Universidade Federal
de Ouro Preto

Enya Luísa Gomes dos Santos - 19.2.4201
Nathann Zini dos Reis - 19.2.4007
Vitória Maria Silva Bispo - 19.2.4109

TRABALHO PRÁTICO I - PET & SHOP

Relatório apresentado por exigência da disciplina BCC221 - Programação Orientada a Objetos, da Universidade Federal de Ouro Preto.
Professor: Guillermo Camara Chavez

INTRODUÇÃO

O trabalho prático tem como objetivo a implementação de um sistema de gerenciamento de um Pet Shop. Os serviços oferecidos pelo estabelecimento são *venda de produtos* para cães, gatos e pássaros, *consultas* e *banho e tosa*. Para operar esse sistema, é necessário que haja três tipos de usuários: o *administrador*, que possui acesso às principais funções do sistema, os *vendedores*, que possuem acesso às funções referente à venda de produtos e serviços e os *veterinários*, que possuem acesso apenas ao cadastro de clientes e à ordem de serviço.

Através disso, foi identificado os principais métodos e propriedades de cada entidade desse sistema do Pet Shop para implementar classes e construir objetos para o funcionamento do software.

ARQUITETURA E IMPLEMENTAÇÃO

A arquitetura do projeto foi implementada pensando na melhor forma seguindo as instruções do enunciado. Dentro das três principais classes do programa, *Administrador, Vendedor e Veterinário,* há funções que esses são permitidos de executar durante a execução do sistema, e algumas dessas funções necessitam do relacionamento entre outras classes para ser implementada. A lógica utilizada na arquitetura segue a lógica do menu.

Registros: A ideia dessa classe foi centralizar todos os dados do programa em um único lugar. Um único objeto dessa classe é criada e passada como referência para todos os outros métodos para ter acesso aos vectors com os dados do programa;

Login: Criamos uma classe template para login que funciona para qualquer tipo de dado que for fornecido como parâmetro. No caso, cada classe chama as funções do Login passando o objeto ;

Usuário: A classe Usuário é a classe base abstrata para as outras pessoas do sistema (veterinário, vendedor e administrador). Ela não é instanciada, pois há métodos que somente serão implementados nas classes que derivarem dela;

Administrador: Uma das classes que deriva da classe abstrata Usuário, administrador é basicamente responsável pelas principais funções do sistema. É o único que consegue realizar cadastros de funcionários, gera relatório do sistema,

visualiza as pessoas cadastradas no sistema, bem como as contas, pois o mesmo é responsável por pagá-las, bem como comprar produtos para a empresa. Para realizar todas as funções acima ele tem que está logado;

Vendedor: Classe derivada da classe abstrata Usuário. Essa classe é responsável por vender produtos para cães, gatos e pássaros e serviços de consulta, banho e tosa. Além disso, o vendedor possui permissão para visualizar os clientes, produtos, criar ordem de serviço e cadastrar clientes no sistema. A classe só consegue realizar as operações se estiver acessado o sistema com o login definido previamente pelo administrador;

Veterinário: Classe derivada da classe abstrata Usuário, o veterinário é responsável por realizar o serviço contratado pelo cliente. Para isso, ele tem acesso às ordens de serviço e ao final do tratamento, gera o relatório referente ao serviço prestado para aquela ordem de serviço. Além disso, ele pode visualizar os clientes cadastrados. Para realizar todas as funções acima ele tem que está logado;

Venda: Classe criada para realizar as vendas. Como o Pet Shop oferece diferentes tipos de serviços, utilizamos sobrecarga de função e operador para tratar essa especificidade;

Serviço: Classe criada para registrar os serviços que o pet & shop oferece, ela é acessada pela ordem de serviço e venda;

Produto: Classe criada para registrar os produtos vendidos pelo Pet Shop. Essa classe é constantemente utilizada pelo vendedor no momento de realizar as vendas. Outra classe que a utiliza é a Administrador, que sempre atualiza ao fazer compra de novos produtos;

