

UNIVERSIDADE PAULISTA – UNIP

CIÊNCIAS DA COMPUTAÇÃO

ARNON DAMASCENO NEVES DE SOUZA

RANYELLE EVELIN LISBOA BARBOSA

RODRIGO ZAGO DE SA GOUVEIA

VINICIUS DINIZ CARLOS PRUDENCIO

DESENVOLVIMENTO DE SOFTWARE UTILIZANDO
CONCEITOS DE PROGRAMAÇÃO ORIENTADA A OBJETOS

SÃO PAULO - SP

2017

ARNON DAMASCENO NEVES DE SOUZA
RANYELLE EVELIN LISBOA BARBOSA
RODRIGO ZAGO DE SA GOUVEIA
VINICIUS DINIZ CARLOS PRUDENCIO

DESENVOLVIMENTO DE SOFTWARE UTILIZANDO
CONCEITOS DE PROGRAMAÇÃO ORIENTADA A OBJETOS

Atividade prática supervisionada e apresentada à
graduação de Ciências da Computação, para fins de
conhecimento na área.

Orientador: Uanderson Celestino.

SÃO PAULO - SP

2017

DEDICATÓRIA

Os autores do presente trabalho dedicam-no aos três sustentáculos de toda e qualquer aprendizagem humana: ao Criador, à Família e aos Mestres.

AGRADECIMENTOS

Este trabalho é fruto das lições dos Mestres e da colaboração de autores inspirados em aprender, aplicar e contribuir cientificamente.

Destarte, os integrantes desse projeto agradecem os ensinamentos transmitidos pelos quatro Mestres da turma do 3º semestre de 2017 do curso de Ciências da Computação da Universidade Paulista – UNIP – Unidade Paraíso:

Ao Professor Uanderson Celestino, pelo incentivo à elaboração de uma Atividade Prática Supervisionada que preze pela qualidade de seu conteúdo;

Aos Professores Uanderson, Veras e Fábio, pela paciência em prover os instrumentais técnico-científicos necessários para a transformação do aprendizado obtido em conhecimento aplicado;

Aos Professores Moretti, Rafael e Leminski, pelo intuito de sempre nos incentivar a aprender junto com as nossas dificuldades, buscando sempre novos conhecimentos e oportunidades.

*“Se você não encontra o sentido das coisas
é porque este na se encontra, se cria”.*

(Antonie de Saint-Exupéry)

RESUMO

O Java é uma linguagem de Programação Orientada a Objeto (POO) criado pela Sun em 1992, ela é baseada em classes, onde você cria objetos e desenvolve sua programação. Em 2009 a Oracle comprou a Sun e atualmente está na versão oito (8).

Ela oferece vários tipos de dados primitivos que podem ser numéricos (int, float, Double), alfabético (char) ou booleano (verdadeiro ou falso). O foco do Java são aplicações de médio e grande porte, sendo desenvolvido por várias pessoas, e não programas de pequenos portes, sendo desenvolvido por uma ou duas pessoas, porém nada impede que ela seja utilizada e implementada para produzir bons programas.

Antes do Java tinham programas como o C e o Pascal onde temos um código fonte que é compilado para código da máquina de uma plataforma ou sistema operacional. Esse código binário será executado pelo sistema operacional e por esse motivo deve saber conversar com o mesmo, não sendo tão possível executar o mesmo programa em outros sistemas operacionais.

Palavras-chave: Java; Programas; Linguagem Java; Programação Orientada a Objetos.

ABSTRACT

Java is an object-oriented programming language created by Sun in 1992, it is class-based, where you create objects and develop your programming. In 2009 Oracle bought Sun and is currently in version 8.

It offers several types of primitive data that can be Numeric (int, float, double), Alphabetic (char) or Boolean (true or false). The focus of Java are medium and large applications, being developed by several people, not small program developed by one or two people, however there is nothing to prevent it from being used and implemented to produce good programs.

Before the Java had programs like C and Pascal where we have a source code that is compiled to machine code of a platform or operating system. This binary code will be executed by the operating system and for this reason should know that with the same without being so can run the same program in other operating systems.

Keywords: Java; Software; Language Java; Object Oriented Programming.

LISTA DE FIGURAS

Figura 1: Estrutura do programa	18
Figura 2: Estrutura do programa.	19
Figura 3: Estrutura do programa.	20
Figura 4: Estrutura do programa.	21
Figura 5: Estrutura do programa.	23
Figura 6: Estrutura do programa	24
Figura 7: Estrutura do programa.	25
Figura 8: Estrutura do programa.	26
Figura 9: Estrutura do programa.	28
Figura 10: Tela inicial.	55
Figura 11: Tela inicial – menu.	56
Figura 12: Tela inicial – Informações.....	57
Figura 13: Informações – Sobre.....	58
Figura 14: Tela inicial – Informações sobre o programa.	59
Figura 15: Meio ambiente.....	60
Figura 16: Meio ambiente – Melhorando nosso meio ambiente 1.....	61
Figura 17: Meio ambiente- Melhorando nosso meio ambiente 2.....	62
Figura 18: Cidade mais poluídas.....	63
Figura 19- Cidades mais poluídas – Mapa interativo.	64
Figura 20: Cidades mais poluídas – pais - Sobre.....	65
Figura 21: Cidades mais poluídas – Pais - Principais.	66
Figura 22: Cidades mais poluídas – Pais - Sobre.	67
Figura 23: Cidades mais poluídas – Pais – sobre – cidade – interatividade. ..	68
Figura 24: Cidades mais poluídas – Pais –Sobre – Cidades.	69

Figura 25: Cadastro de usuário.....	70
Figura 26: Login.	71

Sumário

1. OBJETIVO DO TRABALHO.....	Erro! Indicador não definido.
2. INTRODUÇÃO.....	2
3. CONCEITOS GERAIS DA PROGRAMAÇÃO ORIENTADA A OBJETO	4
3.1 A Plataforma Java.....	4
• Java 2 Standard Edition(J2SE).....	4
• Java 2 Enterprise Edition (J2EE).	4
• Java 2 Micro Edition (J2ME)..	4
3.2 - A Maquina Virtual Java.....	4
3.3 - Variáveis em Java	5
3.4 - Tipos Numéricos em Java	6
3.5 Comandos de Decisão.	7
• O Comando If..	7
• O Comando Else..	7
• O Comando Switch – Case.....	7
3.6 Comandos de Repetição	8
• Comando While.....	8
• Comando Do-While.....	8
• Comando For.	8
3.7 Projetando Classes.	9
• Métodos.	9
• Construtores.....	9
• Getters e Setters.	9
• Modificadores de Acesso.	9

