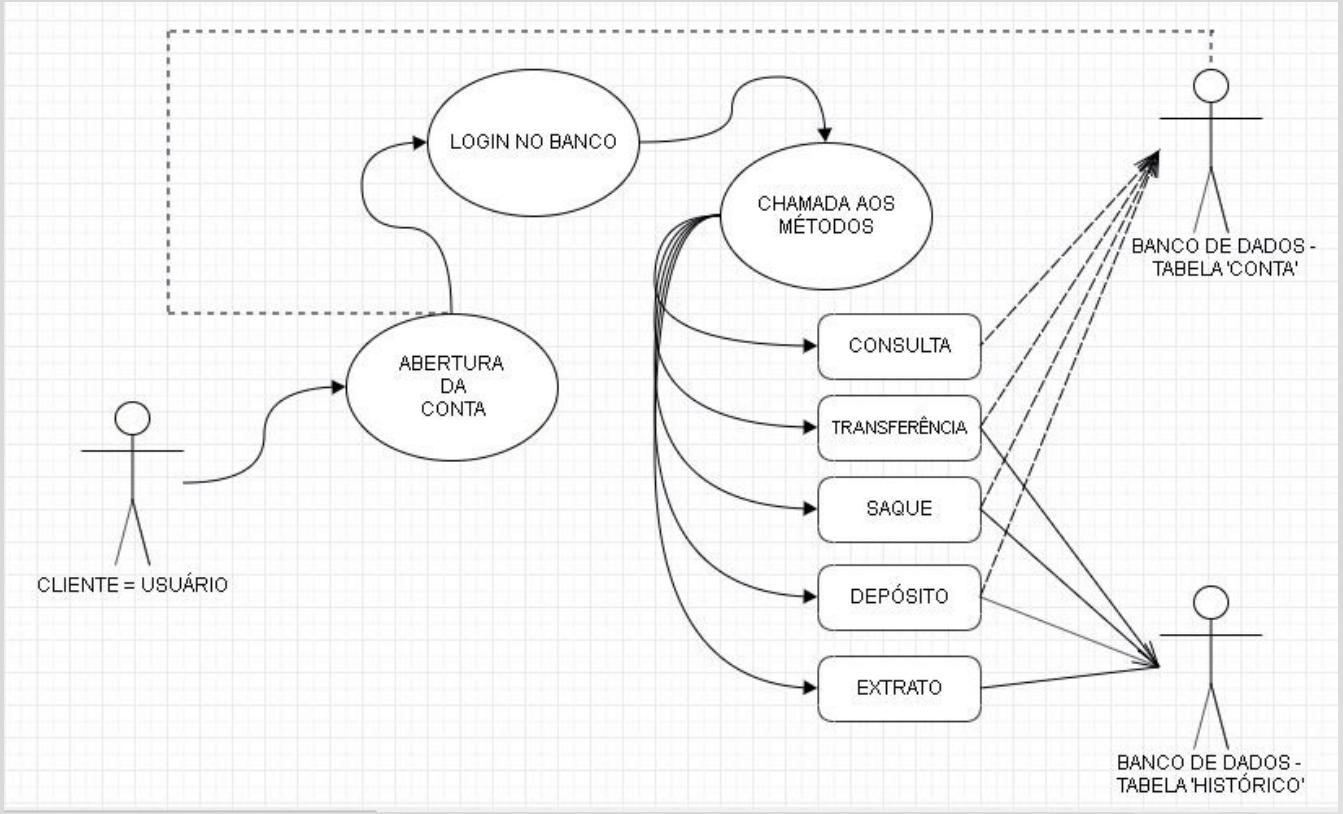
