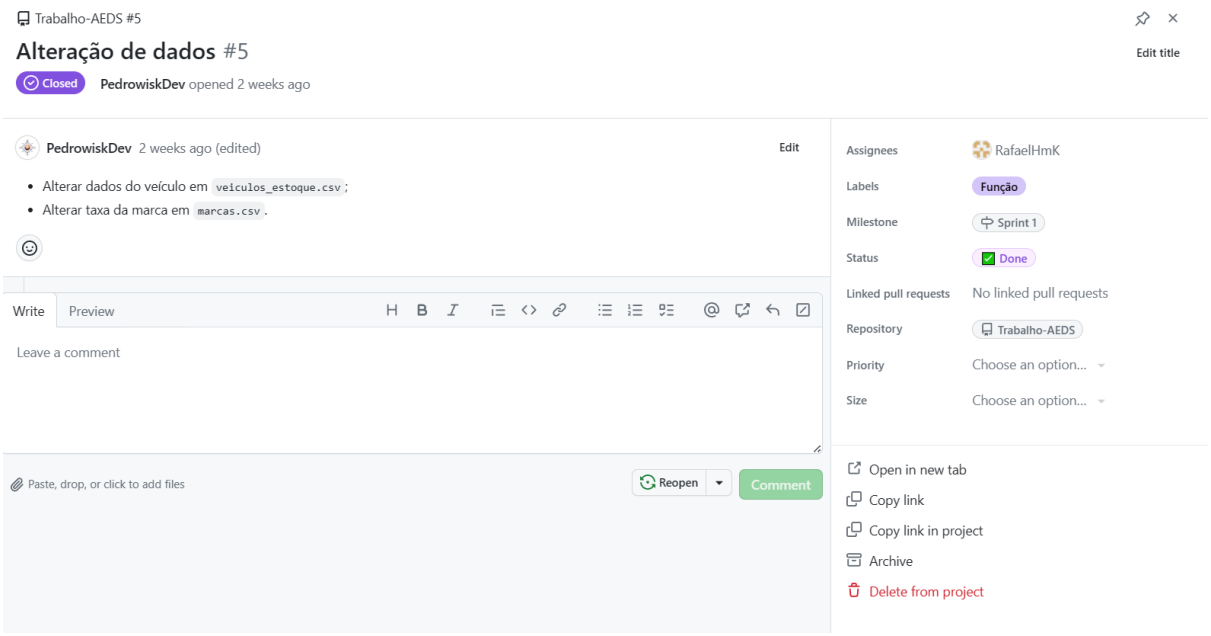
Compra de Veículos



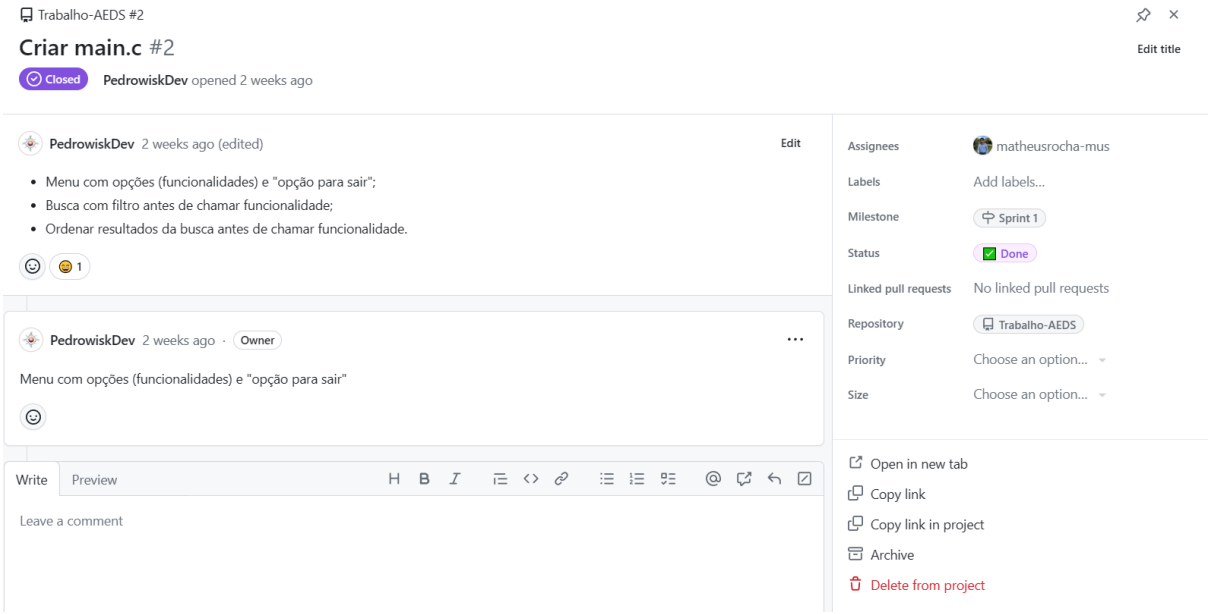
Venda de veículos



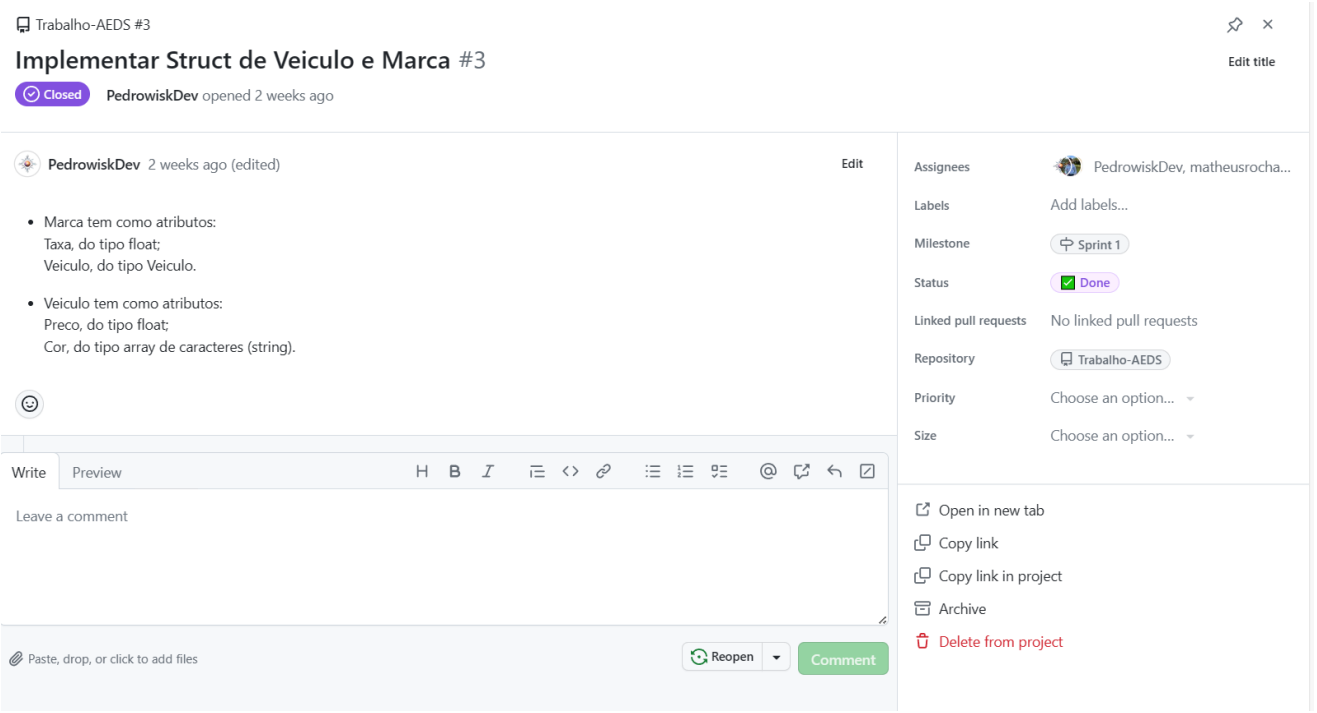
Alteração de Dados



