PMR3402 - Vending Machine

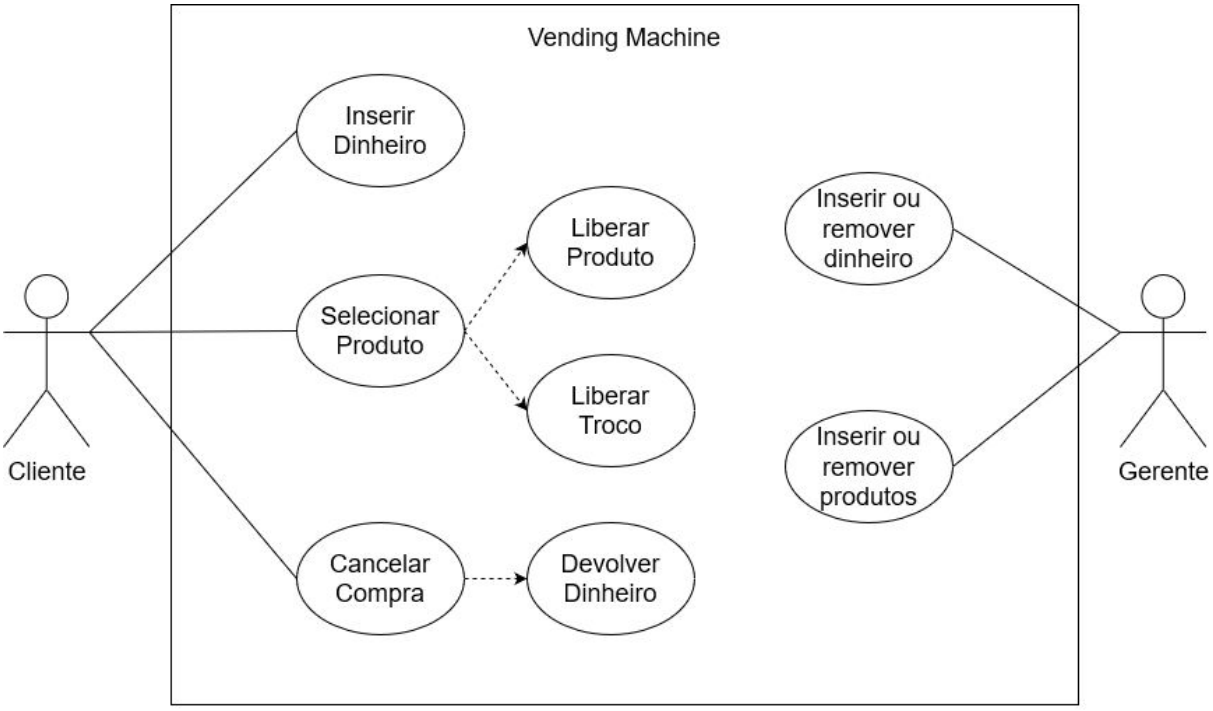
Relatório Final

Gustavo Marangoni Rubo
4584080

Diagramas UML

Casos de uso

A vending apresenta para um cliente a função de trocar dinheiro por produtos, selecionado por ele mesmo.
Um gerente da máquina tem acesso também, por meio de uma chave e trava mestra da máquina, ao cofre e estoque dos produtos. Tal trava é eletromecânica, de forma que dá acesso ao interior da máquina e funciona também como método de autenticação no sistema, similar a uma senha.



