UNIVERSIDADE DE SÃO PAULO (USP)
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO (ICMC)
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO (SCC)

Isabella Arão - 9265732
Marina Fagundes - 9265405
Robson Silva - 7233062

Entrega 1

São Carlos
2023

Isabella Arão - 9265732
Marina Fagundes - 9265405
Robson Silva - 7233062

Entrega 1

Relatório apresentado à disciplina Laboratório de Introdução à Ciência de Computação I, como parte dos requisitos para aprovação na matéria oferecida pela Universidade de São Paulo, na área de Computação.

Prof.: Danilo Hernane Spatti
Disciplina: Laboratório de Introdução à Ciência de Computação I
Turma: Turma SCC 0502

São Carlos
2023

INTRODUÇÃO

O presente trabalho tem como objetivo apresentar a primeira etapa do Projeto Sistema de Gestão dos Usuários do *Uaibank*. O Projeto será realizado em duas partes: Modelagem do Problema e Implementação.

Para esta primeira etapa, Modelagem do Problema, será apresentado um documento contendo um fluxograma em que ilustra as sequências do desenvolvimento geral do projeto, além dos pseudocódigos apontando as funcionalidades do Sistema. O uso do fluxograma e do pseudocódigo deverão, de antemão, mostrar a lógica, por meio de algoritmos, das soluções pensadas para este problema.

PSEUDOCÓDIGO

ALGORITMO_CADASTRO

INÍCIO

DECLARE int opcao, quant, quant_total, quant1, i, id;

//será feita alocação dinâmica para os vetores a seguir, uma vez que não existe tamanho definido para eles:

DECLARE string nome[], nome_correto[];

DECLARE int idade[], idade_correta[], id[];

DECLARE real saldo[], saldo_correto[];

quant1 ← 0;

quant ← 0;

quant_total ← quant + quant1;

id ← 0;

MOSTRE (“1. Inserção de um novo usuário

2. Inserção de vários usuários

3. Busca de um usuário por id

4. Transferência entre usuários

5. Remoção de um usuário por id”);

Leia opcao;

SE opcao == 1 ENTÃO:

quant ← quant + 1;

quant_total ← quant + quant1;

```
PARA i ← (quant_total - quant) ATÉ (quant_total - 1) FAÇA:
        id ← id + 1;
        id[i] ← id;
        LEIA nome[i];
        ENQUANTO número de caracteres de nome [i] >
        100 FAÇA
        // ainda aprenderemos como implementar o comando acima.
                MOSTRE (“Erro! O nome deve conter menos de 100
                caracteres. Escreva novamente”);
                LEIA nome[i];
        FIMENQUANTO
        nome_correto[i] = nome [i];
FIMPARA
PARA i ← (quant_total - quant) ATÉ (quant_total - 1) FAÇA:
        LEIA idade[i];
        ENQUANTO idade[i] < 0 FAÇA
                MOSTRE (“Erro! A idade não pode ser negativa.
                Escreva novamente”);
                LEIA idade[i];
        FIMENQUANTO
        idade_correta[i] = idade[i];
FIMPARA
PARA i ← (quant_total - quant) ATÉ (quant_total - 1) FAÇA:
        LEIA saldo[i];
```

```
            ENQUANTO saldo[i] < 0 FAÇA
                MOSTRE (“Erro! O saldo não pode ser negativo.
                Escreva novamente”);
                LEIA saldo[i];
            FIMENQUANTO
            saldo_correto[i] = saldo[i];
        FIMPARA
        PARA i ← (quant_total - quant) ATÉ (quant_total - 1) FAÇA:
            MOSTRE (“Usuário inserido com id”, id[i]);
        FIMPARA
FIMSE
```