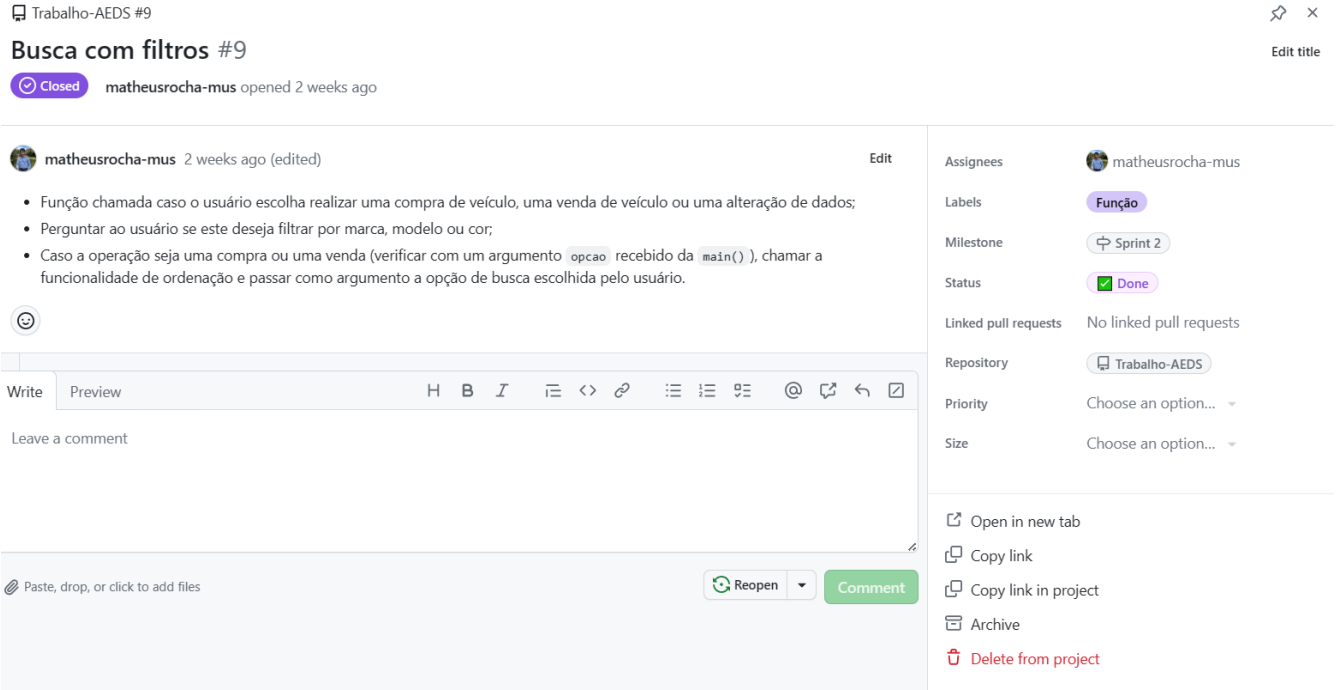
Criar main.c



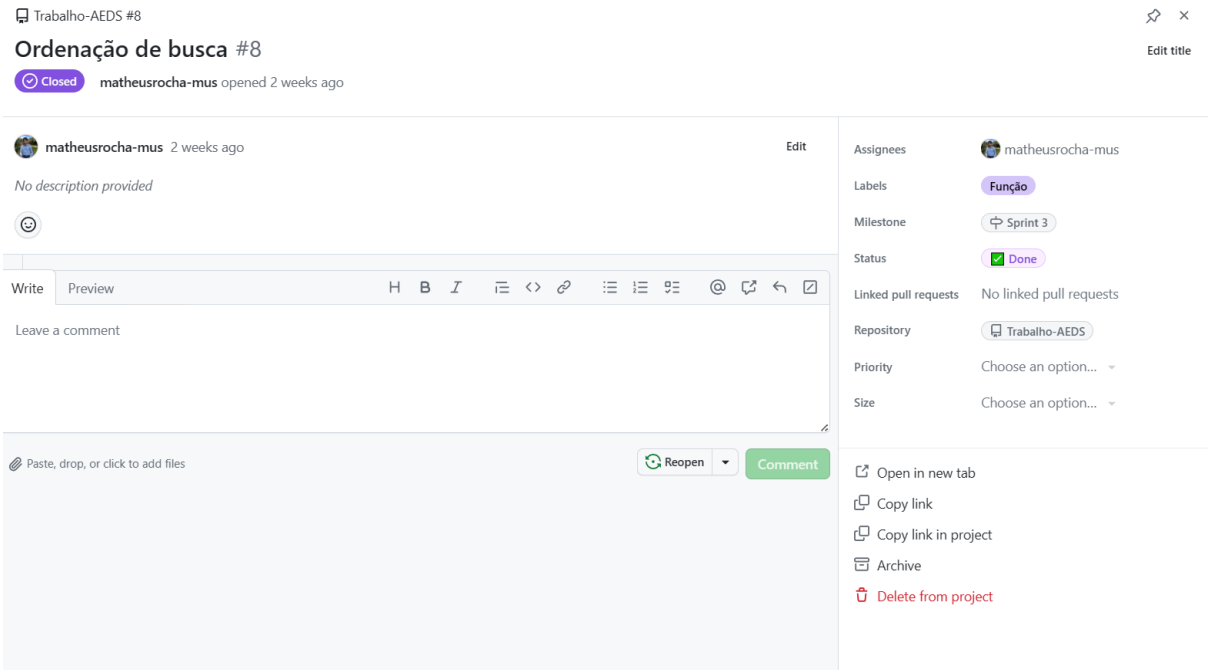
Implementar struct de veículos e marca



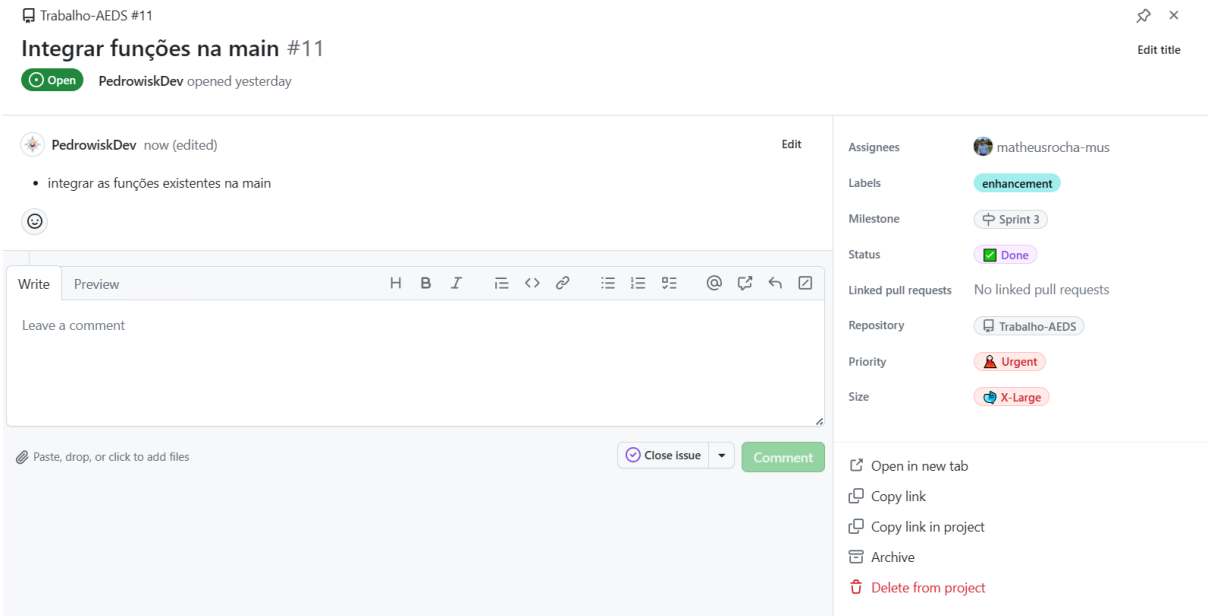
