**PROJETO INTERDISCIPLINAR**

# Sistema de reserva de passagens

## Aéreas e Rodoviárias


**Alunos:**

| RGM | Nome |
|---|---|
| 31183263 | Daniel Silveira Kohlrausch Flores |
| 31319432 | Raphael de Celles Silva Brasileiro |
| 32007876 | Vinicius Neves Pereira |
| 31731155 | Nayara de Asis Micheletti |
| 31433588 | Fellipe Ângelo Andrade Souza |

São Paulo

2022

**UNIVERSIDADE CRUZEIRO DO SUL**

**PROJETO INTERDISCIPLINAR**

# Sistema de reserva de passagens

## Aéreas e Rodoviárias

Trabalho apresentado como parte do requisito para aprovação na Disciplina de Projeto Interdisciplinar do curso de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Cruzeiro do Sul.

**Orientadores:** Prof. Cristiane Camilo Hernandez e Prof. Jessica Barbara da Silva Ribas

São Paulo

2022

**Sumário**

# 1. APRESENTAÇÃO:

## 1.1 Justificativa e Motivação

O projeto interdisciplinar tem por objetivo a aplicação tanto teórica quanto prática dos conhecimentos absorvidos dos alunos participantes, tendo como ferramenta de avaliação de desempenho dos mesmos nas matérias trabalhadas durante o calendário universitário. A motivação para a escolha do tema deste projeto veio através de discussões entre o grupo sobre a alta demanda e os altos valores de preços em plataformas de viagens que, consequentemente, foi concretizada a ideia de uma construção de um sistema com o mesmo propósito: simular uma reserva de passagens aéreas / rodoviárias.

## 1.2 Dados do Sistema.

O sistema foi desenvolvido baseando-se nos conteúdos e conhecimentos adquiridos durante as aulas, portanto, é um sistema bem simples, solicitando dados básicos para cadastro de um usuário (como *Nome*, *CPF*, *Endereço* e *Telefone*) seguido dos destinos e os meios de transporte disponíveis ficando a critério do usuário de decidir a melhor opção.

# 2 REQUISITOS DE TÉCNICA DE DESENVOLVIMENTO DE ALGORITMOS

Foram utilizadas noções básicas para estruturar o sistema em linguagem em formato de Pseudocódigo (linguagem para compreensão simples do software). Abaixo, destaca-se conceitos relevantes que foram utilizadas na construção do sistema.

```
//cadastra dados
para(i=0;i<qtdpassagem;i++){
    escreva("Digite os dados para compra da pessagem");
    escreva("Digite seu nome completo: ");
    leia(nome[i]);
    verificaIdade();
    escreva("Digite seu endereço ex: rua,av,alameda: ");
    leia(endereco[i]);
    escreva("Digite o numero: ");
    leia(numero[i]);
    escreva("Digite o complemento: ");
    leia(completo[i]);
    escreva("Digite o cep: ");
    leia(cep[i]);
    escreva("Digite o cpf: ");
    leia(nCpf[i]);
    escreva("Digite email: ");
    leia(email[i]);
    escreva("Digite o telefone: ");
    leia(telefone[i]);
    escreva ("\t\t\t\t Compre sua Passagem\n");
    escreva ("Digite o local de origem: ");
    leia (localEmbarque[i]);
    escreva ("Digite o destino: ");
    leia (destino[i]);
    verificaOA();
    verificaIdaEV();
    verificaPol();
    verificaH();
}
//mostra dados
para(i=0;i<qtdpassagem;i++){
    //Dados Cliente
    escreva("\t\t\t ** Cliente **\n" +"Nome: " + nome[i] + "\nData de nascimento: " +  diaNasc[i] + "/" + mesNasc[i] + "/" +
    anoNasc[i] + "\nNumero do CPF:" + nCpf[i] + "\nEndereço: " + endereco[i] + "\nNumero: " + numero[i] + "\nComplemento: " +
    complemento[i] + "\nCEP: " + cep[i] + "\nE-mail: " + email[i] + "\nTelefone: " + telefone[i]);
    //Dados da Passagem
    escreva("\t\t\t ** Passagem **\n" +"\nLocal de Origem: "+ localEmbarque[i]+ "\nDestino: "+ destino[i] +"\nTipo de Transporte: "
    +onibusAviao[i]+ "\nEscolhido: "+ escolha[i] + "\nData de ida: " + diaIda[i] + "/" + mesIda[i] + "/"+ anoIda[i] +"\nData de Volta: "
    + diaVolta[i] + "/" + mesVolta[i] + "/"+ anoVolta[i] +"\nPoltrona: "+ poltrona[i] + "\nHorario: "+ horario[i] + " horas"+ "\nPreço: "
    + preco+ "\nCodigo Passagem: "+geradorCod());
}
```

