

Trabalho Prático

PDS II

Coleta Seletiva

Daniel Oliveira Souza

Felipe Costa Gomes

Maria Fernanda Fávaro de Almeida

Vitor Hugo Lacerda Lana

Introdução:

O programa desenvolvido foi baseado na proposta do Trabalho Prático da disciplina PDS II.

Desse modo, o sistema tem a funcionalidade principal de auxiliar pessoas e instituições a participarem do processo de coleta seletiva da região onde o software estiver disponível.

Primeiramente, para ter acesso ao sistema, o usuário deve se cadastrar e efetuar o login.

Depois disso, ele terá acesso aos menus: Material, Local e Agendamento de Coletas. Estes possuem as funcionalidades gerais do programa que serão descritas mais afundo posteriormente.

Essencialmente, após o cadastro, o usuário, pode inserir no sistema os seus materiais disponíveis para a coleta e também cadastrar locais onde as doações podem ser realizadas. Além disso, com a funcionalidade dos agendamentos, ele tem a possibilidade de escolher uma data, horário e local específico para que a ocorra contribuição.

Implementação:

No processo de implementação do programa, ele foi dividido em cinco grandes partes principais:

* Menu Principal
* Agendamento
* Local
* Material
* Usuário

A seguir, cada uma delas será apresentada detalhadamente.

Menu Principal:

O menu tem a funcionalidade principal de fazer a interface usuário-sistema.

Constituído pelas funções:

* Pessoa \*MenuUsuario()

Responsável pelo cadastro de usuários e pelo login. Ela retorna o usuário que está logado naquele momento.

* void MenuPrincipal(Pessoa \*usuario)

Responsável por conjugar a distribuição dos outros menus.

* void MenuMaterial(Pessoa \*usuariologado)

Responsável por chamar funções da classe CadastroMaterial (apresentada posteriormente), como cadastrar, excluir e visualizar os materiais cadastrados.

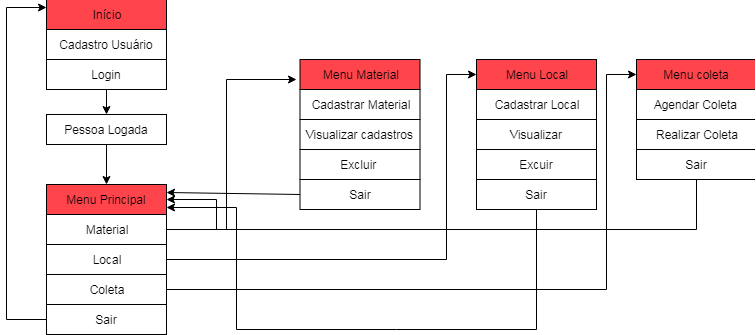
* void MenuLocal()

Responsável por chamar funções da classe CadastroLocal.

* void MenuCadastroColeta(Pessoa \*usuario)

Responsável por manipular o processo de agendamento de coletas.

O diagrama a seguir mostra como as funções foram interligadas:



Agendamento:

A parte do programa destinada ao agendamento é encarregada de pegar os dados relacionados a esse aspecto e fazer a interface entre o agendamento de coleta e a realização da mesma. O primeiro é feito pelo doador e o segundo pelo receptor.

O doador escolhe o material a ser doado e o usuário que irá recebe-lo. Com isso, armazena-se esses dados em um vetor.

Depois disso, o usuário receptor, utilizando sua conta afirmara se a coleta foi realizada com sucesso ou não.

Local:

A fatia destinada ao local possui as classes: Local e CadastroLocal.

A classe local é responsável por armazenar o nome do local cadastrado e tem funções básicas como get e set para eventuais manipulações.

A classe CadastroLocal tem as funções:

* void cadastro()

Responsável por criar um novo local e adicioná-lo a um vetor da classe.

* int escolheponto()

Retorna os locais cadastrados no vetor para que o usuário possa escolher algum para fazer determinada função.

* void imprime\_locais()

Tem a única funcionalidade de imprimir na tela.

* void excluir\_local().

Usa a função escolheponto para que o usuário exclua determinado local de coleta.

Material:

A parte “Material” do software é formada pelas classes que descrevem os materiais passíveis de coleta e pela classe CadastroMaterial.

