Bruno Henrique de Carvalho Scaglione - 10335812

PMR 3402
Sistemas Embarcados
Vending Machine

Brasil

3 de fevereiro de 2021

Lista de ilustrações

Figura 1 – Casos de Uso 6
Figura 2 – Diagrama de Estados 6
Figura 3 – Digrama de Componentes - Parte 1 7
Figura 4 – Digrama de Componentes - Parte 2 8
Figura 5 – Digrama de Componentes - Parte 3 8
Figura 6 – Diagrama de Sequência - Inserir Dinheiro 9
Figura 7 – Diagrama de Sequência - Efetuar Compra (Parte 1) 9
Figura 8 – Diagrama de Sequência - Efetuar Compra (Parte 2) 10
Figura 9 – Diagrama de Sequência - Efetuar Compra (Parte 3) 10
Figura 10 – Diagrama de Sequência - Cancelar Compra 11
Figura 11 – Diagrama de Sequência - Cadastrar Troco 11
Figura 12 – Diagrama de Sequência - Cadastrar Produto 12
Figura 13 – Cliente insere dinheiro 42
Figura 14 – Manutenção erra senha 43
Figura 15 – Manutenção acerta senha 43

Sumário

1 VISÃO GERAL 3

2 DIGRAMAS UML 6

3 DEFINICÕES SISTEMA 13

4 LOOP PRINCIPAL 15

5 OBTER EVENTOS 17

6 MATRIZ DA MÁQUINA DE ESTADOS 19

7 EXECUTAR AÇÃO 21

8 COMPONENTES 26

9 PROGRAMA RODANDO 42

10 CONCLUSÃO 44

1 Visão geral

Uma visão geral sobre: como foi feito o programa, como funciona a Vending Machine programada, tanto para o Cliente como para a Manutençao, e como que foi pensada a modelagem de uma Vending Machine real traduzida para o contexto do software apenas.

A parte estrutural do programa foi copiada do exemplo do Alarme provido pelo Professor, com a excessao de pequenas mudanças, que não afetam o funcionamento do programa como um algoritmo de Máquina de Estados, pois este é o objetivo do trabalho.

Após ter rascunhado todos os digramas UML pedidos pelo professor ao longo do semestre, defini os estados, eventos ações do sistema, fiz o código começando na função de obter os eventos, em seguida a matriz da máquina de estados, depois as ações que teriam que ser executadas e por último os componentes. Esta é a mesma sequência que estão os tópicos neste relatório. Este processo, no entanto, foi um tanto quanto iterativo, muitas vezes tive que modificar os diagramas UML e consequentemente o modo como o programa deveria ser estruturado.

Uma observação importante: eu nomeei as ações de acordo com o evento que provoca elas, para facilitar o entendimento do programa. Todas as ações tem o nome do tipo: "APOS_NOMEDOEVENTOAQUI".

Questões de Modelagem da Vending Machine real: eu considerei que a manutenção tem que digitar uma senha no teclado de autenticação, que ficaria atrá da máquina, e se a senha for correta libera a porta de manutenção que da acesso à toda parte interna da máquina, possibilitando à manutenção repor os produtos e o troco. A reposição tanto dos produtos quanto do troco é manual, ou seja, a pessoa da manutenção olha quantos produtos e moedas tem, para saber a quantidade que vai colocar. Após colocar manualmente esses itens, a manutenção tem que cadastrar nos banco de dados de produto e no banco de dados do caixa de troco quais e quanto itens foram colocados. O dinheiro que é obtido pela máquina pelas compras, cai sempre no moedeiro, onde fica armazenado. A retirada desse dinheiro é feita manualmente, e não há nenhum banco de dados controlando essa quantia. Outro aspecto, o momento em que o produto é derrubado, considerei que a maquina inicia um timer(após inciar a derrubada do produto), e quando esse timer estoura, a máquina assume que o produto já foi entregue e pode voltar ao estado inicial IDLE. Utilizei também buffers em alguns componentes, que denotam uma memória interna de pouco espaço de armazenamento e de curto prazo. Esse buffers são zerados entre ciclos de operação.

Em linhas gerais, o funcionamento da Vending Machine ocorre da seguinte forma:

Antes de entrar no loop principal do programa, é printado em tela uma tabela com os produtos, número desse produtos e preço desses produtos; que representa o painel real da Vending Machine. Após entrar no loop principal, como inicialmente não temos nenhum evento interno, a funcao obterEvento(estado) é chamada. No programa original do Alarme, esta funcao nao tinha argumentos, porém do modo que pensei em traduzir o software da Vending machine, não seria necessário escutar todos os eventos possíveis, apenas os que interessam ao estado atual. Inclusive no meu programa tem um teclado interno(dentro da maquina) que só se pode ter acesso na manutenção, então nao pode estar aberto para o usuário criar um evento(usar o teclado) em outros estados.

Agora, o programa escuta eventos que podem acontecer no estado IDLE, que são: o usuário inserir uma moeda ou a manutenção digitar a senha no teclado de autenticação.

A preferencia é dada primeiramente a a detectar moedas, chama-se uma função dos sensores que printa instruções para o usuario inserir uma moeda (representadas pelo usuario digitar teclas) e escuta-se o input do usuário. Se o usuário digitou uma moeda, essa moeda é inserida no buffer interno do sensor, e esse processo pode ir se repetindo num loop. Após ter inserido a primeira moeda, o programa já entra no estado de "recebendo moedas", no qual o usuário pode cancelar a compra, submeter o dinheiro ou digitar o número do produto através do teclado do cliente. É importante ressaltar que embora o usuário tenha acesso a ecolher um produto pelo número, isso não afetará o programa se ele ainda não submeteu o dinheiro. Se o usuário cancelar, ele recupera o dinheiro. Se o usuário submeter o dinheiro, ele recebe o mesmo teclado mais uma vez, e pode teclar o número do produto que deseja. Internamente o programa guarda o dinheiro inserido no banco de dados do caixa de troco

Agora, se o usuário for da manutenção, ele recusa esse primeiro teclado do cliente digitando qualquer tecla que não represente moedas. Após isso, passamos a escutar o evento de senha teclada no teclado de autenticação. Se a manutenção teclar qualquer senha que não seja nula (apenas enter) dispara-se o evento de senha de manutenção submetida.

Voltando ao caso do Cliente (que já digitou o número do produto que deseja). Agora é feita uma verificação em 3 passos. Primeiro se o produto está disponivel, depois se o dinheiro inserido é suficiente e por último se o troco disponível no caixa de troco é suficiente (neste passo, a quantidade de cada moeda que precisa ser devolvida no troco é calculada e armazenada no "buffer de troco a ser devolvido"do caixa de troco. Se algum desses passos falhar, é disparado um evento interno de compra inválida, mostrando

em tela para o usuário que a compra foi inválida. Neste momento o usúario volta ao mesmo estágio de antes, onde poderá escolher outro produto ou cancelar a compra. Se nenhuma das verificações falhar, o troco é separado no caixa de troco, e é disparado o evento interno de compra válida. Neste momento o programa vai deletar, do banco de dados de produtos, o pruduto comprado; retirar o troco do banco de dados do caixa de troco; promover a caída de dinheiro no moedeiro; devolver o troco separado para o cliente e empurrar o produto escolhido pelo cliente. Neste momento, antes das ações mecânicas começarem (atuadores) o timer é iniciado. Após o timer concluir, a máquina entende que pode voltar ao estado inicial IDLE.

Voltando ao caso da manutenção, após a senha ter sido submetida, o programa vai verificar, utilizando o sitema de autenticação, se a senha é correta. Se for correta, gera-se um evento interno de senha válida, se não, gera um evento interno de senha inválida. Se a senha for inválida, nada acontece e o programa volta ao estado inicial. Se a senha for válida, o programa entra no estado de manutenção, onde o usuário (manutenção) pode utilizar o teclado interno para fazer cadastro de produtos e/ou troco. O teclado interno é um evento que passa a ser escutado, onde a funcão do teclado interno printa em tela as instrucoes para a manutenção fazer o cadastro de moedas ou produtos. A manutencao digita a tecla especifica da moeda ou produto, em seguida digita a quantidade; e esses inputs são guardados em buffers do teclado interno. Se os inputs forem corretos(uma tecla que corresponde a uma moeda ou produto, e a quantidade é menor que o máximo de armazenamento de moedas e produtos na máquina), é obtido o evento de submit de troco ou produto. Então, as ações que ocorrem em seguida são de armazenamento do troco ou produto em seus respectivos bancos de dados.

Para encerrar a manutenção, o programa estará escutando o fechamento da porta de manutenção, toda vez que estiver no estado "ocorrendo manutencao". No qual, a funcao que detecta o fechamento da porta de manutencao printa em tela as instruções para que o usuário(manutenção) possa fechar a porta de manutencão. Se o usuário quiser continuar no estado de manutenção, para fazer outros cadastros, ele pode ignorar o procedimento de fechar a porta, digitando outra coisa. Se foi detectado que a porta foi fechada, a porta em seguida é travada e o programa volta ao estado inicial IDLE

