FACULDADE ESTÁCIO TERESINA

CURSO DE CIÊNCIA DA COMPUTAÇÃO

SISTEMA BANCÁRIO – PROJETO GERAL

Acadêmicos: Lyan Kaleu

Lucas Pinto

Cássio Henrique

Felipe Lima

Rodrigo Henrique

Gusthavo Silva

Ronald Santiago

Orientador: Prof. Joselito Mendes de Sousa Junior

TERESINA/PI. 2023

SUMÁRIO

1.0. INTRODUÇÃO

“O dinheiro é o cérebro da empresa, quem controla o dinheiro controla tudo”.

Desde os primórdios da humanidade, o sistema de trocas esteve presente na sociedade. Nesse contexto, podemos observar que qualquer bem material poderia ter valor, no Império Romano por exemplo, o pagamento de soldados era em "sal", diante dessa prática se originou os termos conhecidos hoje como "soldo" e "salário", entretanto, não existia uma maneira de acumular e armazenar esse salário como temos nos sistemas bancários atualmente.

Por muitos anos, o sistema bancário vem se expandindo exponencialmente. No entanto, nem todos possuem recursos suficientes para atender às demandas do mercado bancário que evolui conforme a sociedade. Pensando nisso, a proposta deste trabalho é criar um sistema bancário que possa auxiliar e instruir a população na melhor forma de gerir, investir e proteger seu dinheiro.

2.0. MOTIVAÇÃO ACADÊMICA E JUSTIFICATIVA

Tendo em vista que o Plano Nacional de Educação - PNE 2014 - 2024, a Extensão na Educação Superior é entendida como um processo interdisciplinar, político educacional, cultural, científico, tecnológico, que ajuda a promover a interação transformadora entre a Instituição de Ensino Superior (IES) e os outros setores da sociedade. A extensão é caracterizada pela produção e aplicação do conhecimento, em articulação permanente com o ensino e a pesquisa e envolve toda a comunidade acadêmica, na figura dos docentes, discentes e corpo técnico-administrativo.

Após identificado a situação problema descrita acima considera-se o tema importante, pois o Sistema Bancário permite uma melhor gestão dos recursos financeiros, visto que é possível alocar capitais investidos, transferências, fundos de conta poupança e corrente, de forma eficaz, facilitando o armazenamento e a busca de informações em situações futuras.

Isso poderá futuramente ser melhorado, servindo como base para novas pesquisas e implementações, sendo relevante para a comunidade acadêmica no processo de aplicação prática dos conteúdos desenvolvidos no âmbito teórico-científico.

3.0. OBJETIVOS

3.1. Como objetivo geral, podemos citar o SFN (Sistema Financeiro Nacional), como exemplo a ser seguido para atender as demandas de um usuário, visto que realiza a fiscalização, regulamentação e execução de operações referentes à circulação da moeda e do crédito dentro da economia.

3.2. Como objetivos específicos, temos a prática dos conceitos abordados na disciplina de Programação Orientada a Objetos em Java, além de uma melhor organização das informações de um banco.

4.0. RESULTADOS ESPERADOS

Espera-se que, com este trabalho, possamos apresentar um sistema que impacte na realidade das pessoas que dependem de sistemas bancários na nossa cidade.

Com o sucesso da nossa proposta, podemos implementar novidades e expandir a utilização do sistema por todo o Estado.

5.0. METODOLOGIA

A proposta é implementar um sistema bancário que simule as relações existentes entre contas. O projeto deve ter, no mínimo, as classes: CONTABANCARIA, CONTACORRENTE e CONTAPOUPANÇA, além do USUÁRIO. A classe ContaBancaria é a superclasse, enquanto as classes ContaCorrente e ContaPoupança herdam da classe ContaBancaria. Além disso, a classe ContaBancaria é uma classe abstrata e implementa a interface MovimentacaoBancaria.

O sistema também deverá fazer algumas operações básicas, como cadastrar uma nova pessoa na instituição (aluno ou funcionário), excluir uma pessoa e atualizar os dados desta.

As operações e classes citadas anteriormente são essenciais para um bom funcionamento da nossa proposta. No entanto, temos as funcionalidades mínimas e necessárias de cada classe individualmente:

- A classe de ContaBancaria deve fornecer um construtor para inicializar esses atributos e métodos getters e setters para acessá-los. Além disso, ela deve implementar os métodos da interface MovimentacaoBancaria.

- A classe de ContaCorrente deve herdar da classe ContaBancaria. Ela deve fornecer um atributo privado (LimiteChequeEspecial (double)). Ela deve fornecer um construtor para inicializar os atributos herdados da classe ContaBancaria e o atributo LimiteChequeEspecial. Além disso, ela deve sobrescrever o método Sacar(valor) para permitir saques, mesmo que o saldo seja insuficiente, desde que o valor esteja dentro do limite do cheque especial. Caso contrário, a exceção SaldoInsuficienteException deve ser lançada.

- A classe de ContaPoupanca deve ser uma classe que herda da classe ContaBancaria. Ela deve fornecer um atributo privado (taxaRendimento (double)). Ela deve fornecer um construtor para inicializar os atributos herdados da classe ContaBancaria e o atributo taxaRendimento. Ela deve conter um método calcularRendimento() que não recebe parâmetros e retorna um valor double.

- A classe Usuário deve permitir que a pessoa acesse cadastre uma conta e interaja com todas as demais classes. Sendo possível realizar os procedimentos mostrados acima de acordo com a escolha do usuário. Esta foi a apresentação básica do nosso trabalho. Todas as classes, atributos e

métodos que puderam ser visualizados anteriormente, são requisitos mínimos de funcionamento do nosso sistema. Assim, tentaremos viabilizar a implementação de funcionalidades extras, além de uma interface gráfica eficaz para a utilização dos recursos oferecidos.

5.1. Identificação do público participante (cenário e amostra)

O públicoalvo do nosso projeto são escolas da cidade de Teresina – Piauí, preferencialmente ligadas ao Município e Estado, visto que possuem condições mais precárias de funcionamento, face às escolas da rede privada de ensino.

A escola utilizada como base de apresentação do sistema é a Unidade Escolar Prefeito Freitas Neto. A Unidade Escolar Prefeito Freitas Neto é uma escola pública em Teresina/PI, no bairro Buenos Aires. Oferece educação especial, ensino fundamental, ensino fundamental - anos finais 6º ao 9º e ensino médio. A escola também oferece toda a estrutura necessária para o conforto e desenvolvimento educacional dos seus alunos, como por exemplo: Alimentação, Auditório, Laboratório de informática, Pátio Coberto, Área Verde, Quadra Esportiva Coberta, Biblioteca, Berçário, Quadra Esportiva Descoberta, Parquinho, Sala de leitura, Refeitório, Laboratório de ciências, Sala de professores, Pátio Descoberto, Banda larga, Internet, Lixo reciclável.

5.2. Descrição da atuação da equipe de trabalho

O desenvolvimento do Sistema Bancário durante a matéria Programação Orientada a Objetos JAVA serviu para discutir ideias e delimitar funções de cada um dos membros do grupo. Lyan Kaleu foi o responsável pela criação do código fonte e da interface gráfica, Rodrigo Henrique pelo desenvolvimento do site que serviu e serve de suporte para o sistema bancário, Gusthavo Lima produziu o diagrama que deu início ao projeto, Cássio Henrique realizou a criação dos slides para apresentação, enquanto Ronald Santiago apresentou e por fim Lucas Pinto e Felipe Lima fizeram o relatório geral do projeto.