*Figura 1 - Para o cadastro de cliente, solicitamos os dados básicos do cliente. Já para a compra de passagens, de início, solicitamos apenas o local de origem e destino para, mais adiante, solicitar mais detalhes sobre a passagem.*

```
//metodos

inicio
 //verifica a idade
    void verificaIdade(){

        escreva("Digite apenas o dia de nascimento");
        leia(diaNasc[i]);
        enquanto(diaNasc[i]>31 ou diaNasc[i]<0){
            escreva("Você digitou o dia incorreto !!!");
            escreva("Digite apenas o dia de nascimento");
            leia(diaNasc[i]);
        }
        escreva("Digite mês de nascimento");
        leia(mesNasc[i]);
        enquanto(mesNasc[i]>12 ou mesNasc[i]<0){
            escreva("Você digitou o dia incorreto !!!");
            escreva("Digite o mês de nascimento");
            leia(mesNasc[i]);
        }
        escreva("Digite o ano de nascimento");
        leia(anoNasc[i]);
            enquanto(anoNasc[i]>2022 ou anoNasc[i]<1902){
            escreva("Você digitou o ano incorreto !!!");
            escreva("Digite o ano de nascimento");
            leia(anoNasc[i]);
        }

    }
fim
```

```
inicio
   //verifica Dia Mês e Ano da Volta
    void verificaDMAV(){
        escreva ("Digite o dia de Volta");
        leia(diaVolta[i]);
        enquanto(diaVolta[i]>31 ou diaVolta[i]<0){
            escreva("Você digitou o dia incorreto");
            escreva ("Digite o dia de Volta");
            leia(diaVolta[i]);
        }

        escreva ("Digite o numero do mês de Volta");
        leia(mesVolta);
        enquanto (mesVolta[i]>12 ou mesIda[i]<0){
            escreva ("Você digitou o mês incorreto");
            escreva ("Digite o numero do mês de Volta");
            leia(mesVolta[i]);
        }

        escreva ("Digite o ano de Volta");
        leia(anoVolta[i]);
        enquanto(anoVolta[i]<2022){
            escreva ("Você digitou o ano incorreto
            \n por enquanto ainda não fazemos viagem para o passado :( ");
            escreva ("Digite o numero do ano de Volta");
            leia(anoVolta[i]);
        }
    }
fim
```

*Figura 2 - Para a confirmação e verificação de dados, realizamos a utilização de vetores.*

```
inicio
 //calcula o preço da passagem
    real precoP(){
        preco= preco*precot;
        retorno preco;
    }

fim
```