2 Digramas UML

Figura 1 – Casos de Uso

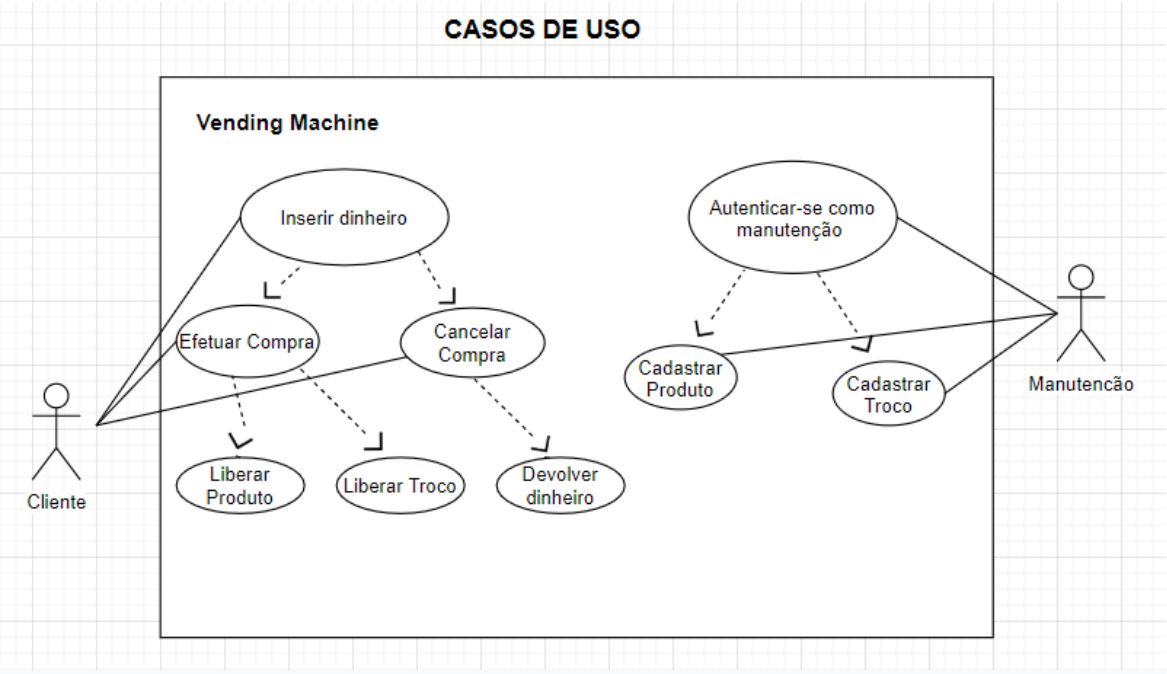


Figura 2 – Diagrama de Estados

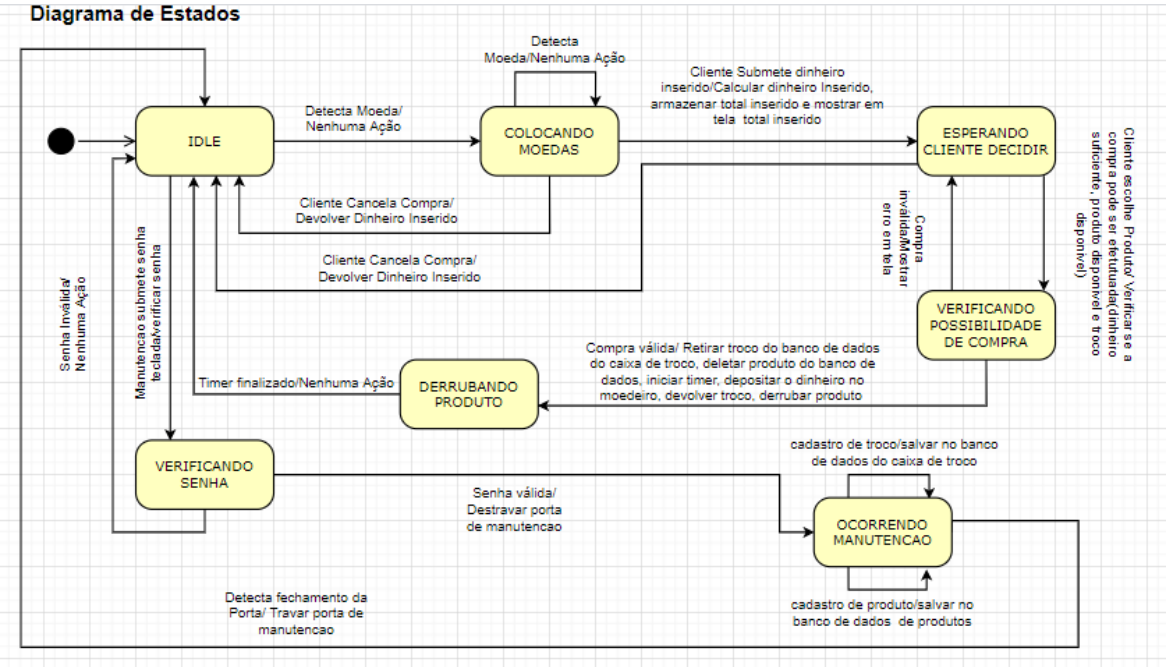


Figura 3 – Digrama de Componentes - Parte 1

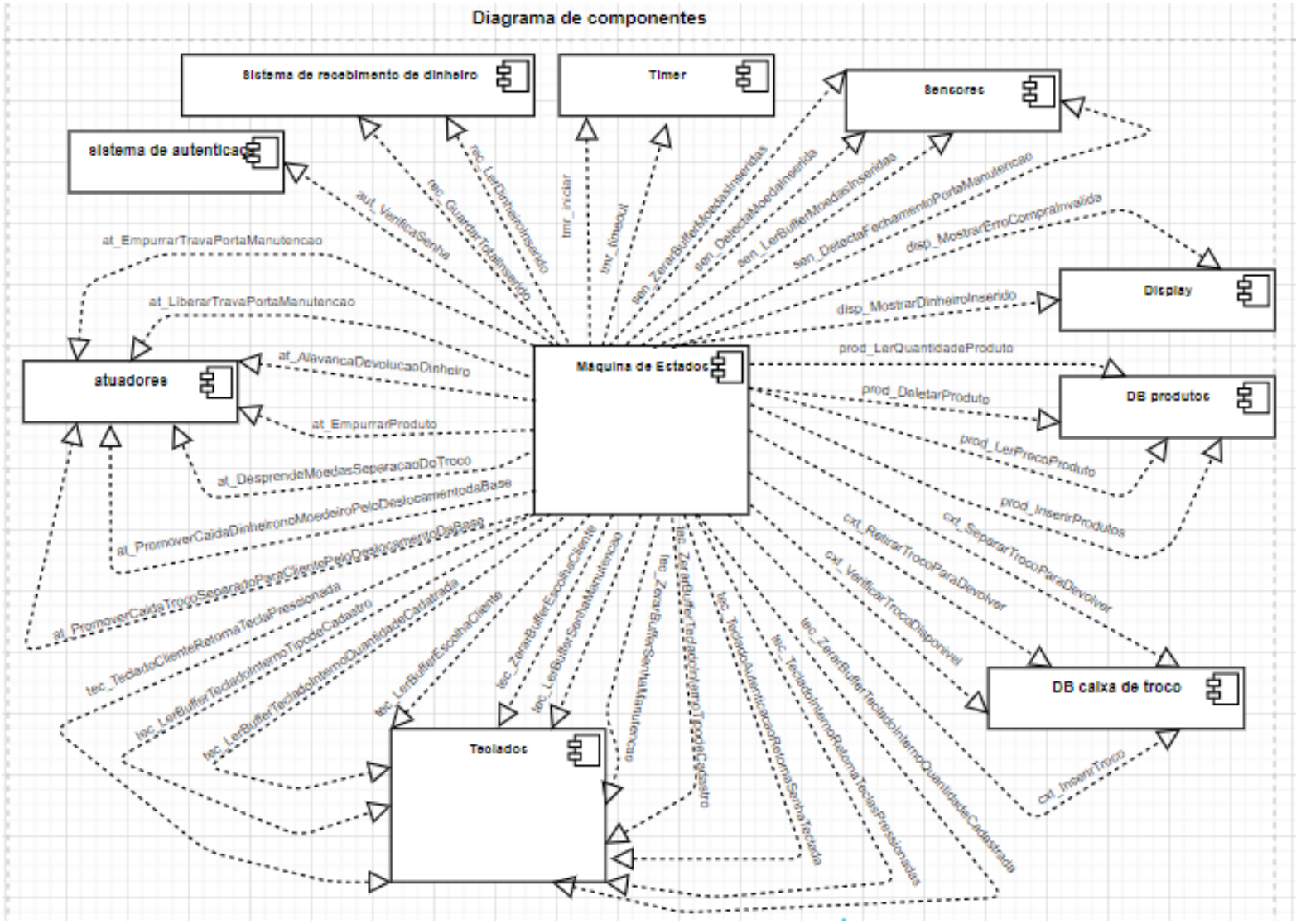


Figura 4 – Digrama de Componentes - Parte 2

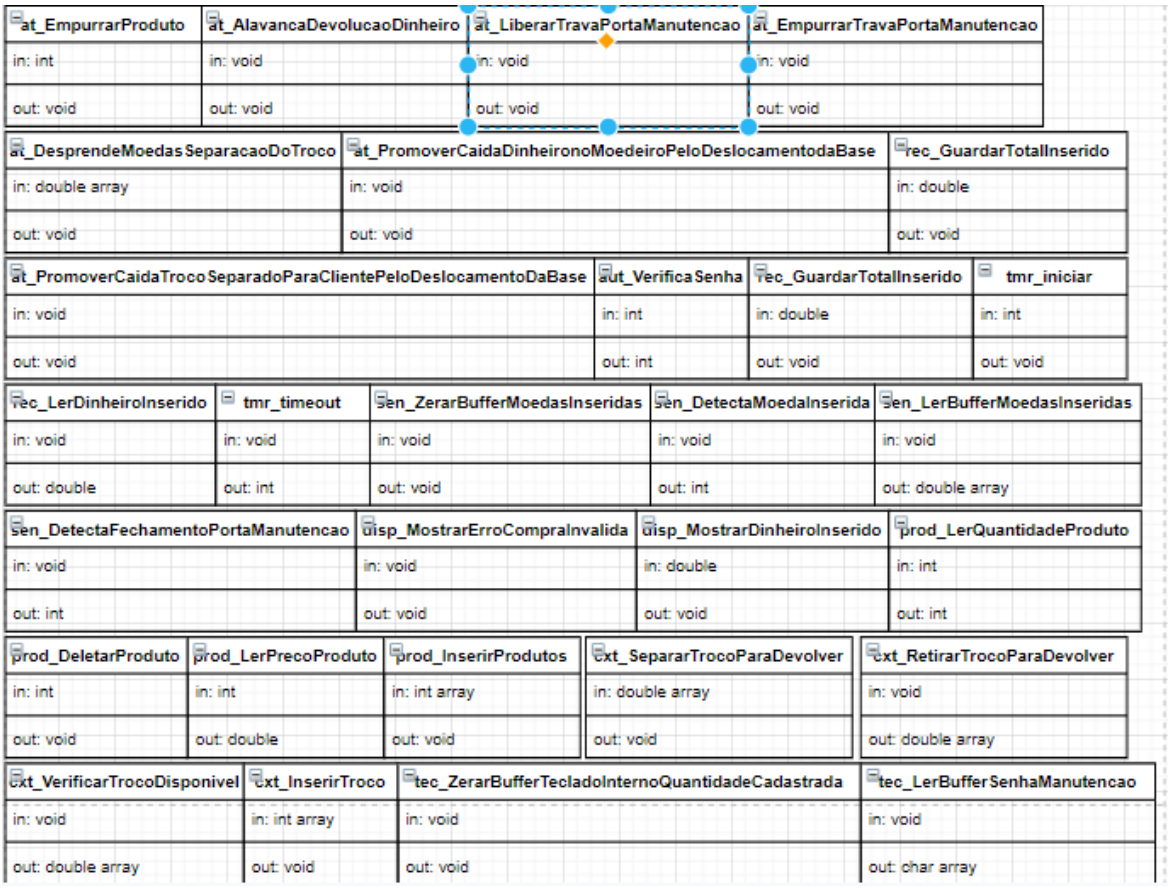


Figura 5 – Digrama de Componentes - Parte 3

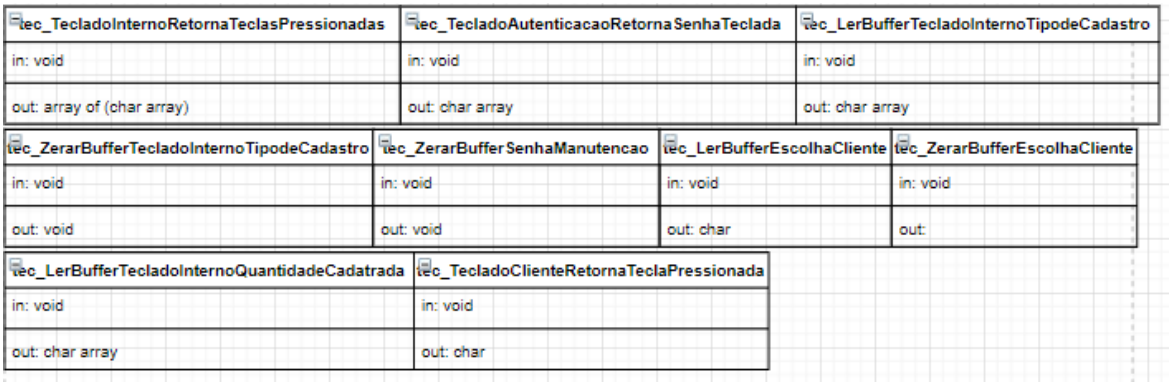


Figura 6 – Diagrama de Sequência - Inserir Dinheiro

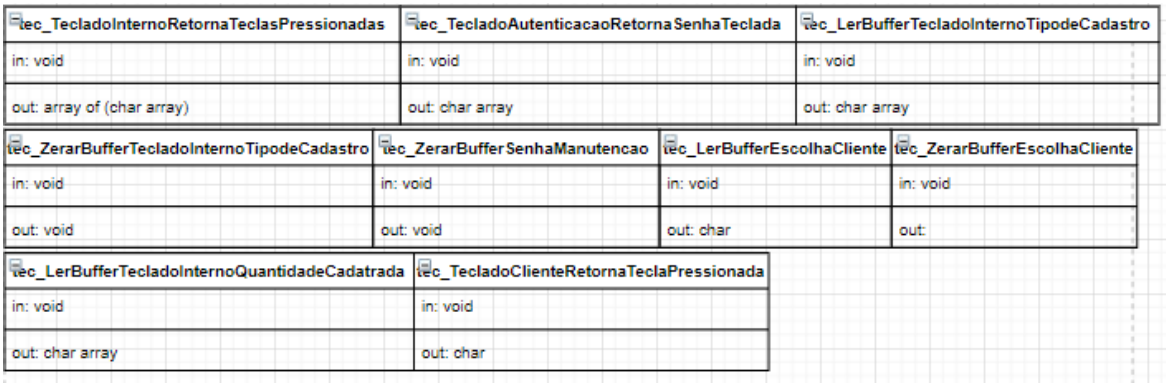


Figura 7 – Diagrama de Sequência - Efetuar Compra (Parte 1)