Assim, para organizar os materiais, criou-se uma classe mãe chamada Material, que possui um construtor com o nome (tipo), a quantidade e uma breve descrição sobre ele. Além disso, há uma função “void modo\_de\_armazenamento()” susceptível de ser sobrescrita nas classes filhas.

As classes que herdam de material são: Papel, Vidro, Plastico, Metal, Oleo.

Nelas, não há nenhuma funcionalidade extra. Entretanto, elas possuem um modo de armazenamento diferente, logo, com o uso de virtual e override, aplicando os conceitos de polimorfismo, cada uma imprime uma instrução própria.

A classe CadastroMaterial possui um mapa de vetores do tipo material. A chave do mapa é o nome da pessoa que está logada. Esse modo de armazenamento possibilita que os materiais que o usuário tem acesso são somente os dele.

Ademais, essa classe possui as seguintes funções:

* void cadastrar\_material (Pessoa\* usuario\_logado)

Responsável por pedir as informações do material a ser cadastrado para o usuário e adicionar o novo material ao vetor. Além de imprimir o modo de armazenamento daquele material para instruir o usuário.

* void imprimir\_materiais\_cadastrados(Pessoa\* usuario\_logado).

Imprime os materiais que a pessoa logada no momento cadastrou.

* void excluirmaterial(int posicao, Pessoa\* usuario\_logado)

Exclui determinado material escolhido pelo usuário.

* std::vector<Material\*> retorna\_materiais(Pessoa \*usuario\_logado)

Retorna o vetor de materiais do usuário logado para eventuais funcionalidades no agendamento de coleta.

Usuário:

Assim como as outras partes do software que foram descritas acima, o fragmento usuário tem a classe Usuario, que possui os atributos do usuário, e a classe CadastroUsuario.

Esta possui as funções:

* Pessoa\* login()

Responsável por efetuar o login e retornar a pessoa logada para ser utilizada em outras partes do programa.

* void cadastro();

Responsável por efetuar o cadastro do usuário e salvar as informações em um map com a chave sendo o nome do usuário e o conteúdo sendo a pessoa cadastrada.

* bool find(std::string)

Utilizada na função login para verificar se o nome dado para acessar o sistema está cadastrado no map de pessoas.

* void imprimepessoas()

Imprime o nome de todos os usuários cadastrados. Utilizada no login para mostar quais nomes estão passiveis de acessar o sistema.

* void sair()

Permite o retorno para o menu inicial para ser possível acessar outra conta.

Testes:

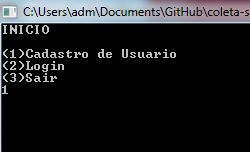
Com as funcionalidades apresentadas, para demonstrar o funcionamento do sistema no geral, na fase “teste” ocorrerá uma simulação do cadastro de 2 usuários arbitrários (1 doador e outro receptor).

O doador cadastrará 2 materiais, o receptor vai pegar um deles e outro será excluído.

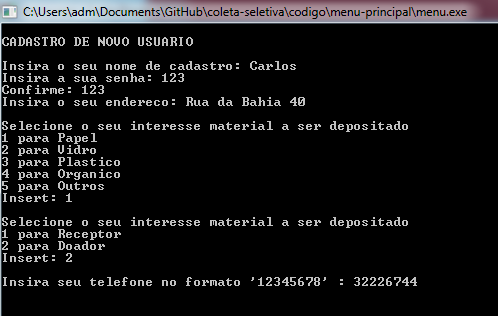
Ademais, será necessário cadastrar um ponto de coleta e um agendamento para a simulação total.

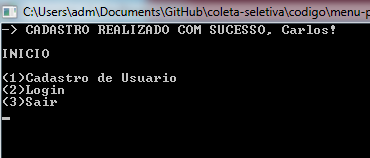
Desse modo, será possível caminhar pelos vários fragmentos do código.

