Pontifícia Universidade Católica de Minas Gerais

PUC Minas

Disciplinas: Fundamentos de Engenharia de Software
Algoritmos e Estruturas de Dados I
Curso: Engenharia de Software
Entrega: 02/07/2022 (FES)
Valor: 10 pontos (FES) – 10 pontos (AEDsI)

Observações*:*

- O trabalho poderá ser feito em **grupos de até 3 alunos**.
- Cópias de trabalho receberão nota **ZERO**.
- O trabalho será avaliado em **10 pontos**.
- O programa deve ser feito na linguagem de programação C.
- As informações mencionadas e manipuladas neste trabalho deverão ser armazenadas em arquivo(s) de **acesso direto**. Portanto, deverá ser feita leitura e escrita em arquivos.
- O trabalho deverá ser entregue pelo Canvas até o dia **02/07/2022** às **23:59 horas**.
- O grupo deve preparar uma apresentação gravada com a participação de todos os seus componentes. Essa apresentação também deverá ser entregue no Canvas.
- Deverá ser entregue o **projeto completo** do programa, a **documentação**, os **arquivos** contendo os testes realizados e a apresentação gravada.
- Em caso de dúvida, entre em contato com seu professor.

Locadora LocaMais

LocaMais é uma locadora de veículos que tem como objetivo atender bem e fidelizar seus clientes. A princípio a LocaMais está trabalhando apenas com locações baseadas em diárias, ou seja, a quilometragem usada pelo cliente é livre. A locadora está localizada em diversas cidades do estado de Minas Gerais, no entanto, a filial de Confins está com sérios problemas, pois ainda não implantou um sistema de *software*para realizar seus controles e gerenciamento de locações de veículos. Muitas vezes um cliente liga solicitando a locação de um veículo e a atendente não consegue saber se será possível atendê-lo. Sem falar que não é possível ter um controle de quantas locações foram feitas por um determinado cliente e premiá-lo de alguma forma. Diante dos problemas vividos pela LocaMais, a locadora resolveu contratar uma empresa desenvolvedora de *software* (vocês). Sendo assim, é necessário compreender a real necessidade da locadora e desenvolver um *software* específico. A seguir foi descrito como deverá ser o sistema, bem como suas restrições.

O sistema

Deseja-se cadastrar os clientes, os veículos e as locações da locadora. As informações que devem ser cadastradas são:

CLIENTE = código, nome, endereço, telefone
LOCAÇÃO = código da locação, data de retirada, data de devolução, seguro, quantidade de dias, código do cliente, código do veículo
VEÍCULO = código do veículo, descrição, modelo, cor, placa, valor diária, quantidade de ocupantes, *status*

Considere as seguintes restrições: ** *Não se esqueça de sempre validar essas restrições*

Para cadastrar uma locação, primeiro é necessário que o cliente e o veículo estejam cadastrados. As locações devem ser cadastradas apenas para veículos com *status* disponível, sendo que os possíveis *status* para o veículo são: alugado, disponível e em manutenção. Além disso, não deve ser feita mais de uma locação para um mesmo veículo em um mesmo período (data de retirada e data de devolução). As locações podem ser feitas no horário comercial (de 8:00 às 18:00 horas).

1. Implemente uma função para cadastrar um cliente. Esta função deve garantir que não haverá mais de um cliente com o mesmo código. Se quiser pode gerar o código automaticamente.

2. Implemente uma função para cadastrar um veículo. Esta função deve garantir que não haverá mais de um veículo com o mesmo código. Se quiser pode gerar o código automaticamente.

3. Implemente uma função que cadastre uma locação. Para cadastrar a locação, o sistema deve receber do usuário o código do cliente que deseja alugar o veículo, a data de retirada, a data de devolução do veículo e a quantidade desejada de ocupantes. A partir disso, o sistema deve encontrar um veículo que esteja disponível para a necessidade do cliente e calcular a quantidade de diárias com base nas datas de retirada e devolução. O cliente também pode contratar um seguro para o veículo alugado, caso queira. O valor do seguro é de R$ 50,00 e será acrescido ao valor total pago pela locação do veículo.

4. Implemente uma função que dê baixa em uma determinada locação, calcule e mostre o valor total a ser pago por um determinado cliente. Lembre-se de alterar o *status* do veículo para disponível. Além disso, será necessário solicitar a data de entrega do veículo para validar com a data de devolução prevista. Também será preciso calcular multa por atraso, caso o cliente não entregue o veículo no dia combinado (considere como multa o valor de 5% do valor total da locação + R$ 30,00 por dia de atraso).