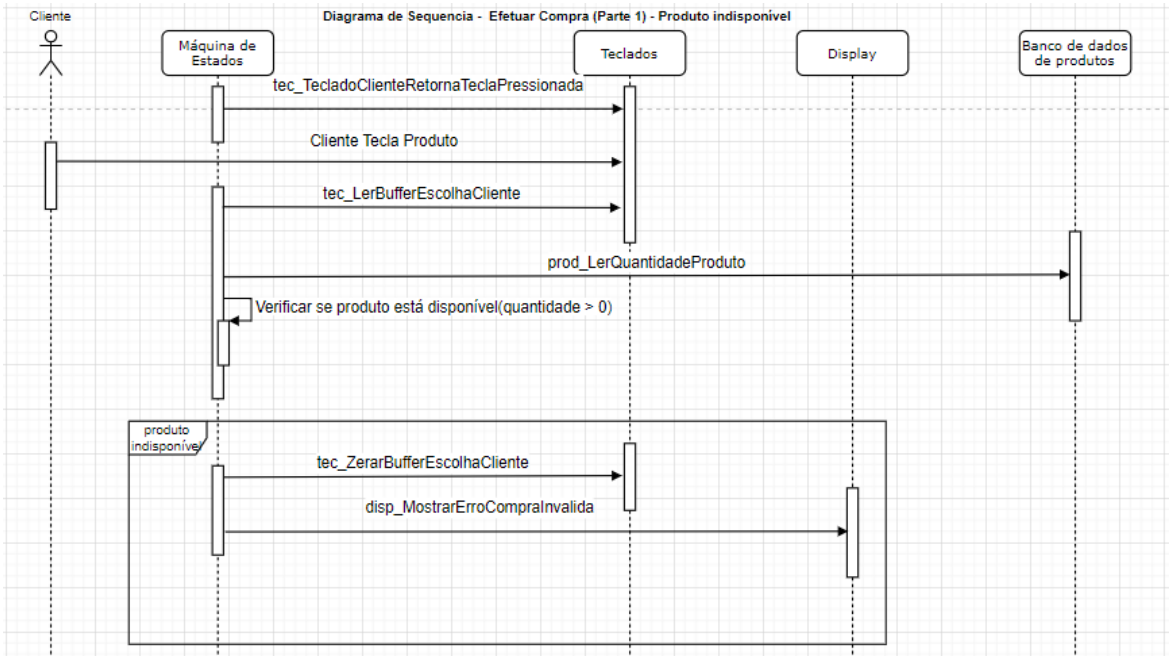


Figura 8 – Diagrama de Sequência - Efetuar Compra (Parte 2)

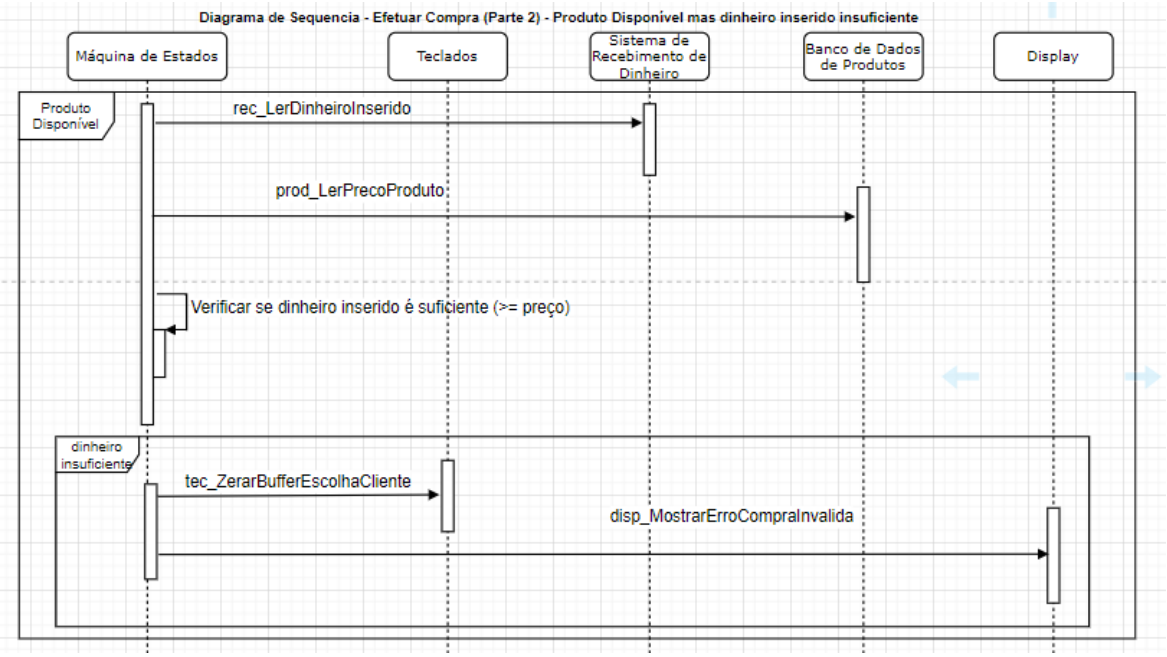


Figura 9 – Diagrama de Sequência - Efetuar Compra (Parte 3)

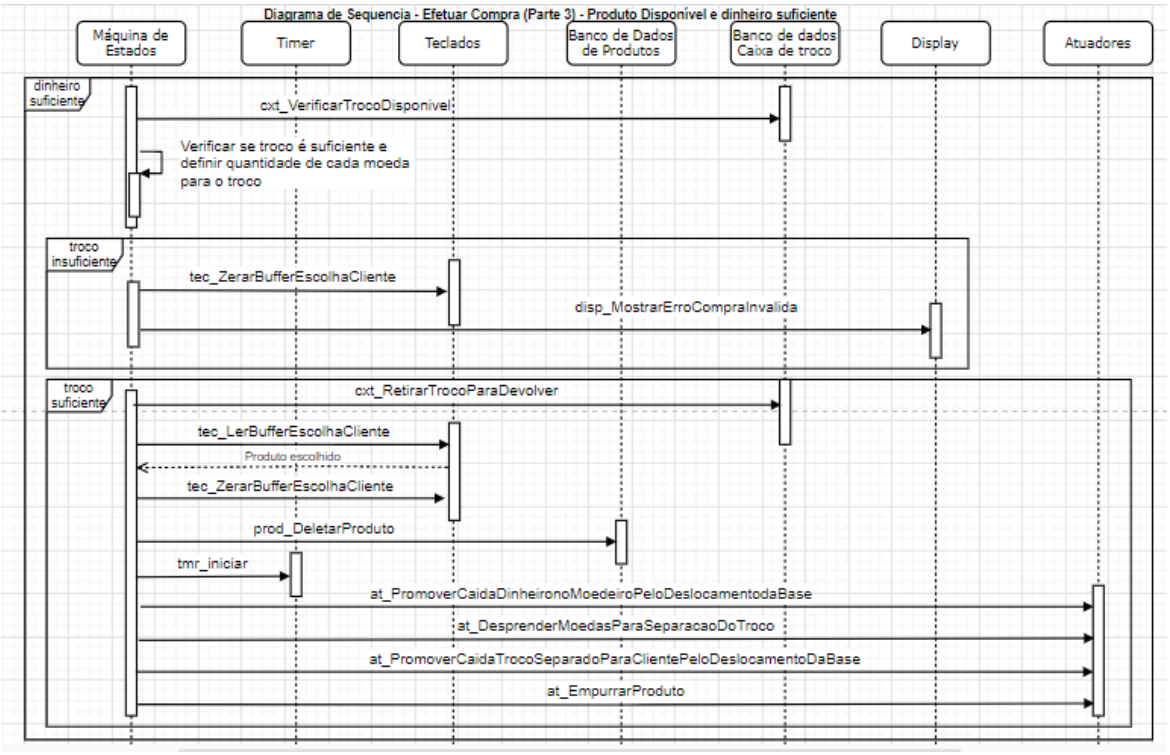


Figura 10 – Diagrama de Sequência - Cancelar Compra

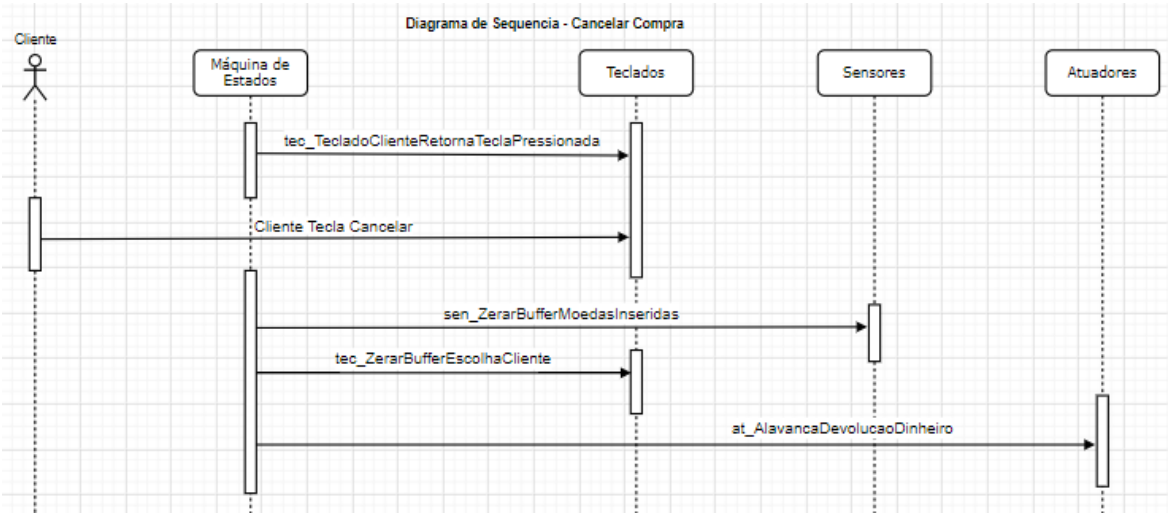


Figura 11 – Diagrama de Sequência - Cadastrar Troco

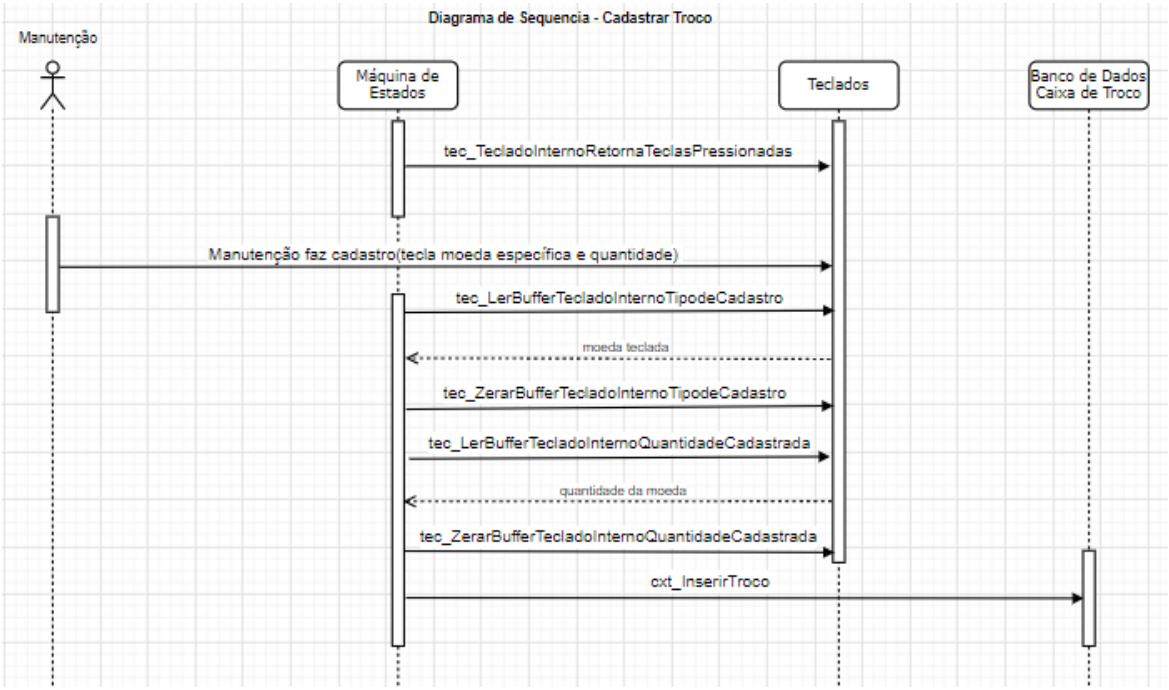
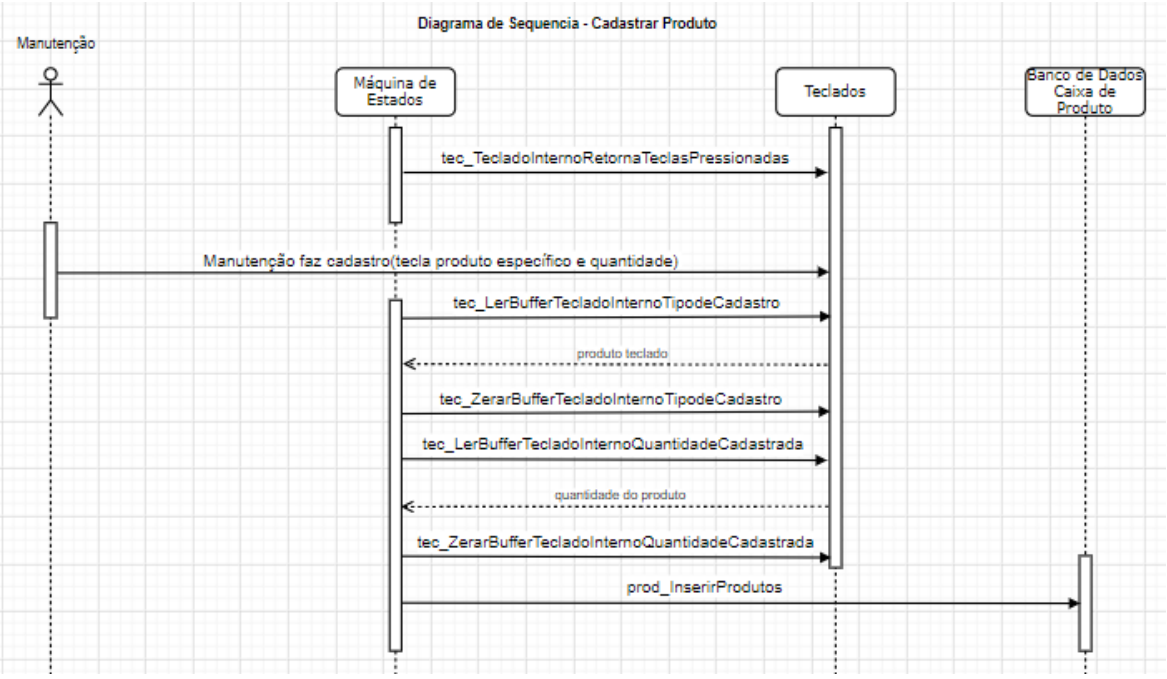