Máquina de Estados

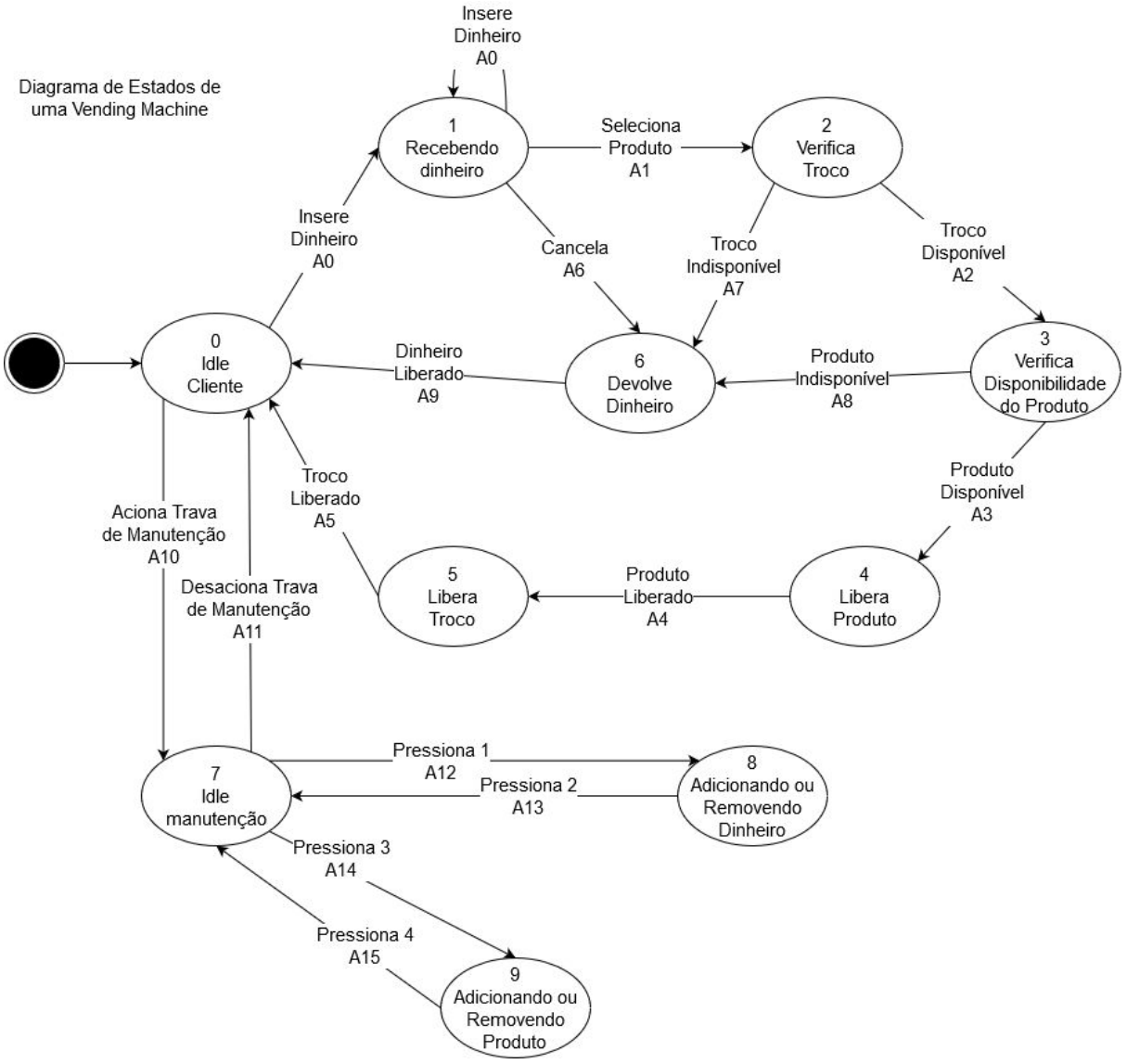


Diagrama de Componentes

Temos os componentes:

- Máquina de Estados
- Display
- Teclado Numérico (para funções de manutenção)
- Teclado de Produtos
- Sistema de Armazenamento e Descarga de Produtos
- Cofre
- Trava eletromecânica de manutenção
- Coletor e Contador de Dinheiro