OrdemServico: Classe criada para armazenar todos os serviços solicitados pelos clientes. Guarda as informações do tipo de serviço, tipo do animal, qual foi o cliente, a data, e o relatório final (que é gerado pelo veterinário);

DataHorario: Classe desenvolvida para registrar data e hora, com propriedades de dia, mês, ano, hora e minuto. Tal classe é bastante usada dentro de outras classes presentes no programa;

Conta: Classe desenvolvida para representar as contas que foram geradas através da compra de novos produtos para o Pet Shop. A classe que a utiliza constantemente é a Administrador;

Cliente: Classe criada para identificar os clientes no momento da criação de uma ordem de serviço gerada pelo vendedor;

Main: Responsável por toda a execução do programa. Também controla a classe Registro, criando seu objeto para a execução do programa. Todas as partes de interação com o usuário são feitas na classe main. Desde a leitura dos dados quanto à escrita na tela;

Estilo: Arquivo cabeçalho implementado para utilizar cores e caracteres ANSI no terminal.

INSTRUÇÕES DE COMPILAÇÃO E EXECUÇÃO

Para facilitar na execução do projeto levando em consideração o número de arquivos, foi usado o Makefile para a compilação. Segue abaixo a implementação do Makefile.

```
CC = g++
CFLAGS = -Wall
CNAME = exec

all: Usuario.o Administrador.o Vendedor.o Veterinario.o Venda.o Servico.o
Produto.o OrdemServico.o DataHorario.o Conta.o Cliente.o main.o
 $(CC) $(CFLAGS) -o $(CNAME) ./O/main.o ./O/Usuario.o ./O/Administrador.o
./O/Vendedor.o ./O/Veterinario.o ./O/Venda.o ./O/Servico.o ./O/Produto.o
./O/OrdemServico.o ./O/DataHorario.o ./O/Conta.o ./O/Cliente.o

Usuario.o: Usuario.cpp Usuario.hpp
 $(CC) $(CFLAGS) -o ./O/Usuario.o -c Usuario.cpp

Administrador.o: Administrador.cpp Administrador.hpp
 $(CC) $(CFLAGS) -o ./O/Administrador.o -c Administrador.cpp

Vendedor.o: Vendedor.cpp Vendedor.hpp
 $(CC) $(CFLAGS) -o ./O/Vendedor.o -c Vendedor.cpp

Veterinario.o: Veterinario.cpp Veterinario.hpp
 $(CC) $(CFLAGS) -o ./O/Veterinario.o -c Veterinario.cpp

Venda.o: Venda.cpp Venda.hpp
 $(CC) $(CFLAGS) -o ./O/Venda.o -c Venda.cpp

Servico.o: Servico.cpp Servico.hpp
 $(CC) $(CFLAGS) -o ./O/Servico.o -c Servico.cpp

Produto.o: Produto.cpp Produto.hpp
 $(CC) $(CFLAGS) -o ./O/Produto.o -c Produto.cpp
```

```
OrdemServico.o: OrdemServico.cpp OrdemServico.hpp
 $(CC) $(CFLAGS) -o ./O/OrdemServico.o -c OrdemServico.cpp

DataHorario.o: DataHorario.cpp DataHorario.hpp
 $(CC) $(CFLAGS) -o ./O/DataHorario.o -c DataHorario.cpp

Conta.o: Conta.cpp Conta.hpp
 $(CC) $(CFLAGS) -o ./O/Conta.o -c Conta.cpp

Cliente.o: Cliente.cpp Cliente.hpp
 $(CC) $(CFLAGS) -o ./O/Cliente.o -c Cliente.cpp

main.o: main.cpp
 $(CC) $(CFLAGS) -o ./O/main.o -c main.cpp

clean:
 rm -rf ./O/*.o $(CNAME)

run:
 ./$(CNAME)
```

Graças ao Makefile, a execução do programa é simples:

`make clean`

Comando executado para remover todos os arquivos .o e o executável anteriores caso exista.