Figura 12 – Diagrama de Sequência - Cadastrar Produto



3 Definicões Sistema

Adicionei também variáveis que representam o preço dos produtos

```
#ifndef DEFINICOES_SISTEMA_H_INCLUDED
#define DEFINICOES_SISTEMA_H_INCLUDED

#define true  1
#define false 0

#define PRODUTO_1 1
#define PRODUTO_2 2
#define PRODUTO_3 3
#define PRODUTO_4 4
#define PRODUTO_5 5
#define PRODUTO_6 6
#define PRODUTO_7 7
#define PRODUTO_8 8
#define PRODUTO_9 9

#define PRECO_PRODUTO_1 0.50
#define PRECO_PRODUTO_2 1.50
#define PRECO_PRODUTO_3 0.05
#define PRECO_PRODUTO_4 0.10
#define PRECO_PRODUTO_5 0.30
#define PRECO_PRODUTO_6 2.50
#define PRECO_PRODUTO_7 0.50
#define PRECO_PRODUTO_8 0.80
#define PRECO_PRODUTO_9 2.00


#define NUM_ESTADOS 7
#define NUM_EVENTOS 14

// ESTADOS
#define IDLE 0
#define COLOCANDO_MOEDAS 1
#define ESPERANDO_USUARIO_DECIDIR 2
#define VERIFICANDO_POSSIBILIDADE_COMPRA 3
#define DERRUBANDO_PRODUTO 4
#define VERIFICANDO_SENHA_MANUTENCAO 5
#define OCORRENDO_MANUTENCAO 6
//

// EVENTOS
#define NENHUM_EVENTO -1
#define DETECTOU_MOEDA_ENTRANDO  0
#define USUARIO_SUBMIT_DINHEIRO_INSERIDO 1
#define USUARIO_TECLA_PRODUTO 2
#define COMPRA_VALIDA 3
#define COMPRA_INVALIDA 4
#define USUARIO_CANCELA_COMPRA 5
#define TIMER_FINALIZADO 6
```

```
#define TECLADO_MANUTENCAO_SUBMIT_SENHA_INSERIDA 7
#define SENHA_INVALIDA 8
#define SENHA_VALIDA 9
#define PORTA_DE_MANUTENCAO_FOI_FECHADA 10
#define TECLADO_INTERNO_SUBMIT_CADASTRO_TROCO_EFETUADO 11
#define TECLADO_INTERNO_SUBMIT_CADASTRO_PRODUTO_EFETUADO 12


// ACOES
#define NENHUMA_ACAO -1
#define APOS_USUARIO_SUBMIT_DINHEIRO_INSERIDO 0
#define APOS_USUARIO_TECLA_PRODUTO 1
#define APOS_COMPRA_VALIDA 2
#define APOS_COMPRA_INVALIDA 3
#define APOS_USUARIO_CANCELA_COMPRA 4
#define APOS_TECLADO_MANUTENCAO_SUBMIT_SENHA_INSERIDA 5
#define APOS_SENHA_VALIDA 6
#define APOS_PORTA_DE_MANUTENCAO_FOI_FECHADA 7
#define APOS_TECLADO_INTERNO_SUBMIT_CADASTRO_TROCO_EFETUADO 8
#define APOS_TECLADO_INTERNO_SUBMIT_CADASTRO_PRODUTO_EFETUADO 9

#endif // DEFINICOES_SISTEMA_H_INCLUDED
```

4 Loop Principal

```
int main() {

  int codigoEvento;
  int codigoAcao;
  int estado;
  int eventoInterno;

  estado = IDLE;
  eventoInterno = NENHUM_EVENTO;



  iniciaSistema();
  printf ("Bem Vindo a Vending Machine\n");
  // mostra painel com tabela de: produtos, numeros dos produtos e precos
  painelProdutosPrecos();
  while (true) {
    if (eventoInterno == NENHUM_EVENTO) {
        // modifiquei obterEvento para receber o parametro estado
        // pois nao preciso ouvir todos os eventos, só os que interessam
        // no estado que eu estou, pois por exemplo, no caso do teclado interno
        // nao é realistico a posibilidade de obter um evento sem já ter tido acesso
        // à maquina
        codigoEvento = obterEvento(estado);
    } else {
        codigoEvento = eventoInterno;
    }
    if (codigoEvento != NENHUM_EVENTO)
    {
       codigoAcao = obterAcao(estado, codigoEvento);
       estado = obterProximoEstado(estado, codigoEvento);
       eventoInterno = executarAcao(codigoAcao);
       printf("Estado: %d Evento: %d Acao:%d\n", estado, codigoEvento, codigoAcao);
    }
  } // while true
} // main
```

Painel da Vending Machine:

```
    void painelProdutosPrecos() {
  printf("Painel da Vending Machine");
  printf("Nome do Produto   | número |   Preço  ");
  printf("    Produto 1     |   1    |  0,50 R$" );
  printf("    Produto 2     |   2    |  1,50 R$");
  printf("    Produto 3     |   3    |  0,05 R$");
  printf("    Produto 4     |   4    |  0,10 R$");
  printf("    Produto 5     |   5    |  0,30 R$");
  printf("    Produto 6     |   6    |  2,50 R$");
  printf("    Produto 7     |   7    |  0,50 R$");
  printf("    Produto 8     |   8    |  0,80 R$");
  printf("    Produto 9     |   9    |  2,00 R$");
}
```

5 Obter Eventos

```
int obterEvento(int estadoAtual){
  int evento = NENHUM_EVENTO;

  if (estadoAtual == IDLE || estadoAtual == COLOCANDO_MOEDAS) {
    if (sen_DetectaMoedaInserida())
      return DETECTOU_MOEDA_ENTRANDO;
  }

  if (estadoAtual == IDLE) {
    char *senhaTecladoManutencao = tec_TecladoAutenticacaoRetornaSenhaTeclada();
    int senhaTecladoManutencaoInt = atoi(senhaTecladoManutencao);
    // se usuario (manutencao) digitou alguma coisa
    if (senhaTecladoManutencao != 0)
      return TECLADO_MANUTENCAO_SUBMIT_SENHA_INSERIDA;
  }

  if (estadoAtual == COLOCANDO_MOEDAS || estadoAtual == ESPERANDO_USUARIO_DECIDIR) {
    // cliente tem 3 opcoes( cancelar, submeter, ou usar o teclado numerico para escolher o produto)
    // mas o uso do teclado numerico para escolher o produto só terá uso após ele ter submetido anteriormente
    // e se ele estiver colocando moedas, apenas cancelar tera uso para ele
    char teclaPressionadaCliente = tec_TecladoClienteRetornaTeclaPressionada();
    if (strcmp(teclaPressionadaCliente,'S') == 0)
      return USUARIO_SUBMIT_DINHEIRO_INSERIDO;
    if (strcmp(teclaPressionadaCliente,'C') == 0)
      return USUARIO_CANCELA_COMPRA;
    int digit = atoi(teclaPressionadaCliente);
    if ( 1 <= digit && digit <= 9)
      return USUARIO_TECLA_PRODUTO;
  }

  if (estadoAtual == OCORRENDO_MANUTENCAO) {
  // a manutencao pode escolher uma moeda especifica para repor
  // ou um produto especifico para repor
    char **teclasPressionadasTecladoInterno = tec_TecladoInternoRetornaTeclasPressionadas();
    // retorna array do tipo {escolha, quantidade}

    // maximo 100 moedas de cada. Fica a cargo do repositor saber quanto de cada moeda ele vai por
    // ele vai poder ver a quantidade de moedas em realacao ao espaco disponivel de cada moeda manualamente,
    // pra saber quantas ele vai colocar
    if ( 1 <= atoi(teclasPressionadasTecladoInterno[1]) && atoi(teclasPressionadasTecladoInterno[1]) <= 100) {
      if (
        strcmp(teclasPressionadasTecladoInterno[0], "1m") == 0 ||
        strcmp(teclasPressionadasTecladoInterno[0], "2m") == 0 ||
        strcmp(teclasPressionadasTecladoInterno[0], "3m") == 0 ||
        strcmp(teclasPressionadasTecladoInterno[0], "4m") == 0 ||
        strcmp(teclasPressionadasTecladoInterno[0], "5m") == 0
      )
        return TECLADO_INTERNO_SUBMIT_CADASTRO_TROCO_EFETUADO;
    }
    // maximo 9 produtos de cada. Fic a cargo de repositor saber quanto de cada produto ele vai por
    // ele vai poder ver a quantidade de produtos em realcao ao espaco disponivel para saber quantos ele vai colocar
```

```
    if ( 1 <= atoi(teclasPressionadasTecladoInterno[1]) && atoi(teclasPressionadasTecladoInterno[1]) <= 9) {
      if (
        strcmp(teclasPressionadasTecladoInterno[0], "1p") == 0 ||
        strcmp(teclasPressionadasTecladoInterno[0], "2p") == 0 ||
        strcmp(teclasPressionadasTecladoInterno[0], "3p") == 0 ||
        strcmp(teclasPressionadasTecladoInterno[0], "4p") == 0 ||
        strcmp(teclasPressionadasTecladoInterno[0], "5p") == 0 ||
        strcmp(teclasPressionadasTecladoInterno[0], "6p") == 0 ||
        strcmp(teclasPressionadasTecladoInterno[0], "7p") == 0 ||
        strcmp(teclasPressionadasTecladoInterno[0], "8p") == 0 ||
        strcmp(teclasPressionadasTecladoInterno[0], "9p") == 0
        )
        return TECLADO_INTERNO_SUBMIT_CADASTRO_PRODUTO_EFETUADO;
    }

    // manutencoa pode tambem fechar a porta para encerrar manutencao
    if (sen_DetectaFechamentoPortaManutencao())
      return PORTA_DE_MANUTENCAO_FOI_FECHADA;
  }

  if (estadoAtual == DERRUBANDO_PRODUTO) {
    // ve se o tempo estorou
    if (tmr_timeout())
      // desliga o timer se o tempo estorou
      tmr_iniciar(false);
      return TIMER_FINALIZADO;
  }


  return evento;


} // obterEvento
```