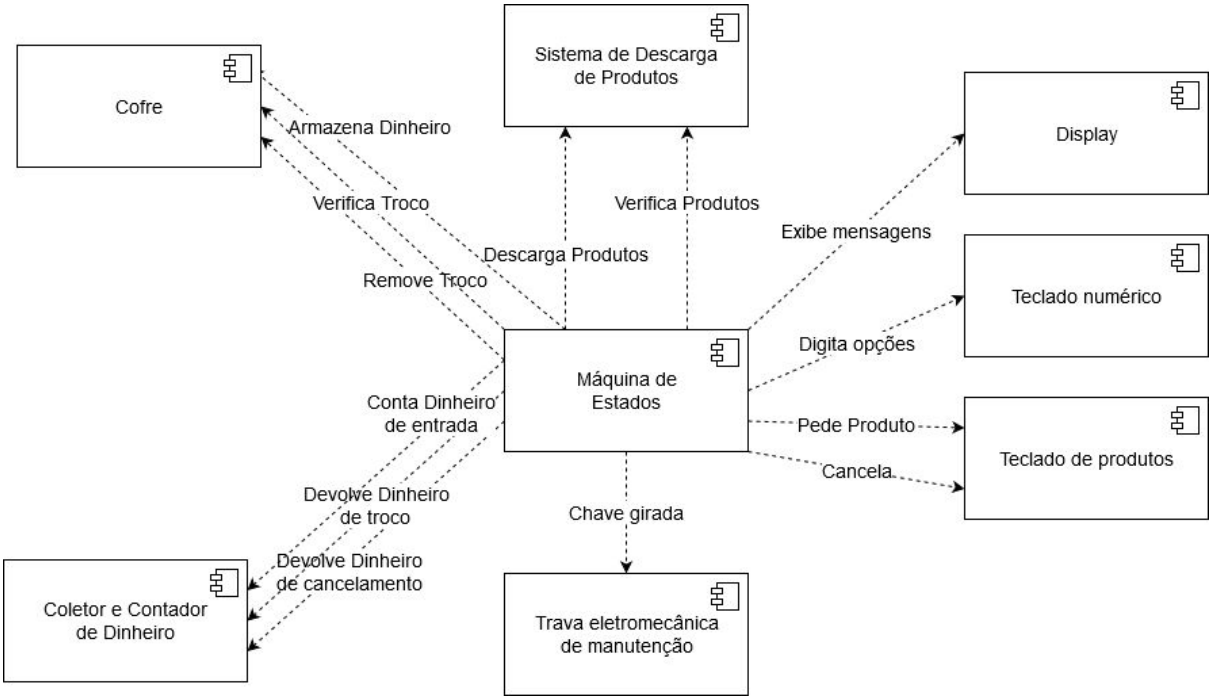
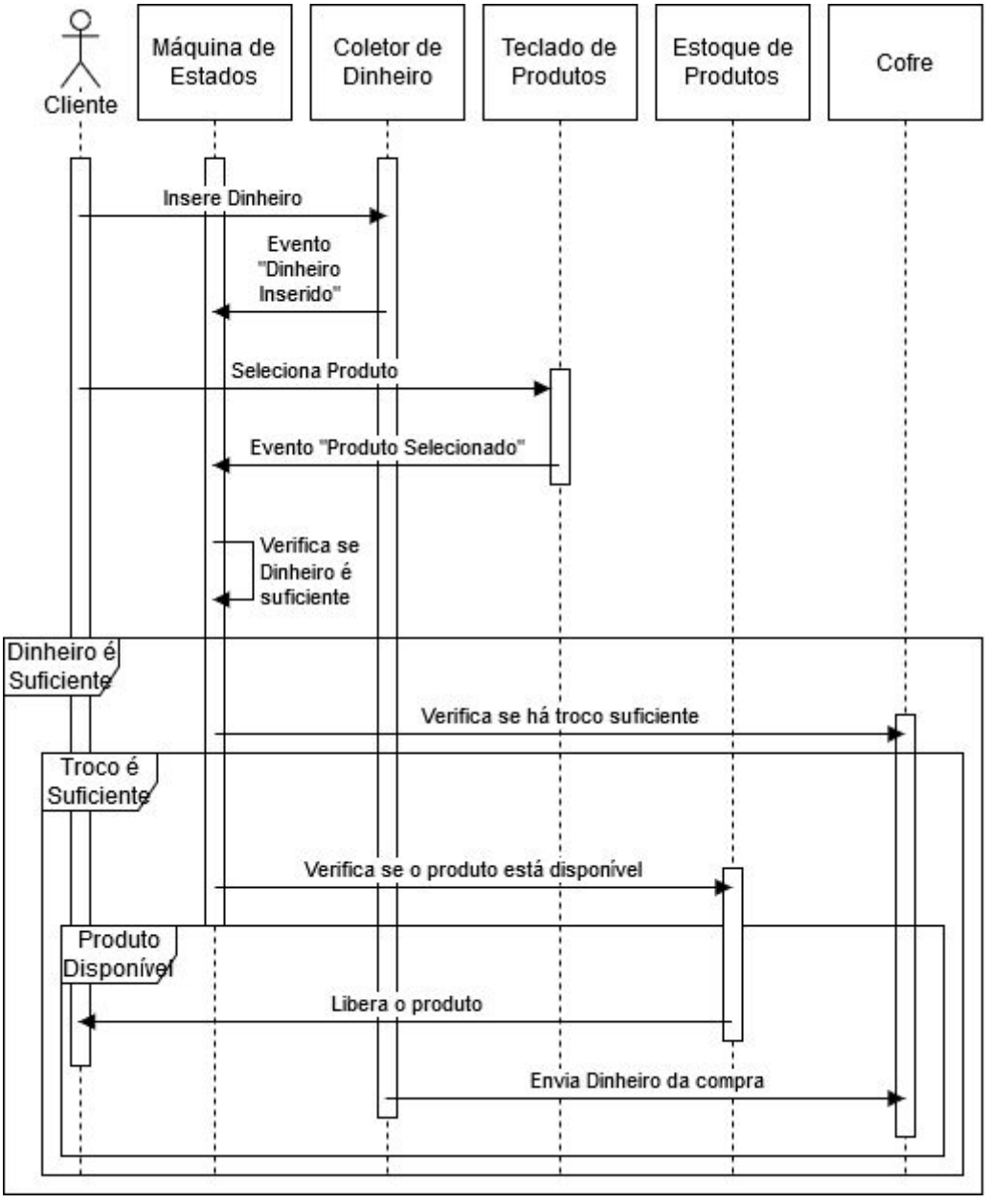


Diagrama de Sequência

Mostro aqui um fluxo de inserção de dinheiro, seleção e compra de produto



Código

Sources

main.c

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

/*
	VENDING MACHINE
*/

#include "definicoes_sistema.h"
#include "display.h"
#include "coletor_dinheiro.h"
#include "teclado_produtos.h"
#include "cofre.h"
#include "descarga_produto.h"

/*****************************************************************
 Estaticos
 *****************************************************************/
  int codigoEvento;
  int codigoAcao;
  int estado;
  int acao_matrizTransicaoEstados[NUM_ESTADOS][NUM_EVENTOS];
  int proximo_estado_matrizTransicaoEstados[NUM_ESTADOS][NUM_EVENTOS];

  dinheiro_armazenado = 5000; //comeca com 50 reais para troco
  dinheiro_buffer = 0;
  int produtos_estoque[] = {2, 5, 3, 4}; //temos 4 produtos
  int produtos_valor[] = {200, 350, 300, 40000};
  char* teclas;