`make all`

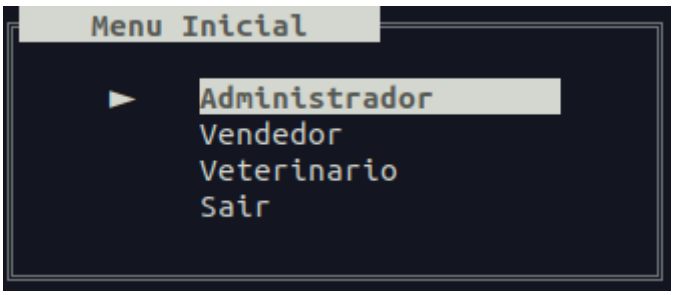
Comando executado criar o executável do programa

`make run`

Comando executado para executar o programa.

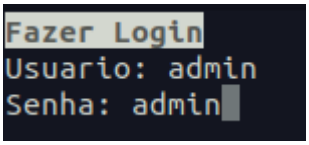
Para navegar por esse menu, também é necessário utilizar as setas de navegação do teclado (UP e DOWN) e enter para confirmar. As telas de menu, com exceção do menu inicial, só podem ser acessadas se a entidade a que se refere tiver passado pelo processo do login.

Menu Inicial



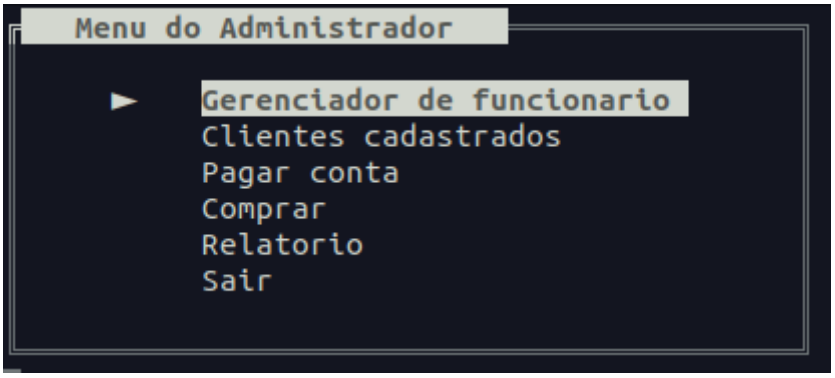
Menu inicial do sistema do Pet Shop, nele o usuário possui a opção de entrar como Administrador, Vendedor, Veterinário ou sair, encerrando a execução do programa. Para navegar entre as operações do menu, basta utilizar as setas de navegação do teclado.

Login do Administrador



Ao acessar o sistema como administrador, é necessário fazer login utilizando “admin” como usuário e senha. O login de administrador é único, ou seja, só é possível entrar como administrador utilizando esses dados.

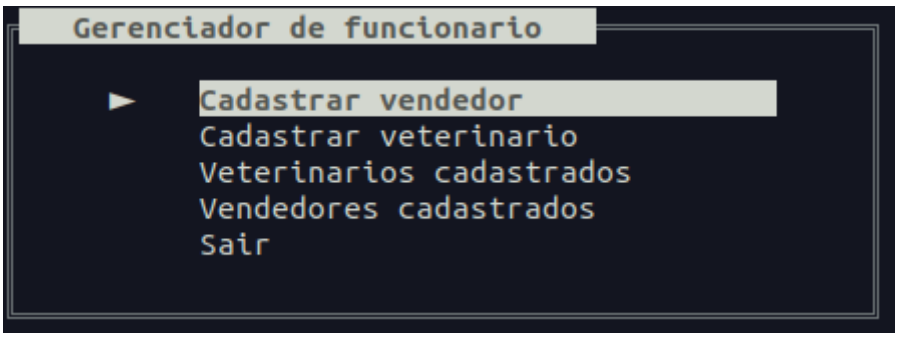
Menu do Administrador



O menu do administrador possui as seguintes funcionalidades: gerenciador de funcionário, clientes cadastrados, pagar conta, comprar, relatório e sair do menu

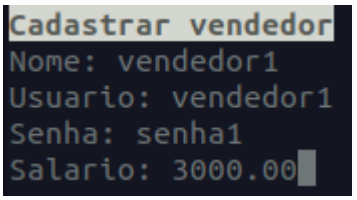
do administrador. O administrador do sistema só consegue realizar essas operações se tiver entrado no sistema com o usuário e senha padrão.

