Universidade Metodista de Angola

Faculdade de Engenharia e Arquitetura Urbanismo

Trabalho de Programação II

Tema:

Sistema Bancário

Curso : Engenharia Informática

Sala:U106

Turma:B

Turno: M

Docente

_____

LUANDA, 2022

UNIVERSIDADE METODISTA DE ANGOLA – U.M.A

IMPLEMENTAÇÃO DE UM SISTEMA BANCÁRIO

Elaborado por:

Manuel Mazambilo Nº40217

Stélvia Taca Nº40749

Tatiana Isidro Nº40149

Vicente António Nº 45185

LUANDA, 2022

Resumo

A programação, às vezes chamada de programação computacional é a forma como nos comunicamos com os computadores. A programação informa à máquina quais ações executar. Criar um código é como definir um conjunto de instruções. Então podemos dizer que programação é um processo de escrita, testes e manutenção de programas de computadores. Esses programas, por sua vez, são compostos por conjuntos de instruções determinados pelo programador que descrevem tarefas a serem realizadas pela máquina e atendem diversas finalidades.

O nosso trabalho resumem- se na implementação de um sistema bancário que tem finalidade de criar contas bancárias, fazer levantamentos, depósitos, transferências e até mesmo retirar extratos bancários .

INDÍCE

Sumário

INTRODUÇÃO

Este presente trabalho encontra-se dividido em dois capítulos que são: **Enquadramento teórico e Aplicação prática do projecto**. No enquadramento teórico iremos abordar sobre conceitos de programação orientada objecto (POO) e abordaremos também sobre as plataformas utilizadas para a concepcão do sistema. No segundo capítulo trataremos sobre as aplicações práticas do projecto, onde demostraremos a codificação do sistema bancário .

CAPÍTLO I

ENQUADRAMENTO TEÓRICO

1-Conceito sobre as linguagem utilizadas

1.1.-JAVA

Para a concepção do projecto em estudo utlizou-se a linguangem programação Java e o MySQL.

JAVA é uma liguagem de programação orientada ao objectos desenvolvida na década de 90 por uma equipa de programadores chefiada por James Gosling.

A história começa em 1991, em San Hill Road empresa filiada a Sun (da qual hoje pertence a empresa Oracle), formado pelo time de engenheiros lliderados por Patrick Naugthon, Sun Fellow e James Gosling.

O grupo estava iniciando um projeto denominado Projeto Green, que consistia na criação de tecnologias modernas de software para empresas eletrônicas de consumo. A ideia principal do Java era que os aparelhos eletrônicos se comunicassem entre si. Por exemplo, o caso de possuir um fogão, você poderia deixar assando sua comida e quando estivesse pronta iria enviar uma mensagem para o microondas ligar e após isso tocar o seu despertador, sendo algo do gênero.

Com o tempo perceberam que não poderiam ficar presos aos sistemas operacionais, até porque os clientes não estavam interessados no tipo de processador que estavam utilizando, e sim na tecnologia. Portanto para o grupo criar uma versão do projeto para cada tipo de sistema era inviável, sendo assim, foi desenvolvido o sistema operacional GreenOS.

A linguagem de programação chamada de Oak (carvalho) foi criada pelo chefe do projeto James Gosling. A explicação da origem do nome foi que enquanto pensava numa estrutura de diretórios para a linguagem, observava pela janela um carvalho. Mas esse nome já estava registrado, então o nome acabou surgindo na cafeteria local da cidade onde tomavam café. “Java”, pois era o nome da terra de origem do café, que os programadores da equipe apreciavam nessa cafeteria, por isso que a logo do Java é um café.

Em 1993, apareceu uma oportunidade para o grupo Green. A empresa FirstPerson junto com a Time-Warner estava pedindo propostas de sistemas operacionais de decodificadores e tecnologias de vídeo sob demanda. Foi na época em que o NCSA apresentou o MOSAIC 1.0, o primeiro navegador gráfico para Web. Então a empresa FirstPerson apostou nos testes da TV da Time-Warner, mas esta empresa acabou escolhendo a tecnologia oferecida pela Silicon Graphics.

Em 1995 a Sun viu uma oportunidade na Web, nessa época nas páginas não existia muita interatividade, apenas conteúdos estáticos eram exibidos. Então nesse ano a Sun anunciou o ambiente Java, sendo um absoluto sucesso, gerando uma aceitação aos browsers populares como o Netscape Navigator e padrões tridimensionais como o VRML (Virtual Reality Modeling Language - Linguagem de Modelagem para a Realidade Virtual).

O Java foi o primeiro a utilizar decodificares de televisões interagindo em dispositivos portáteis e outros produtos eletrônicos de consumo, foi do mesmo jeito que foi

iniciado em 1991, possuindo portabilidade para qualquer ambiente e do desenvolvimento para múltiplas plataformas, em ambientes de eletrônicos de consumo, desde então o Java vem liderando o mercado em termos de linguagem.

1.2-SQL

O **MySQL** é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, como interface. É simplesmente um sistema popular que pode armazenar e gerenciar esses dados para você e é uma solução de base de dados especialmente popular para sites do WordPress. Essa linguagem foi criada nos anos 70 pela IBM para facilitar a implementação e a busca por banco de dados. Com o tempo, o SQL se tornou a linguagem padrão dos SGBDs do mercado, por ser bastante simples e intuitiva.

Os **comandos SQL são** essenciais para uma interação mais rápida em SGBDs (Sistemas de Gerenciamento de Bancos de dados relacionais). Abaixo estão listados os principais comandos:

SELECT: é um dos principais comandos SQL. Através dele é possível retomar registros no banco de dados. Com o SELECT é possível retomar múltiplos registros, informando quantas colunas queremos consultar.

CREAT DATABASE: com este comando é possível criar um banco de dados do zero. Ele é muito importante principalmente quando você precisa reestruturar todo o seu sistema, mas não sabe por onde começar. As informações estão mais fáceis de serem encontradas.

USE e SHOW DATABASES: Esses dois comandos são bastante parecidos, o comando SHOW DATABASES serve para fazer a visualização mais rápida de diferentes bases de dados e já com o comando USE é possível selecionar qual base de dados queremos usar e editar.

INSERT: Serve para inserir dados no banco de dados.

UPDATE: Serve para facilitar a atualização de dados nas tabelas. Esse recurso é bastante simples e de grande utilidade, já que torna possível a alteração de diversos registros com poucos cliques.

DELETE: Serve para excluir um ou mais registros de uma base de dados.

CREATE TABLE: Serve para criar novas tabelas em uma base de dados, conseguindo dividi-las em colunas, onde é possível salvar e referenciar especificações de produtos.

1.2.2-Caracteristícas

O nosso sitema bancário é capaz de fazer cadastramento de clientes, criar contas, aderir a conta poupança, fazer levantamentos, deposito e transferência.

1.2.3-Vantagens

Devido ás novas tecnologias chegando no mercado a corrida rumo à inovação não é diferente para os sistemas bancários. Com a implementação do nosso sistema bancário, o banco terá as seguintes vantagens:

- Redução aos custos operacioanis;
- Diminuição ao tempo de atendimento;
- Aumentára a seguraça no controle financeiro;
- Aprimorará o atendimento ao cliente;
- Permitirá flexibilidade nas transações .

1.2.4-Desvantagens

Como tudo na vida, onde há vantagem ira sempre existir desvantagens. Eis as desvantagem com a implementação de uma base de dados:

- Despesas de implementações;
- Eliminação de cargos ;
- Roubo de dados;
- Invasão de privacidade ;

CAPÍTULO II

APLICAÇÃO PRÁTICA DO PROJECTO

2.1 CÓDIGO JAVA

Os código em java encontra-se dividido em três modulos:DAO, DTO, WIWES

DAO

Conexão da BD com o Java

```
package DAO;

import java.sql.*;

import javax.swing.JOptionPane;

public class ConexaoDB{

        public static Connection conector(){

                java.sql.Connection conexao = null;

                String driver = "com.mysql.jdbc.Driver";

                String url = "jdbc:mysql://localhost:3306/banco";

                String user = "root";

                String pass = "";

                try{

                Class.forName(driver);

                conexao = DriverManager.getConnection(url,user,pass);

                return conexao;

                } catch (ClassNotFoundException | SQLException e){

                System.out.println(e);

                return null; }}}
```

DAO DE CONTA

```
package DAO;

import DTO.CadastroDTO;

import DTO.ContaDTO;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;
```

```
import java.sql.SQLException;
import java.util.ArrayList;
import javax.swing.JOptionPane;
public class ContaDao {
    Connection conn = null;
    PreparedStatement pstm = null;
    ResultSet rs = null;
    ArrayList<ContaDTO> lista = new ArrayList<>();
    public void criarConta(ContaDTO d) {
        String sql = "insert into contacorrente ( Cliente,Titular,Iban,Saldo) values(?,?,?,?)";
        conn = ConexaoDB.conector();
        try {
            pstm = conn.prepareStatement(sql);
            pstm.setInt(1, d.getCliente());
            pstm.setString(2, d.getTitular());
            pstm.setString(3, d.getIban());
            pstm.setDouble(4, d.getSaldo());
            pstm.executeUpdate();
        } catch (SQLException erro) {
            JOptionPane.showMessageDialog(null, "CadastroContaDAO erro" + erro);
        }    }
     public ResultSet listaCliente(){
    conn = ConexaoDB.conector();
    String sql ="select id_Cliente,Nome,NIF from cliente ";
        try {
            pstm =conn.prepareStatement(sql);
            return pstm.executeQuery();
            } catch (SQLException e) {
            JOptionPane.showMessageDialog(null,"ContaDao, listaCliente :" +e);
            return null;
```

```
    } }
   public ArrayList<ContaDTO> listarContas() {
    String sql = "select * from contacorrente ";
    conn = ConexaoDB.conector();
    try {
       pstm = conn.prepareStatement(sql);
       rs = pstm.executeQuery();
       while (rs.next()) {
          ContaDTO ctdto = new ContaDTO();
          ctdto.setNumero_conta(rs.getInt("numero_conta"));
          ctdto.setCliente(rs.getInt("Cliente"));
          ctdto.setTitular(rs.getString("Titular"));
          ctdto.setIban(rs.getString("Iban"));
          ctdto.setSaldo(rs.getDouble("Saldo"));
          lista.add(ctdto);
       } } catch (SQLException erro) {
       JOptionPane.showMessageDialog(null, "listar Cliente " + erro);    }
     return lista;    } }
```