3.8 Orientação a Objeto.....	10
• Herança.....	10
• Polimorfismo.....	10
• Classe Abstrata.....	11
• Interfaces.....	11
• Classes Finais.....	11
3.9 Captura e Tratamento de Erros.....	11
• Try-Catch.....	11
• Finally.....	12
• Throws.....	12
• Throw.....	12
4.0 Banco de dados.....	12
4.1 Modelagem banco de dados.....	12
▪ Modelo físico.....	12
Modelo lógico.....	13
Modelo conceitual.....	13
• Os quatro tipos de atributos.....	13
• Características de uma chave primária.....	14
4.2 Relacionamentos.....	14
4.3 Cardinalidade.....	14
Tipos de cardinalidade.....	15
4.4 Normalização.....	15
• Primeira forma normal.....	15
• Segunda forma normal.....	15
• Terceira forma normal.....	16
4.5 Considerações para programa.....	17

• 5.7 Projeto – estrutura do programa	18
5.8 Relatório com as linhas de código do programa.	29
• Pagina inicial do programa.	29
• A respeito do meio ambiente	34
• Sobre cidades mais poluídas e seu países.	41
5.9 Apresentação do programa funcionando em um computador.	53
• Tela inicial	54
• Tela inicial – Menu.	55
• Tela inicial - Informações.	56
• Tela inicial - Informações – Sobre.	58
• Tela inicial - Informações – informações sobre o programa.	59
• Meio ambiente.	60
• Meio ambiente – Melhorando nosso meio ambiente.	61
• Meio ambiente – Melhorando nosso meio ambiente – desmatamento / sustentabilidade.	62
• Cidades mais poluídas.	63
• Cidades mais poluídas – Mapa interativo.	64
• Cidades mais poluídas – País – Sobre.	65
• Cidades mais poluídas – País – Principais cidades.	66
• Cidades mais poluídas – País – Sobre as cidades.	67
• Cidades mais poluídas – País – Sobre as cidades – Cidade – Interatividade.	68
• Cidades mais poluídas – País – Sobre as cidades – Cidade.	69
• Tela inicial - Cadastro de usuário.	70
• Tela inicial – Login.	71
6. CONCLUSÃO	72
7. REFERÊNCIAS BIBLIOGRÁFICAS	73

8. Fichas de Atividades Praticas Supervisionadas.	74
8.1 RODRIGO ZAGO DE SA GOUVEIA – N989ED-3	75
8.2 VINICIUS DINIZ CARLOS PRUDENCIO – N866CE-2.....	76
8.3 RANYELLE EVELIN LISBOA BARBOSA – N10274-8	77
8.4 ARNON DAMASCENO NEVES DE SOUZA – N1047H-2	78

1. OBJETIVO DO TRABALHO

O presente trabalho tem como objetivo geral o aprendizado da programação orientada a objeto, utilizando a linguagem Java. A linguagem Java nos oferece diversas possibilidades para desenvolver boas aplicações, criar novos softwares e conscientizar a população através dos mesmos, a APS (Atividade Prática Supervisionada) tem como objetivo específico proporcionar a cada um de seus integrantes a oportunidade de, obter conhecimentos teóricos e a aplicação deste, visando desenvolver uma aplicação que utilize esses conhecimentos adquiridos ao longo do curso e trabalho mediante a muita dedicação e muito esforço.

Por fim, está proposto um tema que tem como objetivo informar e alertar os alunos/pessoas sobre os acontecimentos que denigrem o meio ambiente, sugerindo a busca de soluções aos problemas que vem ocorrendo durante vários e vários anos. O presente trabalho cumpre também com outro objetivo: o de contribuir como material inicial para os interessados em adentrar nessa área de especialização.

2. INTRODUÇÃO

A Atividade Prática Supervisionada (APS) do 3º semestre de 2017 estabeleceu a seguinte proposta de trabalho:

O tema terá que ser pertinente à educação ambiental e terá obrigatoriamente que gerar propostas de benefícios enquanto práticas cidadãs ligadas ao meio ambiente que signifiquem soluções para as questões ambientais universais ocasionadas pelas atividades humanas, tais como: O aquecimento global e às mudanças climáticas; À poluição da água dos lençóis freáticos, rios e oceanos; Os danos provocados pelas fontes energéticas não renováveis; entre outras questões ou problemas ambientais que se liguem com respectivas soluções ou caminhos para soluções ambientalmente educativas (UNIP, 2017, p.2).

A Linguagem de Programação Orientada a Objetos é um paradigma de programação de computadores que utiliza classes e objetos, que são criados a partir de modelos, os dados que pertencem a esses modelos são representados por tipos nativos, características da linguagem de programação, porém é possível também encontrar dados já existentes na linguagem por outros modelos criados pelo próprio programador.

Deparamo-nos com várias linguagens de programação, uma mais robusta que a outra, encontramos linguagens voltadas a orientação a objetos entre outras. Cada uma com o seu ideal, de diferentes características e apelos de mercado. A linguagem Java é voltada à orientação a objeto, e por que devemos utiliza-la? A linguagem Java é obrigatoriamente orientada a objeto e de acordo com SANTOS (2003, p. VI) “O código-fonte de um programa ou classe em Java pode ser compilado em qualquer computador, usando qualquer sistema operacional, contanto que este tenha uma máquina virtual Java adequada”, ou seja, utilizando a linguagem Java temos uma vasta possibilidade de execução dos programas desenvolvidos em quaisquer sistemas operacionais nos dando assim a liberdade

de criação sem nos preocupar tanto se o programa rodara nas outras máquinas em que o programa não foi desenvolvido.

Nosso objetivo é mostrar às vintes (20) cidades mais poluídas do mundo e enfatizando os principais problemas enfrentados por cada uma delas, sabemos que não é uma coisa fácil, pois há muitos interesses em volta do tema, mas a conscientização das pessoas é o primeiro passo a ser tomado. O aplicativo visa mostrar onde estão localizadas essas cidades, sua população, seu PIB, entre outras informações.

A linguagem de programação Java é uma das mais utilizadas e vai proporcionar a gente uma maior compreensão sobre o tema do trabalho, por possuir ferramentas que torna a linguagem de fácil compreensão e uma divisão que permite uma maior limpeza do programa. Deixando-o de forma bem natural e expansiva, nos dando também uma vantagem na hora da manutenção da aplicação e da reutilização do seu código para futuros eventos e programas.