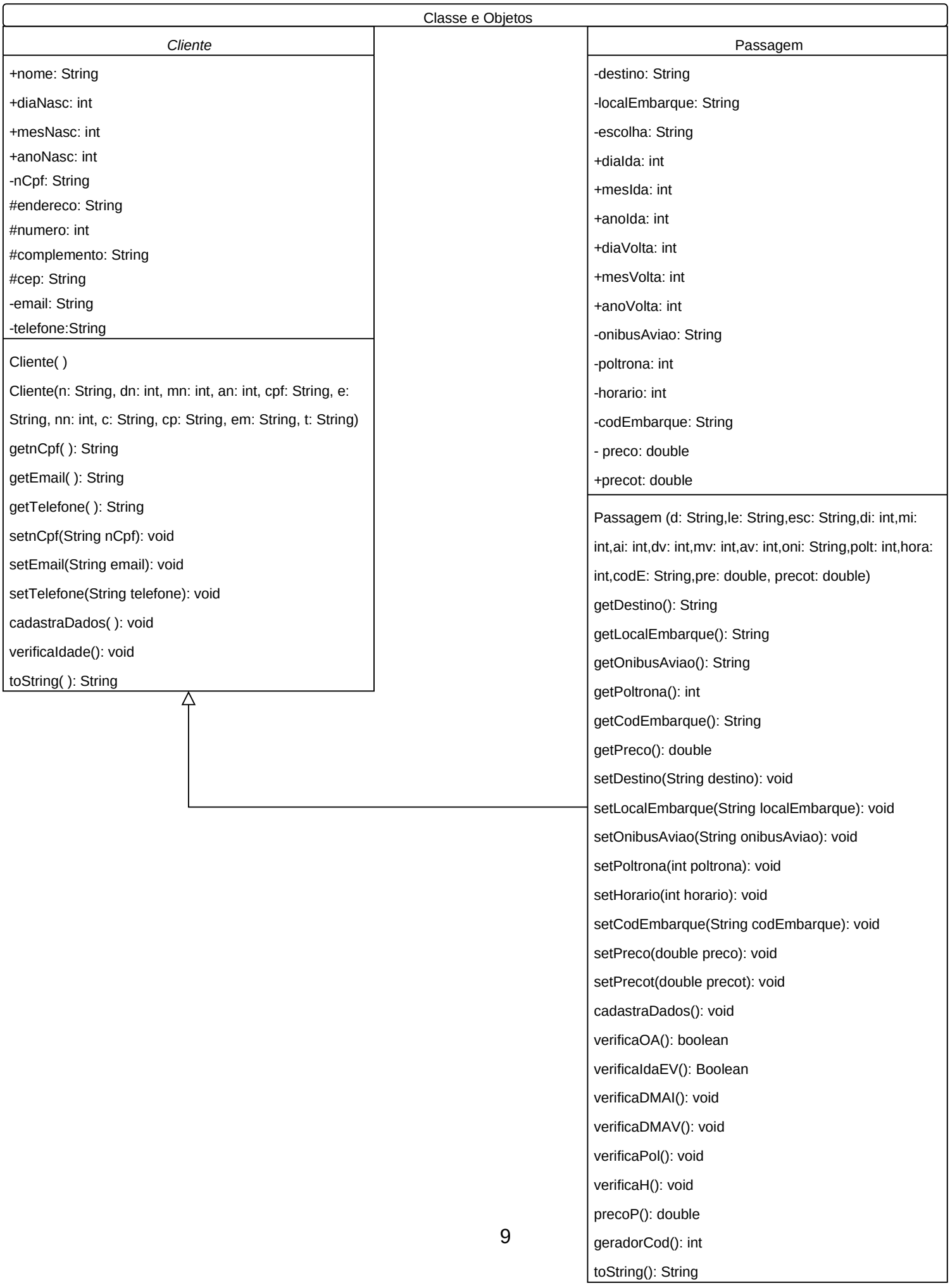
*Figura 3-Para realizar o cálculo do valor das passagens, utilizamos um valor fantasia já definido anteriormente (Ônibus: R$150,00 / Avião: R$500,00) e utilizamos uma formula básica para o cálculo.*

```
inicio
    literal geradorCod(){
        inteiro aleatorio;
        literal s;
        s="";

        para(num=0;num<7;num++){
            aleatorio = rand(100);
            codEmbarque[i]=aleatorio+s;
        }

        retorno codEmbarque;
    }

fim
```

*Figura 4 - Por fim, para gerar o código de embarque para as passagens, utilizamos de um método randomizador para gerar os números aleatoriamente.*

## 3 REQUISITOS DE PROGRAMAÇÃO ORIENTADA A OBJETOS

Utilizando o esquema de Algoritmos em Pseudocódigos e o Diagrama UML (figura abaixo) como auxílio para a construção do sistema em linguagem JAVA, a estrutura de código foi desenvolvida da seguinte forma.


