**PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE ll**

**TRABALHO PRATICO FINAL**

**Grupo (Nome/Matricula/Github):**

-Thiago Leivas /2017098226/thleivaas

-Gustavo Silvestre/2017108787/gustavoSilvestreAC

-Luís Otávio/ 2017089049/luis6alves

-Victor Hugo/ 2018430380/Mashslek

**INTRODUÇÃO**

O objetivo inicial era implementar um sistema que possibilitasse a interação entre doadores e receptor de resíduos. Para tal, o sistema deve ser capaz de realizar o cadastro de dois tipos de usuários (doadores e receptores), englobando seus principais dados (nome, CPF/CNPJ, e-mail, endereço, etc). Após o usuário informar seus dados, ele deverá escolher a partir de uma lista pré-definida o tipo de resíduo que é do seu interesse.

Em seguida, deverá ser cadastrado o ponto de coleta. Para isso, os usuários poderão escolher entre usar o seu endereço, ou o endereço do seu doador/receptor. A realização da coleta ocorrerá no endereço definido, em uma data específica.

**IMPLEMENTAÇÃO**

**1 – Cadastro de usuários:**

Para a realização do cadastro de usuários, foi implementada a classe base “usuário”, que possui os atributos nome, cpf/cnpj, endereço , nome\_residuo, peso\_residuo e dois atributos bool para informar se o usuário é doador ou receptor, pessoa física ou empresa, e métodos para acessar/modificar tais atributos.

A partir da classe usuário, foi implementada a classe derivada “gerenciamento”, usando o conceito de herança. No “.cpp” da classe derivada, foram implementados os métodos da classe base. Para manipular os dados cadastrados, foram usadas funções da biblioteca <list> que permitiram adicionar, remover, imprimir e modificar os usuários cadastrados.

**2 – Cadastro de resíduos:**

Para o cadastro de resíduos, foi implementado o método tipoResiduo da classe gerenciamento, que solicita que o usuário informe, durante o cadastro/modificação, o tipo de resíduo de interesse. Também foi implementada uma aba no menu principal para informar a forma correta de armazenamento dos tipos de resíduos.

**3 – Cadastro do ponto de coleta:**

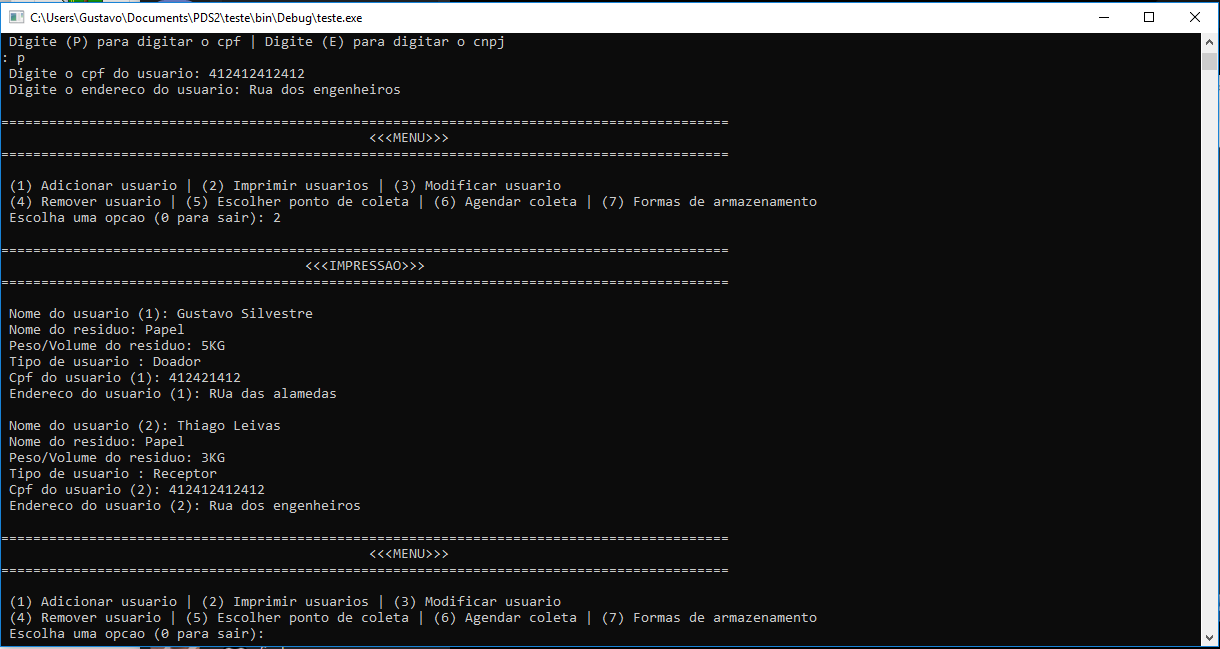
Para o cadastro do ponto de coleta, foi implementada a classe “pontoDeColeta”, derivada da classe gerenciamento. No menu principal, essa classe fornece duas alternativas para o usuário: selecionar um dos endereços cadastrados (doador ou receptor) ou digitar um novo endereço, no qual a coleta deverá ocorrer.