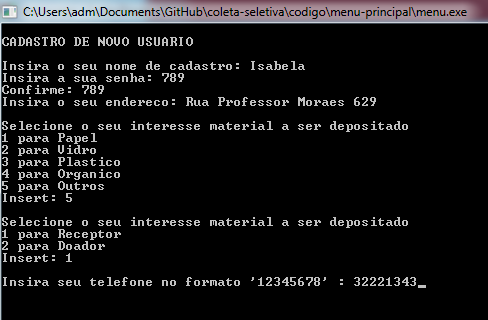
1. Cadastro dos Usuários



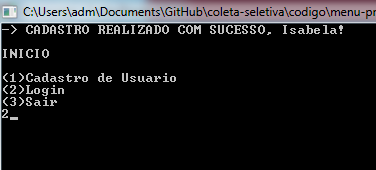
Cadastro do usuário doador:



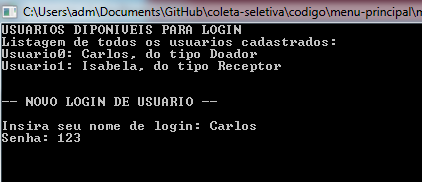
Confirmação do cadastro do doador e cadastro do receptor: 



Confirmação do cadastro do receptor e entrada login do doador:

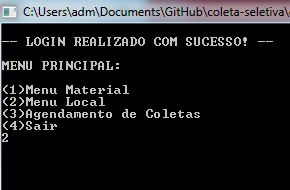


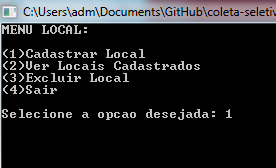
1. Login

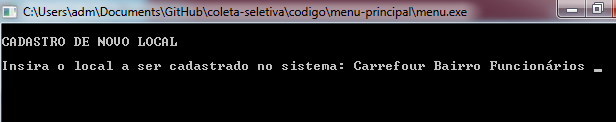


A partir desse momento, todas as funcionalidades serão executadas dentro da conta do usuário “Carlos”, ou seja, o doador.

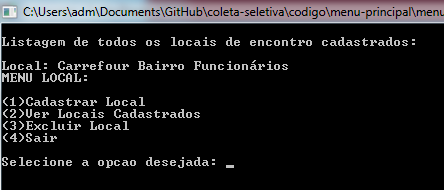
1. Cadastro de Local:





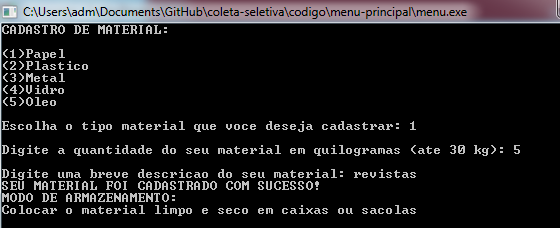


Imprimindo locais disponíveis:

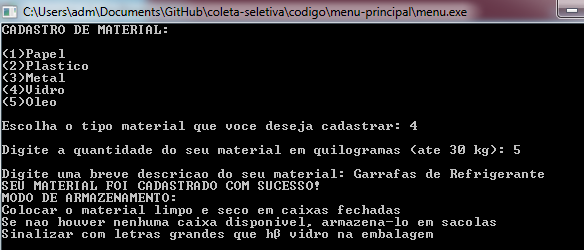


1. Cadastro de Materiais:

Material 1:

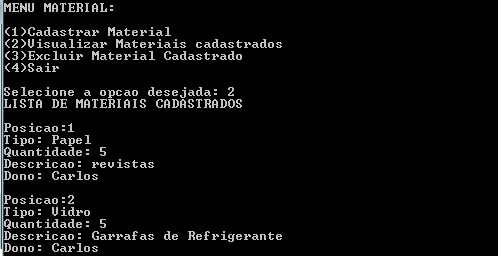


Material 2:

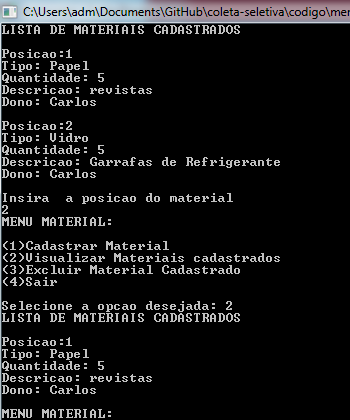


Observa-se que o modo de armazenamento é diferenciado para o papel e para o vidro