Gerenciador de funcionário



O gerenciador de funcionário possui as seguintes funcionalidades: cadastrar vendedor, cadastrar veterinário, vendedores cadastrados e veterinários cadastrados.

Cadastrar vendedor

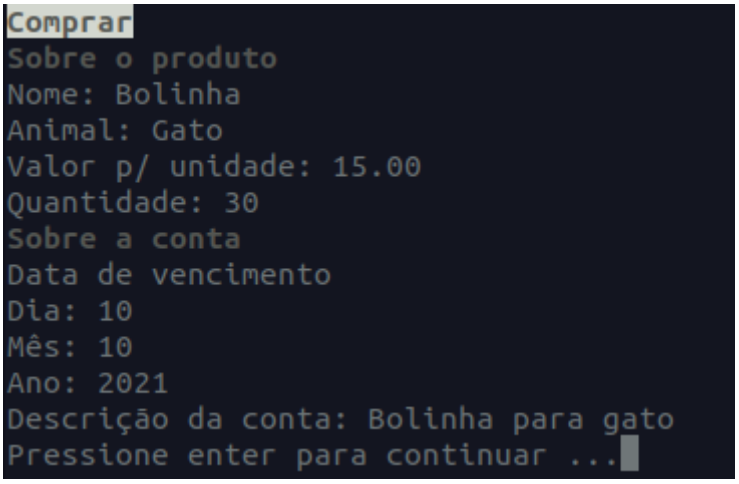


No cadastro de vendedor, é necessário inserir o nome do vendedor, usuário, senha e salário. O usuário e senha serão utilizados para acessar o menu do vendedor.

Vendedores cadastrados

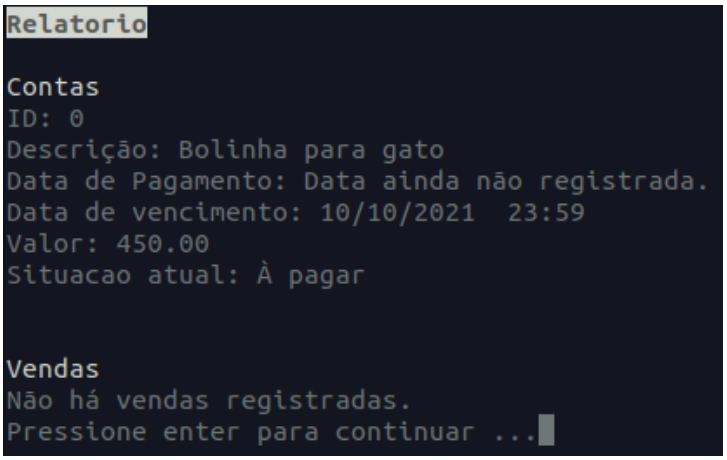
Em vendedores cadastrados, uma lista com os vendedores cadastrados é exibida. Os dados mostrados são: ID, nome, usuário, salário. Para sair da exibição, basta apertar enter.

Comprar



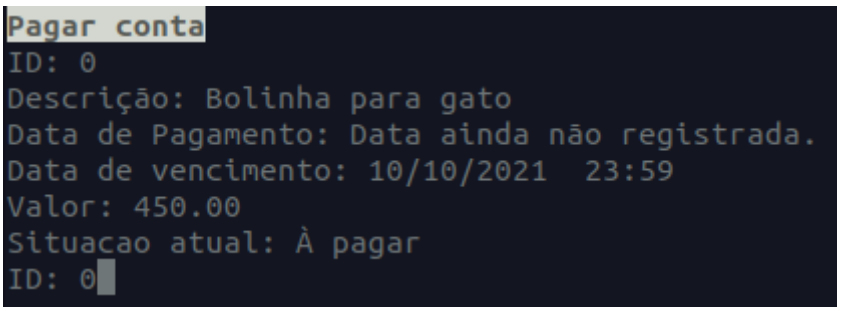
Em comprar, é possível comprar produtos para abastecer a loja do Pet Shop, o administrador deve inserir as informações sobre o produto e sobre a conta. Após realizar a compra, ela fica registrada para pagamento.

