Relatório dos Trabalhos Práticos AEDS III

Barbara L. Araújo[1], **Luisa N. C. S. Souza**[2]

[1]

Abstract. *The purpose of this article is to explain the process of creating a code to register a bank account, based on the development of the CRUD (Create, Read, Update and Delete) responsible for creating the accounts, keeping them updated and, when necessary, deleting, using the method of interleaving the accounts, of the index file by inverted list, of the compression and decompression methods of these files, being them LZW and Hufmman, of the standard matching by KMP and finally of the encryption, by OTP (One Time Pad) , used in the password, the in order to document the code creation process throughout the second half of 2022.*

Resumo. *O artigo tem como propósito explicar o processo de criação de um código para cadastro de uma conta bancária, a partir do desenvolvimento do CRUD (Create, Read, Update e Delete) responsável por criar as contas, manter atualizadas e quando necessario deletar, do metodo de intercalação das contas, do arquivo de indices por lista invertida, dos metodos de compressão e descompressão desses arquivos, sendo eles LZW e Huffman, do casamento de padrões por KMP e por ultimo da criptografia, por OTP (One Time Pad), utilizada na senha, a fim de documentar o processo de criação do código ao longo do segundo semestre de 2022.*

1. Introdução

Um banco precisa de algumas informações do cliente para criar, armazenar, ler, atualizar e deletar uma conta. Sendo assim, o trabalho apresentado desempenha o papel de realizar essas operações com os dados dos clientes, através do CRUD. Para tal, foi utilizada a linguagem de programação JAVA e os dados são armazenados em um documento de texto em formato de hexadecimal. Além disso, foram também apresentados métodos de intercalação das contas e lista invertida. Ademais, para a redução do tamanho do arquivo armazenado, foram trabalhados dois algoritmos de compressão de dados: LZW, com a utilização de um dicionário e Huffman, com a implementação de uma árvore. Com o intuito de identificar, substituir ou criar padrões, o algoritmo de casamento de padrões de KMP foi usado. E por fim, para uma maior segurança na senha criada pelo cliente, a utilização de um sistema de criptografia se mostrou bastante necessária, sendo implementado o método de OTP (One Time Pad).

2. Desenvolvimento

2.1. CRUD

2.1.1. CREATE

O método de CREATE é utilizado para criar uma conta nova no banco de dados do banco. Dessa maneira, ele insere os dados inseridos pelo usuário no arquivo de armazenamento de dados. (FIGURE 1)

2.1.2. READ

O método de READ é utilizado para ler uma conta do banco. Dessa maneira, ele faz uma pesquisa pelo ID da conta a ser lida no arquivo onde os dados estão armazenados e imprime na tela do usuário os dados da determinada conta. (FIGURE 2)

2.1.3. UPDATE

O método de UPDATE é utilizado para atualizar uma conta do banco. Dessa maneira, ele faz uma pesquisa pelo ID da conta a ser atualizada e substitui no arquivo os dados da conta pelas novas informações estabelecidas pelo usuário. (FIGURE 3)