5.3. Tecnologias utilizadas

A principal tecnologia a ser utilizada na nossa proposta é Java, a linguagem de programação referência desta disciplina. Java é uma linguagem de programação orientada a objetos desenvolvida na década de 90 por uma equipe de programadores chefiada por

James Gosling, na empresa *Sun Microsystems*, que em 2008 foi adquirido pela empresa *Oracle Comporation*. Diferente das linguagens de programação modernas, que são compiladas para código nativo, Java é compilada para um *bytecode* que é interpretado por uma máquina virtual (*Java Virtual Machine*, JVM).

A linguagem Java foi projetada tendo em vista os seguintes objetivos:

- Orientação a Objetos - Baseado no modelo de Simular;
- Portabilidade - Independência de plataforma - "escreva uma vez, execute em qualquer lugar" (*"write once, run anywhere"*);
- Recursos de Rede - Possui extensa biblioteca de rotinas que facilitam a cooperação com protocolos TCP/IP, como HTTP e FTP;
- Segurança - Pode executar programas via rede com restrições de execução.

Para o desenvolvimento do sistema, além do Java, utilizamos o Eclipse como interface de desenvolvimento. Eclipse é uma IDE para desenvolvimento Java, porém suporta várias outras linguagens de programação, como C/C++, Python, Kotlin, dentre outras. Ele foi feito em Java e segue o modelo open source de desenvolvimento de software.

O projeto Eclipse foi iniciado na IBM que desenvolveu a primeira versão do produto e doou-o como software livre para a comunidade. O gasto inicial da IBM no produto foi de mais de 40 milhões de dólares. Hoje, o Eclipse é o IDE Java mais utilizado no mundo. Possui como característica marcante o uso da SWT e não do Swing como biblioteca gráfica, a forte orientação ao desenvolvimento baseado em plugins e o amplo suporte ao desenvolvedor com centenas de plug-ins que procuram atender as diferentes necessidades de diferentes programadores.

Outras tecnologias utilizadas no projeto foram as ferramentas Astah Community, LucidChart, Wordpress, Hostinger e domínio.support. Estas são ferramentas gratuitas para a criação de diagramas de classes e casos de uso referentes ao projeto, e as três últimas citadas são as tecnologias utilizadas para o desenvolvimento do site suporte do sistema bancário.

6.0. PLANO DE TRABALHO (CRONOGRAMA)

Elaboração do Diagrama do Projeto	Dia 10/ 09/ 2023 até o dia 18/ 09/ 2023	Definição de classes a serem usadas no código.
Desenvolvimento do Sistema Bancário (Feito em POO em JAVA)	Dia 18/ 09/ 2023 até o dia 19/ 10/ 2023	Após o diagrama, o código foi desenvolvido para funcionar conforme a etapa anterior.
Criação do site suporte para o Sistema Bancário	Dia 21/ 10/ 2023 até o dia 05/ 11/ 2023	Mediante o sucesso das etapas anteriores, um site para auxiliar os usuários foi desenvolvido em WordPress.
Conclusão do Projeto	Dia 05/ 11/ 2023 até o dia 13/ 11/ 2023	O projeto foi finalizado com o relatório geral, o relatório individual de cada membro do projeto e a apresentação na escola.

7.0. REFERENCIAL BIBLIOGRÁFICO

- DEITEL, P. J.; DEITEL, H. M. **Java:** como programar. 10. ed. São Paulo: Pearson, 2017.
- NAUGTHON, Patrick. **Dominando o Java, Guia Autorizado da Sun Microsystems.** [S.l.]: Editora Makron Books. 1997.
- VERSOLATTO, Fabio. **Sistemas orientados a objetos:** conceitos e práticas. 1. ed. Rio de Janeiro: Freitas Bastos, 2023.

FACULDADE ESTÁCIO TERESINA

CURSO DE CIÊNCIA DA COMPUTAÇÃO

RELATÓRIO GERAL DO SISTEMA BANCÁRIO

Acadêmicos: Lyan Kaleu

Lucas Pinto

Cássio Henrique

Felipe Lima

Rodrigo Henrique

Gusthavo Silva

Ronald Santiago

Orientador: Prof. Joselito Mendes de Sousa Junior

TERESINA/PI 2023

SUMÁRIO

1.0. INTRODUÇÃO

“O dinheiro é o cérebro da empresa, quem controla o dinheiro controla tudo”.

Desde os primórdios da humanidade, o sistema de trocas esteve presente na sociedade. Nesse contexto, podemos observar que qualquer bem material poderia ter valor, no Império Romano por exemplo, o pagamento de soldados era em "sal", diante dessa prática se originou os termos conhecidos hoje como "soldo" e "salário", entretanto, não existia uma maneira de acumular e armazenar esse salário como temos nos sistemas bancários atualmente.

Por muitos anos, o sistema bancário vem se expandindo exponencialmente. No entanto, nem todos possuem recursos suficientes para atender às demandas do mercado bancário que evolui conforme a sociedade. Pensando nisso, a proposta deste trabalho é criar um sistema bancário que possa auxiliar e instruir a população na melhor forma de gerir, investir e proteger seu dinheiro.

2.0. MOTIVAÇÃO ACADÊMICA E JUSTIFICATIVA

Tendo em vista que o Plano Nacional de Educação - PNE 2014 - 2024, a Extensão na Educação Superior é entendida como um processo interdisciplinar, político educacional, cultural, científico, tecnológico, que ajuda a promover a interação transformadora entre a Instituição de Ensino Superior (IES) e os outros setores da sociedade. A extensão é caracterizada pela produção e aplicação do conhecimento, em articulação permanente com o ensino e a pesquisa e envolve toda a comunidade acadêmica, na figura dos docentes, discentes e corpo técnico-administrativo.

Após identificado a situação problema descrita acima considera-se o tema importante, pois o Sistema Bancário permite uma melhor gestão dos recursos financeiros, visto que é possível alocar capitais investidos, transferências, fundos de conta poupança e corrente, de forma eficaz, facilitando o armazenamento e a busca de informações em situações futuras.

Isso poderá futuramente ser melhorado, servindo como base para novas pesquisas e implementações, sendo relevante para a comunidade acadêmica no processo de aplicação prática dos conteúdos desenvolvidos no âmbito teórico-científico.

3.0. OBJETIVOS

3.1. Como objetivo geral, podemos citar o SFN(Sistema Financeiro Nacional), como exemplo a ser seguido para atender as demandas de um usuário, visto que realiza a fiscalização, regulamentação e execução de operações referentes à circulação da moeda e do crédito dentro da economia.

3.2. Como objetivos específicos, temos a prática dos conceitos abordados na disciplina de Programação Orientada a Objetos em Java, além de uma melhor organização das informações de um banco.

4.0. METODOLOGIA

A proposta é implementar um sistema bancário que simule as relações existentes entre contas. O projeto deve ter, no mínimo, as classes: CONTABANCARIA,

CONTACORRENTE e CONTAPOUPANÇA, além do USUÁRIO. A classe ContaBancaria é a superclasse, enquanto as classes ContaCorrente e ContaPoupança herdam da classe ContaBancaria. Além disso, a classe ContaBancaria é uma classe abstrata e implementa a interface MovimentacaoBancaria.