Busca com filtros



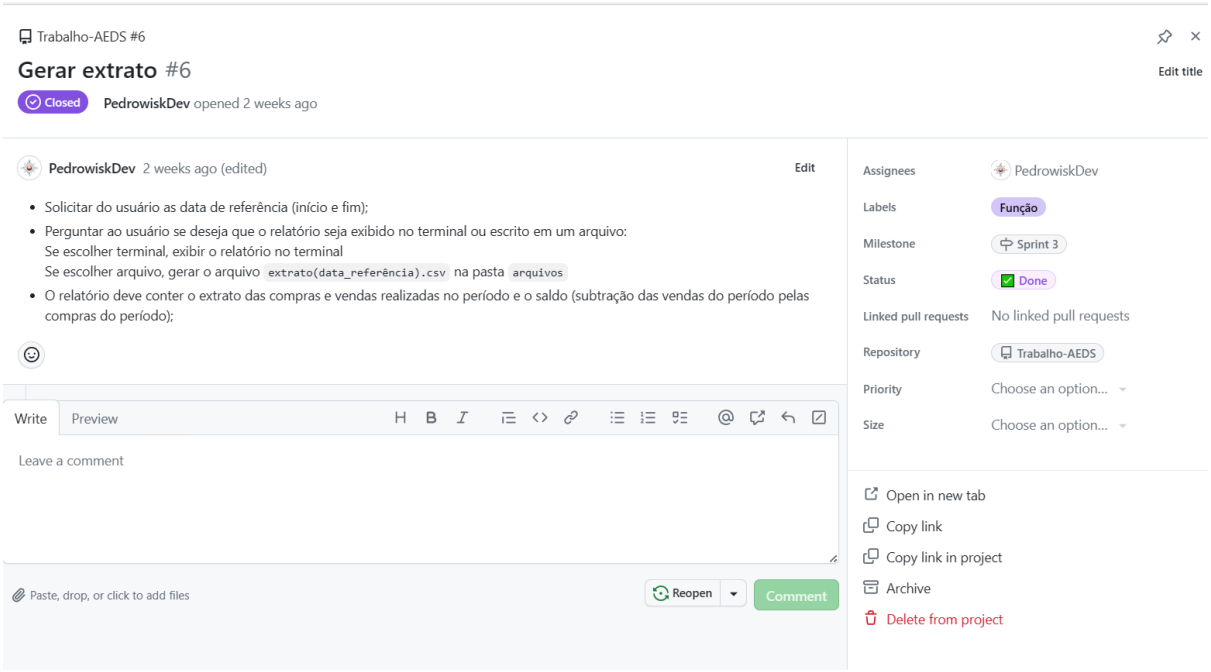
Ordenação por busca



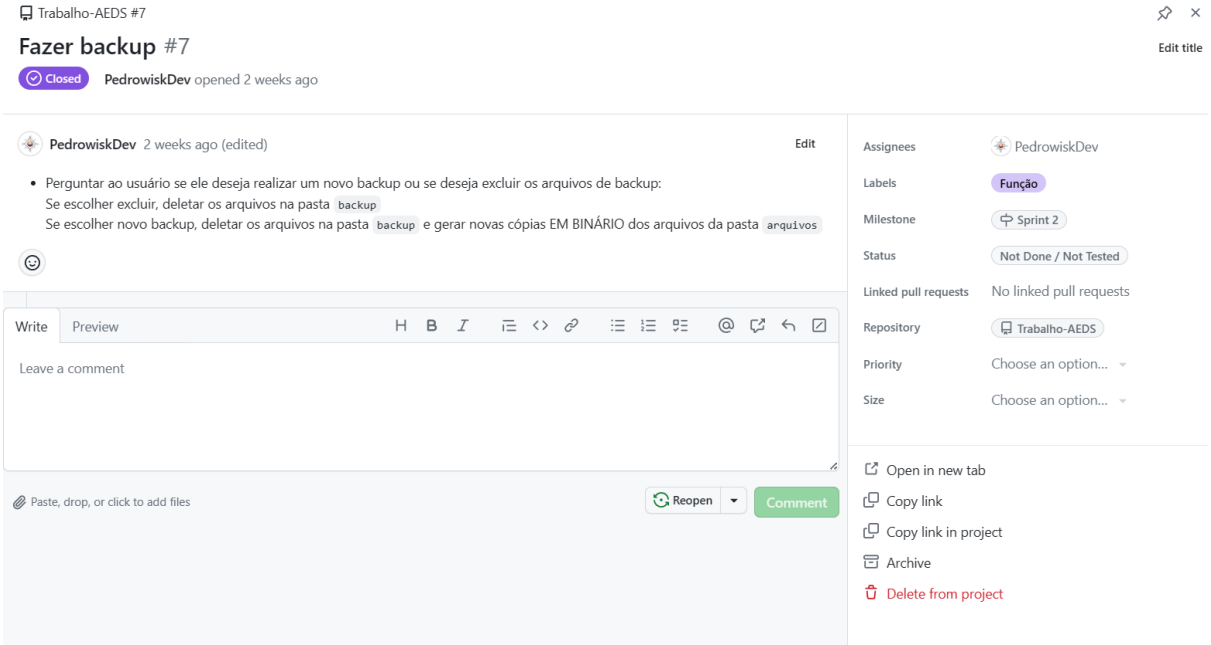
Integrar funções na main



Relatório