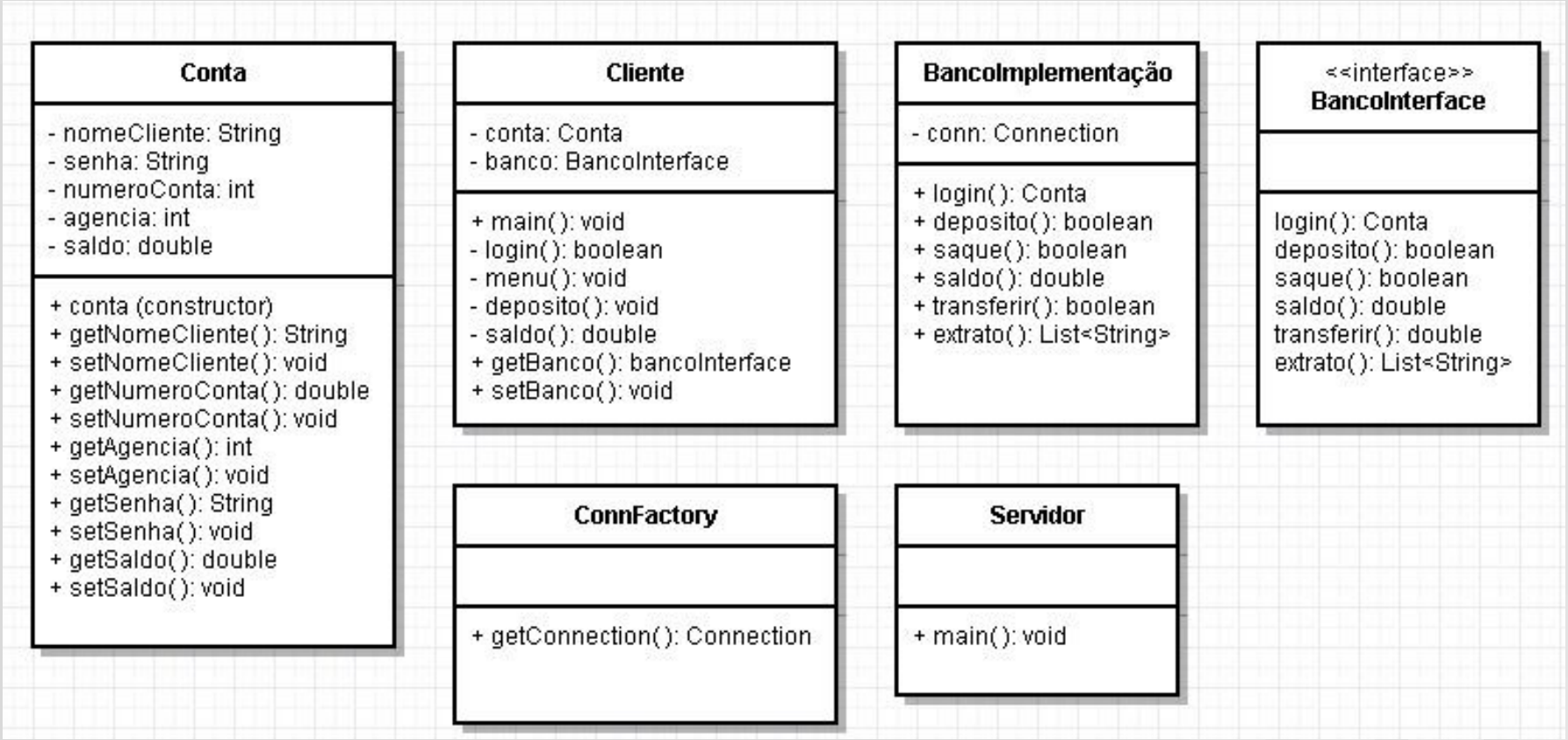
Sistema Bancário Distribuído