SE opcao == 2 ENTÃO:

```
        LEIA quant1;
        quant_total ← quant + quant1;
        PARA i ← quant ATÉ (quant_total - 1) FAÇA:
            id ← id + 1;
            id[i] ← id;
            LEIA nome[i];
            ENQUANTO número de caracteres de nome [i] > 100 FAÇA
                MOSTRE (“Erro! O nome deve conter menos de 100
                caracteres. Escreva novamente”);
                LEIA nome[i];
            FIMENQUANTO
```

```
                nome_correto[i] = nome [i];
                PARA i ← quant ATÉ (quant_total - 1) FAÇA:
                        LEIA idade[i];
                        ENQUANTO idade[i] < 0 FAÇA
                                MOSTRE (“Erro! A idade não pode ser
negativa.
                                Escreva novamente”);
                                LEIA idade[i];
                        FIMENQUANTO
                        idade_correta[i] = idade[i];
                FIMPARA
                PARA i ← quant ATÉ (quant_total - 1) FAÇA:
                        LEIA saldo[i];
                        ENQUANTO saldo[i] < 0 FAÇA
                                MOSTRE (“Erro! O saldo não pode ser
                                negativo. Escreva novamente”);
                                LEIA saldo[i];
                        FIMENQUANTO
                        saldo_correto[i] = saldo[i];
                FIMPARA
                PARA i ← quant ATÉ (quant_total - 1) FAÇA:
                        MOSTRE (“Usuário inserido com id”, id[i]);
                FIMPARA
        FIMPARA
```

```
FIMSE

SE opcao == 3 ENTÃO:
    DECLARE int id_checagem;
    LEIA id_checagem;
    SE id[id_checagem - 1] != 0;
        MOSTRE (“Usuário”,  id[id_checagem - 1], “tem saldo
        de R$”, saldo_correto[id_checagem - 1]);
    SENÃO
        MOSTRE (“Erro: Usuário”, id_checagem, “não encontrado”);
    FIMSE
FIMSE

SE opcao == 4 ENTÃO:
    DECLARE int id_origem, id_destino;
    DECLARE real transferencia;
    LEIA id_origem, id_destino, transferencia;
    ENQUANTO transferencia <=0 FAÇA
        MOSTRE (“Erro! Não é possível realizar
        essa transferência!”);
        LEIA transferencia;
    FIMENQUANTO
    SE  id_origem ==  id_destino ENTÃO
```

```
        MOSTRE (“Erro! Usuário de destino tem que ser diferente  do
        usuário de origem”);
        SENÃO SE  id_origem == 0 ||  id_destino == 0 ENTÃO
                MOSTRE (“Erro! Usuário não existe!”);
SENÃO
        PARA i ← 0 ATÉ (quant_total - 1) FAÇA:
                SE id[i] == id_origem;
                        SE saldo_correto[i] < transferencia ENTÃO
                                MOSTRE (“Erro! Não é possível realizar
                                essa transferência!”);
                        SENÃO
                                saldo_correto[i] = saldo_correto[i] -
                                transferencia;
                                MOSTRE (“Transferência realizada. O
                                saldo atual do usuário”, id_origem, “é
                                R$”, saldo_correto[i]);
                        FIMSE
                FIMSE
                SE id[i] == id_destino;
                        saldo_correto[i] = saldo_correto[i] +
                        transferencia;
                        MOSTRE (“O saldo do usuário”, id_destino, “é
                        R$”, saldo_correto[i]);
                FIMSE
```

FIMPARA

FIMSE

FIMSE

SE opcao == 5 ENTÃO:

DECLARE int id_para_apagar;

LEIA id_para_apagar;

SE id[id_para_apagar -1] == 0;

MOSTRE (“Erro! Usuário”, id_para_apagar, “não existe!”);

SENÃO

id[id_para_apagar - 1] = 0;

MOSTRE (“Usuário”, id_para_apagar, “removido com sucesso”);

FIMSE

SENÃO

MOSTRE (“Erro! Insira a opção desejada novamente.”);

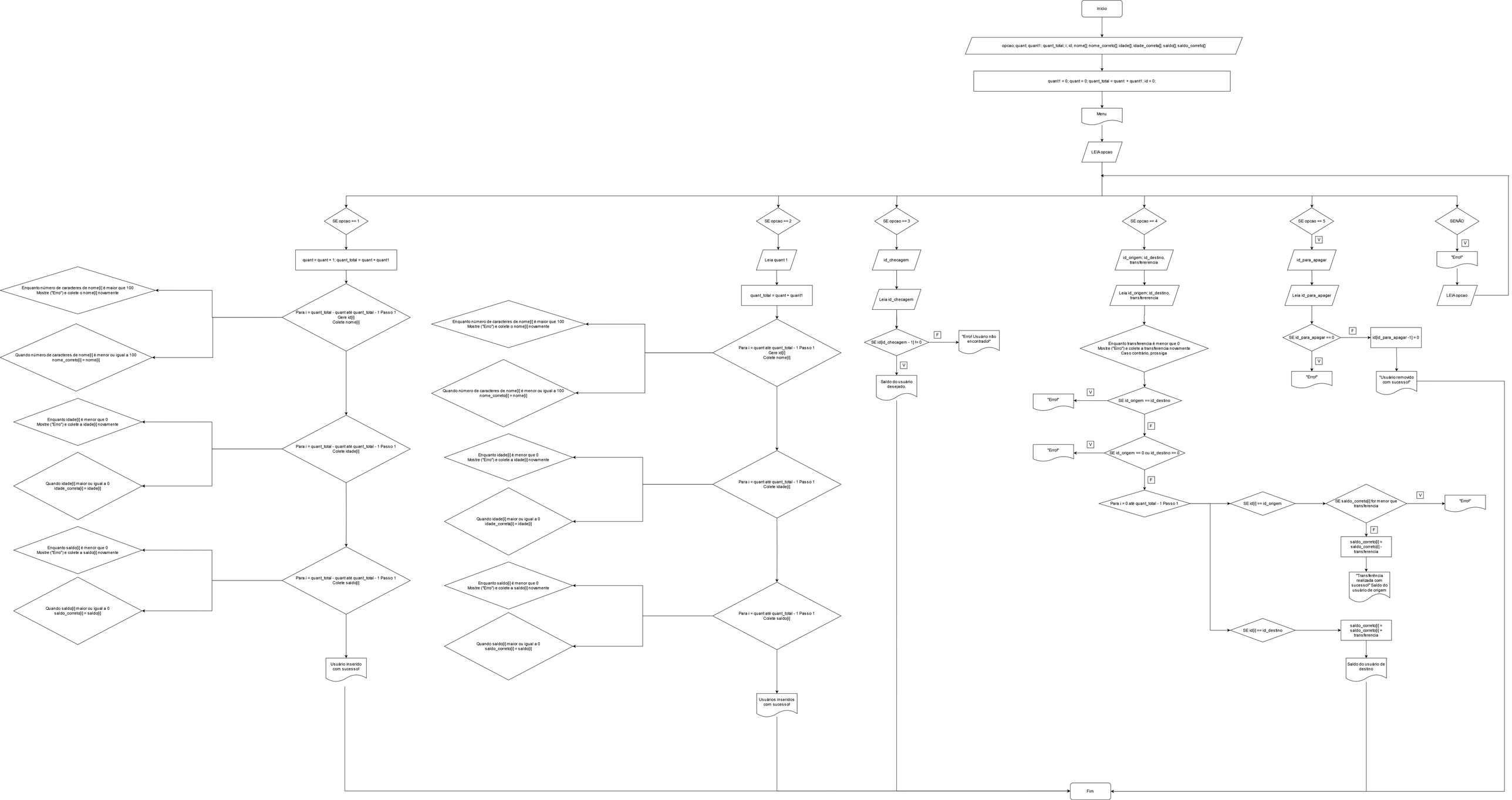
LEIA opcao;