DAO DE CONTA POUPANÇA

```
package DAO;
import DTO.ContaDTO;
import DTO.ContaPoupançaDTO;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import javax.swing.JOptionPane;
public class ContaPoupançaDAO {
   Connection conn = null;
```

```
PreparedStatement pstm = null;
ResultSet rs = null;
ArrayList<ContaPoupançaDTO> lista = new ArrayList<>();
public void criarConta(ContaPoupançaDTO d) {
   String sql = "insert into contapoupança (Cliente,Titular,Saldo,Variação,Limite) values(?,?,?,?,?)";
   conn = ConexaoDB.conector();
   try {
      pstm = conn.prepareStatement(sql);
      pstm.setInt(1, d.getCliente());
      pstm.setString(2, d.getTitular());
      pstm.setDouble(3, d.getSaldo());
      pstm.setDouble(4, d.getVariação());
      pstm.setDouble(5, d.getLimite());
      pstm.executeUpdate();
   } catch (SQLException erro) {
      JOptionPane.showMessageDialog(null, "ContaPoupança erro, criar conta" + erro);  }

 public ResultSet listaCliente(){
conn = ConexaoDB.conector();
String sql ="select id_Cliente,Nome,NIF from cliente ";
   try {
      pstm =conn.prepareStatement(sql);
      return pstm.executeQuery();

   } catch (SQLException e) {
      JOptionPane.showMessageDialog(null,"ContaDao, listaCliente :" +e);
      return null;
   } }
 public ArrayList<ContaPoupançaDTO> listarContas() {
   String sql = "select * from contapoupança ";
```

```
    conn = ConexaoDB.conector();
    try {
       pstm = conn.prepareStatement(sql);
       rs = pstm.executeQuery();
       while (rs.next()) {
          ContaPoupançaDTO ctdto = new ContaPoupançaDTO();
          ctdto.setNumero_conta(rs.getInt("Numero_conta"));
          ctdto.setCliente(rs.getInt("Cliente"));
          ctdto.setTitular(rs.getString("Titular"));
          ctdto.setSaldo(rs.getDouble("Saldo"));
          ctdto.setVariação(rs.getDouble("Variação"));
          ctdto.setLimite(rs.getDouble("Limite"));
          lista.add(ctdto);
       }
    } catch (SQLException erro) {
       JOptionPane.showMessageDialog(null, "listar Contas ,ContaPoupança DAO " + erro); }
    return lista;
}}
```

DAO DE DEPOSITO

```
package DAO;
import DTO.ContaDTO;
import DTO.DepositoDTO;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
import javax.swing.JOptionPane;
public class DepositoDAO {
    Connection conn = null;
    PreparedStatement pstm = null;
    public ResultSet listaCliente(){
    conn = ConexaoDB.conector();
    String sql ="select numero_conta,Cliente,Titular,Saldo from contacorrente ";
       try {
           pstm =conn.prepareStatement(sql);
           return pstm.executeQuery();
       } catch (SQLException e) {
           JOptionPane.showMessageDialog(null,"ListarCliente  Deposito Dao :" +e);
           return null;
       }}
     public void depositar(DepositoDTO d){
  String sql = "insert into deposito ( Numero_Conta,Titular,Valor,Data_Deposito) values(?,?,?,?)";
 conn = ConexaoDB.conector();
 try {
           pstm = conn.prepareStatement(sql);
           pstm.setInt(1, d.getNumero_Conta());
           pstm.setString(2, d.getTitular());
           pstm.setDouble(3, d.getValor());
           pstm.setString(4, d.getData_Deposito());
           pstm.executeUpdate();
      } catch (SQLException erro) {
          JOptionPane.showMessageDialog(null, "Deposito Dao erro" + erro);  }}
  public void novoSaldo(ContaDTO d){
```

```
String sql = "UPDATE `contacorrente` SET `Saldo` = ? WHERE `contacorrente`.`numero_conta` = ?;";
  conn = ConexaoDB.conector();
  try {
     pstm = conn.prepareStatement(sql);
     pstm.setDouble(1, d.getSaldo());
     pstm.setInt(2, d.getNumero_conta());
     pstm.executeUpdate();
   } catch (SQLException erro) {
     JOptionPane.showMessageDialog(null, "Deposito Dao erro, novoSaldo" + erro);
  }}}
```

DAO DE LEVANTAMENTO

```
package DAO;
import DTO.ContaDTO;
import DTO.LevantamentoDTO;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.JOptionPane;
public class LevantamentoDAO {
   Connection conn=null;
   PreparedStatement pstm= null;
    public ResultSet listaCliente(){
   conn = ConexaoDB.conector();
   String sql ="select numero_conta,cliente,Titular,Saldo from contacorrente ";
      try {
         pstm =conn.prepareStatement(sql);
         return pstm.executeQuery();  } catch (SQLException e) {
         JOptionPane.showMessageDialog(null,"ListarCliente  Levantamento Dao :" +e);
```

```
        return null;
    }}
  public void levantar(LevantamentoDTO d){
   String sql = "insert into levantamento ( Numero_Conta,Titular,Valor,Data_Levantamento) values(?,?,?,?)";
    conn = ConexaoDB.conector();
    try {
        pstm = conn.prepareStatement(sql);
        pstm.setInt(1, d.getNumero_Conta());
        pstm.setString(2, d.getTitular());
        pstm.setDouble(3, d.getValor());
        pstm.setString(4, d.getData_Deposito());
        pstm.executeUpdate();

     } catch (SQLException erro) {
        JOptionPane.showMessageDialog(null, "Levantamento Dao erro no levantar" + erro);
    }
  }
    public void novoSaldo(ContaDTO d){
  String sql = "UPDATE `contacorrente` SET `Saldo` = ? WHERE `contacorrente`.`numero_conta` = ?;";
    conn = ConexaoDB.conector();
    try {
        pstm = conn.prepareStatement(sql);
        pstm.setDouble(1, d.getSaldo());
        pstm.setInt(2, d.getNumero_conta());
        pstm.executeUpdate();
     } catch (SQLException erro) {
        JOptionPane.showMessageDialog(null, "Levantamento Dao erro, novoSaldo" + erro);
    }}}
```

DAO DE PAGAMENTO

```
package DAO;
```

```
import DTO.ContaDTO;
import DTO.DepositoDTO;
import DTO.PagamentoDTO;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.JOptionPane;
public class PagamentoDAO {
    Connection conn;
    PreparedStatement pstm;
  public ResultSet listaTipo(){
    conn = ConexaoDB.conector();
    String sql ="select * from tipo_servico ";
       try {
          pstm =conn.prepareStatement(sql);
          return pstm.executeQuery();

       } catch (SQLException e) {
          JOptionPane.showMessageDialog(null,"PagamentoDao listar serviços :" +e);
          return null;
       }  }
  public ResultSet listaContas(){
    conn = ConexaoDB.conector();
    String sql ="select numero_conta,cliente,Titular,Saldo from contacorrente ";
       try {
          pstm =conn.prepareStatement(sql);
          return pstm.executeQuery();
       } catch (SQLException e) {
```

```
            JOptionPane.showMessageDialog(null,"ListarCliente  Pagamentos Dao :" +e);
            return null;       }
}
    public void pagar(PagamentoDTO d){
     String sql = "insert into pagamento_servico ( Numero_Conta,Titular,Serviço,Rupe,Valor,Data_Pagamento)
values(?,?,?,?,?,?)";
        conn = ConexaoDB.conector();
        try {
            pstm = conn.prepareStatement(sql);
            pstm.setInt(1, d.getNumero_conta());
            pstm.setString(2, d.getTitular());
            pstm.setInt(3, d.getServiço());
            pstm.setDouble(4, d.getRupe());
            pstm.setDouble(5, d.getValor());
            pstm.setString(6, d.getData_Pagamento());
            pstm.executeUpdate();
         } catch (SQLException erro) {
            JOptionPane.showMessageDialog(null, "Pagamento DAO pagar" + erro);
        }}
      public void novoSaldo(ContaDTO d){
        String sql = "UPDATE `contacorrente` SET `Saldo` = ? WHERE `contacorrente`.`numero_conta` = ?;";
        conn = ConexaoDB.conector();
        try {
            pstm = conn.prepareStatement(sql);
            pstm.setDouble(1, d.getSaldo());
            pstm.setInt(2, d.getNumero_conta());
            pstm.executeUpdate();
         } catch (SQLException erro) {
            JOptionPane.showMessageDialog(null, "Pagamento Dao erro, novoSaldo" + erro);
        }}    }
```

DAO DE TRANSFERENCIA

```
ackage DAO;

import DTO.ContaDTO;

import DTO.TransferenciaDTO;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import javax.swing.JOptionPane;

public class TransferenciaDAO {

    Connection conn= null;

  PreparedStatement pstm = null;

   public ResultSet listaCliente(){

  conn = ConexaoDB.conector();

  String sql ="select numero_conta,cliente,Titular,Saldo from contacorrente";

     try {

        pstm =conn.prepareStatement(sql);

        return pstm.executeQuery();


     } catch (SQLException e) {

        JOptionPane.showMessageDialog(null,"ListarCliente  Transferencia Dao :" +e);

        return null;

     }}

     public void transferir(TransferenciaDTO d){


   String sql = "insert into transferencia ( Numero_Conta,Titular,Valor,Iban,Data_Transferencia) values(?,?,?,?,?)";

     conn = ConexaoDB.conector();

     try {

        pstm = conn.prepareStatement(sql);

        pstm.setInt(1, d.getNumero_Conta());
```

```
            pstm.setString(2, d.getTitular());
            pstm.setDouble(3, d.getValor());
            pstm.setDouble(4, d.getIban());
            pstm.setString(5, d.getData_Transferencia());
            pstm.executeUpdate();

        } catch (SQLException erro) {
            JOptionPane.showMessageDialog(null, "Transferecia  Dao erro" + erro);
       }
  public void novoSaldo(ContaDTO d){

    String sql = "UPDATE `contacorrente` SET `Saldo` = ? WHERE `contacorrente`.`numero_conta` = ?;";
       conn = ConexaoDB.conector();
       try {
pstm = conn.prepareStatement(sql);
            pstm.setDouble(1, d.getSaldo());
            pstm.setInt(2, d.getNumero_conta());
            pstm.executeUpdate();
        } catch (SQLException erro) {
            JOptionPane.showMessageDialog(null, "Transferencia Dao erro, novoSaldo" + erro);
       }}}
```

DAO DE CADASTRO CLIENTE

```
package DAO;

import DTO.CadastroDTO;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```
import java.util.ArrayList;
import javax.swing.JOptionPane;
public class cadastroDAO {

    Connection conn;
    PreparedStatement pstm;
    ResultSet rs;
    ArrayList<CadastroDTO> lista = new ArrayList<>();

    public void cadastrarfuncionario(CadastroDTO d) {

        String sql = "insert into cliente ( Nome,NIF,Data_Nascimento,Endereço,Telefone,Email) values(?,?,?,?,?,?)";
        conn = ConexaoDB.conector();
        try {

            pstm = conn.prepareStatement(sql);
            pstm.setString(1, d.getNome());
            pstm.setString(2, d.getNif());
            pstm.setString(3, d.getData());
            pstm.setString(4, d.getEndereco());
            pstm.setDouble(5, d.getTelefone());
            pstm.setString(6, d.getEmail());
            pstm.executeUpdate();

        } catch (SQLException erro) {
            JOptionPane.showMessageDialog(null, "cadastroDAO" + erro);
        }

    }
```

```
public ArrayList<CadastroDTO> listarCliente() {
    String sql = "select * from cliente ";
    conn = ConexaoDB.conector();
    try {
        pstm = conn.prepareStatement(sql);
        rs = pstm.executeQuery();
        while (rs.next()) {
            CadastroDTO ctdto = new CadastroDTO();
            ctdto.setId_Cliente(rs.getInt("id_Cliente"));
            ctdto.setNome(rs.getString("Nome"));
            ctdto.setNif(rs.getString("NIF"));
            ctdto.setData(rs.getString("Data_Nascimento"));
            ctdto.setEndereco(rs.getString("Endereço"));
            ctdto.setTelefone(rs.getDouble("Telefone"));
            lista.add(ctdto);
        }   } catch (SQLException erro) {
        JOptionPane.showMessageDialog(null, "listar Cliente " + erro);
    }
    return lista , }}
```