/*****************************************************************
 executarAcao
 Executa uma acao
 Parametros de entrada:
	(int) codigo da acao a ser executada
 Retorno: (int) codigo do evento interno ou NENHUM_EVENTO
*****************************************************************/
int executarAcao(int codigoAcao)
{
	int retval;
	char *ptr;
	int num = strtol(teclas + 1, &ptr, 10);

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

```
        return retval;

        switch(codigoAcao)
        {
        case A01:
        //incrementa buffer de dinheiro
        col_add_buffer(num);
        break;
        case A02:
        //verifica se o dinheiro é suficiente e se há troco
        if (dinheiro_buffer < produtos_valor[num - 1]) {
                dpl_msg("Dinheiro insuficiente.\n");
                break;
        }
        if (cof_verifica_troco(dinheiro_buffer - produtos_valor[num - 1]))
                retval = TROCO_DISPONIVEL;
        break;
        case A03:
        //verifica se ha produto
        if (dsp_verifica_produto(num))
                retval = PRODUTO_DISPONIVEL;
        break;
        case A04:
        //liberar produto
        dsp_descarregar_produto(num);
        retval = PRODUTO_LIBERADO;
        break;
        case A05:
        //liberar troco, adicionar buffer ao armazenamento
        cof_armazena_valor(dinheiro_buffer);
        col_zera_buffer();
        retval = TROCO_LIBERADO;
        break;
        case A06:
        //devolver dinheiro
        col_cancelar();
        retval = CANCELA;
        break;
        } // switch

        return retval;
} // executarAcao

/*****************************************************************************
 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_CLIENTE][INSERE_DINHEIRO] =
RECEBENDO_DINHEIRO;
  acao_matrizTransicaoEstados[IDLE_CLIENTE][INSERE_DINHEIRO] = A01;

  proximo_estado_matrizTransicaoEstados[RECEBENDO_DINHEIRO][INSERE_DINHEIRO] =
RECEBENDO_DINHEIRO;
  acao_matrizTransicaoEstados[RECEBENDO_DINHEIRO][INSERE_DINHEIRO] = A01;

  proximo_estado_matrizTransicaoEstados[RECEBENDO_DINHEIRO][SELECIONA_PRODUTO] =
VERIFICA_TROCO;
  acao_matrizTransicaoEstados[RECEBENDO_DINHEIRO][SELECIONA_PRODUTO] = A02;

  proximo_estado_matrizTransicaoEstados[VERIFICA_TROCO][TROCO_DISPONIVEL] =
VERIFICA_DISPONIBILIDADE_PRODUTO;
  acao_matrizTransicaoEstados[VERIFICA_TROCO][TROCO_DISPONIVEL] = A03;


proximo_estado_matrizTransicaoEstados[VERIFICA_DISPONIBILIDADE_PRODUTO][PRODUTO_DISPONIV
EL] = LIBERA_PRODUTO;
  acao_matrizTransicaoEstados[VERIFICA_DISPONIBILIDADE_PRODUTO][PRODUTO_DISPONIVEL] =
A04;

  proximo_estado_matrizTransicaoEstados[LIBERA_PRODUTO][PRODUTO_LIBERADO] =
LIBERA_TROCO;
  acao_matrizTransicaoEstados[LIBERA_PRODUTO][PRODUTO_LIBERADO] = A05;

  proximo_estado_matrizTransicaoEstados[LIBERA_TROCO][TROCO_LIBERADO] = IDLE_CLIENTE;
  acao_matrizTransicaoEstados[LIBERA_TROCO][TROCO_LIBERADO] = NENHUMA_ACAO;

  proximo_estado_matrizTransicaoEstados[RECEBENDO_DINHEIRO][CANCELA] = DEVOLVE_DINHEIRO;
  acao_matrizTransicaoEstados[RECEBENDO_DINHEIRO][CANCELA] = A06;

  proximo_estado_matrizTransicaoEstados[VERIFICA_TROCO][TROCO_INDISPONIVEL] =
DEVOLVE_DINHEIRO;
  acao_matrizTransicaoEstados[VERIFICA_TROCO][TROCO_INDISPONIVEL] = NENHUMA_ACAO;


proximo_estado_matrizTransicaoEstados[VERIFICA_DISPONIBILIDADE_PRODUTO][PRODUTO_INDISPON
IVEL] = DEVOLVE_DINHEIRO;
  acao_matrizTransicaoEstados[VERIFICA_DISPONIBILIDADE_PRODUTO][PRODUTO_INDISPONIVEL] =
NENHUMA_ACAO;

  proximo_estado_matrizTransicaoEstados[IDLE_CLIENTE][ACIONA_TRAVA_MANUTENCAO] =
IDLE_MANUTENCAO;
  acao_matrizTransicaoEstados[IDLE_CLIENTE][ACIONA_TRAVA_MANUTENCAO] = NENHUMA_ACAO;
```

```
  proximo_estado_matrizTransicaoEstados[IDLE_MANUTENCAO][DESACIONA_TRAVA_MANUTENCAO] =
IDLE_CLIENTE;
  acao_matrizTransicaoEstados[IDLE_MANUTENCAO][DESACIONA_TRAVA_MANUTENCAO] =