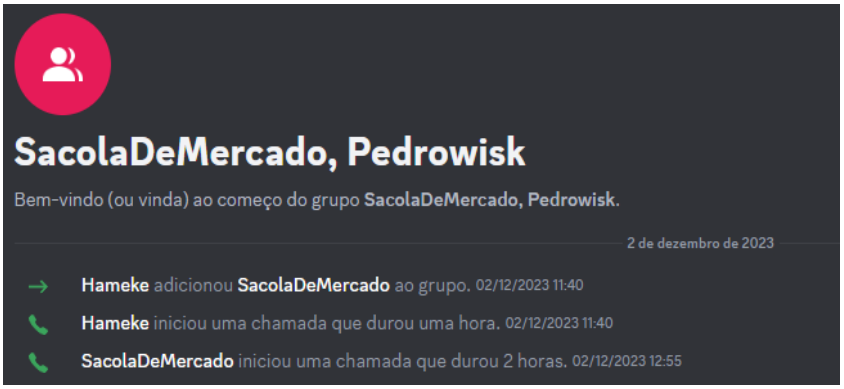
Fazer Backup(não foi realizada a função backup)



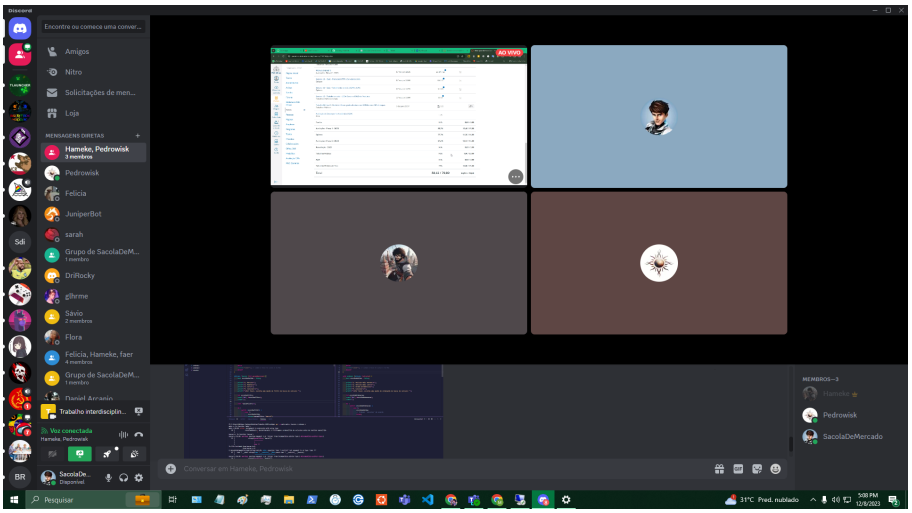
Agora segue prints de reuniões feitas pelos integrantes.