O sistema também deverá fazer algumas operações básicas, como cadastrar uma nova pessoa na instituição (aluno ou funcionário), excluir uma pessoa e atualizar os dados desta.

As operações e classes citadas anteriormente são essenciais para um bom funcionamento da nossa proposta. No entanto, temos as funcionalidades mínimas e necessárias de cada classe individualmente:

- A classe de ContaBancaria deve fornecer um construtor para inicializar esses atributos e métodos getters e setters para acessá-los. Além disso, ela deve implementar os métodos da interface MovimentacaoBancaria.
- A classe de ContaCorrente deve herdar da classe ContaBancaria. Ela deve fornecer um atributo privado (LimiteChequeEspecial (double)).

  Ela deve fornecer um construtor para inicializar os atributos herdados da classe ContaBancaria e o atributo LimiteChequeEspecial. Além disso, ela deve sobrescrever o método Sacar(valor) para permitir saques, mesmo que o saldo seja insuficiente, desde que o valor esteja dentro do limite do cheque especial. Caso contrário, a exceção SaldoInsuficienteException deve ser lançada.

- A classe de ContaPoupanca deve ser uma classe que herda da classe ContaBancaria. Ela deve fornecer um atributo privado(taxaRendimento (double)). Ela deve fornecer um construtor para inicializar os atributos herdados da classe ContaBancaria e o atributo taxaRendimento. Ela deve conter um método calcularRendimento() que não recebe parâmetros e retorna um valor double.
- A classe Usuário deve permitir que a pessoa acesse cadastre uma conta e interaja com todas as demais classes. Sendo possível realizar os procedimentos mostrados acima de acordo com a escolha do usuário

- Em conjunto com a definição das classes, precisamos também montar um diagrama, cujo a função além de dar início ao projeto, é também definir onde vai se encaixar cada classe dentro do desenvolvimento do Sistema Bancário.

Esta foi a apresentação básica do nosso trabalho. Todas as classes, atributos e métodos que puderam ser visualizados anteriormente, são requisitos mínimos de funcionamento do nosso sistema. Assim, tentaremos viabilizar a implementação de funcionalidades extras, além de uma interface gráfica eficaz para a utilização dos recursos oferecido

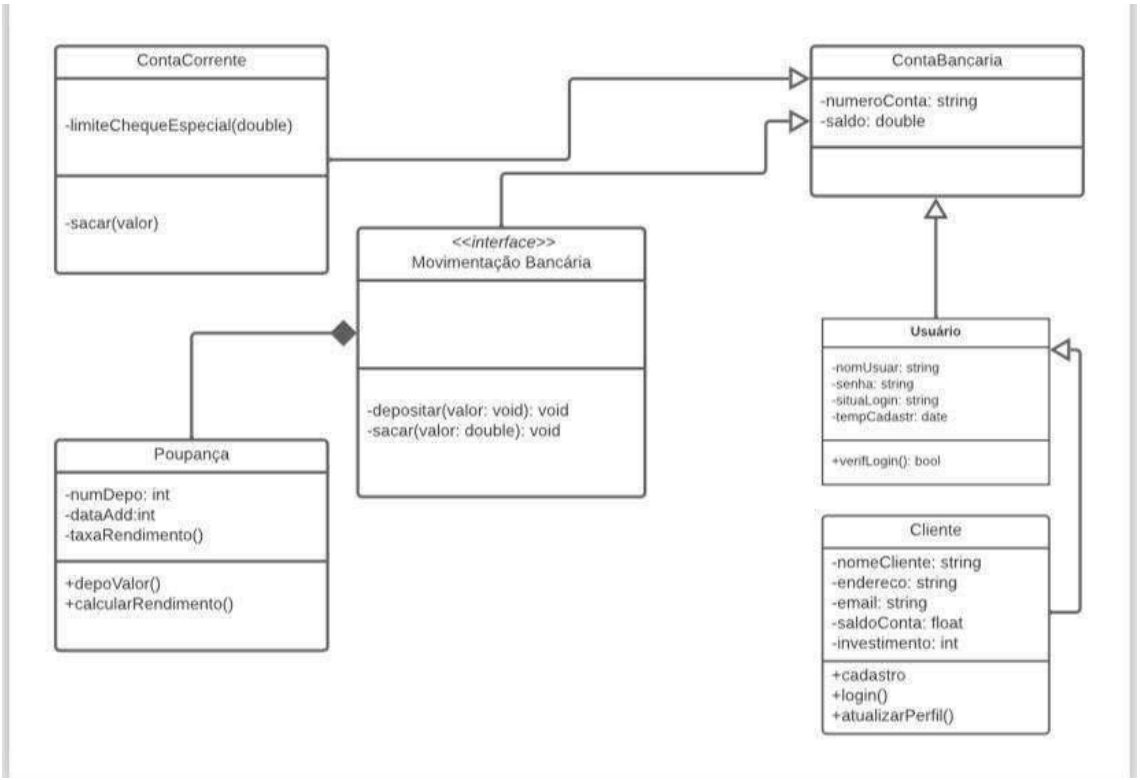


DIAGRAMA DE CLASSES

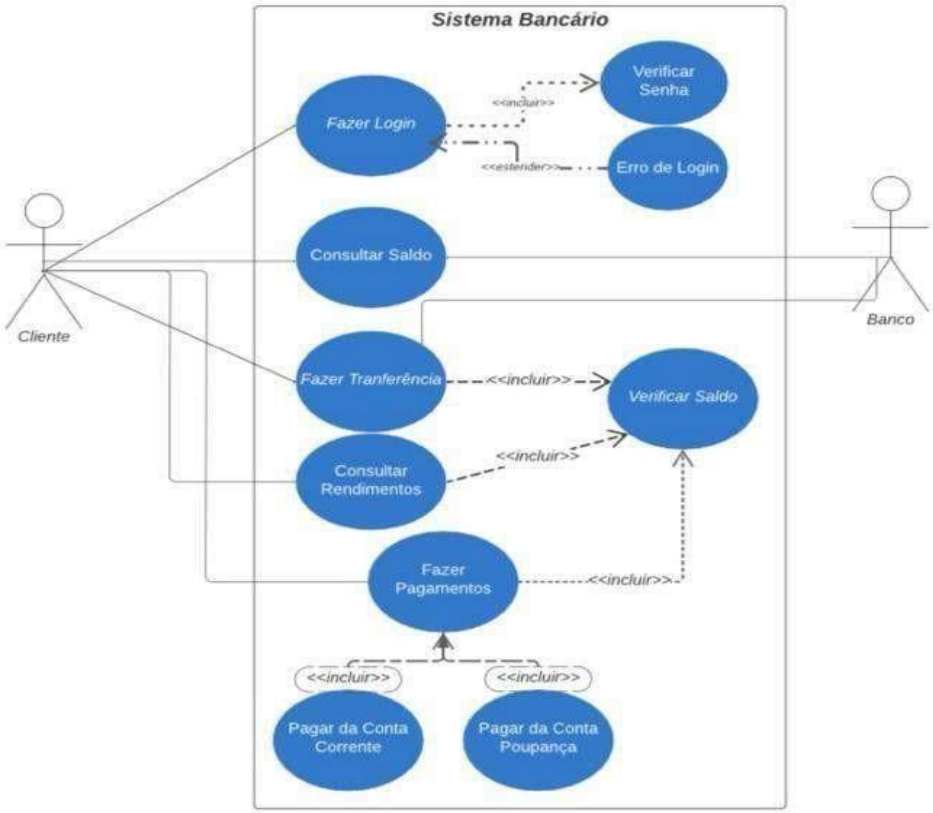


DIAGRAMA DE CASOS DE TESTE

Como podemos ver, nas duas imagens acima vemos dois diagramas: O de classe que vai organizar onde cada vai ficar no desenvolvimento do código e o de Casos de Teste, onde vai fornecer uma visão geral da interação do Banco com o Cliente.