DAO DE USUARIO

```
package DAO;
import DTO.usuarioDTO;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.JOptionPane;

public class usuarioDAO {
```

```
Connection conn;
public ResultSet autenticacaousuario (usuarioDTO a){
   //conn= Conexao.conectaBD();
   try {
      String sql= "select * from funcionario where Nome=? and Senha=?";
      PreparedStatement pstm = conn.prepareStatement(sql);
      pstm.setString(1,a.getNome());
      pstm.setString(2,a.getSenha());
      ResultSet rs= pstm.executeQuery();
      return rs;

   } catch (SQLException erro) {
      JOptionPane.showMessageDialog(null, " usuarioDAO"+erro);
      return null;
   }
} }
```

DTO

DTO DE USUARIO

```
package DTO;
public class usuarioDTO {
   private int id_funcionario;
   private String Nome, Senha;
   public int getId_funcionario() {
      return id_funcionario; }
   public void setId_funcionario(int id_funcionario) {
      this.id_funcionario = id_funcionario;
```

```
    }
    public String getNome() {
        return Nome;
    }
    public void setNome(String Nome) {
        this.Nome = Nome;
    }
    public String getSenha() {
        return Senha;
    }

    public void setSenha(String Senha) {
        this.Senha = Senha;    }}
```

DTO DE CADASTRO

```
package DTO;
public class CadastroDTO {
    private String nome, endereco, nif, data,email;
    private int id_Cliente;
    private double telefone;
    public String getNome() {
        return nome;   }
    public void setNome(String nome) {
        this.nome = nome;
   }
    public String getEndereco() {
        return endereco;
    }
    public void setEndereco(String endereco) {
        this.endereco = endereco;   }
```

```
public String getNif() {
    return nif;
}

public void setNif(String nif) {
    this.nif = nif;
}

public String getData() {
    return data;
}

public void setData(String data) {
    this.data = data;
}

public int getId_Cliente() {
    return id_Cliente;
}
public void setId_Cliente(int id_Cliente) {
    this.id_Cliente = id_Cliente;
}

public double getTelefone() {
    return telefone;
}
```

```
    public void setTelefone(double telefone) {
        this.telefone = telefone;
    }
    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }}
```

DTO DE CONTA

```
package DTO;
public class ContaDTO {
    private int numero_conta,cliente;
    private double Saldo;
    private String Iban,Titular;

    public int getNumero_conta() {
        return numero_conta;
    }
    public void setNumero_conta(int numero_conta) {
        this.numero_conta = numero_conta;
    }

    public int getCliente() {
        return cliente;
    public void setCliente(int cliente) {
        this.cliente = cliente;
```

```
    }
    public double getSaldo() {
        return Saldo;
    }
    public void setSaldo(double Saldo) {
        this.Saldo = Saldo;
    }
    public String getIban() {
        return Iban;
    }
    public void setIban(String Iban) {
        this.Iban = Iban;
    }
    public String getTitular() {
        return Titular;
    }
    public void setTitular(String Titular) {
        this.Titular = Titular;
    }

}
```

DTO DE CONTA POUPANÇA

```
Package DTO;
public class ContaPoupançaDTO {
    private int Numero_conta;
```

```
private int Cliente;

private String Titular;

private double Saldo;

private double Variação;

private double Limite;

public int getNumero_conta() {

   return Numero_conta;

}

public void setNumero_conta(int Numero_conta) {

   this.Numero_conta = Numero_conta;

}

public int getCliente() {

   return Cliente;

}

   public void setCliente(int Cliente) {

   this.Cliente = Cliente;

}

public String getTitular() {

   return Titular;

}

public void setTitular(String Titular) {

   this.Titular = Titular;

}

public double getSaldo() {

   return Saldo;

}

public void setSaldo(double Saldo) {

   this.Saldo = Saldo;

}
```

```
    public double getVariação() {
        return Variação;
    }
    public void setVariação(double Variação) {
        this.Variação = Variação;
    }
    public double getLimite() {
        return Limite;
    }
    public void setLimite(double Limite) {
        this.Limite = Limite;
    }
}
```

DTO DE DEPOSITO

```
package DTO;
public class DepositoDTO {
    private int Numero_deposito,Numero_Conta;
    private String Titular,Data_Deposito;
    private double Valor;

    public int getNumero_deposito() {
        return Numero_deposito;
    }
    public void setNumero_deposito(int Numero_deposito) {
        this.Numero_deposito = Numero_deposito;
    }

    public int getNumero_Conta() {
        return Numero_Conta;
```

```
    }



    public void setNumero_Conta(int Numero_Conta) {
        this.Numero_Conta = Numero_Conta;
    }

    /**
    public String getTitular() {
        return Titular;
    }

    public void setTitular(String Titular) {
        this.Titular = Titular;
    }

    public String getData_Deposito() {
        return Data_Deposito;
    }
    public void setData_Deposito(String Data_Deposito) {
        this.Data_Deposito = Data_Deposito;
    }

    public double getValor() {
        return Valor;
    }
    public void setValor(double Valor) {
        this.Valor = Valor;
    }
}
```

DTO DE LEVANTAMENTO

```
package DTO;

public class LevantamentoDTO {

   private int Numero_levantamento,Numero_Conta;

   private String Titular,Data_Deposito;

   private double Valor;

   public int getNumero_levantamento() {

      return Numero_levantamento;

   }


   public void setNumero_levantamento(int Numero_levantamento) {

      this.Numero_levantamento = Numero_levantamento;   }



   public int getNumero_Conta() {

      return Numero_Conta;

   }


   public void setNumero_Conta(int Numero_Conta) {

      this.Numero_Conta = Numero_Conta;

   }

   public String getTitular() {

      return Titular;

   }

   public void setTitular(String Titular) {

      this.Titular = Titular;

   }

   public String getData_Deposito() {
```

```
        return Data_Deposito;
    }
    public void setData_Deposito(String Data_Deposito) {
        this.Data_Deposito = Data_Deposito;
    }
    public double getValor() {
        return Valor;
    }
    public void setValor(double Valor) {
        this.Valor = Valor;
    }
}
```

DTO DE PAGAMENTO

```
package DTO;
public class PagamentoDTO {
    private int id_pagamento;
    private int Numero_conta;
    private int Serviço;
    private double Rupe;
    private double Valor;
    private String Data_Pagamento,Titular;

    public int getId_pagamento() {
        return id_pagamento;
    }
    public void setId_pagamento(int id_pagamento) {
        this.id_pagamento = id_pagamento;
```

```
}

public int getNumero_conta() {
    return Numero_conta;
}

public void setNumero_conta(int Numero_conta) {
    this.Numero_conta = Numero_conta;
}

public int getServiço() {
    return Serviço;
}

public void setServiço(int Serviço) {
    this.Serviço = Serviço;
}
public double getRupe() {
    return Rupe;
}
public void setRupe(double Rupe) {
    this.Rupe = Rupe;
}

public double getValor() {
    return Valor;
}
public void setValor(double Valor) {
```

```
        this.Valor = Valor;
    }
    public String getData_Pagamento() {
        return Data_Pagamento;
    }
    public void setData_Pagamento(String Data_Pagamento) {
        this.Data_Pagamento = Data_Pagamento;
    }

    public String getTitular() {
        return Titular;
    }
    public void setTitular(String Titular) {
        this.Titular = Titular;
    }
}
```

DTO DE TIPO-CONTAS

```
package DTO;


public class Tipo_ContaDTO {
    private int id_Tipo;
    private String Tipo;
    public int getId_Tipo() {
        return id_Tipo;
    }

    public void setId_Tipo(int id_Tipo) {
```

```
        this.id_Tipo = id_Tipo;
    }

    public String getTipo() {
        return Tipo;
    }

    public void setTipo(String Tipo) {
        this.Tipo = Tipo;
    }

}
```

DTO DE TRANFERÊNCIA

```
package DTO;

public class TransferenciaDTO {
    private int Numero_Transferencia,Numero_Conta;
    private double Valor,Iban;
    private String Titular,Data_Transferencia;

    public int getNumero_Transferencia() {
        return Numero_Transferencia;
    }

    /**
     * @param Numero_Transferencia the Numero_Transferencia to set
     */
```

```
public void setNumero_Transferencia(int Numero_Transferencia) {
    this.Numero_Transferencia = Numero_Transferencia;
}

/**
 * @return the Numero_Conta
 */
public int getNumero_Conta() {
    return Numero_Conta;
}
public void setNumero_Conta(int Numero_Conta) {
    this.Numero_Conta = Numero_Conta;
}

public double getValor() {
    return Valor;
}

public void setValor(double Valor) {
    this.Valor = Valor;
}
public double getIban() {
    return Iban;
}

public void setIban(double Iban) {
    this.Iban = Iban;
}

public String getTitular() {
```

```
        return Titular;
    }

    public void setTitular(String Titular) {
        this.Titular = Titular;
    }

    public String getData_Transferencia() {
        return Data_Transferencia;
    }
    public void setData_Transferencia(String Data_Transferencia) {
        this.Data_Transferencia = Data_Transferencia;
    }
}
```

VIWES

CÓDIGO PARA AUTENTICAÇÃO DO LOGIN

```
private void jbtEntrarActionPerformed(java.awt.event.ActionEvent evt) {

        String sql = " select * from funcionario where Nome=? and Senha=?";
        try {
            pst = conexao.prepareStatement(sql);
            pst.setString(1, textNome.getText());
            pst.setString(2, textSenha.getText());
            rs = pst.executeQuery();
```

```
            if (rs.next()) {
                setVisible(false);
                TelaPrincipal ob = new TelaPrincipal();
                ob.setVisible(true);
                rs.close();
            } else {
                JOptionPane.showMessageDialog(null, "Senha ou Nome errado, Tente Novamente");
                textNome.setText("");
                textSenha.setText("");
            }

        } catch (SQLException | HeadlessException e) {
            JOptionPane.showMessageDialog(null, e);
        }
    }
```