Classe e Objetos
Cliente
+nome: String
+diaNasc: int
+mesNasc: int
+anoNasc: int
-nCpf: String
#endereco: String
#numero: int
#complemento: String
#cep: String
-email: String
-telefone:String
Cliente( )
Cliente(n: String, dn: int, mn: int, an: int, cpf: String, e:
String, nn: int, c: String, cp: String, em: String, t: String)
getnCpf( ): String
getEmail( ): String
getTelefone( ): String
setnCpf(String nCpf): void
setEmail(String email): void
setTelefone(String telefone): void
cadastraDados( ): void
verificaIdade(): void
toString( ): String
Passagem
-destino: String
-localEmbarque: String
-escolha: String
+diaIda: int
+mesIda: int
+anoIda: int
+diaVolta: int
+mesVolta: int
+anoVolta: int
-onibusAviao: String
-poltrona: int
-horario: int
-codEmbarque: String
- preco: double
+precot: double
Passagem (d: String,le: String,esc: String,di: int,mi:
int,ai: int,dv: int,mv: int,av: int,oni: String,polt: int,hora:
int,codE: String,pre: double, precot: double)
getDestino(): String
getLocalEmbarque(): String
getOnibusAviao(): String
getPoltrona(): int
getCodEmbarque(): String
getPreco(): double
setDestino(String destino): void
setLocalEmbarque(String localEmbarque): void
setOnibusAviao(String onibusAviao): void
setPoltrona(int poltrona): void
setHorario(int horario): void
setCodEmbarque(String codEmbarque): void
setPreco(double preco): void
setPrecot(double precot): void
cadastraDados(): void
verificaOA(): boolean
verificaIdaEV(): Boolean
verificaDMAI(): void
verificaDMAV(): void
verificaPol(): void
verificaH(): void
precoP(): double
geradorCod(): int
toString(): String

Classe pai - “Cliente”:

```java
public class Cliente{
    //atributos
    public String nome;
    public int diaNasc;
    public int mesNasc;
    public int anoNasc;
    protected String nCpf;
    protected String endereco;
    protected int numero;
    protected String complemento;
    protected String cep;
    protected String email;
    protected String telefone;
```

```java
//sem retorno e sem parametros
public void cadastraDados(){
    JOptionPane.showMessageDialog(parentComponent: null, message: "\t\t\t\t Digite os dados para comprar sua Passagem ");
    nome = JOptionPane.showInputDialog(parentComponent: null, message: "Digite o nome completo ");
    verificaIdade();
    endereco = JOptionPane.showInputDialog(parentComponent: null, message: "Digite o endereço ex: rua,av,alameda ");
    numero = Integer.parseInt(JOptionPane.showInputDialog(parentComponent: null, message: "Digite o numero "));
    complemento = JOptionPane.showInputDialog(parentComponent: null, message: "Digite o complemento ");
    cep = JOptionPane.showInputDialog(parentComponent: null, message: "Digite o cep ");
    nCpf = JOptionPane.showInputDialog(parentComponent: null, message: "Digite o numero do CPF ");
    email = JOptionPane.showInputDialog(parentComponent: null, message: "Digite o Email ");
    telefone = JOptionPane.showInputDialog(parentComponent: null, message: "Digite o telefone com DDD ");
```