2.1.4. DELETE

O método de DELETE é utilizado para deletar uma conta do banco. Dessa maneira, ele faz uma pesquisa pelo ID da conta a ser deletada e apaga no arquivo os dados da conta, ativando uma lápide (demonstra que uma informação foi excluída do arquivo). (FIGURE 4)

```java
public static boolean create(RandomAccessFile arq, Conta conta) throws Exception {
    boolean resp = false;

    try {
        arq.seek(arq.length()); // Ponteiro vai para o final do arquivo
        arq.writeByte(v: 0); // Lápide
        arq.writeInt(conta.toByteArray().length);// Escreve os dados em bytes

        arq.writeInt(conta.getIdconta());
        arq.writeUTF(conta.getNomePessoa());
        arq.writeInt(conta.getQtd());

        // --- escreve todos os emails ---
        for (int i = 0; i < conta.getQtd(); i++) {
            arq.writeUTF(conta.getEmail()[i]);
        }

        arq.writeUTF(conta.getNomeUsuario());

        key = getChave(conta.getSenha());
        String password = OTP.cifrar(conta.getSenha(), key);
        arq.writeUTF(password);

        arq.writeUTF(conta.getCpf());
        arq.writeUTF(conta.getCidade());
        arq.writeInt(conta.getTransferenciasRealizadas());
        arq.writeFloat(conta.getSaldoConta());

        System.out.println("Senha criptografada: " + password);
        System.out.println("Chave: " + key);

        arq.seek(pos: 0); // Ponteiro vai para o início do arquivo
        resp = true;
    } catch (Exception e) {
        throw new Exception(message: "Erro ao inserir registro");
    }

    return resp;
}
```

(a) FIGURE 1. Código do método de CREATE.

```java
public static Conta readId(RandomAccessFile arq, int pesquisa) throws Exception {

    try {
        Conta conta = new Conta();
        arq.seek(pos: 4); // Ponteiro no inicio do arquivo

        while (arq.getFilePointer() < arq.length()) { // Até o ponteiro chegar ao final do arquivo
            if (arq.readByte() == 0) {
                int tam = arq.readInt();

                conta.setIdConta(arq.readInt());

                if (conta.getIdconta() == pesquisa) { // Caso o id da conta seja igual ao id procurado
                    conta.setNomePessoa(arq.readUTF());

                    String[] emails = new String[1000]; // armazenar os emails

                    // --- ler todos os emails ---
                    for (int i = 0; i < conta.getQtd(); i++) {
                        emails[i] = arq.readUTF();
                    }

                    conta.setEmail(emails);// adiciona os emails

                    conta.setNomeUsuario(arq.readUTF());

                    conta.setSenha(OTP.decifrar(arq.readUTF(), key));

                    conta.setCpf(arq.readUTF());
                    conta.setCidade(arq.readUTF());
                    conta.setTransferenciasRealizadas(arq.readInt());
                    conta.setSaldoConta(arq.readFloat());

                    return conta;
                } else {
                    arq.skipBytes(tam - 4); // Pula o restante do registro
                }
            } else {
                arq.skipBytes(arq.readInt()); // Pula o registro todo
            }
        }

        return null;

    } catch (Exception e) {
        System.out.println(x: "Nao foi possível ler o registro!");
        return null;
    }
}
```

(b) FIGURE 2. Código do método de READ.

2.2. Intercalação

O objetivo do uso da intercalação é de gerar segmentos ordenados maiores na fase de distribuição, requerindo o uso de uma fila de prioridades, como um heap. A classe da intercalação abaixo demonstra como é feito esse processo.(FIGURE 5)

2.3. Lista Invertida

O Índice Invertida tem como propósito buscar texto em arquivos, como ocorre nas maquinas de busca na web. Para cada indice invertido é criado uma lista invertida. Na

(c) FIGURE 3. Código do método de UPDATE.

(d) FIGURE 4. Código do método de DELETE.

Figure 1. FIGURE 5. Código da Terceira Intercalação no método de Intercalar.

lista são retirados os termos não significativos e o conjunto é ordenado. As listas invertidas são adequadas para consultas combinadas, possuem implementação relativamente fácil e é a estrutura de dados mais popular usada em sistemas para recuperação de documentos. (FIGURE 6)