Visualizar materiais cadastrados:



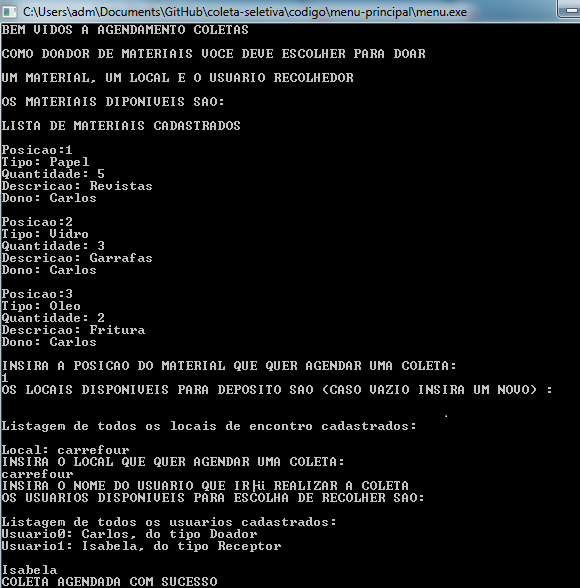
Excluir material e visualizar os cadastrados após a exclusão:



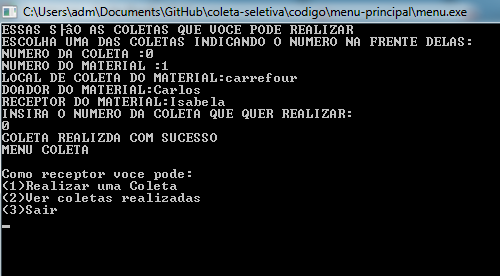
Depois desse processo, 3 materiais foram cadastrados no sistema pelo usuário doador Carlos.

Ademais, o usuário também inseriu um novo ponto de coleta chamado Carrefour.

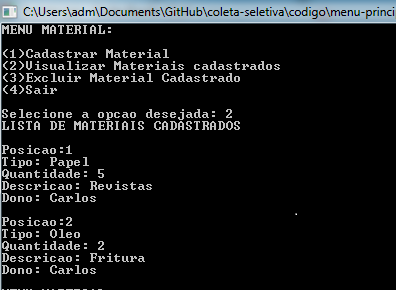
Depois disso, houve o agendamento da coleta:



Depois disso, houve a troca de contas. Assim, a receptora Isabela poderia confirmar a coleta:



Para confirmar o funcionamento correto do sistema, nesse momento seria necessário que o material escolhido para ser coletado não estivesse mais no menu de materiais do usuário Carlos:



Conclusão:

No desenvolvimento do projeto, com o uso de Programação Orientada ao Objeto foi possível ver como as qualidades desse tipo de abordagem, mostradas durante o curso, funcionam na prática.

Como os integrantes trabalharam com partes diferentes do sistema para, aos poucos, fazerem a integralização, o fato da facilidade na manutenção do código foi extremamente importante. Fazer com que todas as partes se encaixassem com pequenas mudanças não ocorreria com a praticidade que ocorreu caso a programação estruturada fosse utilizada.

Ademais, é importante salientar como o uso do Github foi de extrema importância para o bom manuseio do software.

O ponto mais delicado e que gerou mais problemas no trabalho foi utilizar o Windows. Como o VPL do Moodle é baseada no Linux, alguns detalhes de sintaxe, ao se trabalhar com .h ou .cpp mudaram substancialmente. Essas mudanças fizeram o desenvolvimento do software ficar parado por um tempo considerável até que o erro fosse descoberto.

Além disso, a instanciação das classes cadastro para o uso das tads presente nelas foi um detalhe que gerou uma incerteza. Apesar disso, depois do entendimento de como esse processo funcionava, não houve mais nenhum empecilho significativo.

Por fim, é importante salientar que a elaboração desse trabalho prático foi de extrema importância para consolidação dos conhecimentos não só em POO e não só em C++, mas em todos os âmbitos que envolvem programação. E apesar de todos os desafios que o processo de desenvolvimento acarretou, sua conclusão e seu funcionamento conforme os parâmetros inicialmente estipulados foi compensador.