NENHUMA_ACAO;

  proximo_estado_matrizTransicaoEstados[IDLE_MANUTENCAO][PRESSIONA_1] =
ADICIONANDO_REMOVENDO_DINHEIRO;
  acao_matrizTransicaoEstados[IDLE_MANUTENCAO][PRESSIONA_1] = NENHUMA_ACAO;

  proximo_estado_matrizTransicaoEstados[ADICIONANDO_REMOVENDO_DINHEIRO][PRESSIONA_2] =
IDLE_MANUTENCAO;
  acao_matrizTransicaoEstados[ADICIONANDO_REMOVENDO_DINHEIRO][PRESSIONA_2] =
NENHUMA_ACAO;

  proximo_estado_matrizTransicaoEstados[IDLE_MANUTENCAO][PRESSIONA_3] =
ADICIONANDO_REMOVENDO_PRODUTO;
  acao_matrizTransicaoEstados[IDLE_MANUTENCAO][PRESSIONA_3] = NENHUMA_ACAO;

  proximo_estado_matrizTransicaoEstados[ADICIONANDO_REMOVENDO_PRODUTO][PRESSIONA_4] =
IDLE_MANUTENCAO;
  acao_matrizTransicaoEstados[IDLE_MANUTENCAO][PRESSIONA_4] = NENHUMA_ACAO;

} // initStateMachine

/*************************************************************************
 iniciaSistema
 Inicia o sistema ...
 Parametros de entrada: nenhum
 Retorno: nenhum
*************************************************************************/
void iniciaSistema()
{
   iniciaMaquinaEstados();
} // initSystem

/*************************************************************************
 obterEvento
 Obtem um evento, que pode ser da IHM ou do alarme
 Parametros de entrada: nenhum
 Retorno: codigo do evento
*************************************************************************/

int obterEvento(int estado)
{
  int retval = NENHUM_EVENTO;

  teclas = tcp_obterTeclas();

  if (teclas[0] == 'i' && (estado == 0 || estado == 1))
       return INSERE_DINHEIRO;
  if (teclas[0] == 's' && (estado == 1))
       return SELECIONA_PRODUTO;
  if (teclas[0] == 'c' && (estado == 1))
```

```
        return CANCELA;
  if (teclas[0] == 'm' && (estado == 0))
        return ACIONA_TRAVA_MANUTENCAO;
  if (teclas[0] == 'n' && (estado == 7))
        return DESACIONA_TRAVA_MANUTENCAO;

  return retval;

} // obterEvento

/*****************************************************************************
 obterAcao
 Obtem uma acao da Matriz de transicao de estados
 Parametros de entrada: estado (int)
                        evento (int)
 Retorno: codigo da acao
*****************************************************************************/
int obterAcao(int estado, int codigoEvento) {
  return acao_matrizTransicaoEstados[estado][codigoEvento];
} // obterAcao


/*****************************************************************************
 obterProximoEstado
 Obtem o proximo estado da Matriz de transicao de estados
 Parametros de entrada: estado (int)
                        evento (int)
 Retorno: codigo do estado
*****************************************************************************/
int obterProximoEstado(int estado, int codigoEvento) {
  return proximo_estado_matrizTransicaoEstados[estado][codigoEvento];
} // obterAcao


/*****************************************************************************
 Main
 Loop principal de controle que executa a maquina de estados
 Parametros de entrada: nenhum
 Retorno: nenhum
*****************************************************************************/
int main() {

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

  estado = IDLE_CLIENTE;
  eventoInterno = NENHUM_EVENTO;

  iniciaSistema();
  printf ("Vending Machine iniciada\n");
  printf ("Para inserir dinheiro, tecle i e valor a inserir em moedas\n");
```

```
  printf ("Para cancelar, tecle c\n");
  printf ("Para selecionar o produto, tecle s e número do produto");
  printf("Produtos:\n1 - Agua  R$2,00\n2 - Refri R$3,50\n3 - Suco  R$3,00\n4 - Lala
R$40,00\n");

  while (true) {
        if (eventoInterno == NENHUM_EVENTO) {
        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  Dinheiro inserido: %d centavos\n",
estado, codigoEvento, codigoAcao, dinheiro_buffer);
        }
  } // while true
} // main
```