4.1. Identificação do público participante (cenário e amostra)

O público alvo do nosso projeto são escolas da cidade de Teresina – Piauí, preferencialmente ligadas ao Município e Estado, visto que possuem condições mais precárias de funcionamento, face às escolas da rede privada de ensino.

A escola utilizada como base de apresentação do sistema é a Unidade Escolar Prefeito Freitas Neto. A Unidade Escolar Prefeito Freitas Neto é uma escola pública em Teresina/PI, no bairro Buenos Aires. Oferece educação especial, ensino fundamental, ensino fundamental - anos finais 6º ao 9º e ensino médio. A escola também oferece toda a estrutura necessária para o conforto e desenvolvimento educacional dos seus alunos, como por exemplo: Alimentação, Auditório, Laboratório de informática, Pátio Coberto, Área Verde, Quadra Esportiva Coberta, Biblioteca, Berçário, Quadra Esportiva Descoberta, Parquinho, Sala de leitura, Refeitório, Laboratório de ciências, Sala de professores, Pátio Descoberto, Banda larga, Internet, Lixo reciclável.

4.2. Descrição da atuação da equipe de trabalho

O desenvolvimento do Sistema Bancário durante a matéria Programação Orientada a Objetos JAVA serviu para discutir ideias e delimitar funções de cada um dos membros do grupo. Lyan Kaleu foi o responsável pela criação do código fonte e da interface gráfica, Rodrigo Henrique pelo desenvolvimento do site que serviu e serve de suporte para o sistema bancário, Gusthavo Lima produziu o diagrama que deu início ao projeto, Cássio Henrique realizou a criação dos slides para apresentação, enquanto Ronald Santiago apresentou e por fim Lucas Pinto e Felipe Lima fizeram o relatório geral do projeto.

4.3. Tecnologias utilizadas

A principal tecnologia a ser utilizada na nossa proposta é Java, a linguagem de programação referência desta disciplina. Java é uma linguagem de programação orientada a objetos desenvolvida na década de 90 por uma equipe de programadores chefiada por James Gosling, na empresa *Sun Microsystems*, que em 2008 foi adquirido pela empresa

Oracle Comporation. Diferente das linguagens de programação modernas, que são compiladas para código nativo, Java é compilada para um *bytecode* que é interpretado por uma máquina virtual (*Java Virtual Machine*, JVM).

A linguagem Java foi projetada tendo em vista os seguintes objetivos:

- Orientação a Objetos - Baseado no modelo de Simular;
- Portabilidade - Independência de plataforma - "escreva uma vez, execute em qualquer lugar" (*"write once, run anywhere"*);
- Recursos de Rede - Possui extensa biblioteca de rotinas que facilitam a cooperação com protocolos TCP/IP, como HTTP e FTP;
- Segurança - Pode executar programas via rede com restrições de execução.

Para o desenvolvimento do sistema, além do Java, utilizamos o Eclipse como interface de desenvolvimento. Eclipse é uma IDE para desenvolvimento Java, porém suporta várias outras linguagens de programação, como C/C++, Python, Kotlin, dentre outras. Ele foi feito em Java e segue o modelo open source de desenvolvimento de software.

O projeto Eclipse foi iniciado na IBM que desenvolveu a primeira versão do produto e doou-o como software livre para a comunidade. O gasto inicial da IBM no produto foi de mais de 40 milhões de dólares. Hoje, o Eclipse é o IDE Java mais utilizado no mundo. Possui como característica marcante o uso da SWT e não do Swing como biblioteca gráfica, a forte orientação ao desenvolvimento baseado em plugins e o amplo suporte ao desenvolvedor com centenas de plug-ins que procuram atender as diferentes necessidades de diferentes programadores.

Outras tecnologias utilizadas no projeto foram as ferramentas Astah Community, LucidChart, Wordpress, Hostinger e domínio.support. Estas são ferramentas gratuitas para a criação de diagramas de classes e casos de uso referentes ao projeto, e as três últimas citadas são as tecnologias utilizadas para o desenvolvimento do site suporte do sistema bancário.

5.0. RESULTADOS ESPERADOS

Espera-se que, com este trabalho, possamos apresentar um sistema que impacte na realidade das pessoas que dependem de sistemas bancários na nossa cidade.

O sucesso da nossa proposta, podemos implementar novidades e expandir a utilização do sistema por todo o Estado. O nosso sistema também tem o intuito de ser um código acessível para que ele possa ser reutilizado devido a POO, e assim sendo, ele pode também ser aprimorado por outros profissionais da área no momento de sua expansão.

6.0. RESULTADOS OBTIDOS

Após a definição de tarefas, a elaboração e o desenvolvimento do Sistema Bancário, tivemos um resultado muito positivo e eficiente para o problema proposto. Depois de muita discussão acerca sobre o que esperávamos e o que fizemos, segue abaixo os resultados obtidos do nosso Projeto de desenvolvimento de Sistema Bancário:

```java
import java.io.Serializable;

public class ContaBancaria implements MovimentacaoBancaria, Serializable{
    // A classe ContaBancaria implementa a interface MovimentacaoBancaria,
    // o que significa que ela precisa fornecer implementações para todos os métodos definidos na interface.

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private String numeroConta; // Declaração de uma variável de instância 'numeroConta' que irá armazenar o número da conta bancária.
    private double saldo; // Declaração de uma variável de instância 'saldo' que irá armazenar o saldo da conta bancária.
    private String tipoConta;

    public ContaBancaria(String numeroConta, double saldo, String tipoConta) throws IllegalAccessException {
        // Construtor da classe. Recebe o número da conta e o saldo inicial como parâmetros.
        if(numeroConta == null || numeroConta.isEmpty()) {
            throw new IllegalAccessException("O número da conta não pode ser vazio ou nulo."); // Se o número da conta for nulo ou vazio, uma exceção IllegalAccessException é lançada.
        }
        this.numeroConta = numeroConta; // Inicializa o número da conta com o valor fornecido.
        this.saldo = saldo; // Inicializa o saldo com o valor fornecido.
        this.tipoConta = tipoConta;
    }

    // Getters e setters
    public String getNumeroConta() {
        return numeroConta;
    }

    public void setNumeroConta(String numeroConta) {
        this.numeroConta = numeroConta;
    }

    public double getSaldo() {
        return saldo;
    }

    public void setSaldo(double saldo) {
        this.saldo = saldo;
    }

    public String getTipoConta() {
        return tipoConta;
    }

    public void setTipoConta(String tipoConta) {
        this.tipoConta = tipoConta;
    }

    @Override // Implementação do método 'depositar' que adiciona o valor ao saldo.
    public void depositar(double valor) {
        this.saldo += valor;

    }

    @Override
    public void sacar(double valor) throws SaldoInsuficienteException{
        if(valor > saldo) {
            throw new SaldoInsuficienteException("Saldo insuficiente para saque."); // Se o valor do saque for maior que o saldo, uma exceção SaldoInsuficienteException é lançada.
        }
        this.saldo -= valor; // Caso contrário, o valor é subtraído do saldo.
    }

}
```