2.4. Compressão de Dados

2.4.1. LZW

O metodo de dicionário do LZW equivale a utilizar um dicionário modificado que foi inicializado com todos os símbolos do alfabeto. A classe com o código de LZW consiste na compressão e descompressão dos dados de uma conta bancária fornecida peo usuário. A compressão começa com iniciazação do dicionário (foi utilizado a função Map e Hash Map) e dentro de loop, que repete até o fim do texto, deve-se achar a maior string W

```
public static boolean listaInvertida(RandomAccessFile arq, String name) throws Exception {
    Conta conta = new Conta(); // cria uma nova conta
    RandomAccessFile list = new RandomAccessFile("listName", "rw");
    if (list.length() != 0) { // verifica se o arquivo ta vazio
        list.setLength(0); // zera o arquivo
    }

    System.out.println("\nNome: " + name);
    list.writeUTF(name); // Escreve o nome no arquivo

    arq.seek(4); // Ponteiro no inicio do arquivo
    int cont = 0; // Conta os registros
    while (arq.length() != arq.getFilePointer()) { // Enquanto nao for igual ao final do arquivo
        double ponteiro = arq.getFilePointer(); // Pega o atual ponteiro
        if (arq.readByte() == 0) {
            arq.readInt(); // tamanho
            int id = arq.readInt(); // id
            String n = arq.readUTF(); // nome

            String[] emails = new String[1000]; // armazenar os emails
            for (int i = 0; i < conta.getQtd(); i++) {
                emails[i] = arq.readUTF(); // email
            }

            arq.readUTF(); // user
            arq.readUTF(); // senha
            arq.readUTF(); // cpf
            arq.readUTF(); // cidade
            arq.readInt(); // transacoes
            arq.readFloat(); // saldo
            if (n.equals(name)) { // Se o nome for igual ao passado por parametro
                System.out.println("ID: " + id + "- Posicao: " + (int) ponteiro); // imprime a posicoa e o id do
                                                                                  // registro
                list.writeInt(id); // Escreve o id no arquivo
                list.writeInt((int) ponteiro); // Escreve a posicao do registro no arq da lista
                cont++;
            }
        } else { // Caso o registro esteja inativo
            arq.skipBytes(arq.readInt()); // Pula o registro
        }
    }

    System.out.println("Quantidade de registros: " + cont);
    list.writeInt(cont);// Escreve a qntd no arquivo

    list.close();

    return true;
```

Figure 2. FIGURE 6. Código do método de lista invertida.

existente no dicionário, escrever o indice de W na saída, olhar o próximo caráter A que não fez parte de W, escrever WA no dicionário caso W e A já façam parte do dicionário e por último avançar para a posição de A, montando assim a tabela do dicionário. (FIGURE 7)

```
/** Compress a string to a list of output symbols. */
public static List<Integer> compressao(String text) {
    // Dicionario
    int dicioTam = 256;
    Map<String,Integer> dicionario = new HashMap<String,Integer>();
    for (int i = 0; i < 256; i++)
        dicionario.put("" + (char)i, i);

    String w = "";
    List<Integer> resp = new ArrayList<Integer>();
    for (char c : text.toCharArray()) {
        String wc = w + c;
        if (dicionario.containsKey(wc))
            w = wc;
        else {
            resp.add(dicionario.get(w));
            // Add wc to the dicionario.
            dicionario.put(wc, dicioTam++);
            w = "" + c;
        }
    }

    // Output the code for w.
    if (!w.equals(anObject: ""))
        resp.add(dicionario.get(w));
    return resp;
}
```

Figure 3. FIGURE 7. Código de Compressão por LZW.

Já na descompressão deve-se novamente incializar o dicionário (com os símbolos básicos) decodificar o 1º índice, escrevê-lo na saída e armazená-lo em W, colocar W? no dicionário e em um loop que repete até o fim dos indices: decodificar o primeiro símbolo S do próximo índice, trocar o ? da última entrada no dicionário por S, decodificar o resto do índice, escrevê-lo na saída e armazená-lo em W e por último colocar W? no dicionário.