Com o passar dos anos, nos deparamos com inúmeros casos de desastres ambientais, portanto o nosso projeto dispõe-se a fazer um aplicativo informativo, que leva informações as pessoas e possíveis formas de tratamentos dos problemas pré-estabelecidos.

3. CONCEITOS GERAIS DA PROGRAMAÇÃO ORIENTADA A OBJETO

Desenvolvida pela Sun Microsystems em 1995, apesar de ter sido uma linguagem nova foi aceita espetacularmente pelos programadores do mundo inteiro, espalhou-se e se tornou famosa como nunca antes ocorreu com uma linguagem de programação. De acordo com Braz (2011, p. 4) “Um fator que colaborou com isso, é o fato da linguagem possuir vantagens agregadas tais como: orientação a objetos, independência de plataforma, multitarefa, robusta, segura e distribuída.” Contudo o Java foi e ainda é uma linguagem famosa e utilizada no mundo todo pela vasta vantagem que a mesma oferece.

3.1 A Plataforma Java

A tecnologia Java se divide em três (3) plataformas com objetivos específicos, são eles: Java 2 Standard Edition(J2SE), Java 2 Enterprise Edition (J2EE) e o Java 2 Micro Edition (J2ME).

- **Java 2 Standard Edition(J2SE)** – são ferramentas e APIs (Application Program Interface) essenciais para quaisquer aplicações feitas em Java incluindo para as outras plataformas, essa plataforma do Java é suficiente para desenvolver aplicações desktop com ou sem interface gráfica, pois ela possibilita inúmeros recursos para implementar uma aplicação.
- **Java 2 Enterprise Edition (J2EE)** – O J2EE serve para o desenvolvimento de aplicações distribuídas englobando tecnologias como RMI(Remote Method Invocation), EJB(Enterprise JavaBeans), CORBA(Common Object Request Broker Architecture), JMS(Java Message Service) etc.. possibilitando assim uma criação mais ampla
- **Java 2 Micro Edition (J2ME)** – Essa plataforma é utilizada para o desenvolvimento de aplicações para aparelhos portáteis tais como palms, celulares, eletrodomésticos entre outros.

3.2– A Máquina Virtual Java

Segundo Braz (2011, p. 5) “O JRE é um conjunto de programas que possibilita executar aplicações Java. O coração do JRE é a Máquina Virtual Java ou Java Virtual Machine (JVM).” Com a JVM (Java Virtual Machine) possibilitando uma das características mais impressionantes da linguagem Java a visibilidade do mesmo se torna mais ampla e concreta, pois tem uma vasta expansão na portabilidade do código. Abaixo demonstraremos um pouco como compreender a funcionalidade da JVM.

- Processo de Compilação: o programa é compilado para o código de máquina da plataforma que será executado, pois é compilado para bytecode, pois ele é genérico podendo assim ser executado sem problema em qualquer plataforma, já que não é específico para nenhum sistema operacional em particular.
- Execução: O bytecode é interpretado pela Java Virtual Machine existindo uma JVM para cada plataforma onde a tecnologia Java poderá ser executada e a mesma deve encontrar-se instalada no computador no qual será executado um programa ou aplicação Java.

3.3 – Variáveis em Java

De acordo com Braz (2011, p.13) “Como Java é uma linguagem fortemente tipada, **todas as variáveis devem ser declaradas antes de serem usadas**. O tipo de uma variável determina o tipo de informação que pode ser armazenada nela.” Ou seja, o Java nos disponibiliza uma série de instrumentos que podem ser utilizados de maneira simples possibilitando a adaptação de uma ou mais informação contida em um programa, as variáveis que definimos torna a visibilidade e a praticidade de um programa mais natural.

Ao declararmos uma variável é reservada uma porção de memória a qual os dados serão armazenados e mantidos em segurança. Toda variável recebe as seguintes características:

- Qualquer variável deve ser obrigatoriamente declarada antes de ser usada.
- Deverá possuir nome, tipo e um escopo.
- Variáveis declaradas dentro de métodos são variáveis locais, já no corpo do programa são atributos.
- Toda variável é obrigatoriamente inicializada.

- Iniciar com letras de a-z/A-Z, underline (_) ou um sinal de dólar (\$), os demais subsequentes podem conter números. Como toda linguagem de programação possui suas próprias palavras-chaves, a linguagem Java não fica atrás, assim não sendo possível a utilização da mesma em seu programa.

3.4 – Tipos Numéricos em Java

Na linguagem Java, um valor faz referência a um objeto ou a um dos oito (8) tipos de dados primitivos. São divididos da seguinte forma:

- Seis são tipos numéricos, quatro inteiros e dois para pontos flutuantes.

Ao utiliza-lo devemos levar em consideração a alocação que cada um dos tipos possui, para que na hora de chama-los não ocorrer um overflow, isso dar-se por um calculo exceder o intervalo do tipo numérico.

Um fator predominante é que o overflow não é um problema para o ponto flutuante de dupla precisão, pois o seu alcance e reserva que ele aloca é o suficiente para muitos casos de operações. Já erros de arredondamento são um dos casos sérios com os pontos flutuantes, pois pode ocorrer quando se faz uma conversão entre números binários e decimais ou entre números inteiros e flutuantes. Para resolver esse problema podemos utilizar o tipo long, pois alocamos mais espaço para as operações acontecerem.

Abaixo uma tabela especificando o tipo, a descrição e o tamanho de cada um dos tipos primitivos.

Tipos Primitivos		
Tipo	Descrição	Tamanho
Int	Tipo inteiro, com intervalo -2.147.483.648. . . 2.147.483.647 (cerca de 2 bilhões)	4 bytes
Byte	Tipo que descreve um único byte, com intervalo -128. . . 127	1 byte
Short	Tipo inteiro curto, com intervalo -32768. . . 32767	2 bytes
Long	Tipo inteiro longo, com intervalo -9.223.372.036.854.775.808. . . 9.223.372.036.854.775.807	8 bytes
Double	Todo ponto flutuante de dupla precisão, com um intervalo de aproximadamente $\pm 10^{308}$ e aproximadamente 15 dígitos	8 bytes

	decimais significativos	
Float	Todo ponto flutuante de precisão simples, com um intervalo de aproximadamente $\pm 10^{38}$ e aproximadamente 7 dígitos decimais significativos	4 bytes
Char	Tipo caractere, representando unidades de código no esquema de codificação Unicode	2 bytes

3.5 Comandos de Decisão If – Else, Switch – Case.

Os programas que desenvolvemos muitas vezes tem a necessidade de tomar uma decisão, seja ela favorável ou não ao usuário, levando sempre em conta uma condição, tornando-a verdadeira ou falsa. Porque utilizamos os comandos de decisão? Exatamente para guiar o usuário a passos adiante e manter um controle maior sobre o programa em si.