6 Matriz da Máquina de Estados

```
/*****************************************************************
 iniciaMaquina de Estados
 Carrega a maquina de estados
 Parametros de entrada: nenhum
 Retorno: nenhum
*****************************************************************/
void iniciaMaquinaEstados() {
  int i;
  int j;

  for (i=0; i < NUM_ESTADOS; i++) {
    for (j=0; j < NUM_EVENTOS; j++) {
       acao_matrizTransicaoEstados[i][j] = NENHUMA_ACAO;
       proximo_estado_matrizTransicaoEstados[i][j] = i;
    }
  }
  proximo_estado_matrizTransicaoEstados
    [IDLE][DETECTOU_MOEDA_ENTRANDO] = COLOCANDO_MOEDAS;
  // acao_matrizTransicaoEstados
  //  [IDLE][DETECTOU_MOEDA_ENTRANDO] = NENHUMA_ACAO;

  proximo_estado_matrizTransicaoEstados
    [IDLE][TECLADO_MANUTENCAO_SUBMIT_SENHA_INSERIDA] = VERIFICANDO_SENHA_MANUTENCAO;
  acao_matrizTransicaoEstados
    [IDLE][TECLADO_MANUTENCAO_SUBMIT_SENHA_INSERIDA] = APOS_TECLADO_MANUTENCAO_SUBMIT_SENHA_INSERIDA;

  // proximo_estado_matrizTransicaoEstados
  //  [COLOCANDO_MOEDAS][DETECTOU_MOEDA_ENTRANDO] = COLOCANDO_MOEDAS;
  // acao_matrizTransicaoEstados
  //  [COLOCANDO_MOEDAS][DETECTOU_MOEDA_ENTRANDO] = NENHUMA_ACAO;

  proximo_estado_matrizTransicaoEstados
    [COLOCANDO_MOEDAS][USUARIO_SUBMIT_DINHEIRO_INSERIDO] = ESPERANDO_USUARIO_DECIDIR;
  acao_matrizTransicaoEstados
    [COLOCANDO_MOEDAS][USUARIO_SUBMIT_DINHEIRO_INSERIDO] = APOS_USUARIO_SUBMIT_DINHEIRO_INSERIDO;

  proximo_estado_matrizTransicaoEstados
    [COLOCANDO_MOEDAS][USUARIO_CANCELA_COMPRA] = IDLE;
  acao_matrizTransicaoEstados
    [COLOCANDO_MOEDAS][USUARIO_CANCELA_COMPRA] = APOS_USUARIO_CANCELA_COMPRA;

  proximo_estado_matrizTransicaoEstados
    [ESPERANDO_USUARIO_DECIDIR][USUARIO_TECLA_PRODUTO] = VERIFICANDO_POSSIBILIDADE_COMPRA;
  acao_matrizTransicaoEstados
    [ESPERANDO_USUARIO_DECIDIR][USUARIO_TECLA_PRODUTO] = APOS_USUARIO_TECLA_PRODUTO;

  proximo_estado_matrizTransicaoEstados
    [ESPERANDO_USUARIO_DECIDIR][USUARIO_CANCELA_COMPRA] = IDLE;
  acao_matrizTransicaoEstados
    [ESPERANDO_USUARIO_DECIDIR][USUARIO_CANCELA_COMPRA] = APOS_USUARIO_CANCELA_COMPRA;
```

```
  proximo_estado_matrizTransicaoEstados
    [VERIFICANDO_POSSIBILIDADE_COMPRA][COMPRA_VALIDA] = DERRUBANDO_PRODUTO;
  acao_matrizTransicaoEstados
    [VERIFICANDO_POSSIBILIDADE_COMPRA][COMPRA_VALIDA] = APOS_COMPRA_VALIDA;

  proximo_estado_matrizTransicaoEstados
    [VERIFICANDO_POSSIBILIDADE_COMPRA][COMPRA_INVALIDA] = ESPERANDO_USUARIO_DECIDIR;
  acao_matrizTransicaoEstados
    [VERIFICANDO_POSSIBILIDADE_COMPRA][COMPRA_INVALIDA] = APOS_COMPRA_INVALIDA;

  proximo_estado_matrizTransicaoEstados
      [DERRUBANDO_PRODUTO][TIMER_FINALIZADO] = IDLE;
  // acao_matrizTransicaoEstados
  //  [DERRUBANDO_PRODUTO][TIMER_FINALIZADO] = NENHUMA_ACAO;

  proximo_estado_matrizTransicaoEstados
    [VERIFICANDO_SENHA_MANUTENCAO][SENHA_VALIDA] = OCORRENDO_MANUTENCAO;
  acao_matrizTransicaoEstados
    [VERIFICANDO_SENHA_MANUTENCAO][SENHA_VALIDA] = APOS_SENHA_VALIDA;

  proximo_estado_matrizTransicaoEstados
    [VERIFICANDO_SENHA_MANUTENCAO][SENHA_INVALIDA] = IDLE;
  // acao_matrizTransicaoEstados
  //  [VERIFICANDO_SENHA_MANUTENCAO][SENHA_INVALIDA] = NENHUMA_ACAO;

  proximo_estado_matrizTransicaoEstados
    [OCORRENDO_MANUTENCAO][APOS_TECLADO_INTERNO_SUBMIT_CADASTRO_PRODUTO_EFETUADO] = OCORRENDO_MANUTENCAO;
  acao_matrizTransicaoEstados
    [OCORRENDO_MANUTENCAO][APOS_TECLADO_INTERNO_SUBMIT_CADASTRO_PRODUTO_EFETUADO] =
      APOS_TECLADO_INTERNO_SUBMIT_CADASTRO_PRODUTO_EFETUADO;

  proximo_estado_matrizTransicaoEstados
    [OCORRENDO_MANUTENCAO][APOS_TECLADO_INTERNO_SUBMIT_CADASTRO_TROCO_EFETUADO] = OCORRENDO_MANUTENCAO;
  acao_matrizTransicaoEstados
    [OCORRENDO_MANUTENCAO][APOS_TECLADO_INTERNO_SUBMIT_CADASTRO_TROCO_EFETUADO] =
      APOS_TECLADO_INTERNO_SUBMIT_CADASTRO_TROCO_EFETUADO;

  proximo_estado_matrizTransicaoEstados
    [OCORRENDO_MANUTENCAO][PORTA_DE_MANUTENCAO_FOI_FECHADA] = IDLE;
  acao_matrizTransicaoEstados
    [OCORRENDO_MANUTENCAO][PORTA_DE_MANUTENCAO_FOI_FECHADA] = APOS_PORTA_DE_MANUTENCAO_FOI_FECHADA;

} // initStateMachine
```