cofre.c

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

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

int dinheiro_armazenado;

void cof_armazena_valor (int val) {
        dinheiro_armazenado += val;
        dpl_msg("Compra confirmada.\n");
}

int cof_verifica_troco (int troco) {
        if (dinheiro_armazenado >= troco) return true;
        else {
        dpl_msg("Não temos troco no momento.");
        return false;
        }
}

void cof_remove_troco (int troco) {
        dinheiro_armazenado -= troco;
}
```

coletor_dinheiro.c

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

#include "definicoes_sistema.h"
#include "coletor_dinheiro.h"

int dinheiro_buffer;  //dinheiro no buffer de entrada

void col_add_buffer(int val) {
	dinheiro_buffer += val;
	dpl_msg("Dinheiro adicionado.\n");
}

void col_zera_buffer() {
	dinheiro_buffer = 0;
	dpl_msg("Retire seu troco.\n");
}

void col_cancelar() {
	dinheiro_buffer = 0;
	dpl_msg("Compra cancelada.\n");
	dpl_msg("Retire seu dinheiro.\n");
}
```

descarga_produto.c

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

#include "definicoes_sistema.h"
#include "descarga_produto.h"

int produtos_estoque[4];

int dsp_verifica_produto (int num) {
	if (produtos_estoque[num - 1] > 0) {
	dpl_msg("Produto disponivel\n");
	return true;
	}
	else {
	dpl_msg("Produto em falta\n");
	return false;
	}
}

void dsp_descarregar_produto(int num) {
	produtos_estoque[num - 1] --;
	dpl_msg("Produto descarregado.\n");
}
```

display.c

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

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

void dpl_msg(char *s) {
	printf("%s", s);
}
```

teclado_produtos.c

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

#include "definicoes_sistema.h"
#include "teclado_produtos.h"

char buf[10];
char* tcp_obterTeclas()
{
	//Produtos aceitos: 1, 2, 3, 4
	printf("obter teclas:");
	scanf("%s", buf);
	return buf;
}
```

Headers

definicoes_sistema.h

```c
#ifndef DEFINICOES_SISTEMA_H_INCLUDED
#define DEFINICOES_SISTEMA_H_INCLUDED

#define true  1
#define false 0

#define NUM_ESTADOS 10
#define NUM_EVENTOS 16

// ESTADOS
#define IDLE_CLIENTE                      0
#define RECEBENDO_DINHEIRO                1
#define VERIFICA_TROCO                    2
#define VERIFICA_DISPONIBILIDADE_PRODUTO  3
#define LIBERA_PRODUTO                    4
#define LIBERA_TROCO                      5
#define DEVOLVE_DINHEIRO                  6
#define IDLE_MANUTENCAO                   7
```

```
#define ADICIONANDO_REMOVENDO_DINHEIRO      8
#define ADICIONANDO_REMOVENDO_PRODUTO       9

// EVENTOS
#define NENHUM_EVENTO               -1
#define INSERE_DINHEIRO             0
#define SELECIONA_PRODUTO           1
#define TROCO_DISPONIVEL            2
#define PRODUTO_DISPONIVEL          3
#define PRODUTO_LIBERADO            4
#define TROCO_LIBERADO              5
#define CANCELA                     6
#define TROCO_INDISPONIVEL          7
#define PRODUTO_INDISPONIVEL        8
#define DINHEIRO_LIBERADO           9
#define ACIONA_TRAVA_MANUTENCAO     10
#define DESACIONA_TRAVA_MANUTENCAO 11
#define PRESSIONA_1                 12
#define PRESSIONA_2                 13
#define PRESSIONA_3                 14
#define PRESSIONA_4                 15

// ACOES
#define NENHUMA_ACAO -1
#define A01  0 //incrementar buffer de dinheiro
#define A02  1 //verificar se há troco
#define A03  2 //verificar se há produto
#define A04  3 //liberar produto
#define A05  4 //liberar troco
#define A06  5 //devolver dinheiro


#endif // DEFINICOES_SISTEMA_H_INCLUDED
```

cofre.h

```
#ifndef COFRE_H_INCLUDED
#define COFRE_H_INCLUDED

void cof_armazena_valor (int val);

int cof_verifica_troco (int troco);

void cof_remove_troco (int troco);

#endif // COFRE_H_INCLUDED
```

coletor_dinheiro.h

```
#ifndef COLETOR_DINHEIRO_H_INCLUDED
#define COLETOR_DINHEIRO_H_INCLUDED
```

```
int dinheiro_buffer;  //dinheiro no buffer de entrada, em centavos

void col_add_buffer();

void col_zera_buffer();

void col_cancelar();

#endif // COLETOR_DINHEIRO_H_INCLUDED
```

descarga_produto.h

```
#ifndef DESCARGA_PRODUTO_H_INCLUDED
#define DESCARGA_PRODUTO_H_INCLUDED

int dsp_verifica_produto (int num);

void dsp_descarregar_produto(int num);

#endif // DESCARGA_PRODUTO_H_INCLUDED
```

display.h

```
#ifndef DESCARGA_PRODUTO_H_INCLUDED
#define DESCARGA_PRODUTO_H_INCLUDED

int dsp_verifica_produto (int num);

void dsp_descarregar_produto(int num);