Márcio Rodrigues Filho
Lorenzo Antônio Leite
Lucas Wallace Nascimento

Casos de Uso

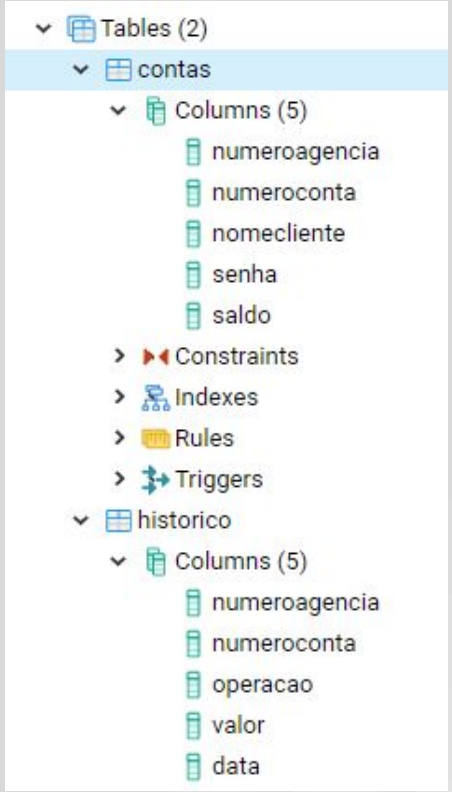

LOGIN NO BANCO
CHAMADA AOS MÉTODOS
ABERTURA DA CONTA
CLIENTE = USUÁRIO
CONSULTA
TRANSFERÊNCIA
SAQUE
DEPÓSITO
EXTRATO
BANCO DE DADOS - TABELA 'CONTA'
BANCO DE DADOS - TABELA 'HISTÓRICO'