*Figura 5 -Para o cadastro de cliente (sendo “Cliente” a classe pai), solicitamos os dados básicos dividindo entre dados públicos (nome e data de nascimento) e protegidos (número de CPF, endereço completo, CEP, e-mail e telefone). Utilizamos um método sem retorno.*

```java
//verifica dia mes e ano de Nascimento
public void verificaIdade(){

    //verificão dia de Nascimento
    diaNasc= Integer.parseInt(JOptionPane.showInputDialog(parentComponent: null, message: "Digite apenas o dia de Nascimento"));
        while(diaNasc>31 || diaNasc<0){
            JOptionPane.showMessageDialog(parentComponent: null,message: "Você digitou o dia incorreto");
            diaNasc= Integer.parseInt(JOptionPane.showInputDialog( message: "Digite apenas o dia de Nascimento "));
        }

    //verificão mes de Nascimento
    mesNasc= Integer.parseInt(JOptionPane.showInputDialog( message: "Digite o numero do mês de Nascimento"));
        while(mesNasc>12 || mesNasc<0){
            JOptionPane.showMessageDialog(parentComponent: null,message: "Você digitou o mês incorreto");
            mesNasc= Integer.parseInt(JOptionPane.showInputDialog( message: "Digite o numero do mês de Nascimento"));
        }

    //verificão ano de Nascimento
    anoNasc= Integer.parseInt(JOptionPane.showInputDialog( message: "Digite o ano de Nascimento"));
        while(anoNasc<1902 || anoNasc>2022){
            JOptionPane.showMessageDialog(parentComponent: null,message: "Você digitou o ano incorreto ");
            anoNasc= Integer.parseInt(JOptionPane.showInputDialog( message: "Digite o Ano de Nascimento"));
        }
```

*Figura 6 - Para realizar a verificação da idade do cliente (se é válida ou não), utilizamos o método “while”.*

```java
//sem retorno e sem parametro
public String toString(){
    String aux = " ";
    if(aux == " "){ }
    JOptionPane.showMessageDialog(parentComponent: null,"\t\t\t ** Cliente **\n" +"Nome: " + nome + "\nData de
    nascimento: " +  diaNasc + "/" + mesNasc + "/" +
    anoNasc + "\nNumero do CPF:" + nCpf + "\nEndereço: " + endereco + "\nNumero: " + numero + "\nComplemento: "
    + complemento + "\nCEP: "
    + cep + "\nE-mail: " + email + "\nTelefone: " + telefone);
    return aux;

}
```

*Figura 7 - A impressão dos dados é realizada através da função toString, onde mostrará uma janela com todos os dados inseridos pelo cliente.*

## Classe filho – “Passagem”:

```java
public class Passagem extends Cliente{

    //Atributos
    private String destino;
    private String localEmbarque;
    public String escolha;
    public int diaIda;
    public int mesIda;
    public int anoIda;
    public int diaVolta;
    public int mesVolta;
    public int anoVolta;
    private String onibusAviao;
    private int poltrona;
    private int horario;
    private int codEmbarque;
    private double preco;
    public double precot;

//Métodos

//Cadastro
public void cadastraDados(){
    super.cadastraDados();
    JOptionPane.showMessageDialog(parentComponent: null, message: "\t\t\t\t Compre sua Passagem ");
    localEmbarque= JOptionPane.showInputDialog( message: "Digite o local de Origem");
    destino = JOptionPane.showInputDialog( message: "Digite o destino");
    verificaOA();
    verificaIdaEV();
    verificaPol();
    verificaH();


}
```