(FIGURE 8)

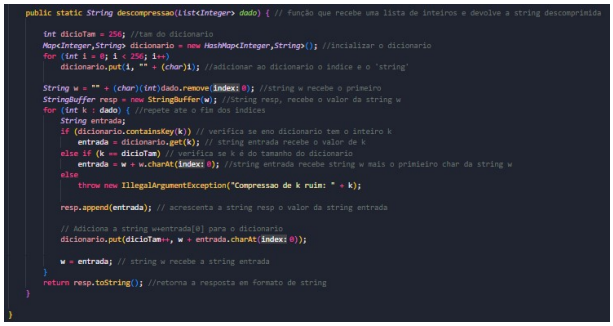


Figure 4. FIGURE 8. Código de Desompressão por LZW.

2.4.2. Huffman

A classe Huffman realiza a compressão de dados de uma conta do banco, utilizando do conceito de árvore, com nós e folhas. Com isso, uma árvore balanceada é criada e os nós da direita recebem valor 1 e os da esquerda recebem valor 0. Dessa maneira, a árvore é percorrida de modo que todos os caracteres utilizados no texto a ser comprimido possuem uma sequência de 0 ou 1, como por exemplo, o nome da pessoa pode ser Barbara e apresentar a sequência de 1100101110100. (FIGURE 9)

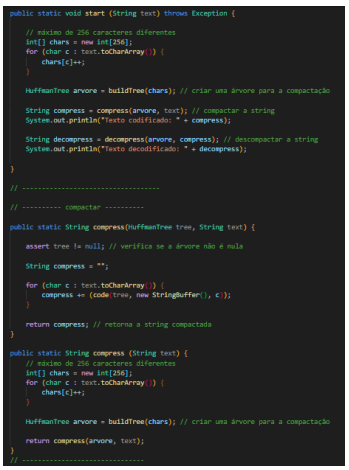


Figure 5. FIGURE 9. Código do método de compressão por HUFFMAN.

2.5. Casamento de Padrão - KMP

O metodo de casamento de padrões, KMP, em termos teóricos, foi responsável por introduzir o conceito de autômato de estados, sendo usado para a busca de cadeias, de modo que procura a ocorrência de uma palavra W dentro de um ”texto” S. A pesquisa dentro do metodo ocorre de maneira que campara-se o primeiro caractere do padrão com o primeiro do texto , e isso ocorre para todos os caracteres até que o padrão seja encontrado. Se durante a comparação, um carctere for distinto do outro, eu vou dar um shift no meu padrão, para realizar novas comparações, contudo, a parte que antes já fora comparada (prefixo do padrão) eu posso reutiliza-la na comparação se esta já foi encontrada no texto. Ou seja,

quando ocorrer um erro de comparação eu nunca irei retroceder no texto e sim no padrão. Dessa maneira a classe implementada no código abaixo, permite que um texto padrão, fornecido pelo usuário na tela de comando, seja buscado/procurado no texto original, que seria uma string contendo todos os dados da conta bancária de uma pessoa já cadastrada. (FIGURE 10)

Figure 6. FIGURE 10.Código do casamento de padrões por KMP.

2.6. Criptografia - OTP

A classe OTP realiza a criptografia das senhas das conta do banco, de maneira que assim que são criadas são armazenadas criptografadas e, quando são lidas, retornam ao texto como foi inserido pelo usuário na criação ou atualização da conta.

O tipo de criptografia utilizado foi o de OTP, ou seja, One Time Pad, porque é uma criptografia simétrica, sendo uma cifra de fluxo. Com isso, a criptografia é feita bit por bit, uma vez que cada caractere é substituido por outro de acordo com a soma entre o número representado pelo caractere da senha e o número representado pelo caractere da chave aleatória, sendo a chave de mesmo tamanho da senha. Esse ciframento foi comprovado matematicamente como inquebrável, portanto apresenta uma segurança muito grande, o que torna extremamente viável e necessário para o uso de uma senha de um banco. (FIGURE 11)