CRIANDO METODOS

MÉTODO PARA CADASTRAR

```
    public void cadastrarClientes() {

        try {
            String nome, endereco, nif, email;
            double telefone;
            nome = textNome.getText();
            nif = textNif.getText();
            email = txt_email.getText();
            endereco = textEndereco.getText();
            telefone = Double.parseDouble(textTelefone.getText());
```

```
            String dia, mes, ano;
            dia = jcmbdia.getSelectedItem().toString();
            mes = jcmbMes.getSelectedItem().toString();
            ano = jcmbAno.getSelectedItem().toString();
            int anoverif = Integer.parseInt(ano);
            int idade;
            idade = 2022 - anoverif;
            if (idade < 18) {
               JOptionPane.showMessageDialog(null, "Idade Inválida para criar Conta");
            } else {
               String data = dia + "/" + mes + "/" + ano;
               CadastroDTO d = new CadastroDTO();
               d.setNome(textNome.getText());
               d.setNif(textNif.getText());
               d.setData(data);
               d.setEndereco(textEndereco.getText());
               d.setTelefone(Double.parseDouble(textTelefone.getText()));
               d.setEmail(email);
               cadastroDAO e = new cadastroDAO();
               e.cadastrarfuncionario(d);
               JOptionPane.showMessageDialog(null, "Cadastrado com Sucesso", "Alerta",
JOptionPane.INFORMATION_MESSAGE);
               textEndereco.setText("");
               textNif.setText("");
               textNome.setText("");
               textTelefone.setText("");
               txt_email.setText("");
            }

        } catch (NumberFormatException | HeadlessException e) {
```

```
            JOptionPane.showMessageDialog(null, "Cadastro formulario, cadastrar Cliente" + e);
    }


      }
```

METODO PARA LISTAR CLIENTES

```
Vector<Integer> id_Cliente = new Vector<Integer>();
      Vector<String> NIF = new Vector<String>();


      public void listarCliente() {
          try {
              ContaDao objcontaDao = new ContaDao();
              ResultSet rs = objcontaDao.listaCliente();
              while (rs.next()) {
                  id_Cliente.addElement(rs.getInt(1));
                  cmbTitular.addItem(rs.getString(2));
                  NIF.addElement(rs.getString(3));
              }
          } catch (SQLException e) {
              JOptionPane.showMessageDialog(null, "Erro no formulario do tipo de contas, listar Cliente" + e);
          }
```

METODO PARA CADASTRAR CONTA CORRENTE

```
public void cadastrarContas() {
          try {
              String Iban = txtiban.getText();
              String Titular;
              int cliente = 0;
              double saldo = 0;
              cliente = id_Cliente.get(cmbTitular.getSelectedIndex() - 1);
              saldo = Double.parseDouble(txtSaldo.getText());
```

```
            Titular = (String) cmbTitular.getSelectedItem();
            if (saldo < 5000) {
                JOptionPane.showMessageDialog(null, "Erro, Valor a depositar insuficiente");
                txtSaldo.setText("");
            } else {
                ContaDTO cndto = new ContaDTO();
                cndto.setCliente(cliente);
                cndto.setTitular(Titular);
                cndto.setIban(Iban);
                cndto.setSaldo(saldo);
                ContaDao cndoa = new ContaDao();
                cndoa.criarConta(cndto);
                JOptionPane.showMessageDialog(null, "Conta criada com sucesso!!");
                txtSaldo.setText("");
                txtiban.setText("");
                jtxtNIF.setText("");
            }
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(null, "Erro no formulario de contas, cadastrar Contas" + e);
        }
    }
```

METODO PARA CADASTRAR CONTA POUPANÇA

```
public void cadastrarContasPoupança() {
        try {

            String Titular;
            int cliente = 0;
            double saldo = 0,variação,limite;
            cliente = id_Cliente.get(cmbTitular.getSelectedIndex() - 1);
            saldo = Double.parseDouble(txtSaldo.getText());
```

```
        Titular = (String) cmbTitular.getSelectedItem();
        variação = Double.parseDouble(txtVariação.getText());
        limite =  Double.parseDouble(txtLimite.getText());
        if (saldo < 5000) {
            JOptionPane.showMessageDialog(null, "Erro, Valor a depositar insuficiente");
            txtSaldo.setText("");
        } else {
            ContaPoupançaDTO cndto = new ContaPoupançaDTO();
            cndto.setCliente(cliente);
            cndto.setTitular(Titular);
            cndto.setSaldo(saldo);
            cndto.setLimite(limite);
            cndto.setVariação(variação);
            ContaPoupançaDAO cndoa = new ContaPoupançaDAO();
            cndoa.criarConta(cndto);
            JOptionPane.showMessageDialog(null, "Conta criada com sucesso!!");
            txtSaldo.setText("");
            jtxtNIF.setText("");
            txtLimite.setText("");
            txtSaldo.setText("");
            txtVariação.setText("");
        }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(null, "Erro no formulario de contas poupança, cadastrar Contas" + e);
    }

}
```

METODO PARA MOSTRAR TITULAR DA CONTA

```
Vector<String> Nome = new Vector<String>();
```

```
    Vector<Integer> NumeroCliente = new Vector<Integer>();
    public Vector<Double> SaldoActual = new Vector<Double>();

    public void mostrarTitular() {
        jtxtCliente.setText(String.valueOf(NumeroCliente.get(jcmbNumeroConta.getSelectedIndex() - 1)));
        jtxtTitular.setText(String.valueOf(Nome.get(jcmbNumeroConta.getSelectedIndex() - 1)));

    }
}
```

METODO PARA LISTAR CONTA

```
public void listarContas() {
        try {
            DepositoDAO objlevaDao = new DepositoDAO();
            ResultSet rs = objlevaDao.listaCliente();
            while (rs.next()) {
                jcmbNumeroConta.addItem(rs.getInt(1));
                NumeroCliente.addElement(rs.getInt(2));
                Nome.addElement(rs.getString(3));
                SaldoActual.addElement(rs.getDouble(4));
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Erro no formulario do Deposito Listar Contas" + e);
        }
    }
```

METODO PARA DEPOSITAR

```
public void depositar() {
        try {
            String dia, mes, ano;
            dia = jcmbdia.getSelectedItem().toString();
            mes = jcmbMes.getSelectedItem().toString();
```

```
ano = jcmbAno.getSelectedItem().toString();
String data = dia + "/" + mes + "/" + ano;
String Nomeu;
double Valor, valorantigo, novovalor;
int Numero_Conta;
Nomeu = jtxtTitular.getText();
Valor = Double.parseDouble(jtxtValor.getText());
valorantigo = SaldoActual.get(jcmbNumeroConta.getSelectedIndex() - 1);
if (Valor < 0) {
   JOptionPane.showMessageDialog(null, "Valor introduzido Inválido");
   jtxtValor.setText("");
} else {
   novovalor = Valor + valorantigo;
   Numero_Conta = (int) jcmbNumeroConta.getSelectedItem();
   DepositoDTO depdto = new DepositoDTO();
   ContaDTO contadto = new ContaDTO();
   depdto.setNumero_Conta(Numero_Conta);
   depdto.setTitular(Nomeu);
   depdto.setValor(Valor);
   depdto.setData_Deposito(data);
   DepositoDAO dpdao = new DepositoDAO();
   contadto.setSaldo(novovalor);
   contadto.setNumero_conta(Integer.parseInt(jtxtCliente.getText()));
   dpdao.novoSaldo(contadto);
   dpdao.depositar(depdto);
   JOptionPane.showMessageDialog(null, "Deposito concluido");
   jtxtCliente.setText("");
   jtxtTitular.setText("");
   jtxtTitular1.setText("");
   jtxtValor.setText("");
```

```
            }
        } catch (HeadlessException | NumberFormatException e) {
            JOptionPane.showMessageDialog(null, "Erro no formulario" + e);
        }
    }

}
```

METODO PARA LEVANTAMENTO

```
public void levantar() {
        try {
            String dia, mes, ano;
            dia = jcmbdia.getSelectedItem().toString();
            mes = jcmbMes.getSelectedItem().toString();
            ano = jcmbAno.getSelectedItem().toString();
            String data = dia + "/" + mes + "/" + ano;
            String Nome;
            double Valor, valorantigo, novovalor;;
            int Numero_Conta;
            Nome = jtxtCliente.getText();
            Valor = Double.parseDouble(jtxtValor.getText());
            valorantigo = SaldoActual.get(jcmbNumeroConta.getSelectedIndex() - 1);
            if (Valor > valorantigo) {
                if (Valor < 0) {
                    JOptionPane.showMessageDialog(null, "Impossivel realizar a operação");
                    jtxtValor.setText("");
                } else {
                    JOptionPane.showMessageDialog(null, "Impossivel realizar a operação");
                    jtxtValor.setText("");
                }
            } else {
```

```
            novovalor = valorantigo - Valor;
            Numero_Conta = (int) jcmbNumeroConta.getSelectedItem();
            LevantamentoDTO depdto = new LevantamentoDTO();
            ContaDTO contadto = new ContaDTO();
            depdto.setNumero_Conta(Numero_Conta);
            depdto.setTitular(Nome);
            depdto.setValor(Valor);
            depdto.setData_Deposito(data);
            LevantamentoDAO dpdao = new LevantamentoDAO();
            contadto.setSaldo(novovalor);
            contadto.setNumero_conta(Integer.parseInt(jtxtCliente.getText()));
            dpdao.novoSaldo(contadto);
            dpdao.levantar(depdto);
            JOptionPane.showMessageDialog(null, "Levantamento concluido");
            jtxtCliente.setText("");
            jtxtTitular.setText("");
            jtxtValor.setText("");
          }
      } catch (HeadlessException | NumberFormatException e) {
          JOptionPane.showMessageDialog(null, "Erro no formulario Levantar " + e);
      }
  }
```

METODO PARA LISTAR-SERVIÇOS

```
Vector<Integer> id_servico = new Vector<Integer>();
   Vector<String> Nome = new Vector<String>();
   Vector<Integer> NumeroCliente = new Vector<Integer>();
   public Vector<Double> SaldoActual = new Vector<Double>();

   public void listarTipoServico() {
      try {
```

```
        PagamentoDAO objpagaDao = new PagamentoDAO();
        ResultSet rs = objpagaDao.listaTipo();
        while (rs.next()) {
            id_servico.addElement(rs.getInt(1));
            jcmbServico.addItem(rs.getString(2));
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "Erro no formularioServiços listar servicos" + e);
    }

}
```