Relatório



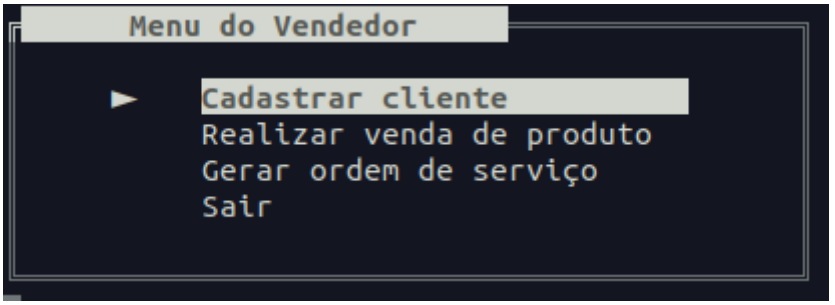
O relatório gerado pelo administrador exibe todas as contas que a loja possui e todas as vendas realizadas anteriormente.

Pagar conta



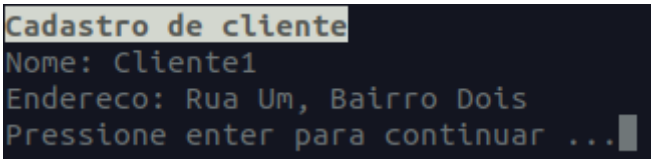
Para pagar a conta, apenas o administrador pode fazer, é exibido todas as contas que ainda há de ser pagas e então é digitado o ID referente a conta que deseja pagar.

Menu do Vendedor



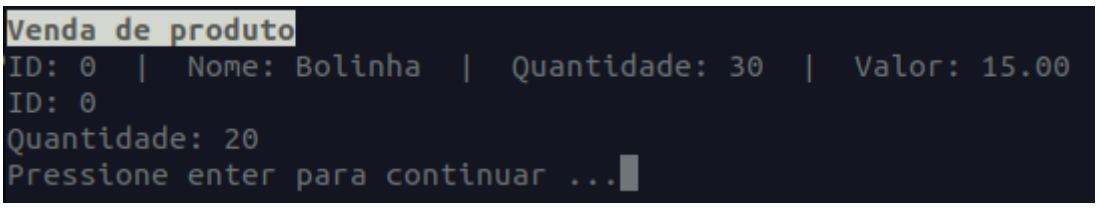
Menu exibido quando o vendedor realiza o login no sistema

Cadastro de cliente



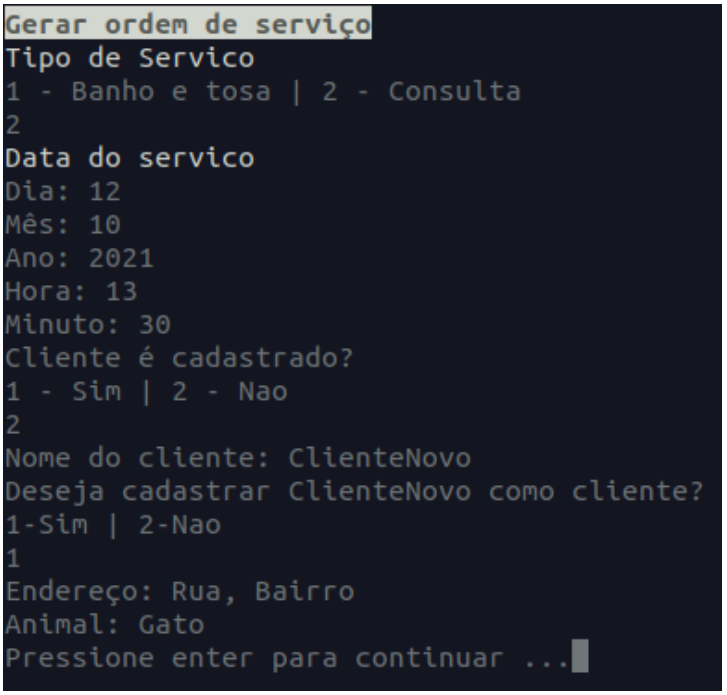
Funcionalidade do vendedor, permite que ele cadastre um cliente no sistema