- **O Comando If** – utilizado para implementar uma decisão, composto por duas partes, condição e corpo, quando a condição é verdadeira, o corpo, local onde desenvolve a tomada de decisão, é executado, permitindo assim a execução de diferentes ações com base em uma condição verdadeira.
- **O Comando Else** – executado juntamente com o IF, caso uma condição seja verdadeira, o IF automaticamente executara o corpo da tomada de decisão, e se ela não for verdadeira? Como funciona? O programa parará? Dará erro? Na verdade não, é aí que vem o else, para continuar a validação daquilo que é preciso na hora da implementação, se a condição for verdadeira execute isso, se não for execute aquilo ou termine.
- **O Comando Switch – Case** – utilizado para avaliação de uma expressão que contenha vários resultados possíveis, é bastante útil, pois não é necessária a repetição de vários comandos de repetição, basta escolher e ir declarando o que tem que ser feito, levando em consideração que a expressão do switch só deve resultar em char, byte, short ou int. a estrutura dessa tomada de decisão se desenvolve na seguinte maneira, switch(expressão){
case 0:(condição)

```
break; }
```

A condição case, segue a seguinte linha de raciocínio, caso o usuário escolha a condição estabelecida na parte case, o código que será executado é o que está dentro dela, separando assim muitas opções possíveis.

3.6 – Comandos de Repetição While, Do-While, For.

Comandos utilizados para a repetição de uma ação várias e várias vezes, sem a necessidade da duplicação da mesma linha de código. Possibilitando assim um encurtamento do código-fonte e uma visibilidade melhor do que está ocorrendo.

- **Comando While** – utilizando a implementação de um loop para a execução de um bloco de comando muitas vezes é a função que o while enquanto a condição estabelecida for verdadeira, a forma básica da instrução que o ele tem é a seguinte: while(condição) comando_ou_bloco. A condição que o while recebe é de formado booleano ou uma expressão cujo resultado seja booleano.
- **Comando Do-While** – esse comando é utilizando quando deseja que o corpo do laço seja executado pelo menos uma vez, isto faz com que a variável de inicio receba o valor falso, garantindo a execução do laço. A condição da instrução que ele tem é a seguinte: do {comando_ou_bloco} while(condição);.
- **Comando For** – esse comando possui uma parte inicial com a inicialização das variáveis, seguida pela expressão da comparação e a parte do incremento ou decremento das variáveis do laço. Essa estrutura é controlada por contadores que se agrupam na inicialização das variáveis, sendo controlada por uma única instrução. Estrutura: for (inicialização; condição; iteração) { comandos; }.

3.7 – Projetando Classes - Métodos, Construtores, Getters e Setters, Modificadores de acesso.

Projetar uma classe às vezes pode ser um desafio quando não são desenvolvidos corretamente, ou às vezes os resultados que esperamos não tem uma qualidade, abaixo encontraremos alguns objetivos e conceitos que devemos ter claro na hora da criação de qualquer programa.

- **Métodos** – Todo e qualquer objeto que é definido num programa orientado a objeto, é composto de métodos, levando em consideração que os mesmo devem ser sempre definidos dentro de uma classe.
- **Construtores** – é desenvolvido do mesmo modo que uma função, porém este método leva consigo o nome que a classe criada.
- **Getters e Setters** – utilizados para obter uma proteção maior na hora da criação das variáveis, dando um acesso não direto a elas, na hora da implementação, ao colocar uma variável como **private**, ela só poderá ser acessada através dos métodos getters e setters.
- **Modificadores de Acesso** – na linguagem Java o acesso dos atributos e métodos são controlados através dos modificadores de acesso, e cada um desenvolve uma função, são eles:
 - **public:** método é publico, possibilitando que qualquer um atributo consiga ser modificado ou alterado por quaisquer valor através da mesma classe, derivadas ou qualquer outra, pois ele pode ser utilizado por qualquer um.
 - **protected:** método um pouco mais protegido, podendo ser acessível através da mesma classe ou de classes derivadas dela, ou seja, utilizando o conceito de herança
 - **private:** método mais protegido, os atributos que são declarados privados somente são acessíveis pela mesma classe, tornando-o mais seguro e confiável.

3.8 – Orientação a Objeto – Encapsulamento, Herança, Polimorfismo, Classe Abstrata, Interfaces, Classes Finais.

- **Encapsulamento** – a ideia de encapsular tem como princípio a utilização do objeto sem ter o conhecimento da sua implementação interna, o objetivo principal gira em torno de separar o usuário do objeto, programador do objeto, este trás consigo alguns benefícios que vão desde a possibilidade da alteração de um método ou a estrutura que se encontra escondido no mesmo, até a criação de programas mais modulares e organizados, pois conseguimos reaproveitar o código.
- **Herança** – Com a herança é permitido a passagem de métodos para as classes herdadas sem precisar a duplicidade do código, pois dá a possibilidade ao programador de criar classes somente com as diferenças existentes, implantada na subclasse, estas que são inseridas através de hierarquia de especializações. A classe mais geral é chamada de **superclasse** e a mais especializada **subclasse**.
- **Polimorfismo** – Com a utilização do polimorfismo consegue-se utilizar o mesmo nome para métodos diferentes e a capacidade dos mesmos. O polimorfismo transforma a criação de programas mais claros, pois não é necessário dar nomes diferentes aos métodos pré-estabelecidos. Dividido em duas formas de utilização, são elas:
 - **Polimorfismo Estático**: criação do método com o mesmo nome porém com o corpo diferente, nesse caso dizemos que o método está **sobrecarregado**, no caso do uso desse polimorfismo devemos lembrar que todos os métodos possuem o mesmo nome porém são diferenciados pelos parâmetros que recebem.
 - **Polimorfismo Dinâmico**: associado ao conceito de herança, ocorrendo quando uma subclasse redefine o método existente na superclasse. Nesse caso dizemos que o método foi **sobrescrito** na subclasse. Este polimorfismo ocorre quando a

classe tem a necessidade de um comportamento mais adequado do que o que foi declarado na superclasse.

