

Trabalho Prático Interdisciplinar

Disciplinas: Fundamentos de Engenharia de Software
Algoritmos e Estruturas de Dados I
Curso: Engenharia de Software
Profs.: Eveline Alonso Veloso e Roberto Felipe Rocha
Entrega: 30/06/2023
Valor: 7 pontos (AEDs I)

Observações:

- O trabalho poderá ser feito em **grupos de até 3 alunos**.
- Cópia de trabalho receberão a nota **ZERO**
- O programa deve ser feito na linguagem de programação C
- As informações deverão ser armazenadas em arquivo(s) **acesso direto**, portanto, deverá ser feita leitura e escrita em arquivos.
- O trabalho deverá ser entregue pelo Canvas até o dia **30/06/2023 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 e deve demonstrar todas as funcionalidades do software.
- 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

Salão de Festas Patati Patata

O salão de festas Patati Patata é um salão de festas infantil que foi inaugurado no final de 2021 e, por isso, os donos ainda não haviam se preocupado com a implantação de sistemas para realizar o controle e gestão do negócio. No entanto, alguns problemas já começaram a aparecer, como por exemplo, a marcação de duas festas para um mesmo dia em horários que coincidem. Além disso, alguns dados de clientes e fornecedores, que deveriam ser armazenados e de fácil recuperação, não são feitos. Diante dos problemas vividos pelo salão, os donos resolveram contratar uma empresa desenvolvedora de sistemas (você). Sendo assim, é necessário compreender a real necessidade do salão 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 funcionários, os fornecedores e as festas. As informações que devem ser cadastradas são:

- CLIENTE = código, nome, endereço, telefone, data de nascimento
- FORNECEDOR = código, nome do buffet, telefone
- FESTA = código festa, quantidade de convidados, data, dia da semana, horário (início e fim), tema, código cliente, código do fornecedor

- CONTRATO = número contrato, valor total, desconto, valor final, forma de pagamento, status, código festa

Considere as seguintes **restrições**: para cadastrar uma festa, primeiro é necessário que o cliente esteja cadastrado. As festas devem ser cadastradas apenas em datas/horários que não tenham outras festas que coincidem. Considere que apenas no sábado o horário da festa é definido (fixo), pois acontecem duas festas, uma festa das 12 às 16 horas e outra das 18 às 22 horas. Já nos outros dias o cliente pode escolher o horário de início e fim que desejar, lembrando que a duração das festas é de 4 horas.

**** Não esqueça de sempre validar essas restrições**

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 fornecedor. Esta função deve garantir que não haverá mais de um fornecedor com o mesmo código. Se quiser pode gerar o código automaticamente.
3. Implemente uma função que cadastre uma festa. Para cadastrar uma festa, o sistema deve receber do usuário o código do cliente que deseja realizar a festa, a quantidade de convidados, a data da festa (lembre-se de validar se já existe festa), horário e tema. Cada festa deverá ser cadastrado 1 fornecedor (o fornecedor já deve estar cadastrado no sistema)
4. Implemente uma função para calcular o valor total a ser pago, baseado na **tabela 1** e o valor final a ser pago, que deverá ser calculado baseado na forma de pagamento, para isso, deverá ser usada a **tabela 2**. Lembre-se de atualizar o status do contrato sempre inicialmente para “a pagar”.

Tabela 1

Qtde de convidados	Dia da semana	Valor
30	Segunda a quinta	R\$ 1899,00
30	Sexta a domingo	R\$ 2099,00
50	Segunda a quinta	R\$ 2199,00
50	Sexta a domingo	R\$ 2299,00
80	Segunda a quinta	R\$ 3199,00
80	Sexta a domingo	R\$ 3499,00
100	Segunda a quinta	R\$ 3799,00
100	Sexta a domingo	R\$ 3999,00

Tabela 2

Forma de pagamento	Desconto
A vista	10%
Duas vezes	5%
Três vezes	2%
Quatro ou mais vezes	Sem desconto

5. Implemente uma função que permita atualizar o status do contrato do cliente para pago ou cancelado.
6. Implemente funções para realizar pesquisa no sistema (pelo nome/código do cliente):
 - a. informações de clientes
 - b. informações de fornecedores
7. Implemente uma função (relatório) que mostre na tela todas as festas de um determinado cliente.
8. Implemente uma função (relatório) que mostre na tela as informações de uma determinada festa a partir de uma data. Mostre inclusive as informações referente ao contrato da festa com valor total e final.

Para fazer este 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 funcionário, um fornecedor e uma festa. Além disso, permitir realizar as pesquisas. Este menu deve ficar em loop até o usuário selecionar a opção **SAIR**. Além disso, todas as informações deverão ser armazenadas em arquivo(s) binário com **acesso direto**, portanto, deverá ser feita leitura e escrita em arquivos.

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 função e 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.
- 6- Criar um relatório de execução de testes que contenha os casos de teste, a saída retornada durante sua execução e uma indicação se a função passou ou não no teste. 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 a cada semana. 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 acesso direto 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/>