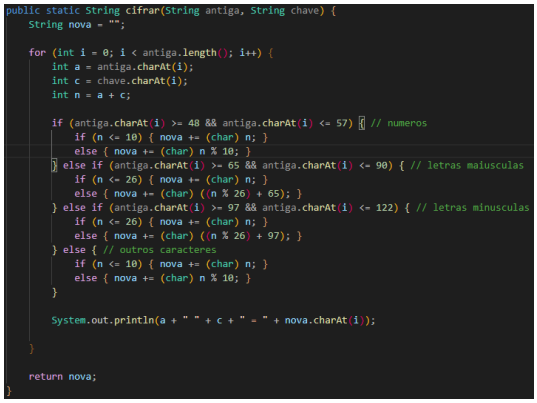
3. Testes e Resultados

Os testes foram realizados de maneira que, a cada entidade que o usuário precisa cadastrar para criar a conta foi sendo, o código especificado foi sendo testado e verficado/conferido se o que foi descrito no código estava sendo realmente feito.

3.1. Resultados

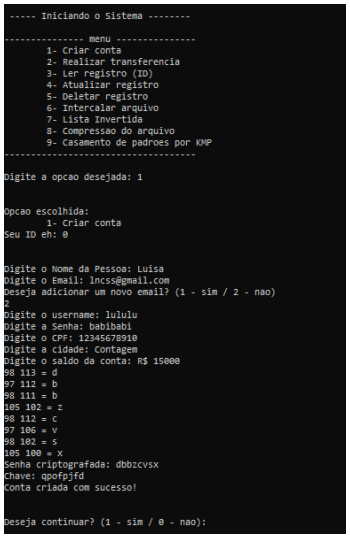
4. Conclusão

Considerando o objetivo do trabalho, pode-se concluir que os sistemas de bancos e outras empresas que guardam uma grande quantidade de dados dos usuários são bastante complexos, contendo muitos elementos e métodos para criar e manipulá-los, além de garantir



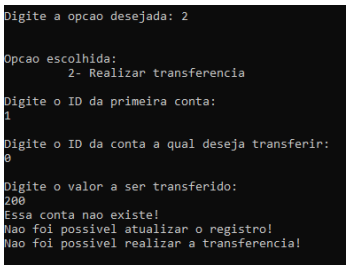
```java
public static String cifrar(String antiga, String chave) {
    String nova = "";

    for (int i = 0; i < antiga.length(); i++) {
        int a = antiga.charAt(i);
        int c = chave.charAt(i);
        int n = a + c;

        if (antiga.charAt(i) >= 48 && antiga.charAt(i) <= 57) { // numeros
            if (n <= 10) { nova += (char) n; }
            else { nova += (char) n % 10; }
        } else if (antiga.charAt(i) >= 65 && antiga.charAt(i) <= 90) { // letras maiusculas
            if (n <= 26) { nova += (char) n; }
            else { nova += (char) ((n % 26) + 65); }
        } else if (antiga.charAt(i) >= 97 && antiga.charAt(i) <= 122) { // letras minusculas
            if (n <= 26) { nova += (char) n; }
            else { nova += (char) ((n % 26) + 97); }
        } else { // outros caracteres
            if (n <= 10) { nova += (char) n; }
            else { nova += (char) n % 10; }
        }

        System.out.println(a + " " + c + " = " + nova.charAt(i));

    }

    return nova;
}
```

Figure 7. FIGURE 11.Código do método de criptografia por OTP.



```
----- Iniciando o Sistema -------

-------------- menu --------------
       1- Criar conta
       2- Realizar transferencia
       3- Ler registro (ID)
       4- Atualizar registro
       5- Deletar registro
       6- Intercalar arquivo
       7- Lista Invertida
       8- Compressao do arquivo
       9- Casamento de padroes por KMP
----------------------------------

Digite a opcao desejada: 1


Opcao escolhida:
       1- Criar conta
Seu ID eh: 0


Digite o Nome da Pessoa: Luisa
Digite o Email: lncss@gmail.com
Deseja adicionar um novo email? (1 - sim / 2 - nao)
2
Digite o username: lululu
Digite a Senha: babibabi
Digite o CPF: 12345678910
Digite a cidade: Contagem
Digite o saldo da conta: R$ 15000
98 113 = d
97 112 = b
98 111 = b
105 102 = z
98 112 = c
97 106 = v
98 102 = s
105 100 = x
Senha criptografada: dbbzcvsx
Chave: qpofpjfd
Conta criada com sucesso!


Deseja continuar? (1 - sim / 0 - nao):
```