METODO PARA PAGAMENTOS-SERVIÇOS

```
public void pagar() {
    try {
        String dia, mes, ano;
        dia = jcmbdia.getSelectedItem().toString();
        mes = jcmbMes.getSelectedItem().toString();
        ano = jcmbAno.getSelectedItem().toString();
        String data = dia + "/" + mes + "/" + ano;
        int Numero_conta, Serviço;
        double Rupe, Valor, valorantigo, novovalor;
        Serviço = id_servico.get(jcmbServico.getSelectedIndex() - 1);
        Numero_conta = (int) jcmbConta.getSelectedItem();
        Valor = Double.parseDouble(jtxtValor.getText());
        valorantigo = SaldoActual.get(jcmbConta.getSelectedIndex() - 1);
        if (Valor > valorantigo) {
            if (Valor < 0) {
                JOptionPane.showMessageDialog(null, "Impossivel realizar a operação");
```

```
            jtxtValor.setText("");
        } else {
            JOptionPane.showMessageDialog(null, "Impossivel realizar a operação");
            jtxtValor.setText("");
        }
    } else {
        novovalor = valorantigo - Valor;
        String titular = jtxtTitular.getText();
        Rupe = Double.parseDouble(jtxtRupe.getText());
        PagamentoDTO pagadto = new PagamentoDTO();
        ContaDTO contadto = new ContaDTO();
        pagadto.setNumero_conta(Numero_conta);
        pagadto.setTitular(titular);
        pagadto.setValor(Valor);
        pagadto.setData_Pagamento(data);
        pagadto.setRupe(Rupe);
        pagadto.setServiço(Serviço);
        PagamentoDAO pagadao = new PagamentoDAO();
        contadto.setSaldo(novovalor);
        contadto.setNumero_conta(Integer.parseInt(jtxtCliente.getText()));
        pagadao.novoSaldo(contadto);
        pagadao.pagar(pagadto);
        JOptionPane.showMessageDialog(null, "Pagamento concluido");
        jtxtCliente.setText("");
        jtxtRupe.setText("");
        jtxtTitular.setText("");
        jtxtValor.setText("00");
    }

} catch (HeadlessException | NumberFormatException e) {
```

```
            JOptionPane.showMessageDialog(null, "Erro no formulario" + e);
        }


    }
```

METODO PARA TRANSFERÊNCIA

```
public void transferir() {
        try {
            String dia, mes, ano;
            dia = jcmbdia.getSelectedItem().toString();
            mes = jcmbMes.getSelectedItem().toString();
            ano = jcmbAno.getSelectedItem().toString();
            String data = dia + "/" + mes + "/" + ano;
            String Nomes;
            double Valor, valorantigo, novovalor, Iban;
            int Numero_Conta;
            Iban = Double.parseDouble(jtxtIban.getText());
            Nomes = jtxtTitular.getText();
            Valor = Double.parseDouble(jtxtValor.getText());
            valorantigo = SaldoActual.get(jcmbConta.getSelectedIndex() - 1);
            novovalor = valorantigo - Valor;
            if (Valor > valorantigo) {
                if (Valor < 0) {
                    JOptionPane.showMessageDialog(null, "Impossivel realizar a operação, Valor Introduzido Inválido");
                    jtxtValor.setText("");
                } else {
                    JOptionPane.showMessageDialog(null, "Impossivel realizar a operação, Valor Introduzido Inválido");
                    jtxtValor.setText("");
                }
            } else {
```

```
            Numero_Conta = (int) jcmbConta.getSelectedItem();
            TransferenciaDTO depdto = new TransferenciaDTO();
            ContaDTO contadto = new ContaDTO();
            depdto.setNumero_Conta(Numero_Conta);
            depdto.setIban(Iban);
            depdto.setTitular(Nomes);
            depdto.setValor(Valor);
            depdto.setData_Transferencia(data);
            TransferenciaDAO dpdao = new TransferenciaDAO();
            contadto.setSaldo(novovalor);
            contadto.setNumero_conta(Integer.parseInt(jtxtCliente.getText()));
            dpdao.novoSaldo(contadto);
            dpdao.transferir(depdto);
            JOptionPane.showMessageDialog(null, "Transferencia concluida");
            jtxtCliente.setText("");
            jtxtIban.setText("");
            jtxtTitular.setText("");
            jtxtValor.setText("");
        }
    } catch (HeadlessException | NumberFormatException e) {
        JOptionPane.showMessageDialog(null, "Erro no formulario Levantar " + e);
    }
}
```

CHAMANDO O METODO NO BOTAO

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    cadastrarClientes();
```

```
        new Contas().setVisible(true);
        this.dispose();
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        cadastrarContas();
   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        cadastrarContasPoupança();
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        depositar();
    }
  private void formMouseMoved(java.awt.event.MouseEvent evt) {
        mostrarTitular   }
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        levantar();   }
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        pagar();
    }
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        transferir();
    }
```

CÓDIGO DA BASE DE DADOS

Para criação de uma base de dados e tabelas temos os seguintes códigos :

Criando a base de dados

```
CREATE DATABASE BANCO  DEFAULT
CREATE TABLE FUNCIONARIO(
ID_FUNCIONARIO INT NOT NULL AUTO_INCREMENT,
NOME VARCHAR (30),
SENHA INT  NOT NULL,
PRIMARY KEY (ID_FUNCIONARIO)
```

```
) DEFAULT CHARSET= UTF8;


  CREATE TABLE CLIENTE(
   ID_CLIENTE  INT NOT NULL AUTO_INCREMENT ,
   NOME VARCHAR (30),
   DATA_NASCIMENTO VARCHAR (20),
   NIF VARCHAR (30),
   ENDEREÇO VARCHAR (20),
   TELEFONE DOUBLE,
   EMAIL VARCHAR (20),
   PRIMARY KEY (ID_CLIENTE)
   ) DEFAULT CHARSET= UTF8;



 CREATE TABLE CONTACORRENTE(
   NUMERO_CONTA  INT NOT NULL AUTO_INCREMENT ,
   CLIENTE INT NOT NULL,
   TITULAR VARCHAR (30) NOT NULL,
   IBAN  VARCHAR (15) NOT NULL,
   SALDO_INICIAL DOUBLE  NOT NULL,
   PRIMARY KEY (NUMERO_CONTA),
  FOREIGN KEY (CLIENTE) REFERENCES CLIENTE  (ID_CLIENTE)


) DEFAULT CHARSET= UTF8;


CREATE TABLE CONTAPOUPANÇA(
   NUMERO_CONTA  INT NOT NULL AUTO_INCREMENT ,
 CLIENTE INT NOT NULL ,
```

```
    SALDO_INICIAL DOUBLE NOT  NULL,
    VARIACAO DOUBLE NOT NULL,
    LIMITE DOUBLE NOT NULL,
PRIMARY KEY (NUMERO_CONTA),
    FOREIGN KEY (NUMERO_CONTA) REFERENCES CLIENTE (ID_CLIENTE)
   ) DEFAULT CHARSET = UTF8;  );

CREATE TABLE LEVANTAMENTO(
    NUMERO_LEVANTAMENTO  INT NOT NULL AUTO_INCREMENT ,
    NUMERO_CONTA INT,
    DATA_LEVANTAMENTO VARCHAR(20),
    TITULAR  VARCHAR ( 20),
    VALOR DOUBLE,
    PRIMARY KEY (NUMERO_LEVANTAMENTO),
    FOREIGN KEY (NUMERO_CONTA) REFERENCES CLIENTE (ID_CLIENTE)
    )DEFAULT CHARSET= UTF8;


CREATE TABLE DEPOSITO (
    NUMERO_DEPOSITO  INT NOT NULL AUTO_INCREMENT ,
    NUMERO_CONTA INT,
    DATA_DEPOSITO VARCHAR(30) NOT NULL,
    TITULAR  VARCHAR ( 20) NOT NULL,
    VALOR DOUBLE,
    PRIMARY KEY (NUMERO_DEPOSITO),
    FOREIGN KEY (NUMERO_CONTA) REFERENCES CLIENTE (ID_CLIENTE)

   ) DEFAULT CHARSET = UTF8;
```

```
CREATE TABLE TRANSFERENCIA (
   NUMERO_TRANSFERENCIA  INT NOT NULL AUTO_INCREMENT ,
   NUMERO_CONTA INT,
   IBAN DOUBLE,
   DATA_TRANSFERENCIA VARCHAR(20),
   TITULAR  VARCHAR ( 20),
   VALOR DOUBLE,
   PRIMARY KEY (NUMERO_TRANSFERENCIA),
   FOREIGN KEY (NUMERO_CONTA) REFERENCES CLIENTE (ID_CLIENTE)
) DEFAULT CHARSET =UTF8;

CREATE TABLE TIPO_SERVIÇO (
   ID_SERVIÇO INT NOT NULL AUTO_INCREMENT ,
  TIPO_SERVIÇO  VARCHAR (30) NOT NULL,
   PRIMARY KEY ( ID_SERVIÇO)
) DEFAULT CHARSET= UTF8;
CREATE TABLE PAGAMENTO_SERVIÇO(
  ID_PAGAMENTO  INT NOT NULL AUTO_INCREMENT ,
   NUMERO_CONTA INT,
   TITULAR VARCHAR (30) NOT NULL,
   SERVIÇO INT NOT NULL,
   DATA_PAGAMENTO VARCHAR(20),
   RUPE  DOUBLE NOT NULL,
   VALOR DOUBLE NOT NULL,
   PRIMARY KEY (ID_PAGAMENTO),
   FOREIGN KEY (NUMERO_CONTA) REFERENCES CLIENTE (ID_CLIENTE),
   FOREIGN KEY (SERVIÇO) REFERENCES tipo_SERVIÇO(ID_SERVIÇO)
) DEFAULT CHARSET= UTF8 ;