7 Executar Ação

```
/*********************************************************************
 executarAcao
 Executa uma acao
 Parametros de entrada:
    (int) codigo da acao a ser executada
 Retorno: (int) codigo do evento interno ou NENHUM_EVENTO
*********************************************************************/
// tem que colocar parametros pra acao aqui tambem
int executarAcao(int codigoAcao)
{
    int retval;

    retval = NENHUM_EVENTO;
    if (codigoAcao == NENHUMA_ACAO)
        return retval;

    switch(codigoAcao) {
    // apos usuario "dizer" que terminou de inserir o dinheiro
      case APOS_USUARIO_SUBMIT_DINHEIRO_INSERIDO:;
          // le uma lista do tipo: {valormoeda1, quantidademoeda1, valormoeda2, quantidademoeda2 ...}
          // apos o sensor ter detectado as moedas inseridas(digitadas) ele atulazou esse buffer(lista) internamente
          double *buffer = sen_LerBufferMoedasInseridas();
          // zera o buffer do sensor para que este possa guardar as moedas inseridas na proxima compra
          sen_ZerarBufferMoedasInseridas();

          // calcula o total inserido (funcao propria da maq estados)
          double totalInserido =
            (buffer[0]*buffer[1]) + (buffer[2]*buffer[3]) + (buffer[4]*buffer[5]) + (buffer[6]*buffer[7]);

          // guarda o total inserido no sistema de recebimento de dinheiro
          rec_GuardarTotalInserido(totalInserido);
          // mostra dinheiro inserido no display para o cliente
          disp_MostrarDinheiroInserido(totalInserido);
          break;
      case APOS_USUARIO_TECLA_PRODUTO:;
          // le o produto o cliente digitou, que o teclad armazenou em seu buffer interno (de 1 a 9)
          char* bufferCliente = tec_LerBufferEscolhaCliente();
          // transforma char em int
          int digit = atoi(bufferCliente);
          // le a quantidade deste produto disponivel no banco de dados de produtos
          int quantidadeDisponivel = prod_LerQuantidadeProduto(digit);

          // agora serao feitas to 3 verificacoes em ordem:
          // 1) Se o produto esta disponivel;
          // 2) se o dinheiro é suficiente;
          // 3) Se tem troco disponivel;

          // [funcao propria da maq. de estados]
          if (quantidadeDisponivel == 0)
            // nao ha mais desse produto
            return COMPRA_INVALIDA;
```

```
else {
  // le o dinheiro que tinhamos guardado no sistema de recebimento de dinheiro
  double valorInserido = rec_LerDinheiroInserido();
  // le o preco do produto no banco de dados de produtos
  double precoProduto = prod_LerPrecoProduto(digit);

  // [funcao propria da maq. de estados]
  if (valorInserido <= precoProduto)
    // dinheiro nao é suficiente
    return COMPRA_INVALIDA;

  else {
    // le o troco disponivel, que eh uma lista do tipo:
    // {valormoeda1, quantidademoeda1, valormoeda2, quantidademoeda2 ...}
    double *trocoDisponivel = cxt_VerificarTrocoDisponivel();

    int moedas5centavos = trocoDisponivel[1];
    int moedas10centavos = trocoDisponivel[3];
    int moedas25centavos = trocoDisponivel[5];
    int moedas50centavos = trocoDisponivel[7];
    int moedas1real = trocoDisponivel[9];

    int trocoMoedas1real = 0;
    int trocoMoedas50centavos = 0;
    int trocoMoedas25centavos = 0;
    int trocoMoedas10centavos = 0;
    int trocoMoedas5centavos = 0;

    // valor de troco que precisamos devolver para cliente
    double trocoNecessario = valorInserido - precoProduto;

    // nossas moedas disponiveis para troco
    double moedas [10] = {
      1, moedas1real,
      0.5, moedas50centavos,
      0.25, moedas25centavos,
      0.1, moedas10centavos,
      0.05, moedas5centavos
    };
    // moedas que vamos separar para o troco
    double moedasParaTroco [10] = {
      1, trocoMoedas1real,
      0.5, trocoMoedas50centavos,
      0.25, trocoMoedas25centavos,
      0.1, trocoMoedas10centavos,
      0.05, trocoMoedas5centavos
    };

    // algoritmo que vai separando moedas para troco e define se troco é suficiente ou nao ()
    // se for possivel, vai cair em algum break
    // se nao, nao é possivel retornar troco com as moedas que temos
    // [funcao propria da maq. de estados]
    int i;
    int quociente;
    for(i=0; i <=10; i+=2) {
```

```
          if (moedas[i+1] != 0)
            quociente = trocoNecessario / moedas[i];
              moedasParaTroco[i+1] = quociente;
            if (i == 0 && quociente == 1)
              break;
            double resto = trocoNecessario - quociente*moedas[i];
            if (resto == 0)
              break;
            trocoNecessario = resto;

            return COMPRA_INVALIDA;

        //guarda as moedas para troco no buffer de devolucao de troco, do caixa de troco
        //se nao precisar de troco, via mandar o array incial com zeros em todas as quantidades
        cxt_SepararTrocoParaDevolver(moedasParaTroco);
        return COMPRA_VALIDA;
        }
      }
    }

case APOS_COMPRA_VALIDA:;
    // retira troco do buffer que guarda o troco a ser devolvido
    double *moedasParaTroco = cxt_RetirarTrocoParaDevolver();
    // le o produto que o cliente digitou antes (de 1 a 9)
    bufferCliente = tec_LerBufferEscolhaCliente();
    // zera o buffer que guardava a escolha do cliente, o produto nesse caso
    // para que outra escolha possa ser guardada na proxima compra
    tec_ZerarBufferEscolhaCliente();
    // converte o char em int
    digit = atoi(bufferCliente);
    // deleta produto do banco de dados
    prod_DeletarProduto(digit);
    // inicia timer, pois vamos mudar o estado de DERRUBANDO_PRODUTO para IDLE baseado em um certo
    // tempo em que vamos poder assumir que o produto foi derrubado e podemos aceitar outras
    // compras ou manutencao
    tmr_iniciar(true);
    // o sistema de recebimento de dinheiro esta acima do moedeiro(internamente na maquina)
    // e o atuador atua de forma a deslocar a base
    // e o dinheiro cair
    at_PromoverCaidaDinheironoMoedeiroPeloDeslocamentodaBase();
    // sabendo a quantidde de cada moeda para o troco, passamos essa infromacao para o sistema de atuadores que pro-
    // moverá a atuação do atuador especifico de cada moeda baseado na quantidade especificada no array de input
    // se nao tiver troco, simplismente via desprender 0 moedas de cada
    at_DesprenderMoedasParaSeparacaoDoTroco(moedasParaTroco);
    // o troco separado ficará intenamente em um local alto, para que com o deslocamente da base, o troco caia para
    // o cliente
    at_PromoverCaidaTrocoSeparadoParaClientePeloDeslocamentoDaBase();
    // passamos a informacao do numero do produto que queremos derrubar para o sistema de atuadores que promoverá
    // a atuação do atuador especifico do produto especificado
    at_EmpurrarProduto(digit);
    break;
case APOS_COMPRA_INVALIDA:;
    // zeramos para que o cliente possa efetuar outra escolha(cancelamento, ou esclher outro produto)
    tec_ZerarBufferEscolhaCliente();
    // mostramos em display que a compra foi invalida
    disp_MostrarErroCompraInvalida();
```

```
    break;
case APOS_USUARIO_CANCELA_COMPRA:;
    sen_ZerarBufferMoedasInseridas();
    tec_ZerarBufferEscolhaCliente();
    // aciona atuador que promove uma alavanca mecanica que promove a devolucao das moedas inseridas, que
    // estavam no sistema de recebimento de dinheiro
    at_AlavancaDevolucaoDinheiro();
    break;
case APOS_TECLADO_MANUTENCAO_SUBMIT_SENHA_INSERIDA:;
    // le senha que foi teclada pela manutencao, e esa armazenada no buffer interno do
    // teclado de autenticacao
    char *bufferManutencao = tec_LerBufferSenhaManutencao();
    // zera o buffer para que proxima senha digitada seja guardada
    tec_ZerarBufferSenhaManutencao();
    int valueBufferManutencao = atoi(bufferManutencao);
    printf("%d", valueBufferManutencao);
    // verifica se senha digitada é a senha do sistema, pelo sistema de autenticacao
    return (aut_VerificaSenha(valueBufferManutencao)) ? SENHA_VALIDA : SENHA_INVALIDA;
case APOS_SENHA_VALIDA:;
    // possibilita abertura da porta de manutencao
    at_LiberarTravaPortaManutencao();
    break;
case APOS_PORTA_DE_MANUTENCAO_FOI_FECHADA:;
    // se foi detecntado que a manutencao fechou a a porta, o travammento ocorre em seguida
    at_EmpurrarTravaPortaManutencao();
    break;
case APOS_TECLADO_INTERNO_SUBMIT_CADASTRO_TROCO_EFETUADO:;
    // essa acao acontece apos a manutencao ter dado submit no cadastro de mais troco
    // numeros de 1m 2m 3m 4m 5m, representarao as moedas de 0.05 centavos a 1 real em ordem

    // le o tipo de cadastro que foi escolhido pela manutencao anteriormente
    // ou seja, nesse caso (troco) tem que ter sido uma das moedas a tecla esclhida
    char *bufferTecladoInternoTipodeCadastroMoeda = tec_LerBufferTecladoInternoTipodeCadastro();
    tec_ZerarBufferTecladoInternoTipodeCadastro();

    // aqui a tranformacao para int,  ainda funciona porque a funcao atoi tranforma em int até encontrar um caracter
    // que nao seja um numero, entao: '1' e '1m' vai dar na mesma
    int valueBufferTecladoInternoTipodeMoeda = atoi(bufferTecladoInternoTipodeCadastroMoeda);
    // quantidade da moeda que esta sendo cadastrada
    char *bufferTecladoInternoQuantidadeCadastradaMoeda = tec_LerBufferTecladoInternoQuantidadeCadastrada();
    tec_ZerarBufferTecladoInternoQuantidadeCadastrada();

    // le a quantidade cadastrada do tipo de moeda escolhido anteriormente
    int valueBufferTecladoInternoQuantidadeCadastradaMoeda = atoi(bufferTecladoInternoQuantidadeCadastradaMoeda);

    // junta as duas informacoes em um array e manda para o caixa de troco, para que seja inserido este troco
    // no banco de dados de troco disponivel da maquina
    int trocoCadastrado [2] =
      {valueBufferTecladoInternoTipodeMoeda, valueBufferTecladoInternoQuantidadeCadastradaMoeda};
    cxt_InserirTroco(trocoCadastrado);
    break;

case APOS_TECLADO_INTERNO_SUBMIT_CADASTRO_PRODUTO_EFETUADO:;
    // muito parecido com a acao anterior, só que vamos inserir produto ao inves de troco, no final
    char *bufferTecladoInternoTipodeCadastroProduto = tec_LerBufferTecladoInternoTipodeCadastro();
    tec_ZerarBufferTecladoInternoTipodeCadastro();
```

```
        int valueBufferTecladoInternoTipodeProduto = atoi(bufferTecladoInternoTipodeCadastroProduto);
        // quantidade da moeda que esta sendo cadastrada
        char *bufferTecladoInternoQuantidadeCadastradaProduto = tec_LerBufferTecladoInternoQuantidadeCadastrada();
        tec_ZerarBufferTecladoInternoQuantidadeCadastrada();
        int valueBufferTecladoInternoQuantidadeCadastradaProduto = atoi(bufferTecladoInternoQuantidadeCadastradaProduto)
        int produtoCadastrado [2] =
        {valueBufferTecladoInternoTipodeProduto, valueBufferTecladoInternoQuantidadeCadastradaProduto};
        // insere o produto com sua quantidade no banco de dados de produtos
        prod_InserirProdutos(produtoCadastrado);
        break;

    } // switch

    return retval;
} // executarAcao
```

8 Componentes

- **atuadores.h**

```
#ifndef ATUADORES_H_INCLUDED
#define ATUADORES_H_INCLUDED

extern void at_DesprenderMoedasParaSeparacaoDoTroco(double* liberacaoMoedas);

extern void at_PromoverCaidaTrocoSeparadoParaClientePeloDeslocamentoDaBase();

extern void at_PromoverCaidaDinheironoMoedeiroPeloDeslocamentodaBase();

extern void at_AlavancaDevolucaoDinheiro();

extern void at_EmpurrarProduto(int produto);

extern void at_EmpurrarTravaPortaManutencao();

extern void at_LiberarTravaPortaManutencao();


#endif
```

- **atuadores.c**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "definicoes_sistema.h"
#include "atuadores.h"

void at_DesprenderMoedasParaSeparacaoDoTroco(double* liberacaoMoedas)
{
    printf("Acionamento do atuador que desprende as moedas pra que sejam separadas para o troco");
}

void at_PromoverCaidaTrocoSeparadoParaClientePeloDeslocamentoDaBase()
{
  printf("Acionamento do atuador que promove caida das moedas separadas anterormente para o troco.");
  printf("Pelo deslocamento da base em que as moedas estao");
}

void at_PromoverCaidaDinheironoMoedeiroPeloDeslocamentodaBase()
{
    printf("Acionamento do atuador que promove caida das moedas, que foram usadas para a compra do produto,");
    printf("no moedeiro(que é só uma caixa que guarda o dinheiro)");
}

void at_AlavancaDevolucaoDinheiro()
{
  printf("Acionamento do atuador que faz uma alavanca mecanica para que o dinheiro inserido seja devolviso");
}
```

```
void at_EmpurrarProduto(int produto)
{
  printf("Acionamento do atuador que empurra o produto de número %d", produto);
}

void at_EmpurrarTravaPortaManutencao()
{
     printf("Acionamento do atuador que empurra empurra a trava da porta de manutencao, promovendo o travamento da p
}

void at_LiberarTravaPortaManutencao()
{
    printf("Acionamento do atuador que libera a tarva da porta de mmanutencao, posibilitando abertura da mesma");
}
```

- **db_caixa_de_troco.h**

```
#ifndef DB_CAIXA_DE_TROCO_H_INCLUDED
#define DB_CAIXA_DE_TROCO_H_INCLUDED

extern double* cxt_VerificarTrocoDisponivel();

extern void cxt_InserirTroco(int* trocoInserido);

extern void cxt_SepararTrocoParaDevolver(double* trocoInserido);

extern double* cxt_RetirarTrocoParaDevolver();


#endif
```

- **db_caixa_de_troco.c**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "definicoes_sistema.h"
#include "db_caixa_de_troco.h"

// {
//   valordamoeda, quantidade
// }

static double TROCO_DISPONIVEL [10] =
{
  0.05, 9,
  0.1, 9,
  0.25, 9,
  0.5, 9,
  1.0, 9
};

static double BUFFER_MOEDAS_PARA_DEVOLVER_TROCO [10] =
{
  0.05, 0,
  0.1, 0,
  0.25, 0,
  0.5, 0,
  1.0, 0
};

double* cxt_VerificarTrocoDisponivel() {
  return TROCO_DISPONIVEL;
}

void cxt_InserirTroco(int* trocoInserido) {
  // vai receber {2, 3} por exemplo
  // vai colocar em TROCO_DISPONIVEL 3 moedas de 10  centavos
  // numeros de 1 a 5 representarao as moedas de 0.05 centavos a 1 real em sequecia crescente
  // so da pra carregar um tipo de moedade cada vez
  int intMoeda = trocoInserido[0];
```

```
  int quantidadeInserida = trocoInserido[1];
  int index;
  // conversao para index do buffer
  index = (intMoeda - 1)*2;
  int quantidadeAtual = BUFFER_MOEDAS_PARA_DEVOLVER_TROCO[index];
  BUFFER_MOEDAS_PARA_DEVOLVER_TROCO[index] = quantidadeAtual + quantidadeInserida;
}

void cxt_SepararTrocoParaDevolver(double* trocoInserido) {
  // input eh array do tipo:
  //{
  //  1, numeromoedas1real, 0.5, numeromoedas50centavos, 0.25, numeromoedas25centavos,
  //  0.10, numeromoedas10centavos, 0.05, numeromoedas5centavos
  //}

  int i;
  int quant = 1;
  // como a ordem das moedas do input esta ao contrario da ordem no Buffer
  for(i=9; i>=0; i=i-2) {
    BUFFER_MOEDAS_PARA_DEVOLVER_TROCO[quant] = trocoInserido[i];
    quant+=2;
  }
}

double* cxt_RetirarTrocoParaDevolver() {
  int i;
  // ler  buffer
  static double moedasParaTroco [10];
  memcpy(moedasParaTroco, BUFFER_MOEDAS_PARA_DEVOLVER_TROCO , 10);
  // zerar buffer
  for(i=1; i<=9; i+=2) {
    BUFFER_MOEDAS_PARA_DEVOLVER_TROCO[i] = 0;
  }
  return moedasParaTroco;
}
```

- **db_produtos.h**

```
#ifndef DB_PRODUTOS_H_INCLUDED
#define DB_PRODUTOS_H_INCLUDED

extern void prod_InserirProdutos(int* produtosInseridos);

extern void prod_DeletarProduto(int produtoDeletado);

extern double prod_LerPrecoProduto(int ProdutoLido);

extern int prod_LerQuantidadeProduto(int ProdutoLido);

#endif
```

- **db_produtos.c**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "definicoes_sistema.h"
#include "db_produtos.h"

// produto, quantidade, preco
static double DATA_PRODUTOS [27] = {
  PRODUTO_1, 9, PRECO_PRODUTO_1,
  PRODUTO_2, 9, PRECO_PRODUTO_2,
  PRODUTO_3, 9, PRECO_PRODUTO_3,
  PRODUTO_4, 9, PRECO_PRODUTO_4,
  PRODUTO_5, 9, PRECO_PRODUTO_5,
  PRODUTO_6, 9, PRECO_PRODUTO_6,
  PRODUTO_7, 9, PRECO_PRODUTO_7,
  PRODUTO_8, 9, PRECO_PRODUTO_8,
  PRODUTO_9, 9, PRECO_PRODUTO_9
};

int prod_LerQuantidadeProduto(int ProdutoLido) {
  // input eh o numero do produto de 1 a 9
  int index;
  // conversao para index do produto em DATA_PRODUTOS
  index = (ProdutoLido -1)*3;
  return DATA_PRODUTOS[index + 1];
}

double prod_LerPrecoProduto(int ProdutoLido) {
  // input eh o numero do produto de 1 a 9
  int index;
  // conversao para index do DATA_PRODUTOS
  index = (ProdutoLido -1)*3;
  return DATA_PRODUTOS[index + 2];
}

void prod_InserirProdutos(int* produtosInseridos) {
  // recebe input do tipo: {6, 5}
  //que significa inserir 5 produtos de número 6
  int index;
```