FIGURA 1

Nessa figura, como é notório, a **ClasseContaBancária** é a classe pai de onde as classes **ContaPoupança** e **ContaCorrente** vão herdar atributos, para assim poderem gerar funcionalidades para o sistema.

```java
public class ContaCorrente extends ContaBancaria implements MovimentacaoBancaria{
    // A classe ContaCorrente estende a classe ContaBancaria e implementa a interface MovimentacaoBancaria.

    /**
     *
     */
    private static final Long serialVersionUID = 1L;
    private double limiteChequeEspecial; // Declaração de uma variável de instância 'limiteChequeEspecial' que irá armazenar o limite do cheque especial.

    // Construtor da classe.
    public ContaCorrente(String numeroConta, double saldo, double limiteChequeEspecial) throws IllegalAccessException {
        super(numeroConta, saldo, numeroConta); // Chama o construtor da classe mãe (ContaBancária) passando o número da conta e saldo.
        this.limiteChequeEspecial = limiteChequeEspecial; // Inicializa o limite do cheque especial com o valor fornecido.
    }

    // Getters e setters
    public double getLimiteChequeEspecial() {
        return limiteChequeEspecial;
    }

    public void setLimiteChequeEspecial(double limiteChequeEspecial) {
        this.limiteChequeEspecial = limiteChequeEspecial;
    }

    // Método para realizar um saque. Pode lançar uma exceção SaldoInsuficienteException.
    public void sacar(double valor) throws SaldoInsuficienteException {
        if(valor > getSaldo() + limiteChequeEspecial) { // Se o valor do saque for maior que o saldo mais o limite do cheque especial...
            throw new SaldoInsuficienteException("Saldo insuficiente para conta"); // ...lança uma exceção indicando saldo insuficiente.
        }
        setSaldo(getSaldo() - valor); // Caso contrário, subtrai o valor do saldo.
    }


}

```

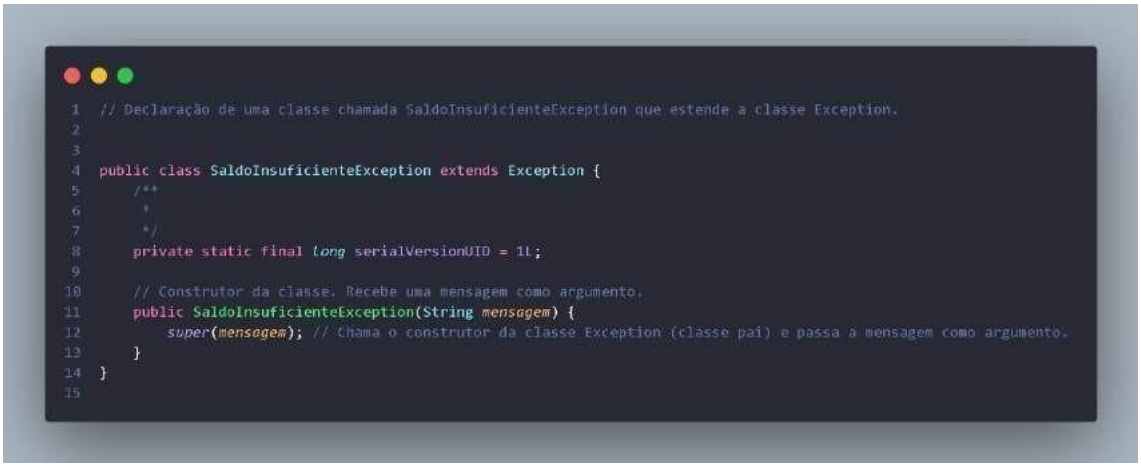
FIGURA 2

```java
// A classe ContaPoupanca extende a classe ContaBancaria.


public class ContaPoupanca extends ContaBancaria {

    /**
     *
     */
    private static final Long serialVersionUID = 1L;
    private double taxaRendimento; // Declaração de uma variável de instância 'taxaRendimento' que irá armazenar a taxa de rendimento da poupança.

    // Construtor da classe.
    public ContaPoupanca(String numeroConta, double saldo, double taxaRendimento) throws IllegalAccessException {
        super(numeroConta, saldo, numeroConta); // Chama o construtor da classe mãe (ContaBancaria) passando o número da conta e saldo.
        this.taxaRendimento = taxaRendimento; // Inicializa a taxa de rendimento com o valor fornecido.
    }

    // Getters e setters
    public double getTaxaRendimento() {
        return taxaRendimento;
    }

    public void setTaxaRendimento(double taxaRendimento) {
        this.taxaRendimento = taxaRendimento;
    }

    // Método que calcula o rendimento da conta poupança, multiplicando o saldo pela taxa de rendimento.
    public double calcularRendimento() {
        return getSaldo() * taxaRendimento;
    }
}

```

FIGURA 3

As figuras seguintes, é observado as classes Filho, Classe **ContaPoupança** e

Classe **ContaCorrente**, onde as mesmas que vão herdar atributos da Classe Pai Classe **ContaBancária** como dito na legenda da figura anterior. Outra coisa notável é método **MovimentacaoContaBancária**, responsável pelos métodos de Depositar e Sacar.



```java
// Declaração de uma classe chamada SaldoInsuficienteException que estende a classe Exception.


public class SaldoInsuficienteException extends Exception {
    /**
     *
     */
    private static final long serialVersionUID = 1L;

    // Construtor da classe. Recebe uma mensagem como argumento.
    public SaldoInsuficienteException(String mensagem) {
        super(mensagem); // Chama o construtor da classe Exception (classe pai) e passa a mensagem como argumento.
    }
}

```

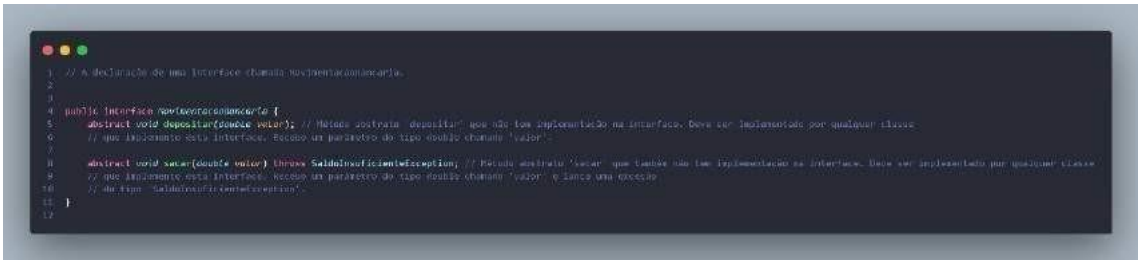
FIGURA 4

A figura 4, mostra uma exceção implementada no código do Sistema, quando um usuário desejar sacar um valor maior que o seu saldo da Conta, o programa retorna '**saldo Insuficiente para conta**'.

```java

public class TesteBanco {

    public static void main(String[] args) throws IllegalAccessException {
        try {
            ContaCorrente contaCorrente = new ContaCorrente("123", 1000.0, 500.0);
            contaCorrente.depositar(200.0);
            contaCorrente.sacar(300.0);

            System.out.println("Saldo da conta corrente: " + contaCorrente.getSaldo());

            ContaPoupanca contaPoupanca = new ContaPoupanca("456", 2000.0, 0.03);
            contaPoupanca.depositar(100.0);

            System.out.println("Saldo da conta poupança: " + contaPoupanca.getSaldo());
            System.out.println("Rendimento da conta poupança: " + contaPoupanca.calcularRendimento());
        } catch(SaldoInsuficienteException e) {
            System.out.println(e.getMessage());
        }
    }
}