5. Implemente funções para realizar pesquisas tanto por clientes quanto porveículos, ou seja, digitando o código,apresente na tela todas as informações do cliente ou veículo correspondente. Deve existir também a possibilidade de listar todos os clientes e todos os veículos.

6. Implemente uma função que mostre na tela todas as locações de um determinado cliente (a pesquisa deve se basear no código do cliente).

7. Implemente uma função que calcule a quantidade de pontos de fidelidade de um cliente. Para cada dia de locação, o cliente ganhará 10 pontos no programa de fidelidade. Lembre-se que é possível ter mais de uma locação para um mesmo cliente. Implemente também uma função para pesquisar os clientes que atingiram um total de 500 pontos no programa de fidelidade, esses clientes serão premiados e receberão um *kit* da LocaMais.

8. Implemente uma função extra, criada pelo grupo. Sejam criativos!

Para fazer esse programa pode ser necessário criar mais funções do que as que estão descritas. Finalmente, faça uma função *main*() que teste o sistema acima. A função *main*() deve exibir um *menu* na tela, com as opções de cadastrar um cliente, um veículo e uma locação. Além disso, permitir realizar as pesquisas e dar baixa em locações. Este *menu* deve ficar em *loop* até o usuário selecionar a opção SAIR. Além disso, todas as informações deverão ser persistidas/armazenadas em arquivo(s) de acesso direto. Portanto, deverá ser feita leitura e escrita em arquivos.

** Não é obrigatória a implementação do programa usando Orientação a Objetos, mas caso o grupo queira usar, é permitido.*

Metodologia

Este é um trabalho interdisciplinar em que você deve planejar, analisar, projetar, implementar e testar uma solução de *software* para o problema apresentado utilizando o Scrum para gerenciar seu progresso.

Inicialmente organize o *backlog* do produto com as funções básicas do sistema. Cada uma das funções será de responsabilidade de um membro do grupo e será desenvolvida em *sprints* de 3 a 4 dias. Seguem algumas sugestões de atividades a serem realizadas nas *sprints*:

1- Definir a assinatura da(s) função(ões). Reflita sobre os parâmetros de entrada e saída da função e comunique aos seus colegas de projeto.

2- Documentar a função indicando seu propósito, e os parâmetros de entrada e saída. O nome da função deve ser escolhido sob o ponto de vista de quem usa a função ou de quem vai chamar a função e deve refletir o que a função faz.

3- Implementar o caso de sucesso da função.

4- Selecionar casos de teste para verificar o funcionamento da função. Um caso de teste deve conter os valores de entrada para a funçãoe a saída esperada.

5- Executar os casos de teste planejados para a função. Inicie fazendo a execução manual de alguns poucos casos de teste. Em seguida implemente a automatização dos testes da função usando a biblioteca *munit*. Especialmente as funções de cálculo de valores (multa, quantidade de pontos no programa de fidelidade, valor total da locação) devem ser testadas de forma automatizada.

6- Criar um relatório de execução de testes que contenha os casos de teste, a saída retornada durante sua execução e uma indicação se o teste passou ou não na função. Isso é feito comparando-se a saída esperada, documentada no caso de teste, com a saída retornada durante a execução da função (esperado x real).

7- Implementar os casos especiais, exceções que possam existir na função. Em seguida, executar os casos de teste anteriores para garantir que as mudanças não quebraram o código anterior que já funcionava. Pense também nos novos casos de teste necessários para a nova versão da função.

O que deve ser entregue para os professores no Canvas

1- A evolução do *backlog* do produto. Indique quais tarefas encontravam-se inicialmente no *backlog* do produto, e em qual *sprint* cada tarefa foi alocada, juntamente com seu responsável.

2- A documentação das funcionalidades do *software*.

3- O planejamento dos casos de teste (entradas, procedimento de teste e saídas esperadas), a implementação dos casos de teste automatizados e o relatório de execução dos testes.

4- O código, em C, das funções e do programa principal, juntamente com o projeto completo do *software*.

5- Arquivos contendo dados já incluídos para teste das funcionalidades.

6- Apresentação gravada em vídeo (*pitch*) mostrando todas as funcionalidades do sistema.

Link para a biblioteca munit: https://nemequ.github.io/munit/