*Figura 8 - Para a reserva de passagens (sendo “Passagem” classe filho de “Cliente”), solicitamos os dados dividindo entre público (a escolha (passagem ida e volta), o preço do transporte e a data das viagens) e privados (o tipo de transporte (ônibus ou avião), destino, local de embarque, preço final das passagens, a poltrona, horário e o código de embarque). Utilizamos um método sem retorno e sem parâmetros para a inserção de dados.*

```java
//Verifica Onibus ou Avião
public boolean verificaOA(){

    //variavel aux
    char verificaOA;

    verificaOA=JOptionPane.showInputDialog( message: "Digite a letra\n (O) para Ônibus \n (A) para Avião").
    toUpperCase().charAt(index: 0);
        while( verificaOA!= 'O' && verificaOA != 'A'){
            JOptionPane.showMessageDialog(parentComponent: null,message: "OPÇÃO INCORRETA!!!");
            verificaOA=JOptionPane.showInputDialog( message: "Digite a letra\n (O) para Ônibus \n (A) para
            Avião").toUpperCase().charAt(index: 0);
        }

    if (verificaOA =='O'){
        onibusAviao ="Ônibus";
        precot = 150;
        JOptionPane.showMessageDialog(parentComponent: null, message: "Você escolheu Ônibus");
        return true;
    }
    else {
        onibusAviao ="Avião";
        precot = 500;
        JOptionPane.showMessageDialog(parentComponent: null, message: "Você escolheu Avião");
        return false;
    }
```

```java
//verifica Dia Mês e Ano da Ida
public void verificaDMAI(){

    //verificão dia de ida
    diaIda= Integer.parseInt(JOptionPane.showInputDialog(parentComponent: null, message: "Digite apenas o dia de
    Ida"));
        while(diaIda>31 || diaIda<0){
            JOptionPane.showMessageDialog(parentComponent: null,message: "Você digitou o dia incorreto");
            diaIda= Integer.parseInt(JOptionPane.showInputDialog( message: "Digite apenas o dia de Ida "));
        }

    //verificão mes de ida
    mesIda= Integer.parseInt(JOptionPane.showInputDialog( message: "Digite o numero do mês de Ida"));
        while(mesIda>12 || mesIda<0){
            JOptionPane.showMessageDialog(parentComponent: null,message: "Você digitou o mês incorreto");
            mesIda= Integer.parseInt(JOptionPane.showInputDialog( message: "Digite o numero do mês de Ida"));
        }

    //verificão ano de ida
    anoIda= Integer.parseInt(JOptionPane.showInputDialog( message: "Digite o ano de Ida"));
        while(anoIda<2022){
            JOptionPane.showMessageDialog(parentComponent: null,message: "Você digitou o ano incorreto ");
            anoIda= Integer.parseInt(JOptionPane.showInputDialog( message: "Digite o Ano de Ida"));
        }
```

*Figura 9 - Para realizar a verificação e validação dos dados preenchidos na passagem, utilizamos o método “while”.*

```java
//calcula o preço da passagem
public double precoP(){
    preco= preco*precot;
    return preco;
}
```

*Figura 10 - Para realizar o cálculo do valor das passagens, utilizamos um valor fantasia já definido anteriormente (Ônibus: R$150,00 / Avião: R$500,00) e utilizamos uma formula básica para o cálculo.*

```java
//para gerar o código de embarque
public int geradorCod(){

        Random aleatorio = new Random();
        int valor = aleatorio.nextInt(bound: 10000);

        codEmbarque = valor;

        return codEmbarque;

}
```

*Figura 11 - Utilizamos de um método randomizador para gerar o código de embarque, sendo os números gerados aleatoriamente.*

```java
//Mostra os dados
public String toString(){

    String aux = " ";
    if(aux == " "){
    super.toString();
    JOptionPane.showMessageDialog(parentComponent: null, "\t\t\t ** Passagem **\n" +"\nLocal de Origem: "+
    localEmbarque+ "\nDestino: "+ destino +"\nTipo de Transporte: "
    +onibusAviao+ "\nEscolhido: "+ escolha + "\nData de ida: " + diaIda + "/" + mesIda + "/"+ anoIda +"\nData de
    Volta: " + diaVolta + "/" + mesVolta + "/"+ anoVolta
    +"\nPoltrona: "+ poltrona + "\nHorario: "+ horario + " horas"+ "\nPreço: "+ preco+ "\nCodigo Passagem: "
    +geradorCod());
    }
    return aux;

}
```

*Figura 12 - A impressão é realizada através da função toString, onde mostrará uma janela com todos os dados referente a passagem.*

Classe principal (main) – “CompraPassagem”:

```java
public class CompraPassagem{
    Run | Debug
    public static void main(String[] args) {
        //variavel Aux
        int qtdpassagem;
        int tipo;
        String resp;
```