Venda de produto



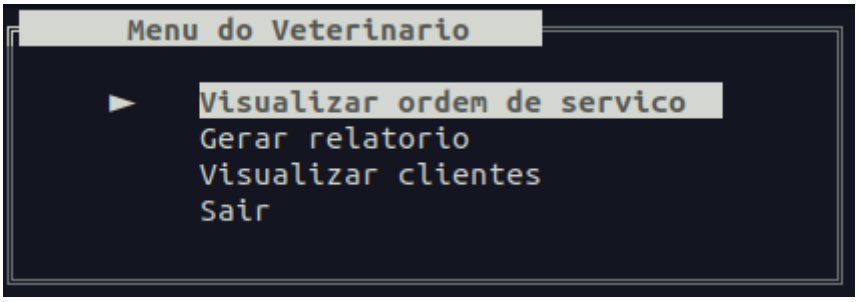
Funcionalidade do vendedor, é exibido todos os produtos que há cadastrado no sistema, tanto quanto a quantidade de cada um e o valor unitário. Então é digitado o ID referente ao produto que deseja vender e a quantidade que será vendida.

Gerar ordem de serviço



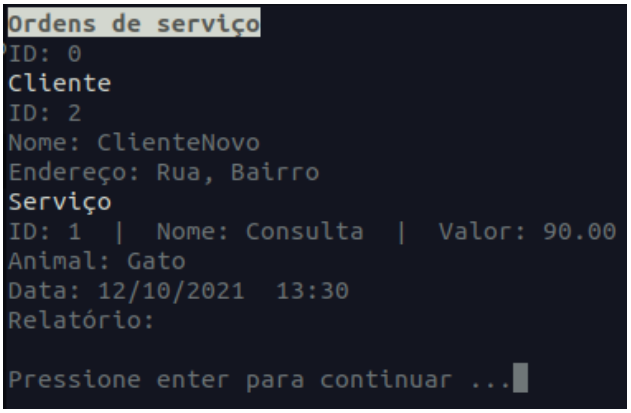
Para o vendedor gerar a ordem de serviço, quando a venda é na verdade um serviço da loja, é exibido os serviços ofertados que deve ser escolhido pelo ID. Logo após, é informado a data referente a quando o serviço será realizado e atribui um cliente a ordem de serviço. Esse cliente pode está pré cadastrado, nesse caso é exibido a lista de cliente cadastrados e é escolhido pelo ID. Caso não esteja pré-cadastrado, há a opção de cadastrar o cliente, nesse caso digita as informações do cliente, ou apenas gerar uma ordem de serviço no nome dele sem que ele seja cadastrado no sistema.

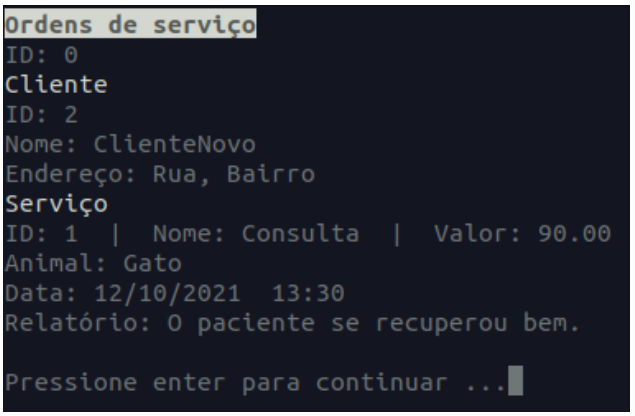
Menu do Veterinário



No menu de veterinário apresenta as opções: Visualizar ordem de serviço, gerar relatório, visualizar clientes, sair. Esse menu é exibido somente se o veterinário conseguir fazer o processo de login.