FIMSE

FIM

FLUXOGRAMA

O fluxograma do projeto se encontra a seguir.


Início
opcao; quant; quant1; quant_total; i; id; nome[]; nome_correto[]; idade[]; idade_correta[]; saldo[]; saldo_correto[]
quant1 = 0; quant = 0; quant_total = quant + quant1; id = 0;
Menu
LEIA opcao
SE opcao == 1
quant = quant + 1; quant_total = quant + quant1
Para i = quant_total - quant até quant_total - 1 Passo 1
Gere id[i]
Colete nome[i]
Enquanto número de caracteres de nome[i] é maior que 100
Mostre ("Erro") e colete o nome[i] novamente
Quando número de caracteres de nome[i] é menor ou igual a 100
nome_correto[i] = nome[i]
Para i = quant_total - quant até quant_total - 1 Passo 1
Colete idade[i]
Enquanto idade[i] é menor que 0
Mostre ("Erro") e colete a idade[i] novamente
Quando idade[i] maior ou igual a 0
idade_correta[i] = idade[i]
Para i = quant_total - quant até quant_total - 1 Passo 1
Colete saldo[i]
Enquanto saldo[i] é menor que 0
Mostre ("Erro") e colete a saldo[i] novamente
Quando saldo[i] maior ou igual a 0
saldo_correto[i] = saldo[i]
Usuário inserido com sucesso!
SE opcao == 2
Leia quant 1
quant_total = quant + quant1
Para i = quant até quant_total - 1 Passo 1
Gere id[i]
Colete nome[i]
Enquanto número de caracteres de nome[i] é maior que 100
Mostre ("Erro") e colete o nome[i] novamente
Quando número de caracteres de nome[i] é menor ou igual a 100
nome_correto[i] = nome[i]
Para i = quant até quant_total - 1 Passo 1
Colete idade[i]
Enquanto idade[i] é menor que 0
Mostre ("Erro") e colete a idade[i] novamente
Quando idade[i] maior ou igual a 0
idade_correta[i] = idade[i]
Para i = quant até quant_total - 1 Passo 1
Colete saldo[i]
Enquanto saldo[i] é menor que 0
Mostre ("Erro") e colete a saldo[i] novamente
Quando saldo[i] maior ou igual a 0
saldo_correto[i] = saldo[i]
Usuários inseridos com sucesso!
SE opcao == 3
id_checagem
Leia id_checagem
SE id[id_checagem - 1] != 0
F
"Erro! Usuário não encontrado!"
V
Saldo do usuário desejado.
SE opcao == 4
id_origem; id_destino, transferencia
Leia id_origem; id_destino, transferencia
Enquanto transferencia é menor que 0
Mostre ("Erro") e colete a transferencia novamente
Caso contrário, prossiga
SE id_origem == id_destino
V
"Erro!"
F
SE id_origem == 0 ou id_destino == 0
V
"Erro!"
F
Para i = 0 até quant_total - 1 Passo 1
SE id[i] == id_origem
SE saldo_correto[i] for menor que transferencia
V
"Erro!"
F
saldo_correto[i] = saldo_correto[i] - transferencia
"Transferência realizada com sucesso!" Saldo do usuário de origem
SE id[i] == id_destino
saldo_correto[i] = saldo_correto[i] + transferencia
Saldo do usuário de destino
SE opcao == 5
V
id_para_apagar
Leia id_para_apagar
SE id_para_apagar == 0
F
id[id_para_apagar -1] = 0
V
"Erro!"
"Usuário removido com sucesso!"
SENÃO
V
"Erro!"
LEIA opcao
Fim