#endif // DESCARGA_PRODUTO_H_INCLUDED
```

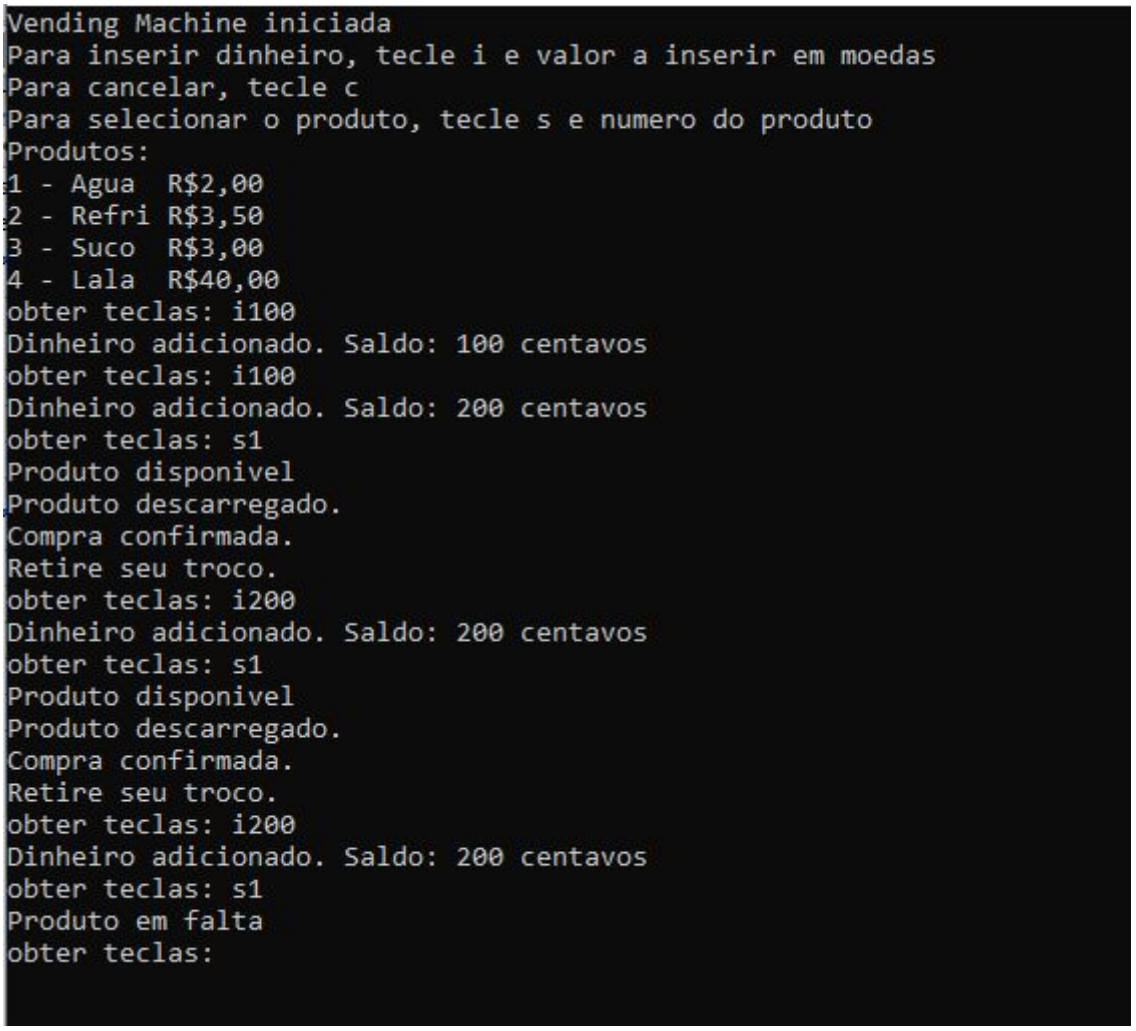
teclado_produtos.h

```
#ifndef TECLADO_PRODUTOS_H_INCLUDED
#define TECLADO_PRODUTOS_H_INCLUDED

extern char* tcp_obterTeclas();