```

FIGURA 5

A imagem acima mostra o código da classe principal que comanda todas as outras classes e serve como diretório principal, A **main**(como chamada na programação), é onde será instanciado os objetos, onde será possibilitado a execução o código.



```
// A declaração de uma interface chamada movimentacaoContaBancaria.


public interface movimentacaoContaBancaria {
    abstract void depositar(double valor); // Método abstrato 'depositar' que não tem implementação na interface. Deve ser implementado por qualquer classe
    // que implemente esta interface. Recebe um parâmetro do tipo double chamado 'valor'.

    abstract void sacar(double valor) throws SaldoInsuficienteException; // Método abstrato 'sacar' que também não tem implementação na interface. Deve ser implementado por qualquer classe
    // que implemente esta interface. Recebe um parâmetro do tipo double chamado 'valor' e lança uma exceção
    // do tipo 'SaldoInsuficienteException'.
}
```

FIGURA 6

A figura 6 mostra apenas o método da **MovimentacaoContaBancária** e como ele funciona.

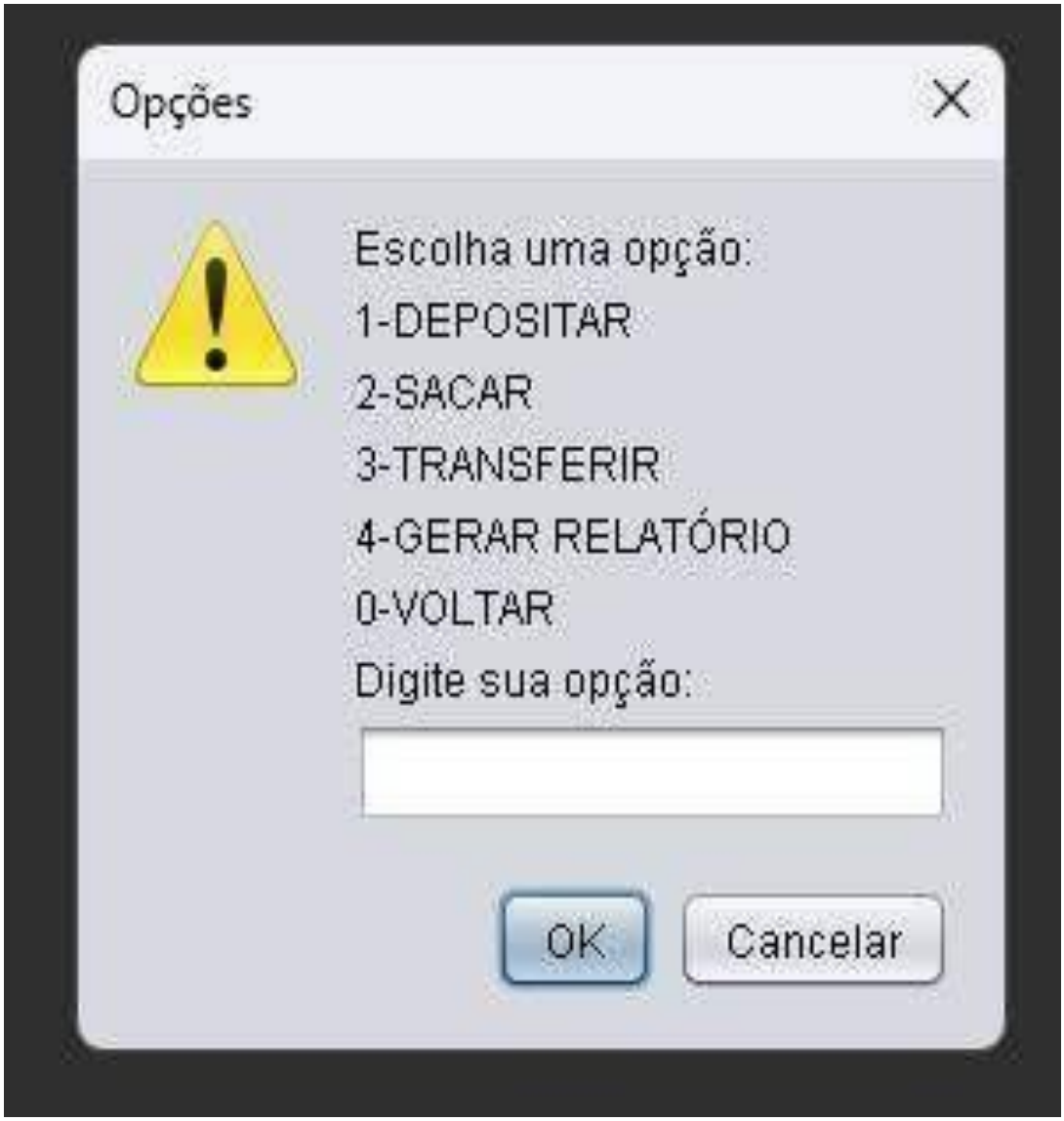


FIGURA 7

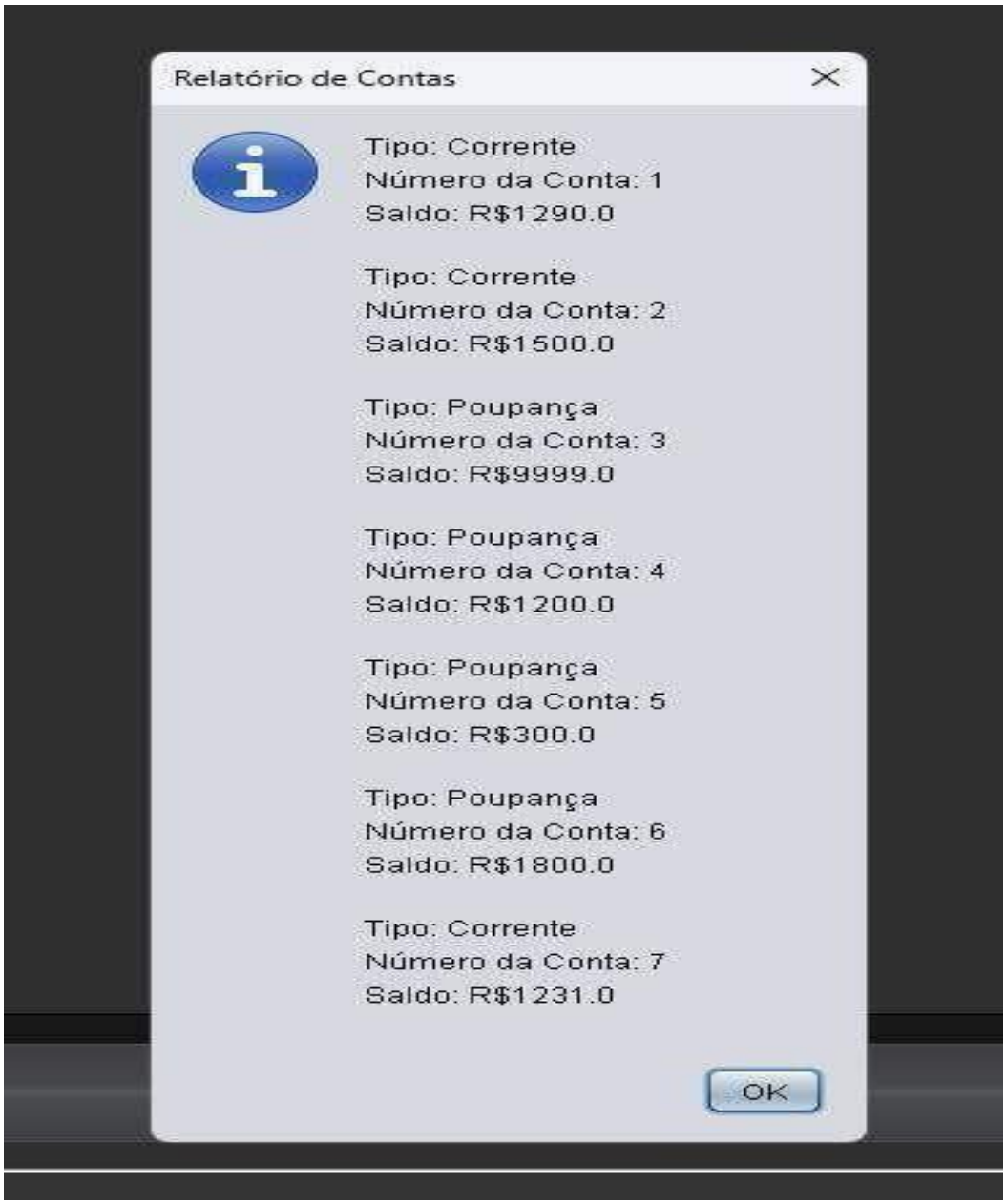


FIGURA 8

As figuras 7 e 8, são responsáveis por mostrar a interface com todas as funcionalidades para um usuário. Na figura 7 é possível observar os métodos (Depositar, Sacar, Transferir, Gerar Relatório e Voltar), enquanto na figura 8 é o relatório das contas existentes no Banco.