Hameke: Rafael Caetano da Silva
SacolaDeMercado: Matheus Caetano Rocha
Pedrowisk: Pedro Henrique Caetano Soares
Os próximos três prints mostram nossas reuniões em sequência de data

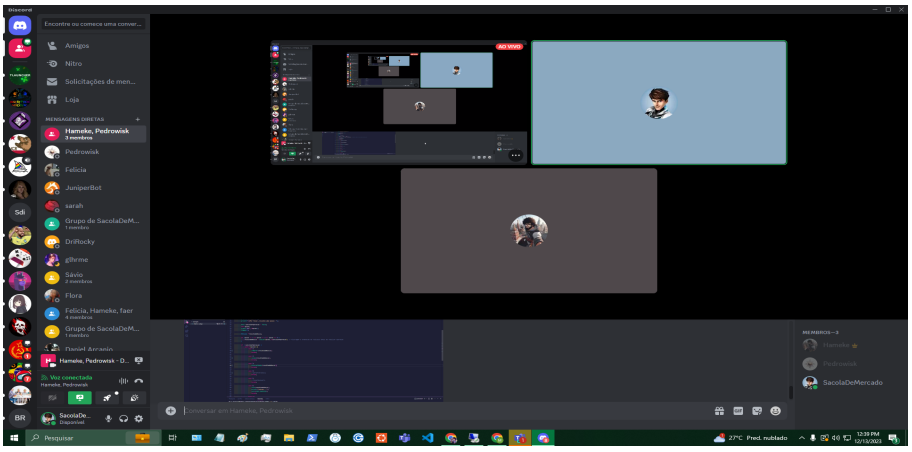
02/12/2023



08/12/2023



13/12/2023



Assinatura das Funções e Parâmetros:

void alterarDados(Veiculo *veiculos, int posicoesPreenchidasVeiculos)

Ela recebe como parâmetros um ponteiro para um array de objetos do tipo **Veiculo** e um inteiro que representa o número de posições preenchidas nesse array. A função é projetada para realizar alterações nos dados dos veículos, utilizando o ponteiro para acessar e modificar as informações pertinentes aos veículos dentro do array, com base no número de posições preenchidas. O corpo da função conteria as instruções específicas para a manipulação dos dados dos veículos.

Veiculo *buscar (int opcaoOperacao, bool *cancelarOperacao, int *posicoesPreenchidasVeiculos)

Ela retorna um ponteiro para um objeto do tipo **Veiculo** e recebe como parâmetros um inteiro **opcaoOperacao**, um ponteiro booleano **cancelarOperacao**, e um ponteiro para inteiro **posicoesPreenchidasVeiculos**. A variável **opcaoOperacao** sugere que a função pode aceitar uma opção como entrada para determinar o tipo de busca a ser realizada. O ponteiro **cancelarOperacao** pode ser utilizado para indicar se a operação deve ser cancelada, enquanto o ponteiro **posicoesPreenchidasVeiculos** pode ser usado para atualizar o número de posições preenchidas no array de veículos após a busca. O corpo da função conteria as instruções específicas para realizar a busca com base na opção fornecida.

void adicionarMarca(const char* marca)

Recebe como parâmetro um ponteiro para uma string constante (**const char* marca**). Essa função tem como objetivo adicionar uma nova marca ao arquivo "marcas.csv". O código primeiro tenta abrir o arquivo para leitura e, em caso de falha, exibe uma mensagem de erro e encerra o programa. Em seguida, verifica se a marca já existe no arquivo. Se a marca já estiver presente, a função encerra sem realizar alterações. Caso contrário, ela fecha o arquivo e o abre novamente em modo de escrita (append), adicionando a nova marca ao final do arquivo. Finalmente, o arquivo é fechado.

Em resumo, a função **adicionarMarca** verifica se a marca já existe no arquivo e, se não existir, a adiciona ao final do arquivo "marcas.csv".

void removerMarca(const char* marca)

Ela recebe um parâmetro, um ponteiro para uma string constante (**const char* marca**), que representa a marca a ser removida do arquivo "marcas.csv". A função realiza as seguintes operações:
Tenta abrir o arquivo "marcas.csv" para leitura e, em caso de falha, exibe uma mensagem de erro e encerra o programa.

Abre um novo arquivo temporário chamado "temp_marcas.csv" para escrita. Em caso de falha, fecha o arquivo original e exibe uma mensagem de erro antes de encerrar o programa.

Percorre o arquivo original linha por linha, ignorando as linhas que contêm a marca a ser removida. As linhas restantes são copiadas para o arquivo temporário.

Fecha os arquivos original e temporário.

Remove o arquivo original "marcas.csv".

Renomeia o arquivo temporário "temp_marcas.csv" para o nome original "marcas.csv".

void adicionarVeiculoEstoque(const Veiculo* veiculo)

Ela recebe um parâmetro, um ponteiro para um objeto do tipo **Veiculo** (**const Veiculo* veiculo**), que representa as informações do veículo a ser adicionado ao arquivo "veiculos_estoque.csv". O código realiza as seguintes operações:
Tenta abrir o arquivo "veiculos_estoque.csv" para escrita em modo append (**"a"**). Em caso de falha, exibe uma mensagem de erro e encerra o programa.

Escreve as informações do veículo no formato CSV (Comma-Separated Values) no arquivo, incluindo a marca, modelo, cor e preço.