*Figura 13 - Para realizar a compra de passagens (sendo “CompraPassagem” a classe principal), definimos a quantidade e o tipo de passagens.*

```
do{
    tipo = Integer.parseInt(JOptionPane.showInputDialog(message: " Bem vindo ao
    programa de Compra de Passagens\n 1 para Comprar \n 2 para Sair"));
    switch(tipo){
        case 1:
        //Cria o objeto
        Cliente acervo[];
        //instanciação do vetor
         acervo = new Cliente[60];

         qtdpassagem = Integer.parseInt(JOptionPane.showInputDialog(
         message: "Digite a quantidade de Passagens que deseja comprar"));

         while (qtdpassagem>60){
             JOptionPane.showMessageDialog(parentComponent: null, message: "No momento
             só temos 60 Passagens disponíveis\n contando com 30 lugares para Onibus\n
             30 lugares para Avião");
             qtdpassagem = Integer.parseInt(JOptionPane.showInputDialog(
         message: "Digite a quantidade de Passagens que deseja comprar"));
         }
```

```
for (int i = 0; i < qtdpassagem; i++) {

    acervo[i]= new Passagem(n: null, i, i, i, cpf: null, e: null, i, c: null,
    cp: null, em: null, t: null, destino: null, localEmbarque: null,
    escolha: null, i, i, i, i, i, i, onibusAviao: null, i, i, i, qtdpassagem,
    i);

     acervo[i].cadastraDados();
     acervo[i].toString();
}

    break;
case 2:
System.exit(status: 0);
    break;
default:
    JOptionPane.showMessageDialog(parentComponent: null,message: "Opção
    Invalida");
    break;
```

```
         System.exit(status: 0);
             break;
         default:
             JOptionPane.showMessageDialog(parentComponent: null,message: "Opção
             Invalida");
             break;

     }
     resp = JOptionPane.showInputDialog(message: "Deseja Continuar? (S/N)");
}while(resp.equalsIgnoreCase(anotherString: "s"));
```

*Figura 14 - Para finalizar o programa e concluir a compra de passagens, utilizamos as estruturas de repetição (do/while, switch (com os cases), for e vetores) para confirmar se o usuário realmente deseja finalizar o programa.*

## 5 CONSIDERAÇÃO FINAIS

Durante a elaboração do projeto, houveram algumas dificuldades que interferiam o desenvolvimento, uma delas foi a criação de uma variável “CPF” (servindo para o usuário digitar o número do documento), tendo muitos questionamentos referente ao tipo em que seria declarada e sua construção dentro do bloco de programação.

Sentimos que o grupo colaborou bem ao desenvolver este projeto (assim como equipes do mundo corporativo devem ser), discutimos assuntos pertinentes e atuais que, querendo ou não, afetam nosso cotidiano e entramos em um consenso justo e prático, nada muito além do permitido.

## 6 BIBLIOGRAFIA


- “Decolar.com”. Disponível em https://www.decolar.com. Acessado em: Setembro de 2022.
- “123Milhas.com”. Disponível em https://123milhas.com. Acessado em: Setembro de 2022.
- FORBELLONE, A. L. V.; EBERSPACHER, H. F. Lógica de Programação: A Construção de Algoritmos e Estrutura de Dados. 3. ed. São Paulo: Pearson Prentice Hall, 2008.
- GUEDES, Sergio. Lógica de Programação Algorítmica. São Paulo: Pearson Education do Brasil, 2014 (e-book)
- MANZANO, José Augusto N. G. Algoritmos: lógica para desenvolvimento de programação de computadores. 28. ed. São Paulo: Erica, 2016 (ebook).
- DEITEL, Paul; DEITEL, Harvey. Java: como programar. 10.ed. São Paulo: Pearson, 2016.
- FURGERI, Sérgio. Java 8, ensino didático: desenvolvimento e implementação de aplicações. São Paulo: Erica, 2015. (e-book).
- MANZANO, J. A. N. G.; COSTA JUNIOR, R. A. Java 7: programação de computadores - desenvolvimento e implementação de aplicações. São Paulo: Érica, 2011 (e-book)