CREATE TABLE EXTRATO(
```

```
    NUMERO_EXTRATO  INT NOT NULL AUTO_INCREMENT ,
    NUMERO_CONTA INT,
    DATA_EXTRATO VARCHAR(20),
    NUMERO_DEPOSITO INT,
    NUMERO_LEVANTAMENTO INT,
    NUMERO_TRANSFERENCIA INT,
    ÑUMERO_PAGAMENTO INT ,
    PRIMARY KEY (NUMERO_EXTRATO),
    FOREIGN KEY (NUMERO_DEPOSITO) REFERENCES DEPOSITO (NUMERO_DEPOSITO),
    FOREIGN KEY (NUMERO_LEVANTAMENTO) REFERENCES LEVANTAMENTO(NUMERO_LEVANTAMENTO),
    FOREIGN KEY (NUMERO_TRANSFERENCIA) REFERENCES TRANSFERENCIA (NUMERO_TRANSFERENCIA),
    FOREIGN KEY (NUMERO_PAGAMENTO) REFERENCES PAGAMENTO_SERVIÇO (ID_SERVIÇO)
)  DEFAULT CHARSET = UTF8;
```

CONCLUSÃO

Em vista dos argumentos apresentados, concluimos que o sistema bancário é o conjunto de bancos comerciais e instituições financeiras que prestam serviços aos consumidores.