```
  int numProduto = produtosInseridos[0];
  int quantidadeInserida = produtosInseridos[1];
  // conversao para index do DATA_PRODUTOS
  index = (numProduto -1)*3;
  int quantidadeAtual = DATA_PRODUTOS[index + 1];
  DATA_PRODUTOS[index + 1] = quantidadeAtual + quantidadeInserida;
}

void prod_DeletarProduto(int produtoDeletado) {
  // input eh o numero do produto de 1 a 9
  int index;
  // conversao para index do DATA_PRODUTOS
  index = (produtoDeletado -1)*3;
  int quantidade = DATA_PRODUTOS[index + 1];
  DATA_PRODUTOS[index + 1] = quantidade -1;
}
```

- **display.h**

```
#ifndef DISPLAY_H_INCLUDED
#define DISPLAY_H_INCLUDED

extern void disp_MostrarDinheiroInserido(double dinheiroInserido);

extern void disp_MostrarErroCompraInvalida();

#endif
```

- **display.c**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "definicoes_sistema.h"
#include "display.h"

void disp_MostrarDinheiroInserido(double dinheiroInserido) {
  printf("Voce inseriu %f reais", dinheiroInserido);


}

void disp_MostrarErroCompraInvalida() {
  printf("Compra invalida, cancele a compra ou escolha outro produto");
}
```

- **sensores.h**

```
#ifndef SENSORES_H_INCLUDED
#define SENSORES_H_INCLUDED

extern double* sen_LerBufferMoedasInseridas();

extern void sen_ZerarBufferMoedasInseridas();

extern int sen_DetectaMoedaInserida();

extern int sen_DetectaFechamentoPortaManutencao();

#endif
```

- **sensores.c**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "definicoes_sistema.h"
#include "sensores.h"

// {
//   valordamoeda, quantidade
// }

static double BUFFER_MOEDAS_INSERIDAS [10] =
{
  0.05, 0,
  0.1, 0,
  0.25, 0,
  0.5, 0,
  1.0, 0
};

int sen_DetectaMoedaInserida(){
  printf("Para inserir uma moeda de 5 centavos digite '1'  e pressione enter \n");
  printf("Para inserir uma moeda de 10 centavos digite '2' e pressione enter \n");
  printf("Para inserir uma moeda de 25 centavos digite '3' e pressione enter \n");
  printf("Para inserir uma moeda de 50 centavos digite '4' e pressione enter \n");
  printf("Para inserir uma moeda de 1 real digite '5' e pressione enter \n");
  printf("Se voce nao quer inserir moedas digite qualquer outra coisa e pressione enter \n");

  char moeda [100];
  double valor;

  // char c;
  // while((c = getchar()) != '\n' && c != EOF)
  fflush(stdin);
  fgets(moeda, 100, stdin);
  moeda[strcspn(moeda, "\n")] = 0;

  int numMoeda = atoi(moeda);
  if (1 <= numMoeda && numMoeda<= 5) {
    int index;
```

```
    // conversao para index do buffer
    index = 2*numMoeda - 1;
    int quantidadeInseridaAtual = BUFFER_MOEDAS_INSERIDAS[index];
    BUFFER_MOEDAS_INSERIDAS[index] = quantidadeInseridaAtual + 1;
    return true;
  }
  return false;
}


double* sen_LerBufferMoedasInseridas() {
  return BUFFER_MOEDAS_INSERIDAS;
}

void sen_ZerarBufferMoedasInseridas() {
  int i;
  // zerar as quantidadades
  for(i=1; i<=9; i+=2) {
    BUFFER_MOEDAS_INSERIDAS[i] = 0;
  }
}

int sen_DetectaFechamentoPortaManutencao(){
  printf("Se voce esta aqui, é porque terminou ou nao quis mais fazer manutencao (cadastro de produto ou troco) \n")
  printf("Digite 'X' para fechar a porta de manutencao \n");
  printf("Para nao fechar a porta digite qualquer outra coisa \n");
  char fechar [100];

  fflush(stdin);
  fgets(fechar, 100, stdin);
  fechar[strcspn(fechar, "\n")] = 0;

  if (strcmp(fechar, 'X') == 0)
    return true;
  printf("returning false");
  return false;
}
```

- **sist_autenticacao.h**

```
#ifndef SIST_AUTENTICACAO_H_INCLUDED
#define SIST_AUTENTICACAO_H_INCLUDED

extern int aut_VerificaSenha(int senhaTeclada);

#endif
```

- **sist_autenticacao.c**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "definicoes_sistema.h"
#include "sist_autenticacao.h"

int SENHA = 123;

int aut_VerificaSenha(int senhaTeclada) {
 return (senhaTeclada == SENHA) ? true : false;
}
```

- **sist_recebimento_dinheiro.h**

```
#ifndef SIST_RECEBIMENTO_DINHEIRO_H_INCLUDED
#define SIST_RECEBIMENTO_DINHEIRO_H_INCLUDED

extern void rec_GuardarTotalInserido(double valorInserido);

extern double rec_LerDinheiroInserido();

#endif
```

- **sist_recebimento_dinheiro.c**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "definicoes_sistema.h"
#include "sist_recebimento_dinheiro.h"

double VALOR_INSERIDO;

void rec_GuardarTotalInserido(double valorInserido){
  VALOR_INSERIDO = valorInserido;
}

double rec_LerDinheiroInserido(){
  return VALOR_INSERIDO;
}
```

- **teclados.h**

```
#ifndef TECLADOS_H_INCLUDED
#define TECLADOS_H_INCLUDED

extern char* tec_TecladoClienteRetornaTeclaPressionada();
extern char* tec_LerBufferEscolhaCliente();
extern void tec_ZerarBufferEscolhaCliente();

extern char* tec_TecladoAutenticacaoRetornaSenhaTeclada();
extern char* tec_LerBufferSenhaManutencao();
extern void tec_ZerarBufferSenhaManutencao();

extern char** tec_TecladoInternoRetornaTeclasPressionadas();
extern char* tec_LerBufferTecladoInternoTipodeCadastro();
extern void tec_ZerarBufferTecladoInternoTipodeCadastro();
extern char* tec_LerBufferTecladoInternoQuantidadeCadastrada();
extern void tec_ZerarBufferTecladoInternoQuantidadeCadastrada();

#endif
```

- **teclados.c**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "definicoes_sistema.h"
#include "teclados.h"

static char BUFFER_ESCOLHA_CLIENTE [100];

static char BUFFER_SENHA_MANUTENCAO [100];

static char BUFFER_TECLADO_INTERNO_TIPO_DE_CADASTRO [100];

static char BUFFER_TECLADO_INTERNO_QUANTIDADE_CADASTRADA [100];

char* tec_TecladoClienteRetornaTeclaPressionada() {
  printf("Teclado disponivel para o Cliente \n");
  printf("Digite 'S' e pressione enter se voce quer submeter o dinheiro inserido \n");
  printf("Digite 'C' se voce quer cancelar a compra e obter o dinheiro de volta \n");
  printf("O teclado numerico tambem esta disponivel \n");
  printf("Atencao: Esse teclado so servira de uso apos o usuario ter submetido o dinheiro \n");
  printf("Digite o numero do produto que deseja comprar (1-9) se voce ja submeteu o dinheiro \n");
  static char escolha [100];

  fflush(stdin);
  fgets(escolha, 100, stdin);
  escolha[strcspn(escolha, "\n")] = 0;

  strcpy(BUFFER_ESCOLHA_CLIENTE, escolha);
  return escolha;
}

char* tec_LerBufferEscolhaCliente() {
    static char buffer_escolha_cliente [100];
```

```
    strcpy(buffer_escolha_cliente, BUFFER_ESCOLHA_CLIENTE);
  return buffer_escolha_cliente;
}

void tec_ZerarBufferEscolhaCliente() {
  strcpy(BUFFER_ESCOLHA_CLIENTE, "0");
}

char* tec_TecladoAutenticacaoRetornaSenhaTeclada() {
  printf("Teclado Numerico de Autenticacao Para Manutencao \n");
  printf("Se voce for da manutencao, digite a senha neste teclado, para abrir a porta de manutencao \n");
  printf("Se voce nao quer se autenticar, tecle apenas enter \n");
  char senha [100];
  fflush(stdin);
  fgets(senha, 100, stdin);
  // no fgets sempre fica um \n do enter no final
  senha[strcspn(senha, "\n")] = 0;
  strcpy(BUFFER_SENHA_MANUTENCAO, senha);
  // sÃ³ pra poder colocar em uma variavel static, pois se devolvermos um
  // pointer de variaveis desse escopo, esse pointer vai ficar vazio pois
  // as variaveis do escopo da funcao sao liberadas do stack
  static char* senhastatic;
  senhastatic  = senha;
  // BUFFER_SENHA_MANUTENCAO = senha; nao pode ser feito em C, arrays can only can be assigned in initialization
  return senhastatic;
}

char* tec_LerBufferSenhaManutencao() {
  static char buffer_senha_manutencao [100];
  strcpy(buffer_senha_manutencao, BUFFER_SENHA_MANUTENCAO);
  return buffer_senha_manutencao;
}

void tec_ZerarBufferSenhaManutencao() {
  strcpy(BUFFER_SENHA_MANUTENCAO, "0");
}

char** tec_TecladoInternoRetornaTeclasPressionadas() {
  printf("Este e o teclado interno da Vending Machine \n");
  printf("Se voce esta aqui e porque ja se autenticou e conseguiu acesso interno \n");
  printf("Aqui voce vai cadastrar o troco ou produtos \n");
  printf("Se voce ja terminou todo o cadastro ou nao quer fazer o cadastro: \n");
  printf("--> digite qualquer outra coisa. Para que voce possa fechar a porta de manutencao \n");
  printf("Funciona da seguinte forma: \n");
  printf(" 1) Digita a tecla do produto especifico ou moeda especifica + enter \n");
  printf("2) Digita quantidade do produto/moeda escolhido anteriormente + enter \n");
  printf("Teclado e composto por: Teclas das Moedas, Teclas dos Produtos e Teclado numerico \n");
  printf("Para cadastrar moedas: \n");
  printf("5 centavos => digite '1m' e pressione enter \n");
  printf("10 centavos => digite '2m' e pressione enter \n");
  printf("25 centavos => digite '3m' e pressione enter \n");
  printf("50 centavos => digite '4m' e pressione enter \n");
  printf("1 real => digite '5m' e pressione enter \n");
  printf("Para cadastrar troco:");
  printf("Produto 1 => digite '1p' \n");
  printf("Produto 2 => digite '2p' \n");
```

```
  printf("Produto 3 => digite '3p' \n");
  printf("Produto 4 => digite '4p' \n");
  printf("Produto 5 => digite '5p' \n");
  printf("Produto 6 => digite '6p' \n");
  printf("Produto 7 => digite '7p' \n");
  printf("Produto 8 => digite '8p' \n");
  printf("Produto 9 => digite '9p' \n");
  printf("Para cadastrar quantidade(de moeda ou produto especifico): utilize o teclado numerico \n");

  char escolha [100];
  fflush(stdin);
  fgets(escolha, 100, stdin);
  escolha[strcspn(escolha, "\n")] = 0;
  strcpy(BUFFER_TECLADO_INTERNO_TIPO_DE_CADASTRO, escolha);

  char quantidade [100];
  fflush(stdin);
  fgets(quantidade, 100, stdin);
  quantidade[strcspn(quantidade, "\n")] = 0;
  strcpy(BUFFER_TECLADO_INTERNO_QUANTIDADE_CADASTRADA, quantidade);

  // retorna array com a escolha cliente(tecla de meoda ou produto especifico)
  // e com aquantidade digitada
  // char** escolhaequantidade;
  char* escolhaequantidade [2];
  strcpy(escolhaequantidade[0], escolha);
  strcpy(escolhaequantidade[1], quantidade);

  static char** escolhaequantidadepointer;
  escolhaequantidadepointer = escolhaequantidade;

  return escolhaequantidadepointer;
}


char* tec_LerBufferTecladoInternoTipodeCadastro() {
  static char buffer_teclado_interno_tipo_de_cadastro [100];
  strcpy(buffer_teclado_interno_tipo_de_cadastro, BUFFER_TECLADO_INTERNO_TIPO_DE_CADASTRO);
  return buffer_teclado_interno_tipo_de_cadastro;
}

void tec_ZerarBufferTecladoInternoTipodeCadastro() {
  strcpy(BUFFER_TECLADO_INTERNO_TIPO_DE_CADASTRO, "0");
}

char* tec_LerBufferTecladoInternoQuantidadeCadastrada() {
  static char buffer_teclado_interno_quantidade_cadastrada [100];
  strcpy(buffer_teclado_interno_quantidade_cadastrada, BUFFER_TECLADO_INTERNO_QUANTIDADE_CADASTRADA);
  return buffer_teclado_interno_quantidade_cadastrada;
}

void tec_ZerarBufferTecladoInternoQuantidadeCadastrada() {
  strcpy(BUFFER_TECLADO_INTERNO_QUANTIDADE_CADASTRADA, "0");
}
```