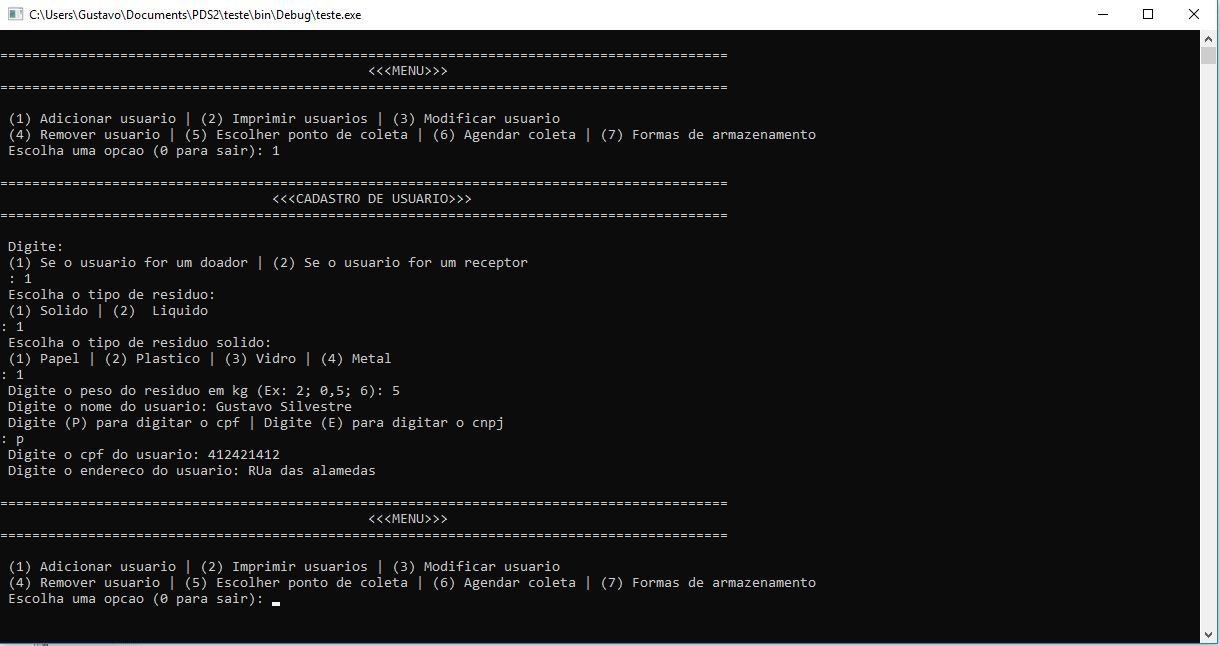
**4 – Agendamento/Realização da coleta:**