- **Classe Abstrata** – Uma classe abstrata é uma classe que não pode ser instanciada, ou seja, não pode ser criado um objeto através dela podendo ser apenas estendida por alguma subclasse mais especializada. Nas classes abstratas podem conter métodos e atributos como em classes convencionais ou métodos abstratos que são obrigatoriamente sobrescritos pelas subclasses. Vale ressaltar que métodos abstratos somente existiram em classes abstratas.
- **Interfaces** – Semelhantes a uma classe abstrata exceto pelo simples fato de que nesta não é possível a implementação de qualquer método concreto ou variável de instancia. Quando uma classe implementa uma interface é obrigatoriamente necessário a implementação de todos os métodos especificados pela interface.
- **Classes Finais** – Representa o final da linhagem, pode ser transformada em objeto, porém essas classes não permitem a passagem de herança, ou seja, o que foi implementado nela, continua nela.

3.9 – Captura e Tratamento de Erros – Try-Catch, Finally, Throws.

Com a captura e o tratamento de erros podemos rodar um código-fonte sem dificuldades, pois ao mesmo tempo em que ele apresente um erro, a função que a linguagem Java disponibiliza o trata, evitando assim o travamento do servidor ou a não compilação do programa. Um programa implementado com as funções de tratamento de erros, tem uma naturalidade maior, uma confiabilidade melhor, por isso é recomendado o uso do mesmo. Encontramos vários blocos de tratamento de erros que nos ajudam a um melhor ajuste e resolução do código.

- **Try-Catch** – este bloco é utilizado para que ocorra o tratamento do erro, necessitamos usar o try-catch, porque ocorre uma operação de conversão do que foi feito para o que precisa ser feito, sendo uma forma mais robusta de tratar os possíveis erros de conversão.

- **Finally** – o bloco finally inclui comandos que liberam recursos que podem ter sido alocados durante a liberação e o processamento do bloco try, sendo ele finalizado com sucesso ou não, porém a presença desse bloco é opcional.
- **Throws** – agindo na assinatura do método ele indica se este método poderá lançar uma exceção que deve ou não ser tratada.
- **Throw** – este método se responsabiliza por lançar uma exceção sem exigir que ela seja tratada pelos seus chamadores, pois o throw transfere o controle do fluxo para os métodos chamadores.

4.0 Banco de dados.

4.1 Modelagem banco de dados.

Existem três modelos de banco de dados.

Modelo físico.

O modelo físico leva em consideração os limites impostos pelo SGBD e pelos requisitos não funcionais dos programas que acessam os dados. Características:

- Elaborado a partir do modelo lógico.
- Pode variar segundo o SGBD.
- Pode ter tabelas físicas.
- Pode ter colunas físicas.
- Possui entidades e atributos.

Entidades.

Qualquer coisa existente no mundo, abstratas ou concretas, na qual se deseja guardar informações

Atributos.

São as propriedades das entidades

Modelo lógico.

O modelo logico leva em conta os limites impostos por algum tipo de tecnologia de banco de dados. Suas características são:

- Deriva do modelo conceitual e visa a representação do negócio.
- Possui entidades associativas no lugar dos relacionamentos.
- Define as chaves primárias das entidades.
- Normalização até a terceira forma normal.
- Adequação ao padrão de nomenclatura.
- Entidades e atributos documentados.

Modelo conceitual.

O modelo conceitual engloba as regras de negócios sem limitação tecnológica ou de implementação. Características :

- Visão geral do negócio.
- Facilitação do entendimento entre usuários e desenvolvedores.
- Possui somente as entidades e atributos principais.
- Pode ter relacionamento de n pra m.

Os quatro tipos de atributos.

- Obrigatórios: Tem que possuir um valor
- Opcional: Pode possuir um valor ou não

- Identificador: Identifica de forma exclusiva cada ocorrência da entidade também conhecida como chave primária
- Chave candidata: Atributos ou grupos de atributos que tem a propriedade de identificar unicamente uma ocorrência da entidade. Pode vir a ser uma chave primária

Características de uma chave primária.

- Não pode haver duas ocorrências de uma mesma entidade com o mesmo conteúdo.
- Não pode ser nula.
- Os atributos identificadores devem ser o conjunto mínimo que pode identificar cada instância de uma entidade
- Não devem ser usadas chaves externas
- Cada atributo deve possuir um tamanho reduzido
- Não deve conter informação volátil

4.2 Relacionamentos.

Um relacionamento pode ser entendido como uma associação entre instâncias de entidades devido às regras de negócio. Normalmente ocorre entre duas ou mais entidades, mas, pode haver dentro da mesma entidade.

Para definir a quantidade de relacionamentos usamos o conceito de cardinalidade.

4.3 Cardinalidade.

A cardinalidade indica quantas ocorrências de uma entidade participam no mínimo e no máximo do relacionamento.

- Mínima: Define o relacionamento entre duas entidades. É obrigatório ou não
- Máxima: Define a quantidade máxima de ocorrências da entidade que pode participar do relacionamento

Tipos de cardinalidade.

- 1 pra 1: A chave primária se desloca do lado obrigatório para o lado opcional
- 1 pra N: A chave primária se desloca do lado 1 pro lado N.
- Nesse caso a chave primária passa a se chamar chave estrangeira.
- N pra N: Leva ao modelo lógico e precisa de mais uma entidade. Chamada de associativa.

4.4 Normalização.

É o conjunto de regras que visa minimizar as anomalias de modificação dos dados e dar maior flexibilidade em sua utilização.

Existem cinco formas normais, porém três são as mais utilizadas.

Primeira forma normal.

Uma relação está na 1FN se somente todos os domínios básicos contiverem apenas valores atômicos.

➤ Procedimentos.

- ✓ Identificar a chave primária da entidade.
- ✓ Identificar o grupo repetitivo e excluir da entidade.
- ✓ Criar uma nova entidade com a chave Primária da entidade anterior e o grupo repetitivo.

A chave primária da nova entidade será obtida pela concatenação da chave primária da entidade inicial e do grupo repetitivo.

Segunda forma normal.

Uma relação está na 2FN se e somente se ela estiver na primeira e todos os atributos não chaves forem totalmente dependentes da chave primária.

➤ Procedimentos.