Fecha o arquivo.

void removerVeiculoOferta(const char* marca, const char* modelo)

Ela recebe dois parâmetros do tipo ponteiro para constante string (**const char* marca, const char* modelo**), representando a marca e o modelo do veículo a ser removido do arquivo "veiculos_oferta.csv". O código realiza as seguintes operações:
Tenta abrir o arquivo "veiculos_oferta.csv" para leitura (**"r"**). Em caso de falha, exibe uma mensagem de erro e encerra o programa.

Abre um novo arquivo temporário chamado "temp_oferta.csv" para escrita (**"w"**). Em caso de falha, fecha o arquivo original e exibe uma mensagem de erro antes de encerrar o programa.

Percorre o arquivo original linha por linha, ignorando as linhas que contêm o veículo a ser removido, com base na marca e no modelo.

Fecha os arquivos original e temporário.

Remove o arquivo original "veiculos_oferta.csv".

Renomeia o arquivo temporário "temp_oferta.csv" para o nome original "veiculos_oferta.csv".

void registrarCompra(const Veiculo* veiculo)

Ela recebe um parâmetro, um ponteiro para um objeto do tipo **Veiculo** (**const Veiculo* veiculo**), representando as informações do veículo a ser registrado no histórico de compras no arquivo "historico_compras.csv". O código realiza as seguintes operações:
Tenta abrir o arquivo "historico_compras.csv" para escrita em modo append (**"a"**). Em caso de falha, exibe uma mensagem de erro e encerra o programa.

Obtém a data e hora local do sistema usando **time** e **localtime**.

Formata a data e hora no formato desejado ("dd/mm/yyyy hh:mm:ss") e escreve no arquivo junto com as informações do veículo.

Fecha o arquivo.

void comprar(Veiculo *veiculosRetornados, int posicoesPreenchidasVeiculos)

Ela recebe dois parâmetros: um ponteiro para um array de objetos do tipo **Veiculo** (**Veiculo *veiculosRetornados**) e um inteiro (**int posicoesPreenchidasVeiculos**), que representa o número de posições preenchidas no array. O código realiza as seguintes operações:
Verifica se há veículos disponíveis para compra. Se não houver, exibe uma mensagem e encerra a função.

Exibe os veículos disponíveis para compra, mostrando suas informações, como marca, modelo, cor e preço.

Solicita ao usuário que escolha o número do veículo desejado para compra ou 0 para cancelar a operação.

Verifica a validade da escolha e executa a operação correspondente.

Se a escolha for válida, a função realiza as seguintes operações:

Adiciona a marca ao arquivo "marcas.csv" se necessário.

Adiciona o veículo ao estoque utilizando a função **adicionarVeiculoEstoque**.

Remove o veículo da oferta utilizando a função **removerVeiculoOferta**.

Registra a compra no histórico utilizando a função **registrarCompra**.

6- Exibe uma mensagem indicando que a compra foi realizada com sucesso.

void removerVeiculoEstoque(const char *marca, const char *modelo)

Ela recebe dois parâmetros do tipo ponteiro para constante string (**const char *marca, const char *modelo**), representando a marca e o modelo do veículo a ser removido do arquivo "veiculos_estoque.csv". O código realiza as seguintes operações:
Tenta abrir o arquivo "veiculos_estoque.csv" para leitura (**"r"**). Em caso de falha, exibe uma mensagem de erro e encerra o programa.

Abre um novo arquivo temporário chamado "temp.csv" para escrita (**"w"**). Em caso de falha, fecha o arquivo original e exibe uma mensagem de erro antes de encerrar o programa.

Percorre o arquivo original linha por linha, ignorando as linhas que contêm o veículo a ser removido, com base na marca e no modelo.

Fecha os arquivos original e temporário.

Remove o arquivo original "veiculos_estoque.csv".

Renomeia o arquivo temporário "temp.csv" para o nome original "veiculos_estoque.csv".

void registrarVenda(const Veiculo *veiculo, float taxa)

Ela recebe dois parâmetros: um ponteiro para um objeto do tipo **Veiculo** (**const Veiculo* veiculo**) e um valor float (**float taxa**), representando as informações do veículo vendido e a taxa aplicada sobre o preço. O código realiza as seguintes operações:
Tenta abrir o arquivo "historico_vendas.csv" para escrita em modo append (**"a"**). Em caso de falha, exibe uma mensagem de erro e encerra o programa.

Obtém a data e hora local do sistema usando **time** e **localtime**.

Calcula o preço final do veículo, aplicando a taxa ao preço original.

Formata a data e hora no formato desejado ("dd/mm/yyyy hh:mm:ss") e escreve no arquivo junto com as informações do veículo, incluindo a marca, modelo, cor e preço final.

Fecha o arquivo.

void venda(Veiculo *veiculosEncontrados, int posicoesPreenchidasVeiculos)

Ela recebe dois parâmetros: um ponteiro para um array de objetos do tipo **Veiculo** (**Veiculo *veiculosEncontrados**) e um inteiro (**int posicoesPreenchidasVeiculos**), representando os veículos disponíveis para venda e o número de posições preenchidas no array. O código realiza as seguintes operações:
Verifica se há veículos disponíveis para venda e se a operação não foi cancelada.

Solicita ao usuário que digite a taxa de venda em formato decimal.

Exibe os veículos disponíveis para venda, mostrando suas informações, como marca, modelo, cor e preço.

Solicita ao usuário que escolha o número do veículo desejado para venda ou 0 para cancelar a operação.

Verifica a validade da escolha e executa a operação correspondente.

Se a escolha for válida, a função realiza as seguintes operações:

Registra a venda utilizando a função **registrarVenda**.

Remove o veículo do estoque utilizando a função **removerVeiculoEstoque**.