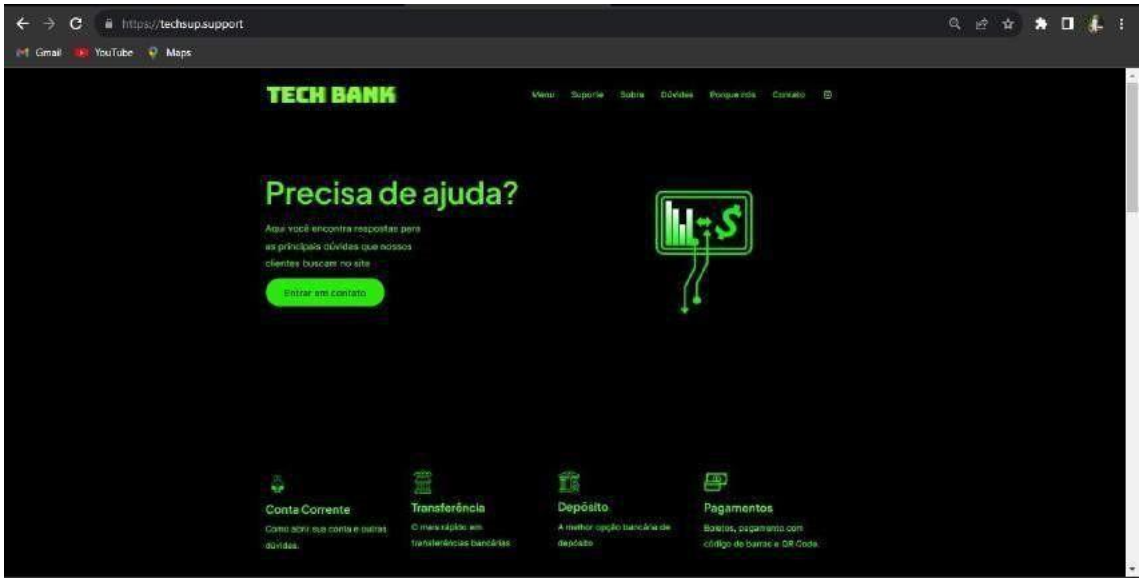


FIGURA 9

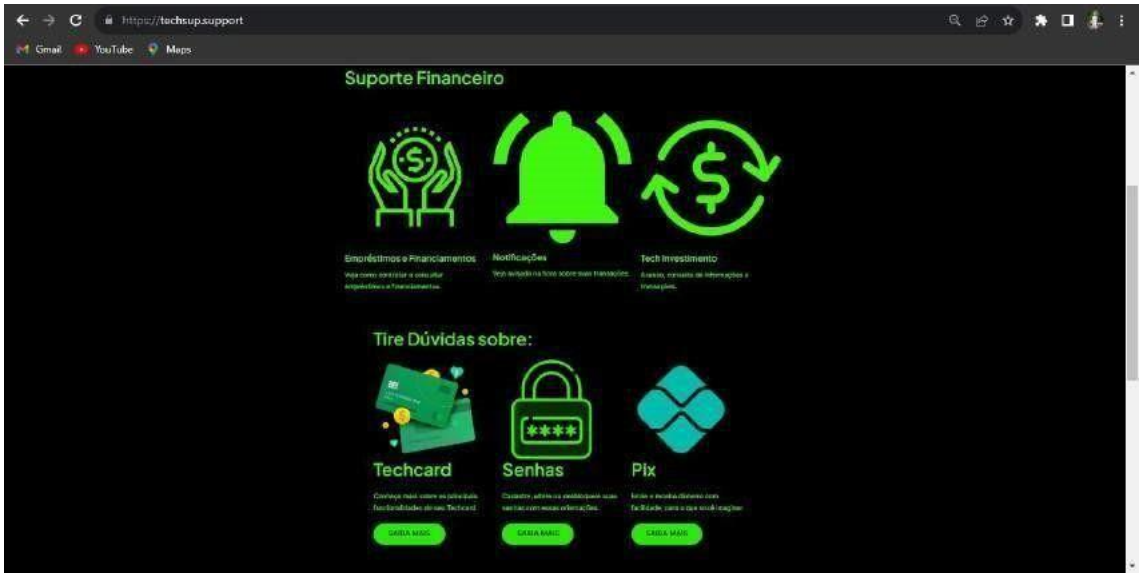


FIGURA 10

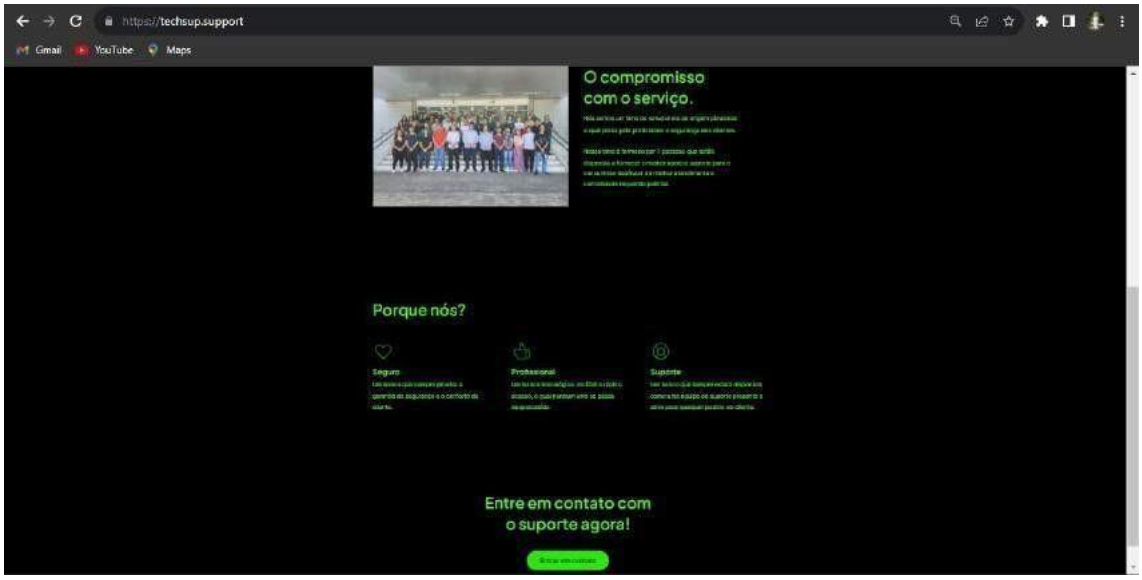


FIGURA 11

As figuras 9, 10, 11 demonstram o site desenvolvido com a tecnologia WordPress que serve como suporte para o Sistema bancário produzido. Ele é um site simples, one page(uma página), que possui um sistema de ancoragem de links, onde um click em cada frase interativa leva a uma região do site.

FIGURA 12

FIGURA 13

As imagens 12 e 13 são referentes ao dia da apresentação, onde a 12 é um recorte de dois dos muitos slides mostrados na **UE Prefeito Freitas Neto** no bairro **Mocambinho** e a figura 13 é uma foto oficial do grupo no dia da apresentação.

7.0. CONCLUSÃO

O desenvolvimento do sistema bancário em Java, fundamentado nos princípios da Programação Orientada a Objetos, demonstrou ser uma escolha sólida e eficaz para atender às demandas complexas e dinâmicas do setor financeiro. Ao longo do projeto, observamos benefícios significativos advindos da modularidade, reutilização de código e da estruturação hierárquica proporcionadas pela abordagem orientada a objetos.

A implementação das funcionalidades principais, como gestão de contas, execução de transações e autenticação de clientes, permitiu criar um sistema robusto e coeso. A

utilização de classes independentes para representar entidades como ContaBancária, ContaPoupança e Usuário facilitou a manutenção do código, promovendo a escalabilidade do sistema.

A estrutura orientada a objetos não apenas simplificou a concepção e implementação do sistema, mas também proporcionou uma visão clara das interações entre os componentes. A encapsulação de dados e métodos em classes contribuiu para a segurança do sistema, reduzindo o risco de manipulações inadequadas.

Os resultados obtidos durante os testes e a avaliação de desempenho validaram a eficácia do sistema. O tempo de resposta das transações foi satisfatório, a capacidade de carga do sistema atendeu às expectativas e os mecanismos de segurança implementados demonstraram efetividade na proteção das informações dos usuários.

Além disso, o feedback positivo dos usuários de teste reforçou a usabilidade e a intuitividade do sistema, indicando que a interface desenvolvida atendeu às expectativas e proporcionou uma experiência positiva aos usuários.

No entanto, reconhecemos que todo sistema possui espaço para aprimoramentos contínuos. As lições aprendidas ao longo do desenvolvimento destacam a importância da manutenção da modularidade e da constante busca por inovações tecnológicas para garantir a relevância do sistema em um ambiente em constante evolução.

Dessa forma, concluímos que o sistema bancário desenvolvido em Java, com base na Programação Orientada a Objetos, atendeu com sucesso aos objetivos propostos. Este projeto não apenas proporcionou uma solução funcional e eficiente, mas também serviu como uma base sólida para futuras expansões e melhorias, reforçando a importância da abordagem orientada a objetos na concepção de sistemas complexos e confiáveis.

8.0. REFERENCIAL BIBLIOGRÁFICO

- DEITEL, P. J.; DEITEL, H. M. **Java:** como programar. 10. ed. São Paulo: Pearson, 2017.