- ✓ Identificar os atributos que não são funcionalmente dependentes de toda a chave primária.
- ✓ Remover todos esses atributos identificados da entidade e criar uma nova entidade com eles.
- ✓ A chave primária da nova entidade será o atributo do qual os atributos removidos são funcionalmente dependentes.

Terceira forma normal.

Uma relação está na 3FN se somente estiver na 2FN e todos os atributos não chaves forem dependentes não transitivos da chave primária

➤ Procedimentos.

- ✓ Identificar todos os atributos que são funcionalmente dependentes de outros atributos não chave
- ✓ Remover e criar uma nova entidade com eles
- ✓ A chave primária da nova entidade será o atributo do qual os atributos removidos são funcionalmente dependentes.

Após a normalização as estruturas dos dados estão projetadas para eliminar as inconsistências e redundâncias dos dados, eliminando dessa forma qualquer problema de atualização e operacionalização do sistema.

4.5 Considerações para programa.

O nosso objetivo é integrar as criações em Java com a modelagem em banco de dados, pois para fazer o aplicativo sobre as cidades mais poluídas do mundo é necessário ter um banco de dados com essas cidades e as informações sobre elas tais como: população, razões da poluição, países em que se encontram, nível de poluição, PIB, entre outras.

Essas informações são necessárias pois servem como parâmetro para avaliar as causas e definir possíveis soluções para cada cidade, tipos alternativos de energias que podem ser utilizados na cidade por exemplo.

Lógico que sabemos que tem vários interesses no em torno do assunto mas aí cabe a cada governo e a cada população fazer a sua parte, esse tipo de tema não será abordado no trabalho, a nossa intenção é apenas mostrar possíveis soluções para cada cidade, a título de ilustração dos problemas que geram tanta poluição, o que provoca problemas de saúde na população.

A poluição é um problema muito grave e deve ter um método de controle, a população também é responsável pela poluição, o principal é conscientizar as pessoas que não devem colaborar com o aumento da poluição, tendo uma conduta mais adequada sobre o tema.

O trabalho é apenas a visão das pessoas do grupo sobre o tema, outras soluções com certeza podem ser utilizadas no tratamento da poluição, o que possibilita ampla discussão sobre o tema debatido, o que pode ser feito, como fazer, quanto tempo levaria, quais os custos, quais os recursos que cada cidade possui pra reverter a situação, com certeza o tema é polêmico mas não temos a intenção de gerar polêmica, apenas dar a nossa visão sobre o tema.

- **5.7 Projeto – estrutura do programa**

Por meio dos paradigmas de orientação a objetos, podemos criar aplicações complexas, que seguem execução de forma cascadeada, as vezes de forma direta e fluente, as vezes inconstante e retornado a pontos específicos, essa é a programação orientada a objetos, que foi empregada de forma prioritária nesta aplicação, mesclando vários aspectos e técnicas de programação, ferramentas de modelagem, e recursos disponíveis no IDE , buscamos confeccionar algo que funcionasse de forma plena e versátil, rápida e bem apresentada e simples mas bem trabalhada.



```
1 package MeioAmbiente;
2
3 //começo do programa, lista com todos os imports usados nessa classe.
4
5 import java.awt.Color;
6 import java.awt.Component;
7 import java.awt.Desktop;
8 import java.awt.EventQueue;
9 import java.awt.Font;
10 import java.awt.event.ActionEvent;
11 import java.awt.event.ActionListener;
12 import java.awt.event.MouseAdapter;
13 import java.awt.event.MouseEvent;
14 import java.net.URI;
15 import javax.swing.ImageIcon;
16 import javax.swing.JButton;
17 import javax.swing.JDesktopPane;
18 import javax.swing.JInternalFrame;
19 import javax.swing.JLabel;
20 import javax.swing.JMenu;
21 import javax.swing.JMenuBar;
22 import javax.swing.JMenuItem;
23 import javax.swing.JPopupMenu;
24 import javax.swing.JTextField;
25 import javax.swing.SwingConstants;
26
27 public class UmPoucoDeCadaPais extends JInternalFrame {
28     /**
29      *
30      * private static final long serialVersionUID = 1L;
31      * //private JPanel contentPane;
32      * //private JTextField txtParaVisualizarA;
33      * private JTextField textFieldVisualizar;
34      *
35      * /**
36      *
37      * * Launch the application.
38      * */
39 }
```

Figura 1: Estrutura do programa

Fonte própria maio de 2017

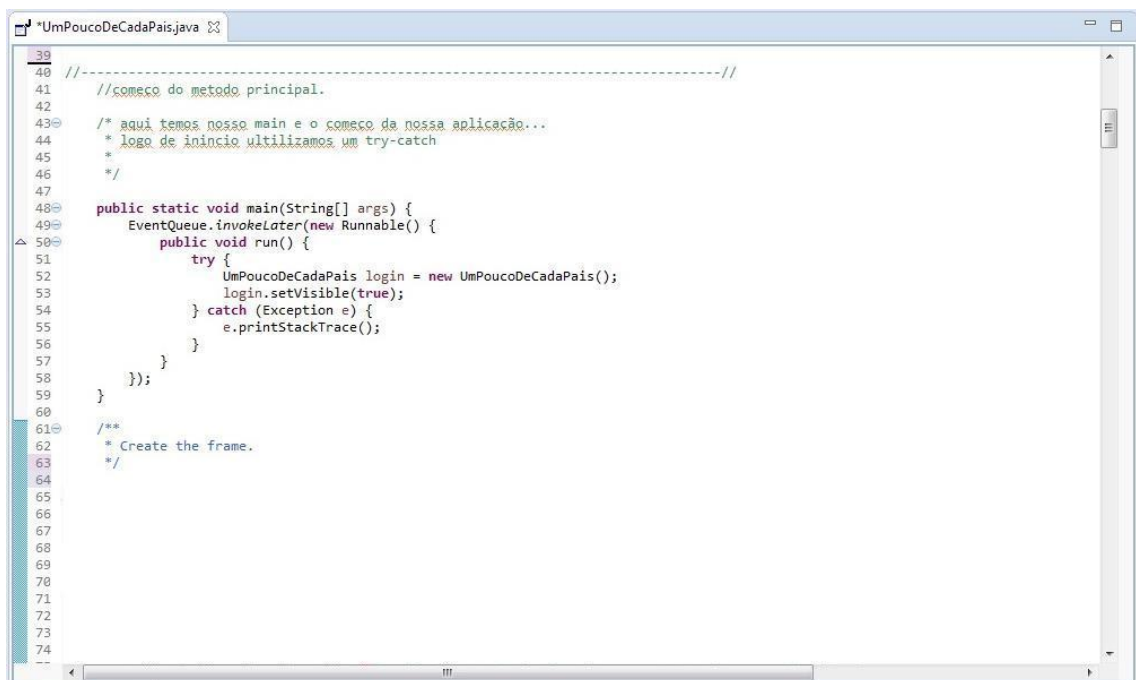
Aqui começamos nossa apresentação, e também a modelagem desta parte do programa, usaremos como material de amostra, a segunda principal janela da nossa aplicação, que poderá ser vista pleno funcionamento e finalizada mais abaixo.