Exibe uma mensagem indicando que a venda foi realizada com sucesso.

void limpar()

Tem como objetivo limpar a tela do console, tornando-a mais legível para o usuário. A implementação usa diretivas de pré-processador (**#ifdef**, **#else**, **#endif**) para realizar a limpeza de tela de acordo com o sistema operacional. Se o código estiver sendo compilado em um ambiente Windows (**_WIN32** definido), ela utiliza **system("cls")** para limpar a tela.

Caso contrário, presume-se que o código esteja sendo compilado em um ambiente Linux ou Mac, onde é utilizado **system("clear")** para realizar a mesma operação.

void ordenar (Veiculo *veiculos, bool *cancelarOperacao)

Ela recebe dois parâmetros: um ponteiro para um array de objetos do tipo Veiculo (Veiculo *veiculos) e um ponteiro booleano (bool *cancelarOperacao), representando os veículos a serem ordenados e uma sinalização para cancelar a operação. O código realiza as seguintes operações:
Exibe um menu de opções de ordenação para o usuário, solicitando a escolha de uma opção.

Lê a escolha do usuário.

Limpa a tela do console.

Utiliza um loop do-while para garantir que o usuário forneça uma escolha válida.

No switch, implementa casos para as opções de ordenação desejadas (mais barato, mais caro, ordem alfabética), realizando as operações de ordenação correspondentes no array de veículos.

Se o usuário escolher cancelar (opção 4), atualiza a variável booleana *cancelarOperacao para true e retorna da função.

Se a escolha não for válida, exibe uma mensagem indicando isso e solicita uma nova escolha.

int lerTransacoesCSV(const char *nomeArquivo, struct Transacao transacoes[])

Tem como objetivo ler transações a partir de um arquivo CSV e armazená-las em uma array de estruturas Transacao. A função retorna o número de transações lidas e preenche o array transacoes com os dados do arquivo. Aqui está uma análise da função:

Const char *nomearquivo: Representa o nome do arquivo CSV a ser lido.

Struct Transacao transacoes[]: É um array de estruturas Transacao que será preenchido com os dados lidos do arquivo CSV.

A função começa tentando abrir o arquivo especificado em modo de leitura ("r"). Se não conseguir abrir o arquivo, exibe uma mensagem de erro, imprime o nome do arquivo com erro e retorna 0, indicando que nenhuma transação foi lida.

Em seguida, a função inicia um loop while que lê cada linha do arquivo usando fgets e armazena os dados nas variáveis correspondentes da estrutura Transacao usando sscanf. Os dados são extraídos da linha do arquivo no formato CSV, onde os campos estão separados por vírgulas. A função continua lendo linhas até atingir o final do arquivo ou até o número máximo de transações (MAX_TRANSACOES).Finalmente, a função fecha o arquivo e retorna o número de transações lidas (i).

void escreverTransacoes(const char *nomeArquivo, struct Transacao transacoes[], int tamanho, const char *dataBusca, int tipotranacao)

tem como objetivo escrever transações em um arquivo, filtrando por uma data específica e tipo de transação. Aqui está uma análise da função:
const char *nomeArquivo: Representa o nome do arquivo onde as transações serão escritas.

struct Transacao transacoes[]: É um array de estruturas **Transacao** contendo as transações a serem escritas no arquivo.

int tamanho: Indica o número total de transações no array **transacoes**.

const char *dataBusca: Representa a data específica pela qual as transações devem ser filtradas.

int tipotranacao: Indica o tipo de transação ('V' para venda, 'C' para compra).

void gerarExtrato()

tem como objetivo gerar um extrato baseado em transações de compra e venda em uma determinada data. Aqui está uma análise da função:
Dois arrays de estruturas **Transacao** são declarados para armazenar transações de venda (**transacoesVenda**) e de compra (**transacoesCompra**).

Duas variáveis (**tamanhoVendas** e **tamanhoCompras**) são usadas para armazenar o número de transações lidas de cada tipo.

Uma variável de caractere (**dataBusca**) é utilizada para armazenar a data inserida pelo usuário.

Um loop **do-while** é usado para permitir que o usuário faça pesquisas repetidas até que opte por não continuar.

O usuário é solicitado a inserir uma data no formato "dd/mm/aaaa".

As transações de venda são lidas do arquivo "historico_vendas.csv" usando a função **lerTransacoesCSV** e armazenadas no array **transacoesVenda**.

As transações de compra são lidas do arquivo "historico_compras.csv" usando a função **lerTransacoesCSV** e armazenadas no array **transacoesCompra**.

As transações são escritas no arquivo "relatorio.csv" usando a função **escreverTransacoes**. O tipo de transação ('V' para venda ou 'C' para compra) é passado como parâmetro.

Um loop **for** é utilizado para imprimir as transações de venda encontradas para a data inserida pelo usuário. Se não houver transações de venda, é exibida uma mensagem de erro.

Outro loop **for** é utilizado para imprimir as transações de compra encontradas para a data inserida pelo usuário. Se não houver transações de compra, é exibida uma mensagem de erro.

Uma mensagem é exibida indicando que os dados foram salvos com sucesso no arquivo "relatorio.csv".

O usuário é perguntado se deseja fazer outra pesquisa. O loop continua se a resposta for 'S' ou 's'.

Casos de teste:

Teste 1: Menu de Operações				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**
números inteiros.	números de 1 a 6.	exibição do menu de opções de busca.	números menores que 0 e maiores que 6,e letras.	Mensagem de “opção inválida”,Erro e interrupção do programa

Relatório de Execução de Testes		
Entradas	**Resultado**	**Aprovado?**
0	Mensagem de “opção inválida”, limpa do console e re-exibição do menu de operações	Sim
1	Exibição do menu de opções de busca	Sim
2	Exibição do menu de opções de busca	Sim
3	Exibição do menu de opções de busca	Sim
4	Exibição do menu de relatório	Sim
5	Exibição do do menu de backup	Sim
6	Exibição de mensagem de saída e encerramento do programa	Sim
7	Mensagem de “opção inválida” no terminal, limpa do console e re-exibição do menu de operações	Sim
‘a’	Erro e interrupção do programa	Não