- NAUGTHON, Patrick. **Dominando o Java, Guia Autorizado da Sun Microsystems.** [S.l.]: Editora Makron Books. 1997.
- VERSOLATTO, Fabio. **Sistemas orientados a objetos:** conceitos e práticas. 1. ed. Rio de Janeiro: Freitas Bastos, 2023.

FACULDADE DE ESTÁCIO DE SÁ

CIÊNCIA DA COMPUTAÇÃO

LYAN KALEU MENESES DE SOUSA – 202303596511

Sistema bancário – Relatório individual

TERESINA

2023

SUMÁRIO

1.0. Introdução:

O objetivo do meu sistema bancário Java foi simular as interações entre vários tipos de contas bancárias. As classes ContaBancaria, ContaCorrente, ContaPoupanca e Usuario foram implementadas no projeto. ContaCorrente e ContaPoupanca herdam a superclasse da classe ContaBancaria. A interface MovimentacaoBancaria é executada pela classe ContaBancaria abstrata.

2.0. Revisão da Literatura:

Os princípios de herança, abstração e interface foram usados para definir a estrutura do sistema bancário. Os conceitos de classes abstratas e interfaces são essenciais para garantir uma estrutura hierárquica e a implementação de métodos obrigatórios para as classes filhas. A estratégia de exceções foi utilizada para lidar com situações de saldo insuficiente durante operações bancárias.

3.0. Metodologia:

- **Interface MovimentacaoBancario**: Define método abstratos **depositar(valor: double)** e **sacar(valor: double).**

- **Classe ContaBancaria:**
  − Atributos privados: **numeroConta (String)** e **saldo (double)**.
  − Construtor para inicializar atributos.
  − Métodos getters e setters para acessar atributos.
  − Implementação dos métodos da interface **MovimentacaoBancaria**.

- **Classe ContaCorrente:**
  − Herda de **ContaBancaria**.
  − Atributo privado: **limiteChequeEspecial (double)**.
  − Construtor para inicializar atributos herdados e atributo específico.
  − Sobrescrita do método **sacar(valor)** para permitir saques dentro do limite do cheque especial.

- **Classe ContaPoupanca:**
  − Herda de **ContaBancaria**.
  − Atributo privado: **taxaRendimento (double)**.
  − Construtor para inicializar atributos herdados e atributo específico.
  − Método **calcularRendimento ()** para calcular o rendimento.

- **Classe TesteBanco:**
  − Método **main** para criar objetos de **ContaCorrente** e **ContaPoupanca**.
  − Realização de movimentações bancárias (depósitos e saques) e cálculo de rendimento na conta poupança.

4.0. Resultados:

Os resultados demonstraram a funcionalidade do sistema bancário em Java, permitindo operações básicas de depósito, saque e cálculo de rendimento para diferentes tipos de contas. A estrutura de herança e a utilização de interfaces garantiram uma organização lógica e reutilizável do código.

5.0. Conclusão:

O projeto demonstrou a aplicação de conceitos fundamentais de programação orientada a objetos ao criar um sistema bancário em Java. A criação de classes e interfaces permite a extensão do sistema adicionando facilmente novos tipos de conta ou funcionalidades. A robustez do sistema foi aumentada pela manipulação de exceções, que evita erros durante as transações bancárias.
Esta experiência melhorou minhas habilidades em desenvolvimento de sistemas escaláveis e modulares, bem como meu entendimento de interfaces, herança e abstração em Java.