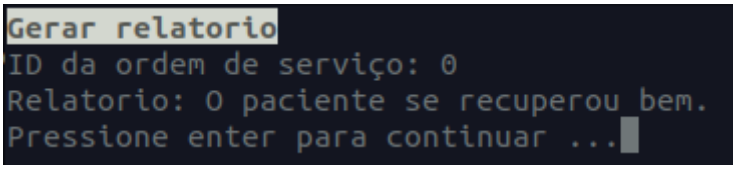
Ordens de serviços





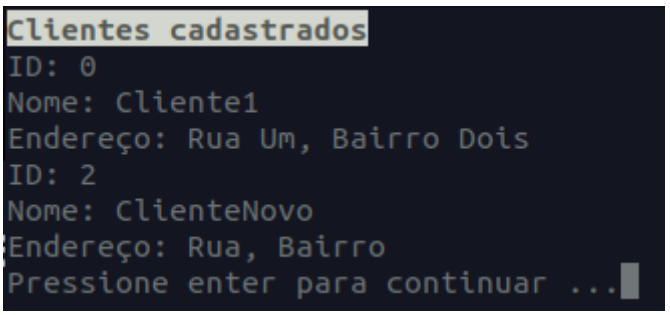
Lista todas as ordens de serviços de consulta gerada pelo vendedor, a imagem da esquerda apresenta uma ordem de consulta sem o relatório, a imagem da direita apresenta a mesma ordem de serviço porém com a consulta registrada pela tela a seguir.

Gerar relatório



Através do ID de uma ordem de serviço de consulta, já cadastrada, é possível que o veterinário registre um relatório, que consiste em uma string.

Clientes cadastrados



Apresenta a listagem de todos os clientes cadastrados.

Todas as opções de menu apresentam a opção sair, nos menus de administrador, vendedor e veterinário, a opção retorna o usuário para o menu inicial. O sair do menu inicial encerra o programa.

RECURSOS DE LINGUAGEM

Bibliotecas:

<termios.h> - Define a estrutura do arquivo *termios*, que fornece a interface do terminal para compatibilidade POSIX. No programa foi usado para construir o menu que é controlado pelas teclas de seta (UP e DOWN) do teclado. Fonte: https://www.ibm.com/docs/en/aix/7.2?topic=files-termiosh-file.

<cstring> - Este arquivo de cabeçalho define várias funções para manipular strings C e matrizes. Foi usado para calcular o tamanho da string com *srtlen*. Fonte: <cstring> (string.h) - C++ Reference.

Além dessas mais específicas, foram utilizadas as seguintes bibliotecas padrões como, *string, vector, iostream e iomanip*.

Recursos externos:

Makefile: O Makefile foi utilizado para facilitar a compilação e a execução no terminal, visto que o número de arquivos *.cpp* e *.hpp* era grande.

DIFICULDADES

O trabalho foi inteiro realizado em conjunto com todos os integrantes do grupo que discutiram todas as soluções para os problemas encontrados.

Dentre as dificuldades encontradas, tivemos um problema com a leitura de strings e outros tipos de dados primitivos da linguagem, porém, depois de bastante calma e conversa entre nós mesmos, foi encontrada a solução para isso.

Também, a criação de uma classe template foi um pouco mais complicada que as demais e buscamos exemplos na internet para ter uma base de criação.

Entretanto, a maior dificuldade encontrada pelo grupo foi conciliar a realização do trabalho prático com os horários dos integrantes, principalmente pelo período EAD estar, em comparação ao que estaria no presencial, que teríamos algumas semanas a mais de aula, mais estreito o prazo de entrega e realização de todas as atividades de todas as matérias.

DECISÕES DO PROJETO

Como mencionado anteriormente, todas as decisões relacionadas ao projeto foram tomadas em conjunto, em reunião com todos os membros do grupo.

REFERÊNCIAS

Disponível em: <Unicode/UTF-8-character table - starting from code position 2500>
Acesso em 21 de Junho de 2021.
Disponível em: <ANSI escape code> Acesso em 21 de Junho de 2021.