- **timer.h**

```
// #ifndef TIMER_H_INCLUDED
// #define TIMER_H_INCLUDED

// extern void time_IniciarTimer();

// extern time_TimerFinalizado();

// #endif

#ifndef TIMER_H_INCLUDED
#define TIMER_H_INCLUDED

/*******************************
 tmr_iniciar
 Aciona ou desaciona o timer
 entradas
   controle: TRUE:liga FALSE:desliga
 saidas
   nenhuma
*******************************/
extern void tmr_iniciar(int controle);

/*******************************
 tmr_timeout
 Retorna se o timer esta em timeout.
 entradas
    nenhuma
 saidas
    FALSE: nao houve estouro do temporizador
    TRUE: houve estouro do temporizador
*******************************/
extern int tmr_timeout();

#endif // TIMER_H_INCLUDED
```

- **timer.c**

```
// #include <stdio.h>
// #include <stdlib.h>
// #include <string.h>

// #include "definicoes_sistema.h"
// #include "timer.h"

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#include "definicoes_sistema.h"
#include "timer.h"

#define TEMPO 10

int tmr_situacao;
time_t horaInicio;
```

```
/*******************************
 tmr_iniciar
 Aciona ou desaciona o timer
 entradas
   controle: TRUE:liga FALSE:desliga
 saidas
   nenhuma
*******************************/
void tmr_iniciar(int controle)
{
   tmr_situacao = controle;
   if (controle)
   {
      horaInicio = time(NULL);
   }
}

/*******************************
 tmr_timeout
 Retorna se o timer esta em timeout.
 entradas
    nenhuma
 saidas
    FALSE: nao houve estouro do temporizador
    TRUE: houve estouro do temporizador
*******************************/
int tmr_timeout()
{
    time_t horaAtual;

    horaAtual = time(NULL);
    if (tmr_situacao == false)
    {
        return false;
    }
    if ((horaAtual - horaInicio) > TEMPO)
    {
        return true;
    }
    return false;
}
```

9 Programa Rodando

Exemplos do programa rodando:

Figura 13 – Cliente insere dinheiro

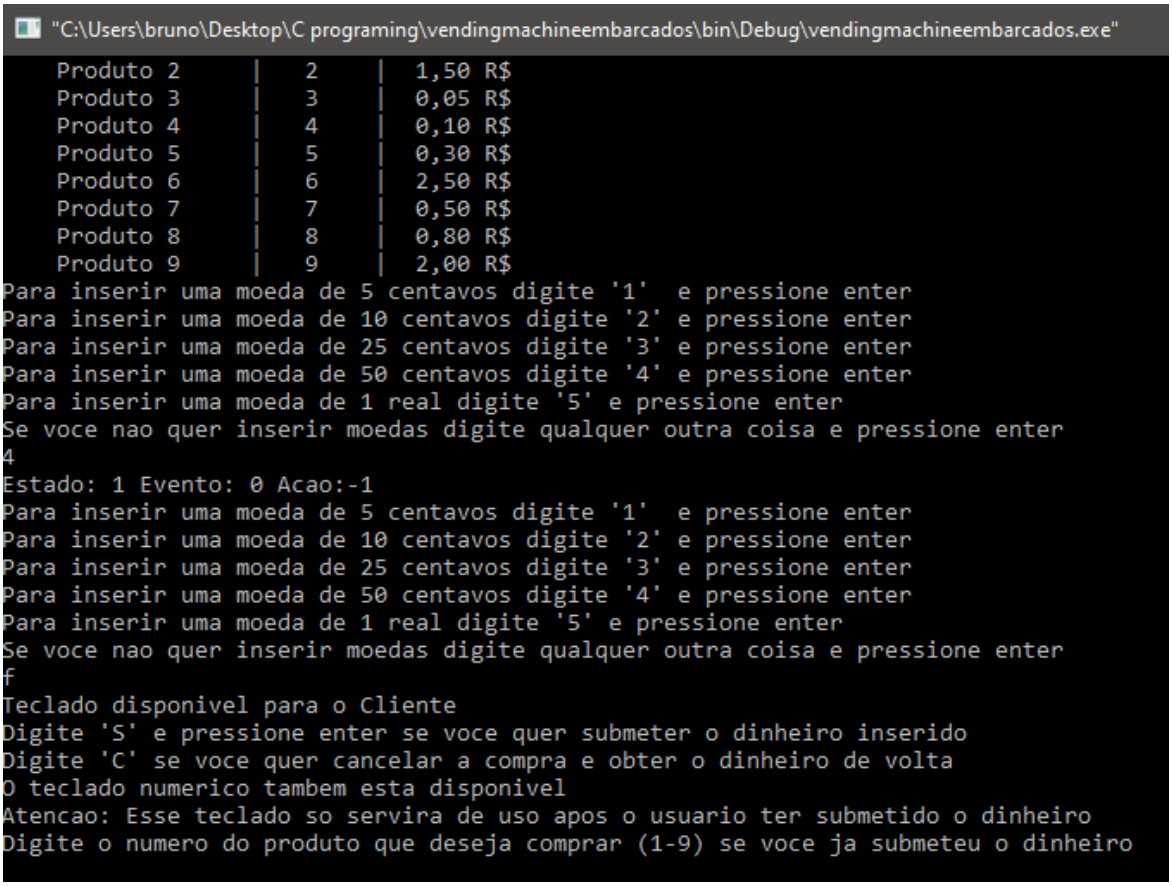


Figura 14 – Manutenção erra senha

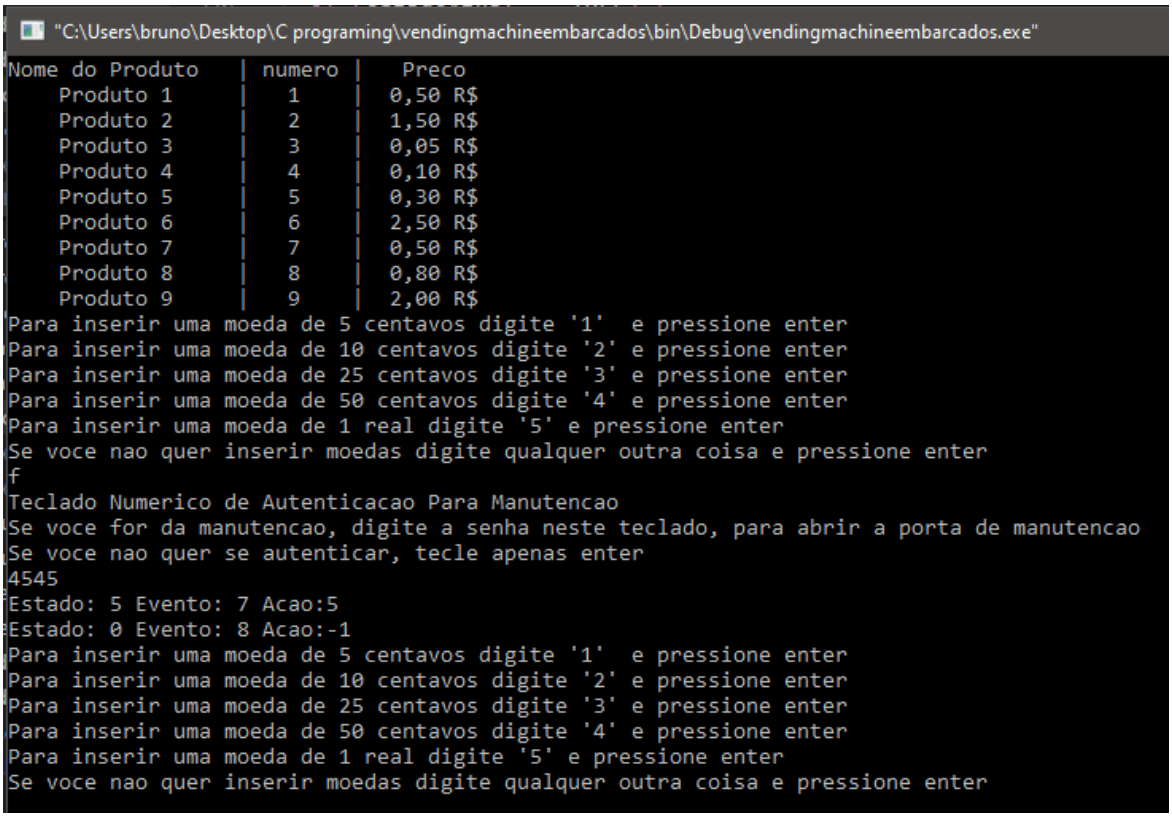
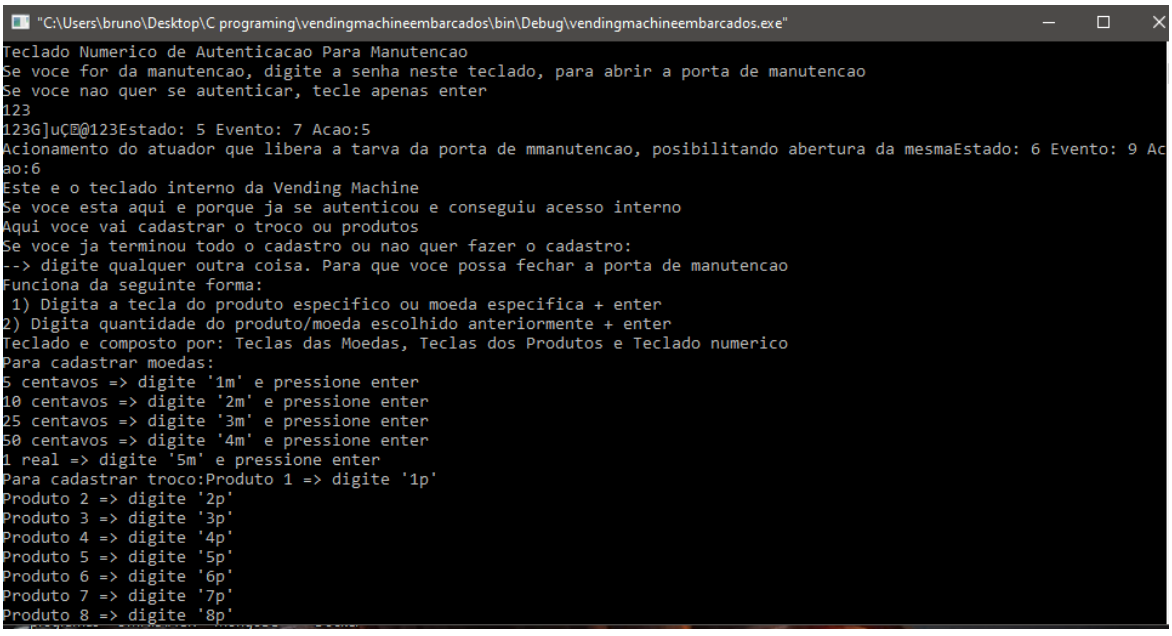


Figura 15 – Manutenção acerta senha



10 Conclusão

A matéria como um todo ajudou a entender como organizar o funcionamento de um sistema real. Os diagramas UML foram muito úteis dar uma 'guia' do que deveria ser feito no código. Porém comecei a perceber que ficou muito complexo para alguns diagramas representarem bem a situação, como no caso do diagrama de componentes, que ficou muito poluído no meu caso.