O sistema bancário tem evoluído bastantePor isso, é possível afirmar que o sistema de banco é o responsável por toda oferta de crédito de um país.

O nosso Sistema em particular é viavel, confiavel, onde os consumidores podem fazer consultas dos seus valores, depositos, levantamentos, pagamentos de servicos e tranferencias bancarias.

REFERÊNCIAS BIBLIOGRÁFICAS

https://www.youtube.com/c/CursoemV%C3%ADdeo.

https://www.youtube.com/channel/UCm-AncOkWSM0_pjw21dWZLw.

https://www.youtube.com/c/Descompila.

ANEXOS

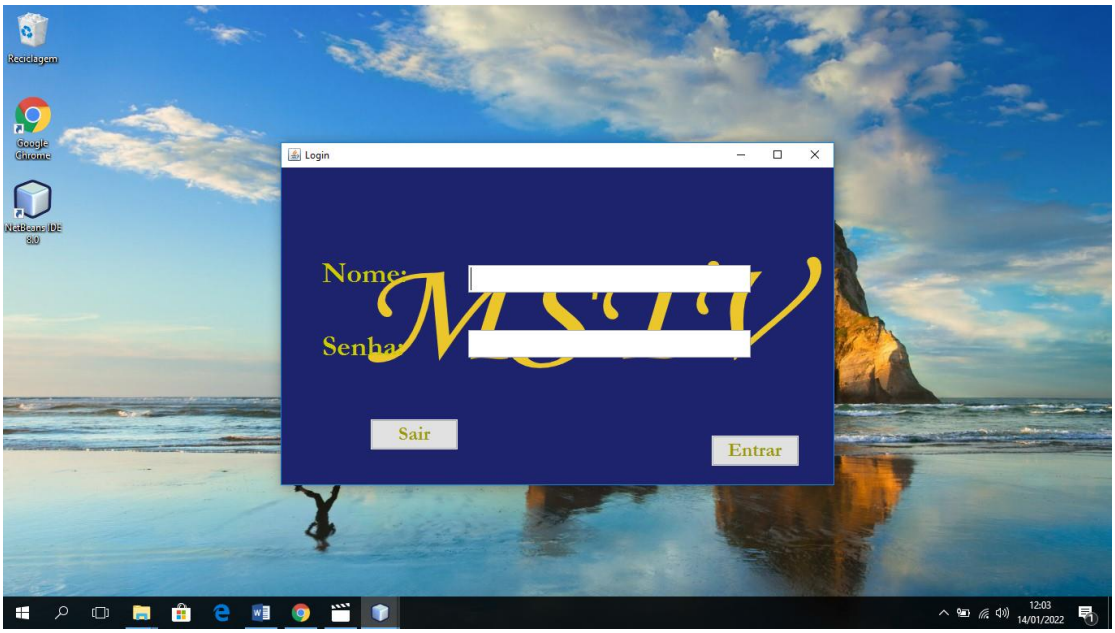

Fig1-Tela Login

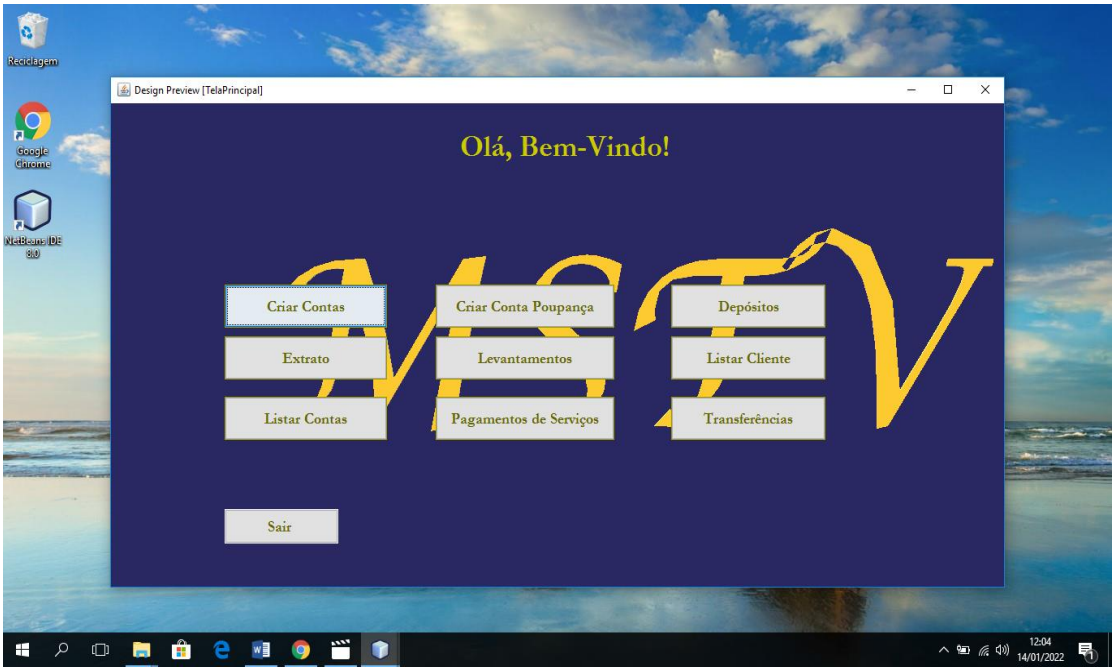

Fig2-Tela Principal

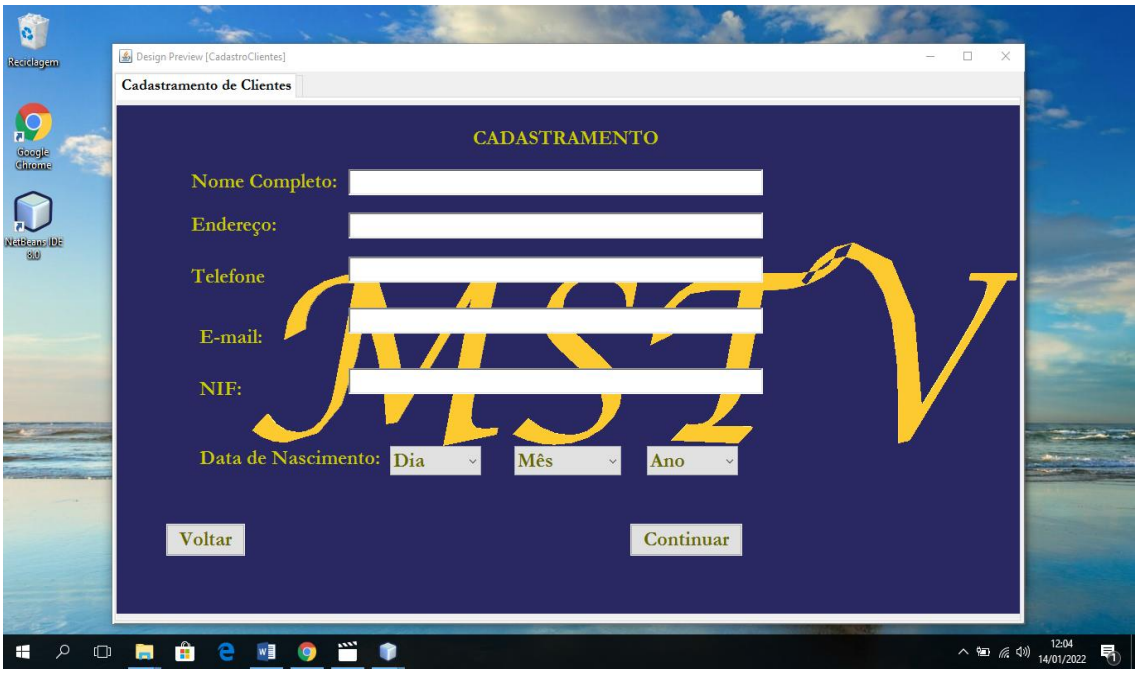

Fig13-Tela cadastramento

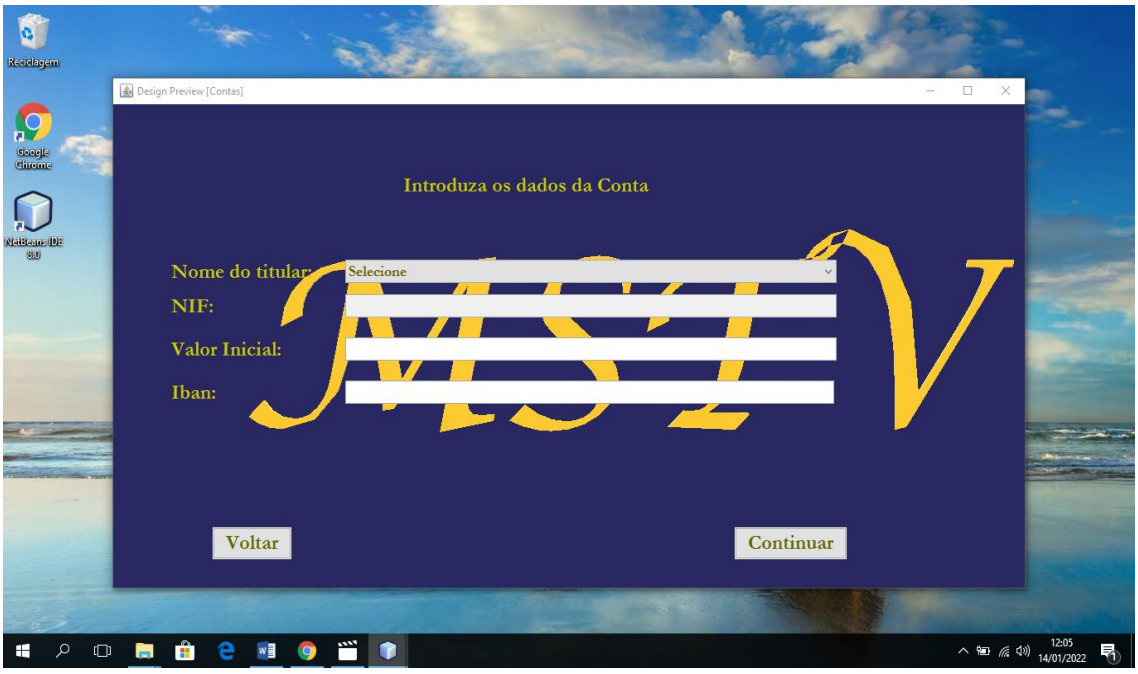

Fig4-Tela criar conta

Fig5-Tela conta poupança