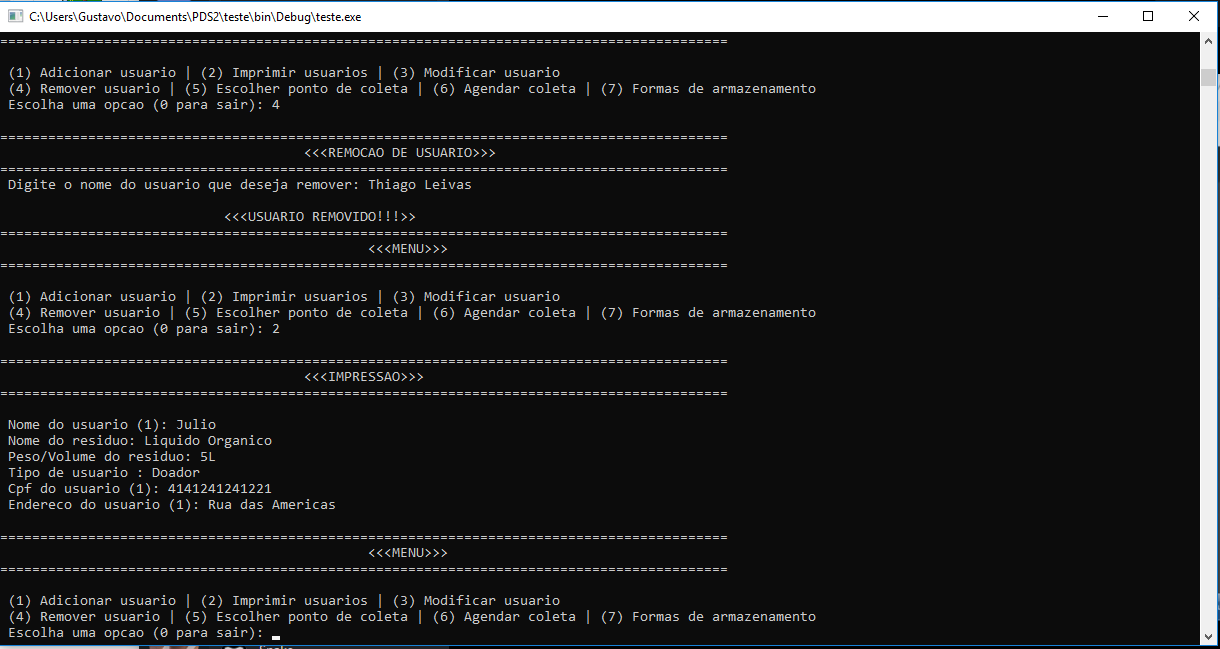
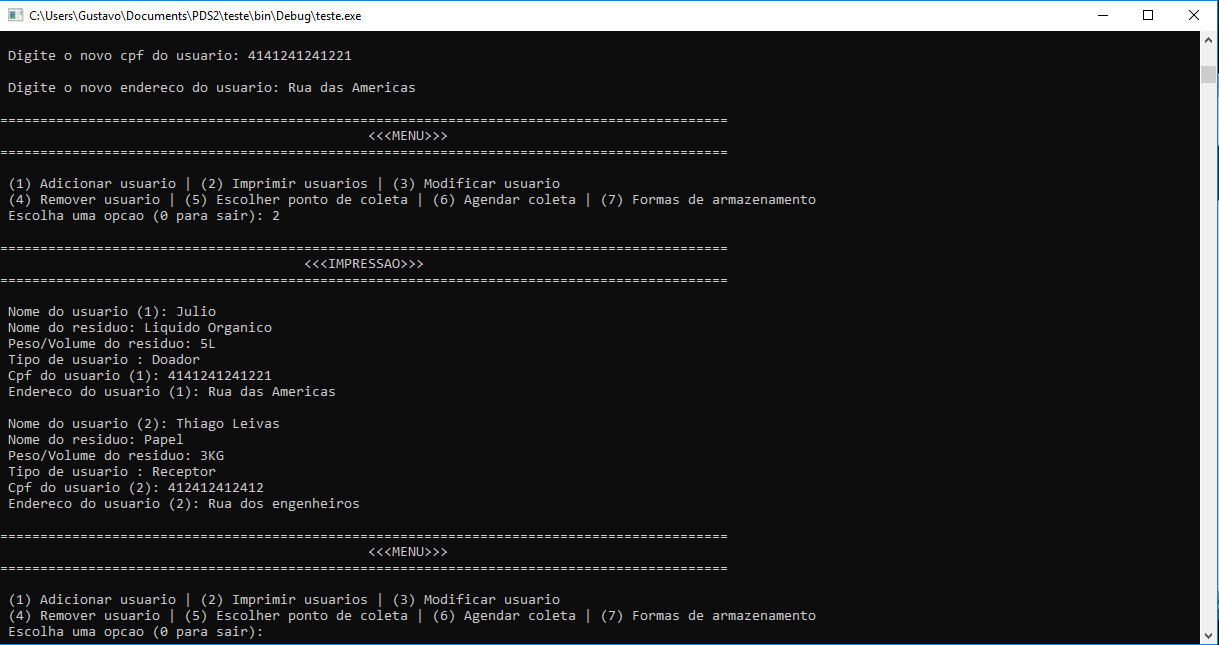
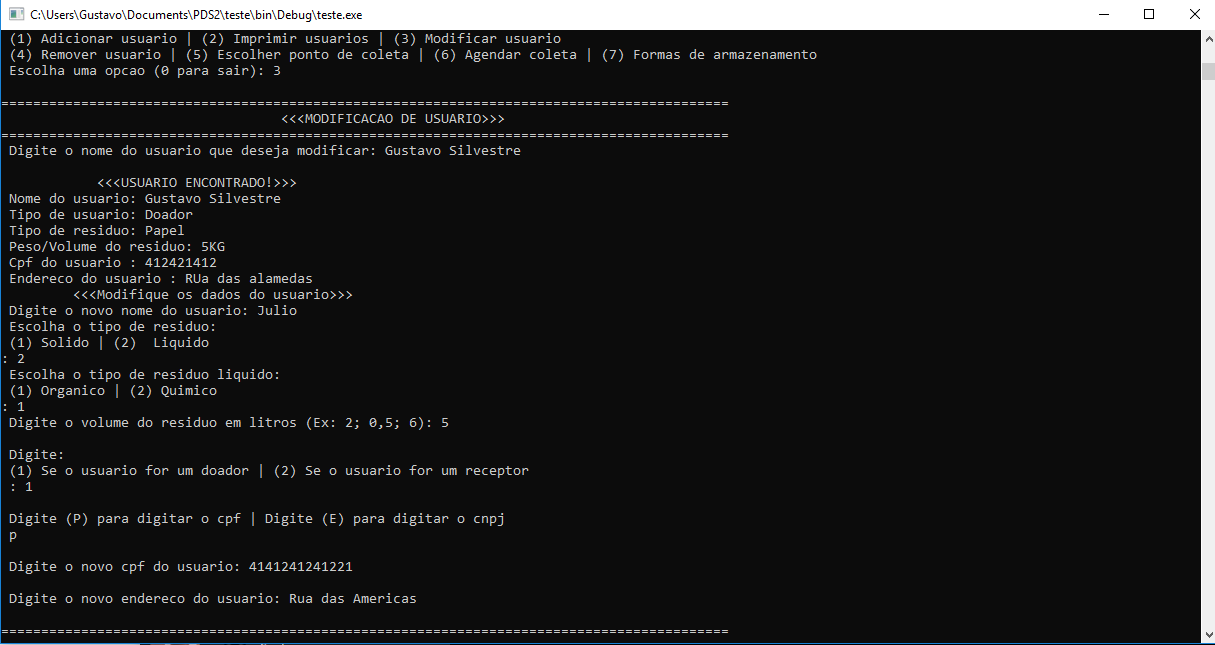
Para o agendamento da coleta, foi implementada a classe “agendamento”, derivada da classe pontoDeColeta. No menu, essa classe fornece três opões: ver o agendamento, criar o agendamento ou confirmar a realização da entrega.