UML

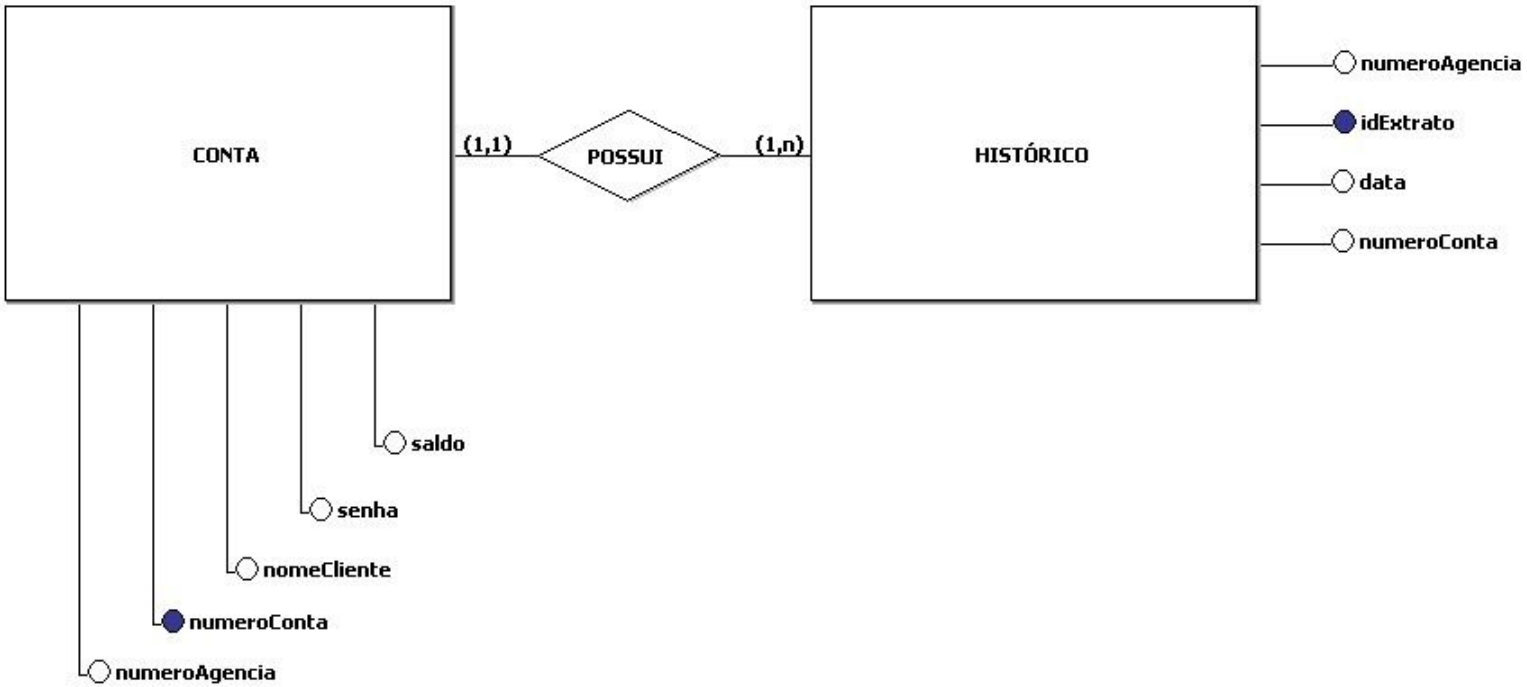

Conta
- nomeCliente: String
- senha: String
- numeroConta: int
- agencia: int
- saldo: double
+ conta (constructor)
+ getNomeCliente(): String
+ setNomeCliente(): void
+ getNumeroConta(): double
+ setNumeroConta(): void
+ getAgencia(): int
+ setAgencia(): void
+ getSenha(): String
+ setSenha(): void
+ getSaldo(): double
+ setSaldo(): void
Cliente
- conta: Conta
- banco: BancoInterface
+ main(): void
- login(): boolean
- menu(): void
- deposito(): void
- saldo(): double
+ getBanco(): bancoInterface
+ setBanco(): void
BancoImplementação
- conn: Connection
+ login(): Conta
+ deposito(): boolean
+ saque(): boolean
+ saldo(): double
+ transferir(): boolean
+ extrato(): List<String>
<<interface>>
BancoInterface
login(): Conta
deposito(): boolean
saque(): boolean
saldo(): double
transferir(): double
extrato(): List<String>
ConnFactory
+ getConnection(): Connection
Servidor
+ main(): void

Duas Tabelas

- Tabela Contas
  - Nome, Agência, Número da Conta, senha, saldo
- Tabela Histórico
  - Agência, Número da Conta, Data (timestamp), Valor da operação, tipo de operação



DER


CONTA
(1,1)
POSSUI
(1,n)
HISTÓRICO
numeroAgencia
idExtrato
data
numeroConta
saldo
senha
nomeCliente
numeroConta
numeroAgencia

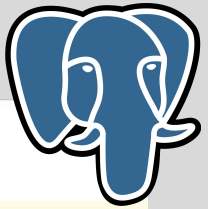
Tecnologias:

JAVA RMI

```java
package Servidor;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.sql.SQLException;
import java.util.ArrayList;

public interface BancoInterface extends Remote {

    Conta login(Conta dadosLogin) throws RemoteException;
    boolean deposito(int Agencia, int numeroConta, double valor) throws RemoteException;
    boolean saque(int Agencia, int numeroConta, double valor) throws RemoteException;
    double saldo(int agencia, int numeroConta) throws RemoteException;
    double transferir(int agenciaOrigem, int numeroContaOrigem, double valor, int agenciaDestino, int numeroContaDestino) throws  RemoteException;
    ArrayList<String> extrato(int Agencia, int numeroConta) throws RemoteException;

}
```

Tecnologias:

JDBC + POSTGRE



```java
package Servidor;

import java.sql.*;

class ConnFactory {

    public Connection getConnection() throws SQLException {
        try {

            Class.forName("org.postgresql.Driver");
            Connection conn;
            conn = DriverManager.getConnection( url: "jdbc:postgresql://localhost/banco?user=postgres&password=admin");
            return conn;

        } catch (ClassNotFoundException e) {
            throw new SQLException(e.getMessage());
        }
    }

    public void runSQL() {

    }
}
```