Como podemos observar na primeira linha temos o package¹ do trabalho, todas as nossas classes encontram-se no mesmo package, logo na linha 5

encontramos o começo da linha de importações, isso faz com que classes, métodos, e algumas funções específicas fiquem disponíveis na classe atual onde se trabalha.

- **JAVA.AWT** (Abstract Windows Toolkit) é uma toolkit gráfica original da linguagem JAVA, atualmente, a AWT faz parte da JFC (**J**ava **F**oundation **C**lasses) uma API padrão, responsável pela parte gráfica IDE / Usuário.
- **JAVA.NET** é uma tecnologia JAVA voltada para a Web, e auxílio de desenvolvimento para aplicações focadas na mesma.
- **JAVA. Swing** é uma biblioteca gráfica que veio a surgir a partir do JAVA 1.2, e acabou superando sua antecessora a **AWT**, principal motivo disso, foi por possuir diversos benefícios que suas antecessoras não possuíam. Hoje em dia o JAVA fornece ao programador a chance de trabalhar com as duas em conjunto.

Após as devidas importações, temos a declaração de duas classes, uma que da inicio a nossa janela em questão.



```
39
40 //-----//
41 //começo do metodo principal.
42
43 /** aqui temos nosso main e o começo da nossa aplicação...
44  * logo de início utilizamos um try-catch
45  */
46
47
48 public static void main(String[] args) {
49     EventQueue.invokeLater(new Runnable() {
50         public void run() {
51             try {
52                 UmPoucoDeCadaPais login = new UmPoucoDeCadaPais();
53                 login.setVisible(true);
54             } catch (Exception e) {
55                 e.printStackTrace();
56             }
57         }
58     });
59 }
60
61 /**
62  * Create the frame.
63  */
64
65
66
67
68
69
70
71
72
73
74
--
```

Figura 2: Estrutura do programa.

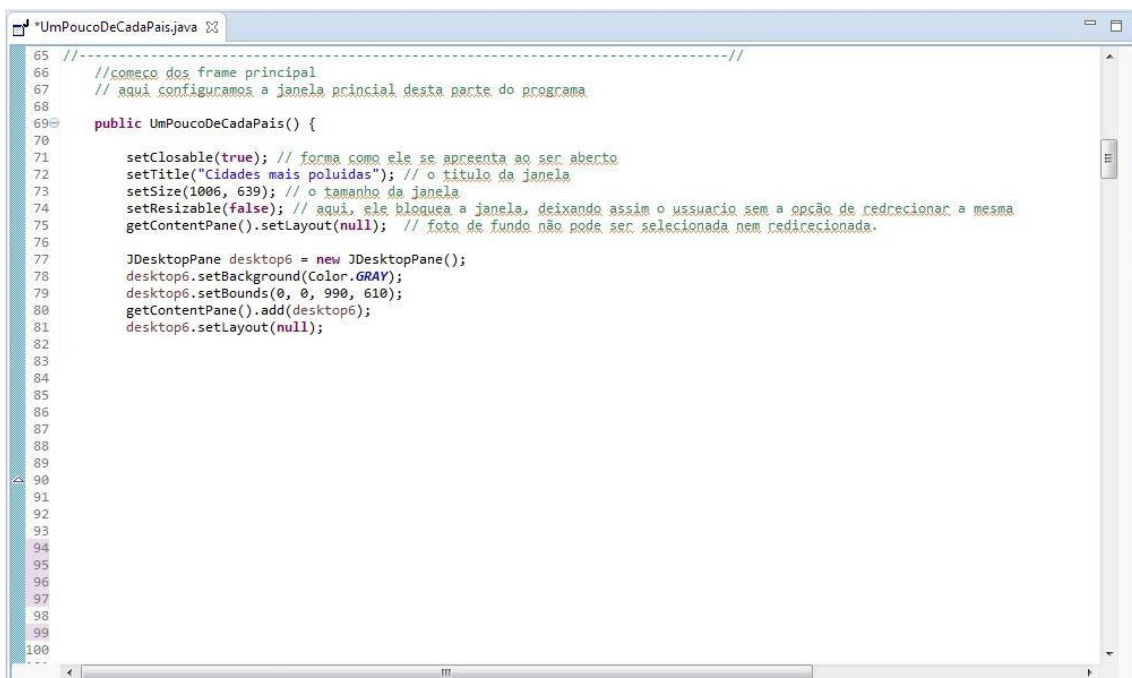
Fonte própria maio de 2017

Na imagem 02 podemos ver o começo do desenvolvimento da classe, com a devida declaração do método Main na linha 48, seguida de um gatilho de evento, e o primeiro de muitos tratamentos de exceções.

Pratica essa extremamente vital, e importante para a integridade da aplicação, o programa não deveria de forma alguma possuir erros na sua estrutura ou execução, mas como sabemos, isso é algo que foge do controle dos programadores, para lidarmos com imprevistos que poderiam resultar em travamentos catastróficos para um servidor, um usuário, ou até para nos mesmos, fazemos uso dessa função indispensável.