#endif // TECLADO_PRODUTOS_H_INCLUDED
```

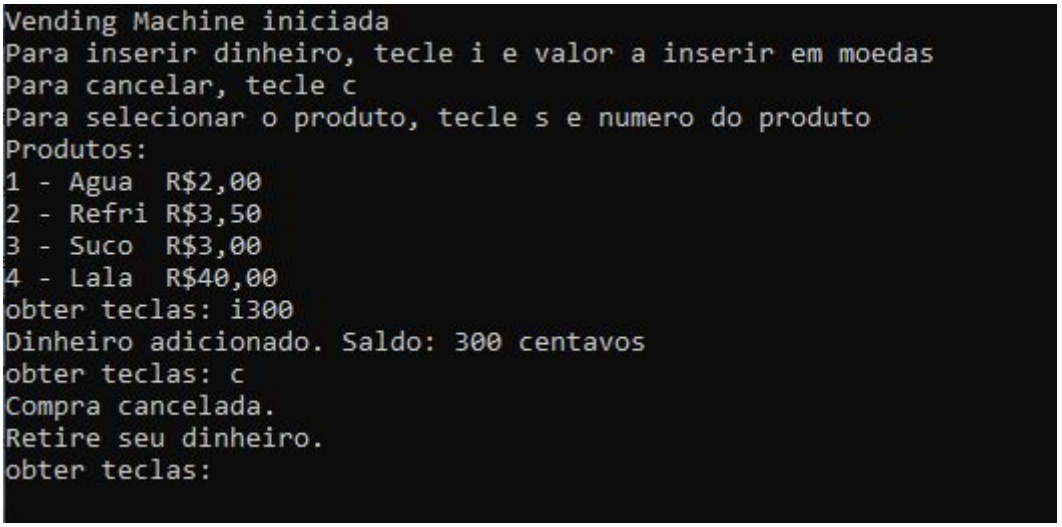
Telas de execução

Fluxo de compra até acabar o estoque de um item

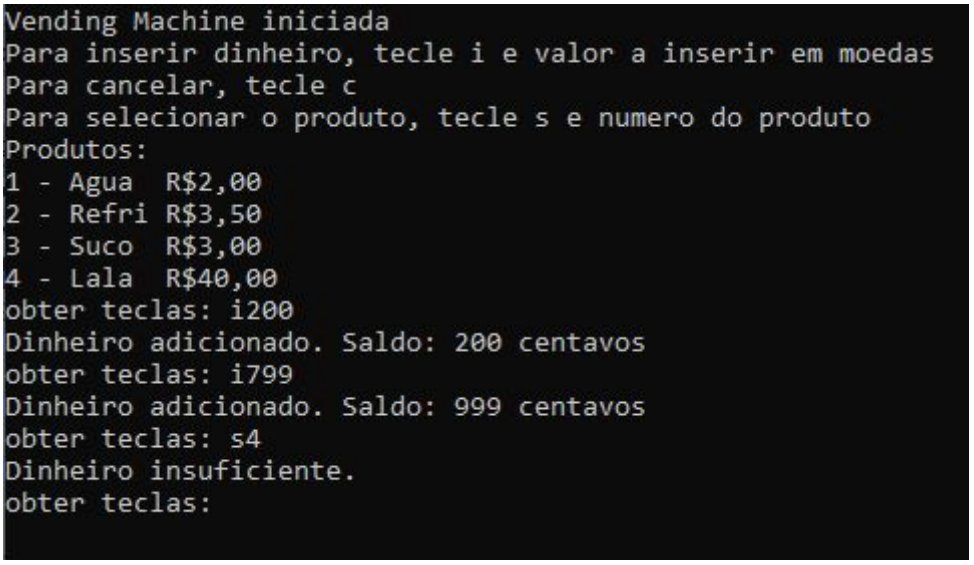
Existiam duas águas (código 1) em estoque. Aqui tento comprar três, e recebo uma mensagem de falta de estoque.



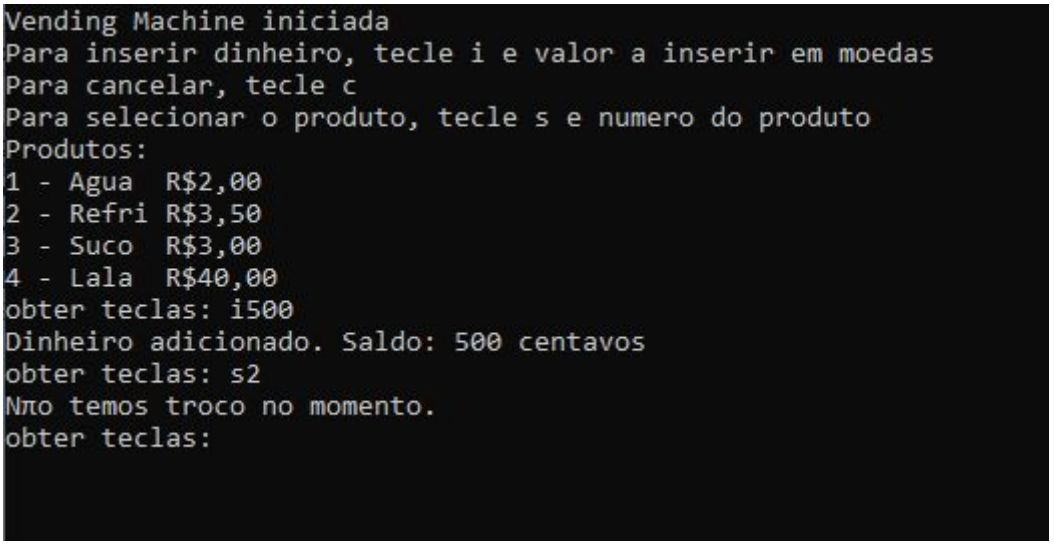
Fluxo de inserção de dinheiro e cancelamento



Fluxo de dinheiro insuficiente



Troco insuficiente na máquina



Comentários sobre o projeto

Vamos admitir dois fatos:

1. **Alunos sempre tentam maximizar a nota, minimizando o esforço.**
2. **Cada um desses relatórios vai acabar com aproximadamente 30 páginas.**

Considerando que a sala tem 60 alunos, vemos claramente que o professor não vai ter capacidade de realmente avaliar a qualidade desses relatórios. Sabendo que nós alunos não vamos ganhar nota por qualidade, tentamos ganhar nota por quantidade (tamanho do trabalho), o que aumenta a carga de correção pro professor.

Nós alunos não temos incentivo de fazer um trabalho de qualidade, e isso é completamente culpa do professor, que estipulou uma entrega enorme do trabalho de meio semestre.

Entregas menores e mais distribuídas são muito mais motivadoras e nos dão mais segurança de que o professor vai ter tempo de avaliar a qualidade dos trabalhos.