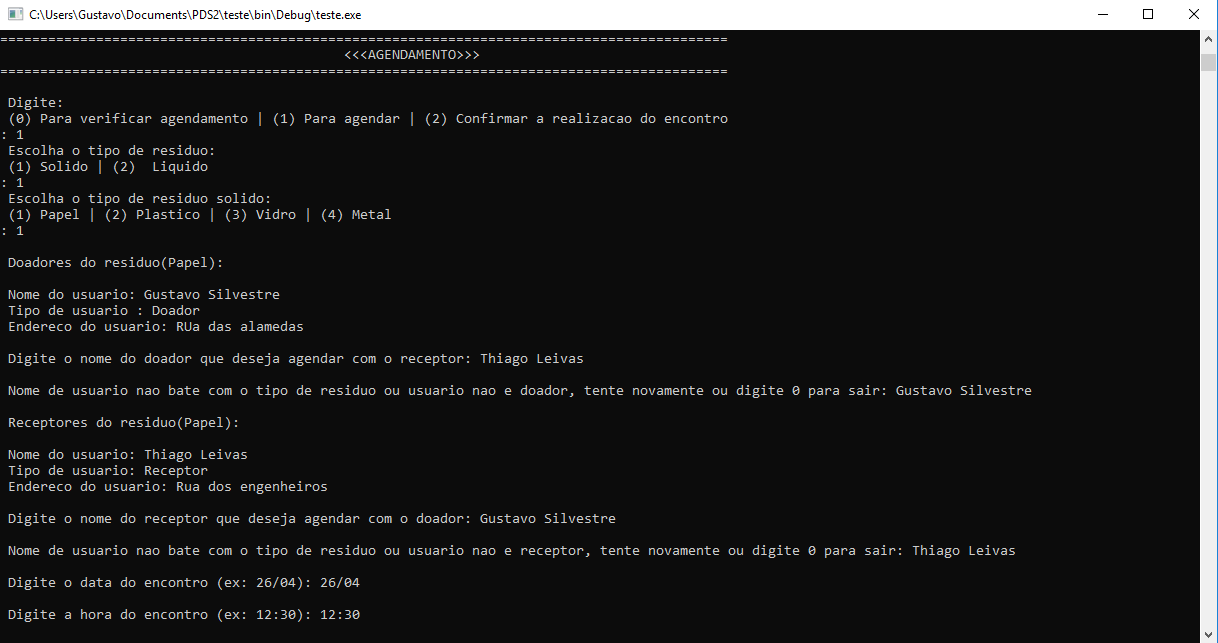
**TESTES**

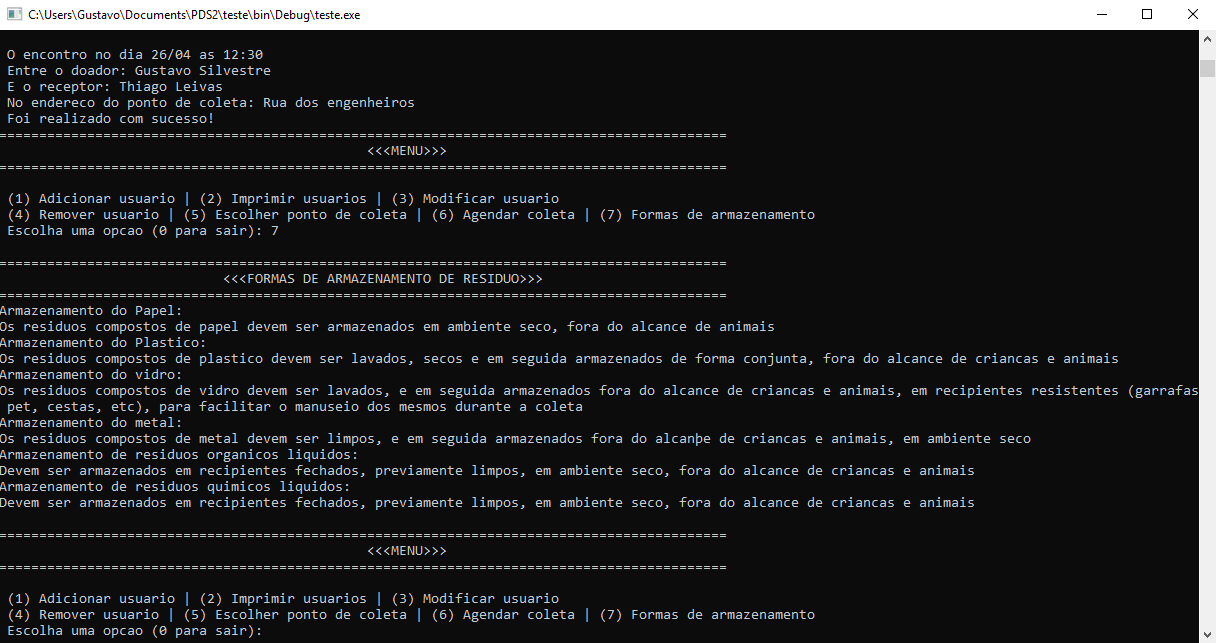
Nos prints abaixo, foram testadas todas as funcionalidades do programa:

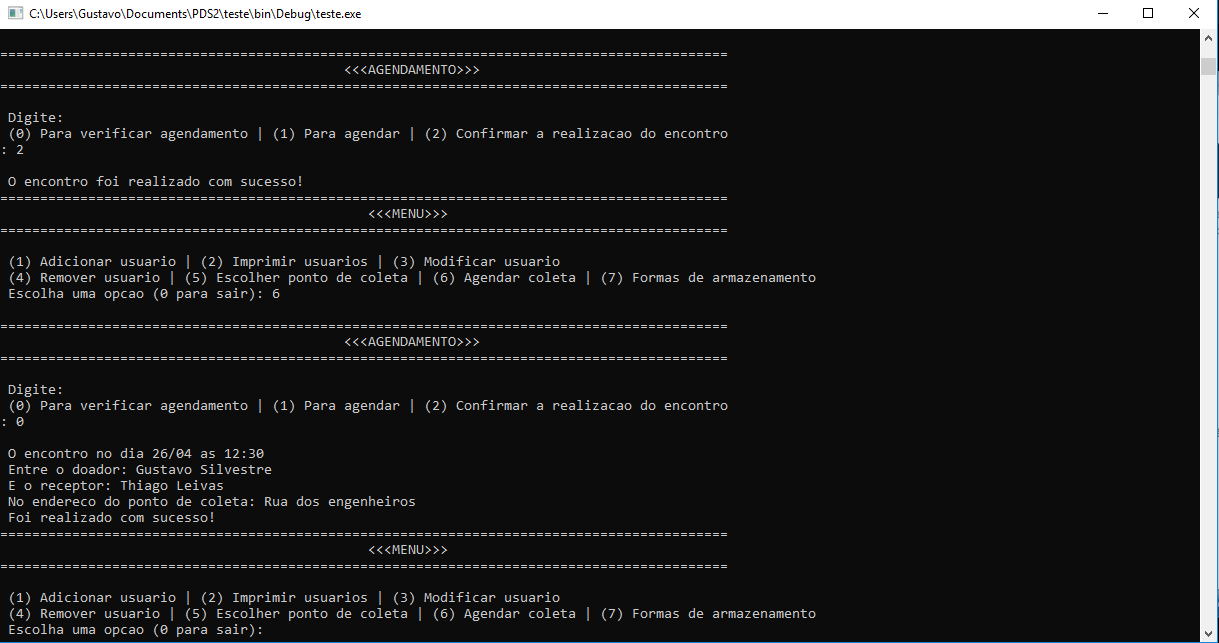
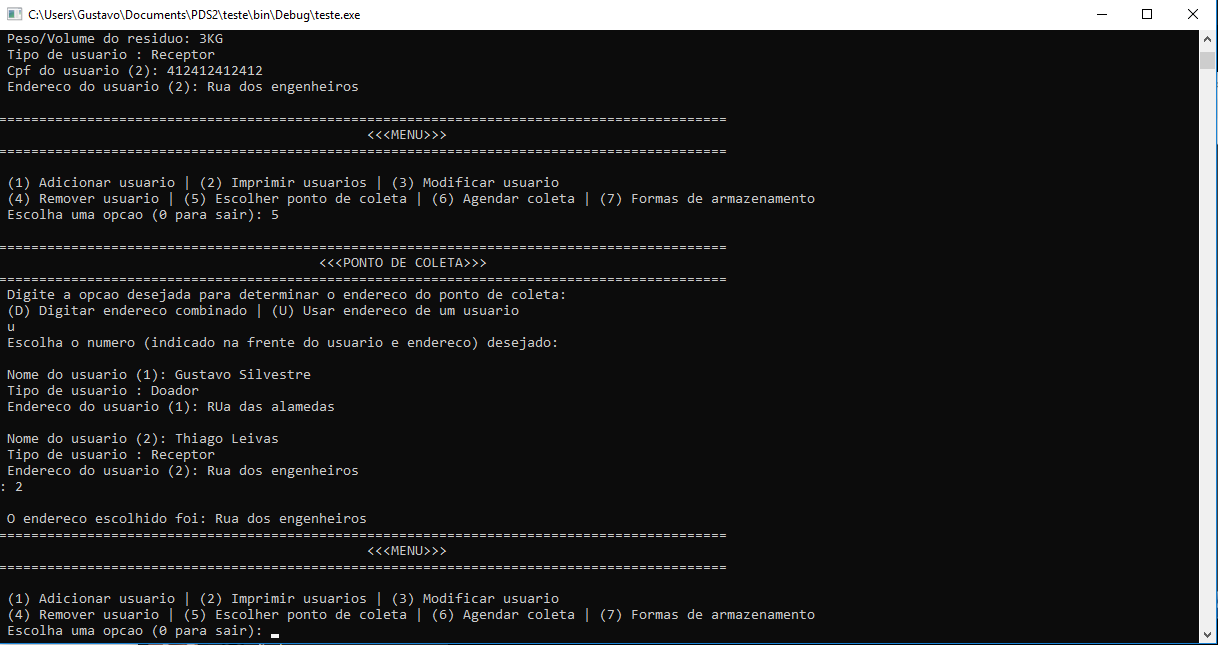
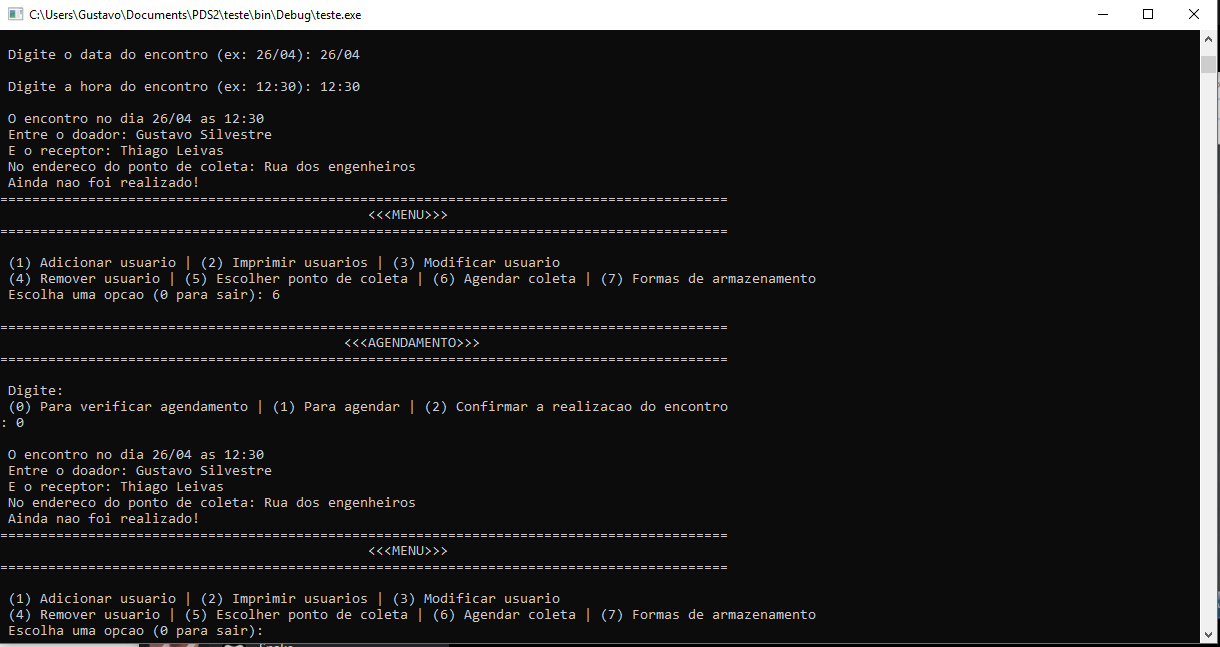
Cadastro de usuários

Impressão de usuários

Modificação/Remoção de usuários

Cadastro do ponto de coleta

Formas de Armazenamento dos tipos de resíduos

Agendamento/Realização da coleta

**CONCLUSÃO**

Entre as principais dificuldades encontradas ao longo do desenvolvimento do programa, se destacaram as seguintes:

.Dificuldade de interligar os módulos do programa;

.Dificuldade no uso e adequação das funções da biblioteca <list>;

.Dificuldade para adaptar o programa à possíveis comportamentos inadequados de usuários.

Apesar das dificuldades, conseguimos por em prática os conceitos de programação orientada ao objeto vistos ao longo do curso, que possibilitaram o uso da criatividade para a resolução de um determinado problema, que no caso era a implementação do programa.

**BIBLIOGRAFIA**

.Slides vistos em sala de aula, disponibilizados no moodle;

. <https://pt.wikipedia.org/wiki/CRUD>;

.http://www.projetoreciclar.ufv.br/;

.https://www.trabalhosgratuitos.com/Humanas/Direito/Formas-de-Armazenamento-e-Tratamento-do-Lixo-848835.html;