Figure 8. Dados da conta fornecidos pelo usuario.



```
Digite a opcao desejada: 2


Opcao escolhida:
         2- Realizar transferencia

Digite o ID da primeira conta:
1

Digite o ID da conta a qual deseja transferir:
0

Digite o valor a ser transferido:
200
Essa conta nao existe!
Nao foi possivel atualizar o registro!
Nao foi possivel realizar a transferencia!
```

Figure 9. Realização de transferência.

a segurança e sigilo desses dados. Portanto, o conteúdo do desenvolvimento do projeto foi feito da melhor maneira possível para assegurar os tópicos abordados anteriormente de um sistema bancário.

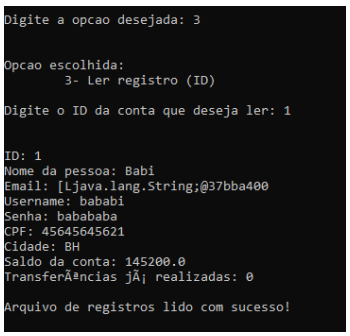


Figure 10. Dados de uma conta lido a partir do fornecimento do ID.

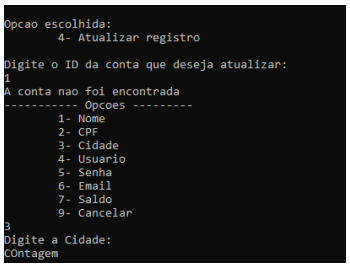


Figure 11. Atualização da entidade cidade de uma conta.

Figure 12. Deletando a conta a partir do seu ID.

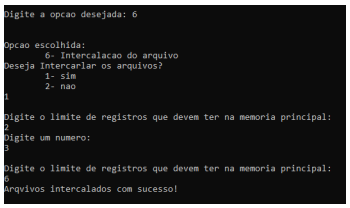


Figure 13. Intercalação do arquivo criado com as contas.

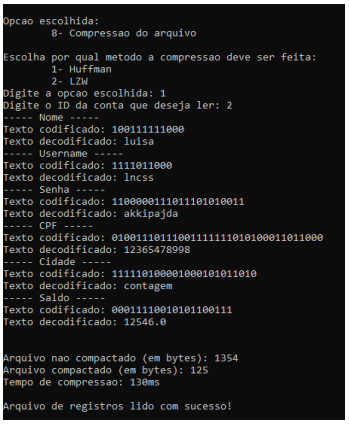


Figure 14. Resultado da compressão e descompressão por HUFFMAN.

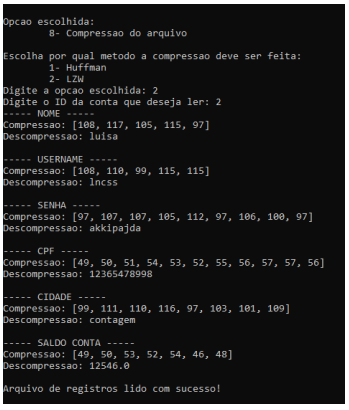


Figure 15. Resultado da compressão e descompressão por LZW.

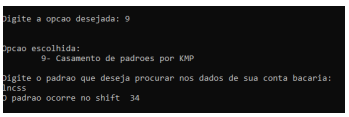


Figure 16. Resultado do casamento de padrões por KMP.