Servidores

```java
public class BalanceamentoServidor extends UnicastRemoteObject implements BancoInterface {
    private static Integer connCounter = 0;
    private BancoInterface banco1, banco2;

    protected BalanceamentoServidor() throws RemoteException, NotBoundException {
        super();
        connCounter++;
        Registry registry1, registry2;
        registry1 = LocateRegistry.getRegistry( host: "localhost",  port: 27017);
        registry2 = LocateRegistry.getRegistry( host: "192.168.0.10",  port: 27017);
        this.banco1 = (BancoInterface) registry1.lookup( name: "banco");
        this.banco2 = (BancoInterface) registry2.lookup( name: "banco");
    }

    public static void main(String[] args) {
        try {
            BalanceamentoServidor bsv = new BalanceamentoServidor();
            Registry registry = LocateRegistry.createRegistry( port: 27015);
            registry.rebind( name: "banco", bsv);
            System.out.println("=== BALANCEAMENTO INICIADO ===");
        } catch (Exception e) {
            System.out.println("Caixa erro: " + e.getMessage());
        }

    }

    @Override
    public Conta login(Conta contaLogin) throws RemoteException {
        connCounter++;
        return (connCounter % 2 == 0) ? this.banco1.login(contaLogin) : this.banco2.login(contaLogin);
    }
```

Implementação no Banco

```java
public boolean deposito(int numeroagencia, int numeroconta, double valor) throws RemoteException {
    try {
        if (this.conn.isClosed()) {
            this.conn = this.db.getConnection();
        }

        if( valor >= 0) {
            StringBuilder sql = new StringBuilder();
            sql.append("UPDATE contas SET saldo=saldo+? WHERE numeroagencia=? AND numeroconta=?");

            PreparedStatement ps = this.conn.prepareStatement(sql.toString());
            ps.setDouble( parameterIndex: 1, valor);
            ps.setInt( parameterIndex: 2, numeroagencia);
            ps.setInt( parameterIndex: 3, numeroconta);
            ps.executeUpdate();

            sql.setLength(0);
            ps.clearParameters();
            sql.append("INSERT INTO historico (numeroagencia, numeroconta, data, valor, operacao) VALUES (?,?,?,?,?)");
            ps = this.conn.prepareStatement(sql.toString());
            ps.setInt( parameterIndex: 1, numeroagencia);
            ps.setInt( parameterIndex: 2, numeroconta);
            ps.setTimestamp( parameterIndex: 3, ts);
            ps.setDouble( parameterIndex: 4, valor);
            ps.setString( parameterIndex: 5,  x: "DEPOSITO");
            ps.executeUpdate();
            return true;
```

Como Fizemos o Extrato

```java
@Override
public ArrayList<String> extrato(int Agencia, int numeroConta) throws RemoteException {
    try {

        if (this.conn.isClosed()) {
            this.conn = this.db.getConnection();
        }

        ArrayList<String> extrato = new ArrayList<~>();

        StringBuilder sql = new StringBuilder();
        sql.append("SELECT data, operacao, valor FROM historico WHERE numeroagencia=? AND numeroconta=? ORDER BY data DESC");
        PreparedStatement ps = this.conn.prepareStatement(sql.toString());
        ps.setInt( parameterIndex: 1, Agencia);
        ps.setInt( parameterIndex: 2, numeroConta);

        ResultSet rs = ps.executeQuery();

        while(rs.next()){
            extrato.add("Data: "+rs.getTimestamp( columnLabel: "data")+" / OPERAÇÃO: "+rs.getString( columnLabel: "operacao")+" / Valor: "+rs.getDouble( columnLabel: "valor"));
        }

        return extrato;

    }catch(Exception e){
        e.printStackTrace();
        return null;
    }

}
```

Vídeo