Teste 2: Menu de Compras/Vendas				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**
números inteiros	números de 1 a 4	exibição da tela da opção escolhida	números menores que 1 e maiores que 4 e letras	Mensagem de “opção inválida”,Erro e interrupção

				do programa
Relatório de Execução de Testes				
Entradas	**Resultado**			**Aprovado?**
0	Mensagem de “opção inválida”, limpa do console e re-exibição do menu de operações			Sim
1	Exibição da tela de marcas			Sim
2	Exibição da tela de modelo			Sim
3	Exibição da tela de cor			Sim
4	Exibição de mensagem de saída e encerramento do programa			Sim
5	Mensagem de “opção inválida” no terminal, limpa do console e re-exibição do menu de operações			Sim
“a”	Erro e interrupção do programa			Não

Teste 3: Menu de Marcas,Modelos,Cores				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**
String	Nome das Marcas	Exibição dos veículos que estiverem presentes em oferta	números	Erro e interrupção do programa
Relatório de Execução de Testes				
Entradas	**Resultado**			**Aprovado?**
“Chevrolet”	Exibição dos modelos da marca escolhida			Sim
“Honda”	Exibição dos modelos da marca escolhida			Sim
“Onix”	Exibição dos modelos do veículo escolhido			Sim
“Civic”	Exibição dos modelos do veículo escolhido			Sim
“Vermelho”	Exibição dos modelos com a cor escolhida			Sim
“Branco”	Exibição dos modelos com a cor escolhida			Sim
“abcd”	Mensagem de “opção inválida” no terminal, limpa do			Sim

	console e re-exibição do menu de operações	
“7”	Erro e interrupção do programa	Não

Teste 4: Alteração de Dados				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**
String,números inteiros	Números entre 0 e a quantidade de veículos encontrados,qualquer string	Exibição dos menus de alteração, e alteração de arquivos	números menores que 0 e maiores que a quantidade de veículos encontrados	Mensagem de “opção inválida” no terminal

Relatório de Execução de Testes		
Entradas	**Resultado**	**Aprovado?**
“0”	Exibição de mensagem de saída e encerramento do programa.	Sim
“1”	Exibição do menu de alteração do veículo escolhido.	Sim
“2”	Exibição do menu de alteração do veículo escolhido.	Sim
“3”	Exibição do menu de alteração do veículo escolhido se não houver opção de escolha exibe mensagem “opção inválida” e encerra o programa.	Sim
“Vermelho”	Altera a cor no arquivo.	Sim
“Chevrolet”	Altera a Marca no Arquivo	Sim
“Onix”	Altera o Modelo no Arquivo	Sim
“7”	Erro e interrupção do programa.	Não
“abcd”	Altera a Marca ou Cor ou Modelo no Arquivo	Sim

Teste 5: Menu de Ordenação				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**
números inteiros	Números entre 1 a 4	Exibição da ordenação escolhida ou fechamento do	números menores que 1 e maiores que 4,letras	Mensagem de “opção inválida” no terminal,Erro e

		programa		interrupção do programa
Relatório de Execução de Testes				
Entradas	**Resultado**			**Aprovado?**
“0”	Mensagem de “opção inválida”, limpa do console e re-exibição do menu de operações			Sim
“1”	Exibição da ordenação desejada			Sim
“2”	Exibição da ordenação desejada			Sim
“3”	Exibição da ordenação desejada			Sim
“4”	Exibição de mensagem de saída e encerramento do programa.			Sim
“5”	Mensagem de “opção inválida”, limpa do console e re-exibição do menu de operações			Sim
“O”	Erro e interrupção do programa			Não

Teste 6: Menu de Relatório				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**
números inteiros	dia/mês/ano	Exibição do relatório do histórico de compras/vendas	strings	Erro e interrupção do programa
Relatório de Execução de Testes				
Entradas	**Resultado**			**Aprovado?**
“11/12/2023”	Exibição da do relatório da data desejada			Sim
“01/10/2020”	Exibição da do relatório da data desejada			Sim
“14/09/2023”	Exibição da do relatório da data desejada			Sim
“11/12/2033”	Exibição da ordenação desejada			Sim
“30/02/2023”	Exibição de mensagem “data inválida” limpa do console e re-exibição do menu de operações			Sim
“04/13/2023”	Mensagem de “data inválida”, limpa do console e			Sim

	re-exibição do menu de operações	
“O”	Erro e interrupção do programa	Não

Teste 7: Menu de Confirmação de Compra/Venda				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**
números inteiros	Números entre 0 a quantidade de opções disponíveis	Confirmação de compra/venda e edição nos arquivos	números menores que 0 e maiores que a quantidade e opções disponíveis,letras	Mensagem de “opção inválida” no terminal,Erro e interrupção do programa
Relatório de Execução de Testes				
Entradas	**Resultado**			**Aprovado?**
“0”	Mensagem de “Compra/Venda cancelada”, limpa do console e encerramento do programa.			Sim
“1”	Exibição de “Compra/Venda efetuada com sucesso”			Sim
“2”	Exibição de “Compra/Venda efetuada com sucesso”			Sim
“3”	Exibição de “Compra/Venda efetuada com sucesso”caso exista uma opção válida			Sim
“4”	Mensagem de “opção inválida” no terminal e encerramento do programa.			Sim
“5”	Mensagem de “opção inválida”no terminal e encerramento do programa.			Sim
“O”	Erro e interrupção do programa			Não

Teste 8: Menu de Taxa				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**
números decimais e inteiros	Números entre 0 a quantidade de taxa desejada	aplica taxa no valor do veiculo	letras	Erro e interrupção do programa
Relatório de Execução de Testes				

Entradas	Resultado	Aprovado?
“0.1”	Adição de 10% de taxa	Sim
“0.2”	Adição de 20% de taxa	Sim
“0.5”	Adição de 50% de taxa	Sim
“3”	Adição de 300% de taxa	Sim
“1”	Adição de 100% de taxa	Sim
“a”	Erro e interrupção do programa	Não