Fig6-Tela de deposito

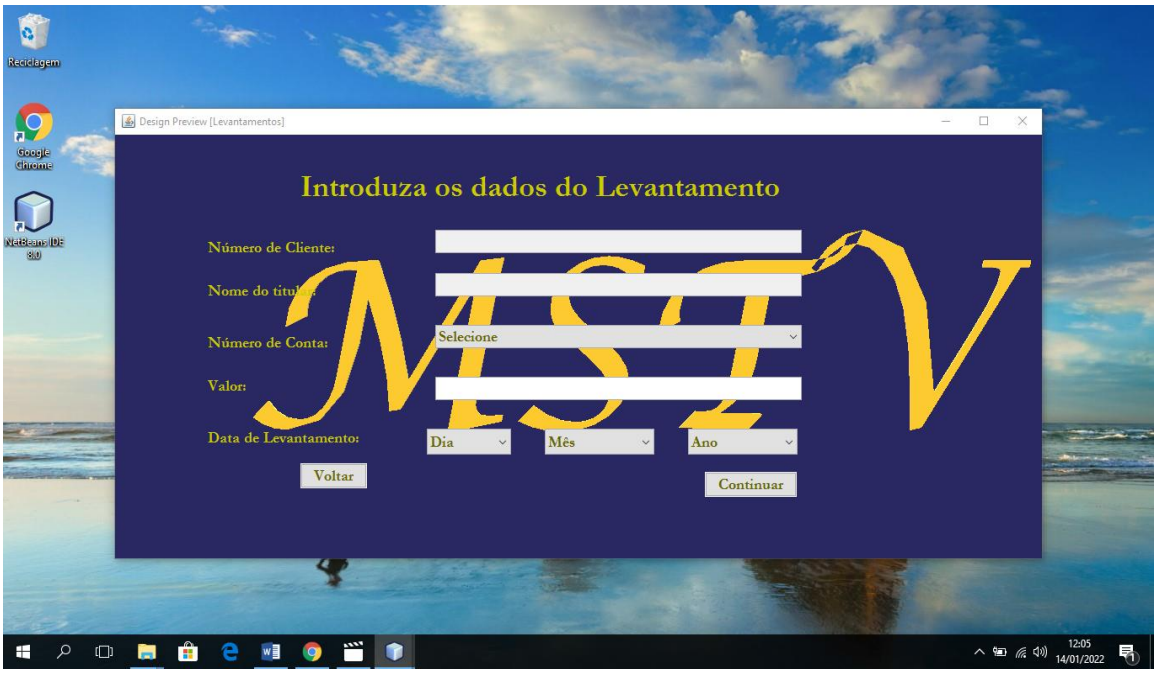

Fig7-Tela de levantamento

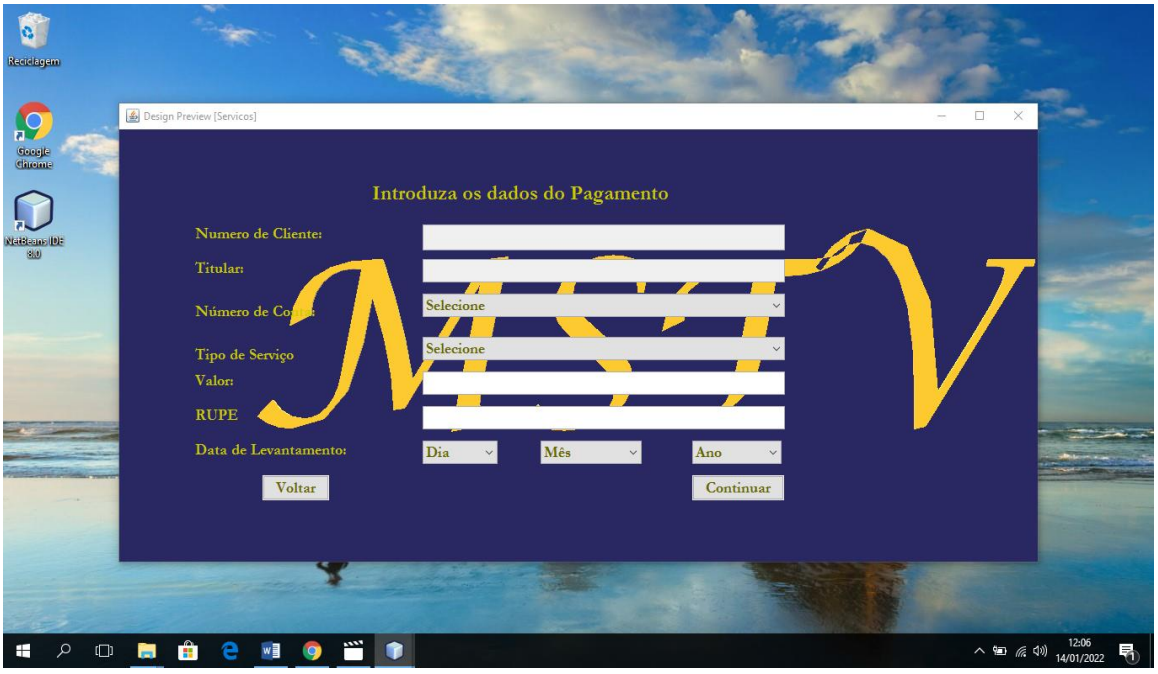

Fig8-Tela de pagamento

Fig9-Tela de transferencia

Fig10-Tela Login funcionando

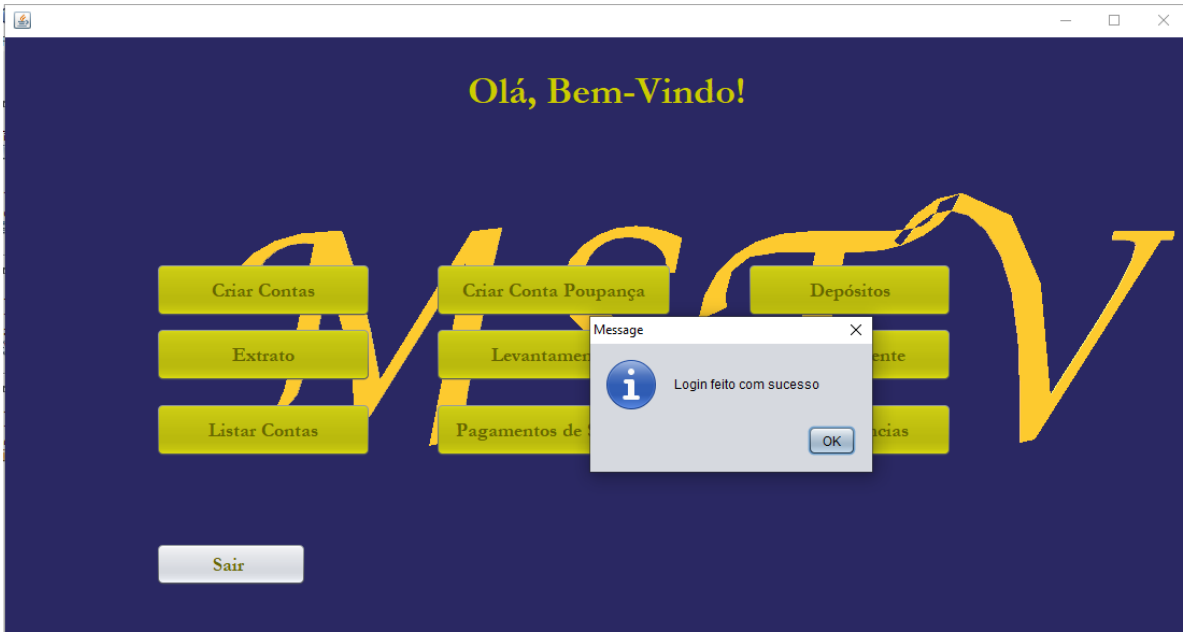

Fig11-Tela principal

Fig12-Tela de Transferencia

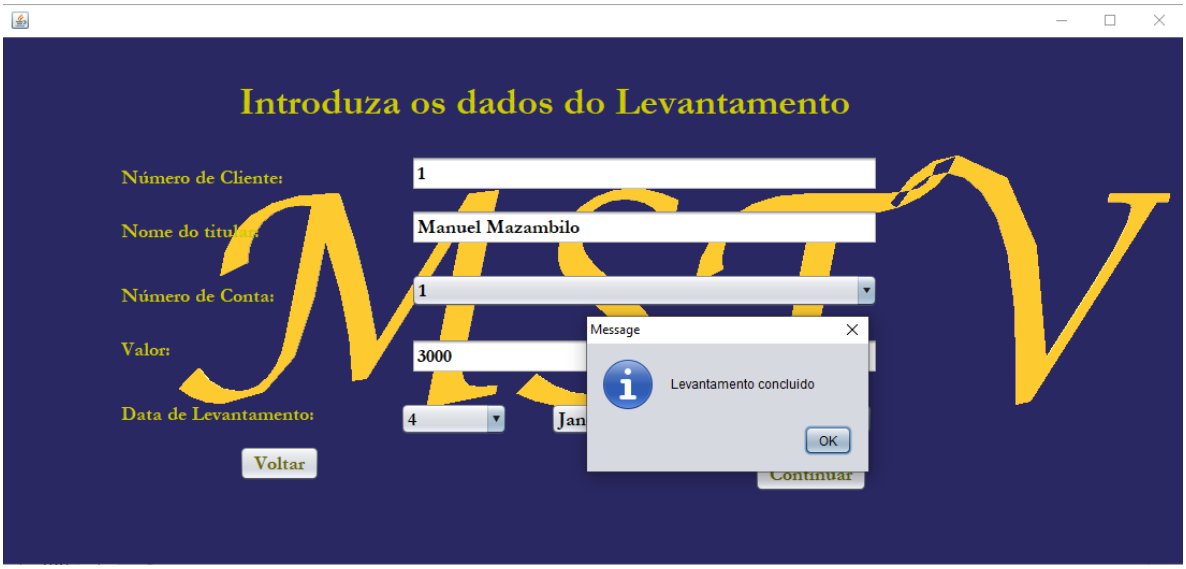

Fig13-Tela de levantamento

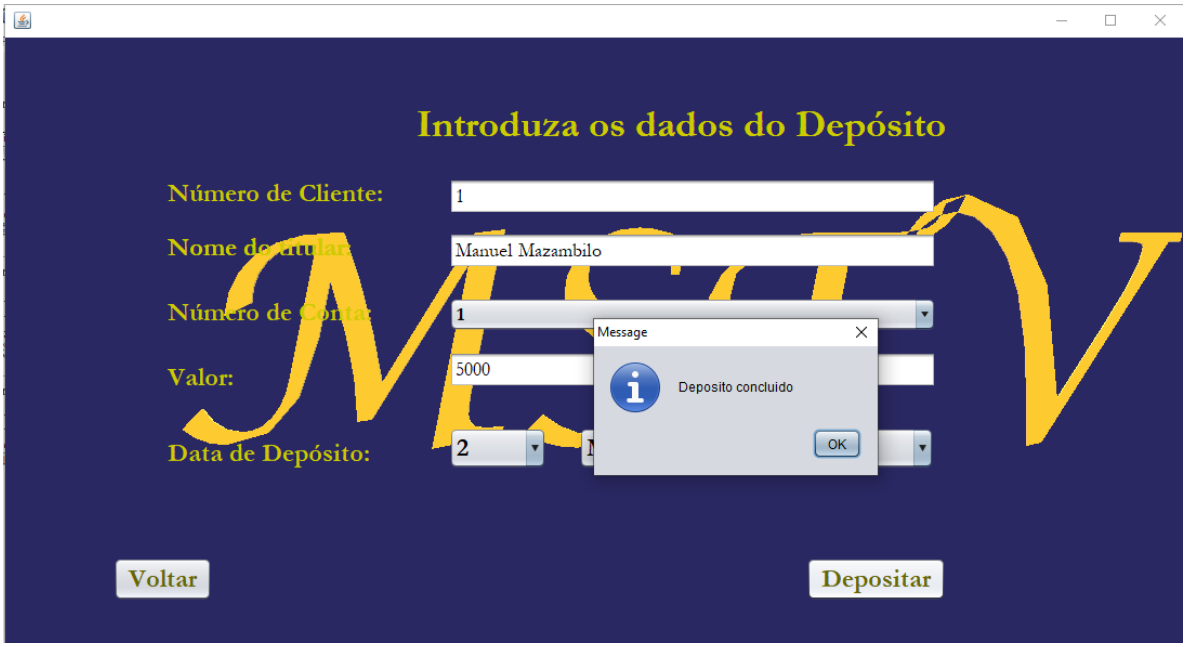

Fig14-Tela de deposito

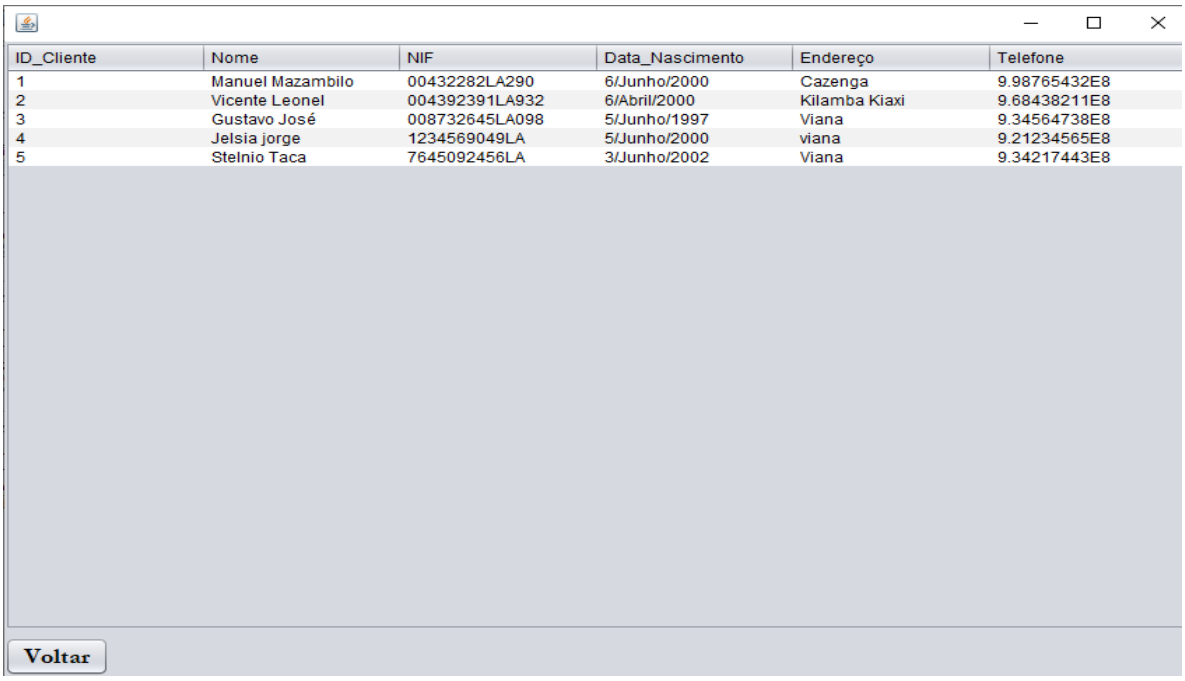

Fig15-Tela de listar cliente

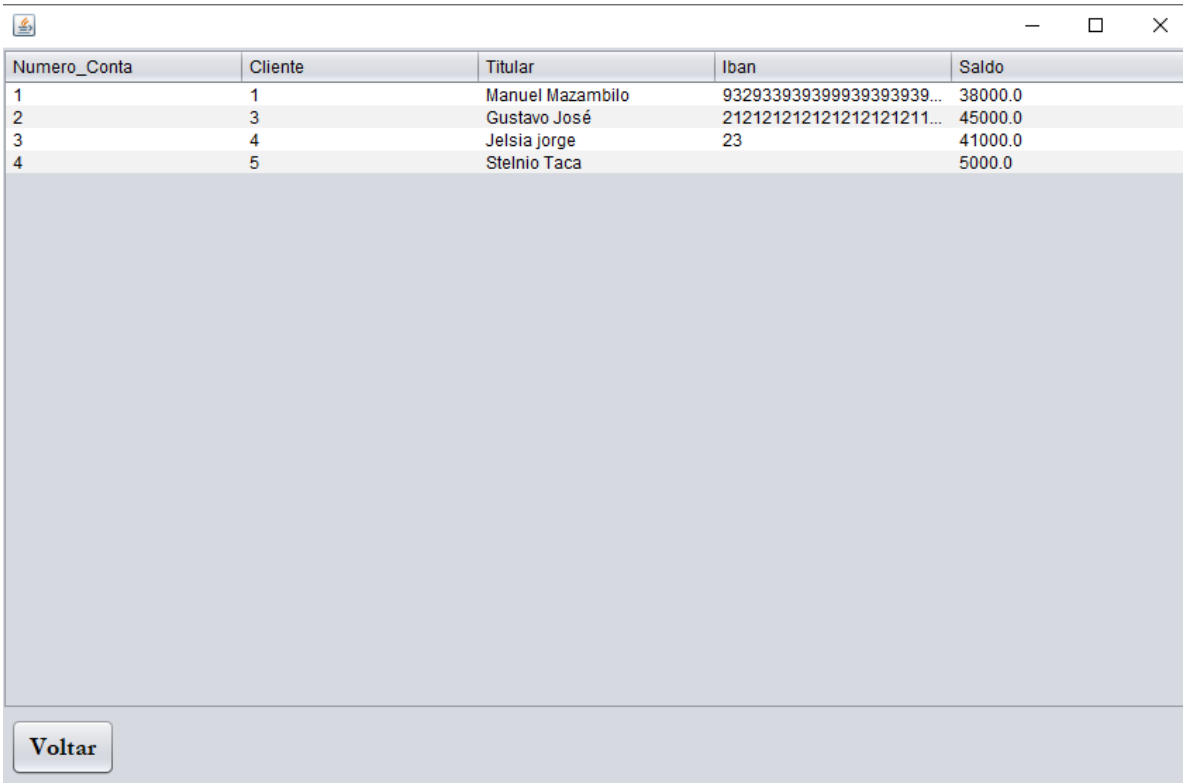

Fig16-Tela de listar conta

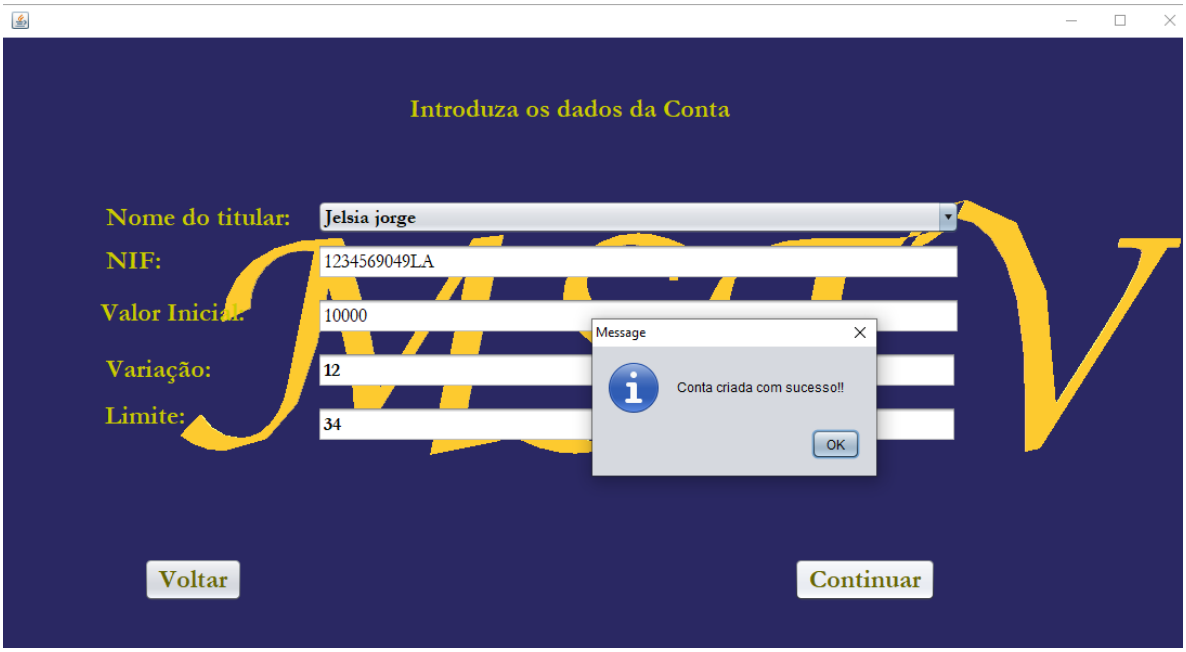

Fig17-Tela de Criar conta poupança

Fig18-Tela de criação de conta a funcionar

Fig19-Tela de cadastramento

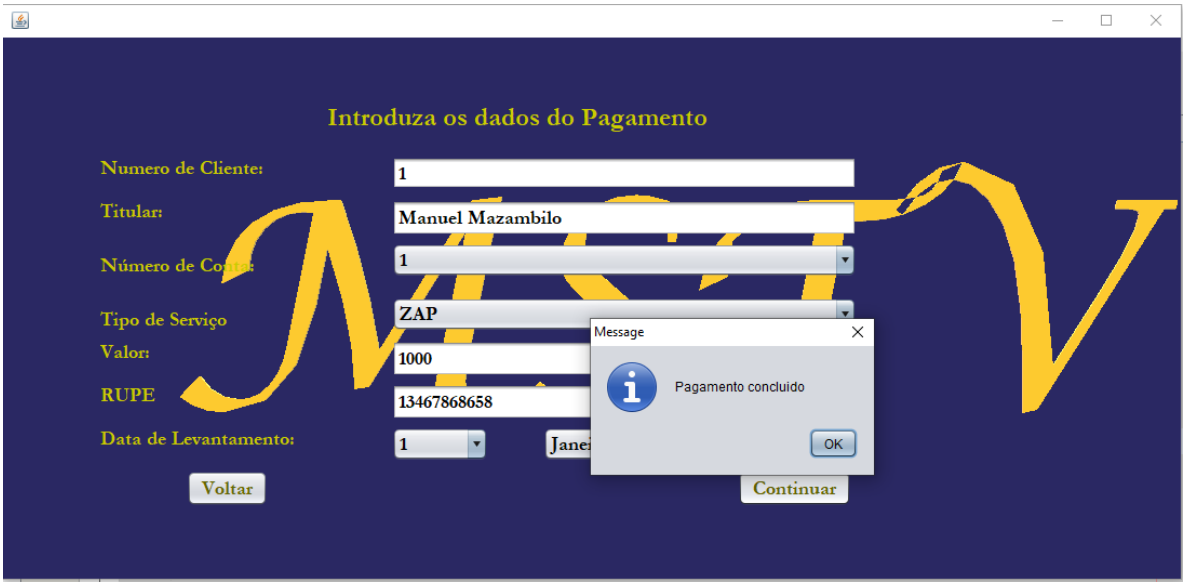

Fig20-Tela de pagamento