A função consiste em, ter possíveis partes do código que podem vir a dar erro, sendo “envoltas” em uma corda de segurança, que no pior dos casos, não travara nosso programa, mas sim, executara a melhor forma que encontramos para resolver os problemas que previmos e já buscamos alguma solução previa que deixaremos

pré-programadas .



```
65 //-----//
66 //começo dos frame principal
67 // aqui configuramos a janela principal desta parte do programa
68
69 public UmPoucoDeCadaPais() {
70
71     setClosable(true); // forma como ele se apresenta ao ser aberto
72     setTitle("Cidades mais poluídas"); // o título da janela
73     setSize(1006, 639); // o tamanho da janela
74     setResizable(false); // aqui, ele bloqueia a janela, deixando assim o usuário sem a opção de redirecionar a mesma
75     getContentPane().setLayout(null); // foto de fundo não pode ser selecionada nem redirecionada.
76
77     JDesktopPane desktop6 = new JDesktopPane();
78     desktop6.setBackground(Color.GRAY);
79     desktop6.setBounds(0, 0, 990, 610);
80     getContentPane().add(desktop6);
81     desktop6.setLayout(null);
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
---
```

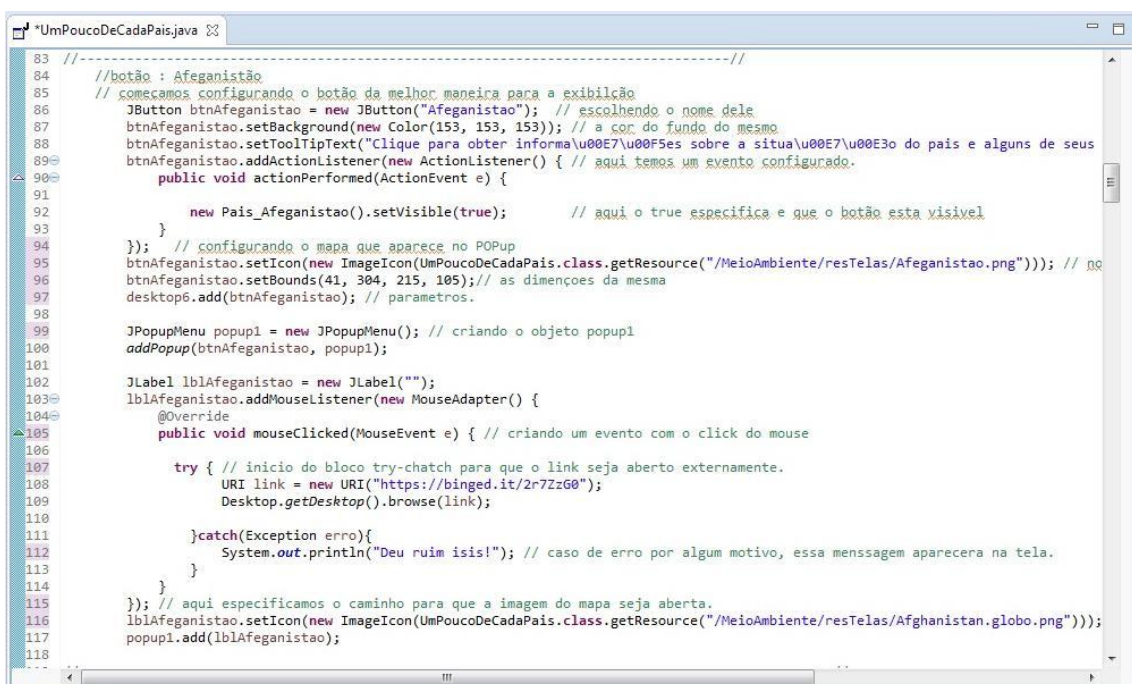
Figura 3: Estrutura do programa.

Fonte própria maio de 2017

Na imagem 03, temos o começo das configurações da janela em si, onde teremos todos os botões, plano de fundo, mapas, e textos.

Começamos o processo com um método, ele que carrega tudo que será exibido.

Primeiramente temos uma sequencia de “Set’s” que definem o comportamento da janela, como ela será aberta, nome, tamanho, se poderá ser redirecionada, e em seguida, temos um objeto do tipo JDesktopPane sendo instanciado, esse objeto que carregara a cor de fundo dos botões, seus tamanhos, se serão sempre visíveis, e suas respectivas localizações na janela atual ao qual pertencem.



```
83 //-----//
84 //botão : Afeganistão
85 // começamos configurando o botão da melhor maneira para a exibição
86 JButton btnAfeganistao = new JButton("Afeganistao"); // escolhendo o nome dele
87 btnAfeganistao.setBackground(new Color(153, 153, 153)); // a cor do fundo do mesmo
88 btnAfeganistao.setToolTipText("Clique para obter informa\u00E7\u00F5es sobre a situa\u00E7\u00E3o do pais e alguns de seus
89 btnAfeganistao.addActionListener(new ActionListener() { // aqui temos um evento configurado.
90     public void actionPerformed(ActionEvent e) {
91         new Pais_Afeganistao().setVisible(true); // aqui o true especifica e que o botão esta visivel
92     }
93 }); // configurando o mapa que aparece no POPup
94 btnAfeganistao.setIcon(new ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/Afeganistao.png"))); // no
95 btnAfeganistao.setBounds(41, 304, 215, 105); // as dimensões da mesma
96 desktop6.add(btnAfeganistao); // parametros.
97
98 JPopupMenu popup1 = new JPopupMenu(); // criando o objeto popup1
99 addPopup(btnAfeganistao, popup1);
100
101 JLabel lblAfeganistao = new JLabel("");
102 lblAfeganistao.addMouseListener(new MouseAdapter() {
103     @Override
104     public void mouseClicked(MouseEvent e) { // criando um evento com o click do mouse
105         try { // inicio do bloco try-chatch para que o link seja aberto externamente.
106             URI link = new URI("https://binged.it/2r7ZzG0");
107             Desktop.getDesktop().browse(link);
108         } catch (Exception erro) {
109             System.out.println("Deu ruim isis!"); // caso de erro por algum motivo, essa mensagem aparecera na tela.
110         }
111     }
112 }); // aqui especificamos o caminho para que a imagem do mapa seja aberta.
113 lblAfeganistao.setIcon(new ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/Afghanistan.globo.png")));
114 popup1.add(lblAfeganistao);
115
116
117
118
```

Figura 4: Estrutura do programa.

Fonte própria maio de 2017

Acima temos a imagem 04, e na mesma, talvez a parte mais complexa da nossa janela, os botões, mas não são todos, escolhemos aqui o primeiro da lista, todos são exatamente igualmente aplicados.

Logo de começo na linha 86 temos a instanciação da classe JButton e a criação do nosso botão como objeto de nome “Afeganistão”, na ordem, tomamos a escolha das cores do mesmo.

Logo após, na linha 88, temos um Popup que carrega um texto específico que será exibido ao usuário dando uma pequena dica, do que poderá ser encontrado ao clicar ali.

Na linha 92, podemos perceber que um novo objeto foi criado, e configurado para se tornar visível assim que o botão receber um click, logo mais falaremos sobre ele .

Na linha 95 continuamos com as configurações do botão, escolhendo em seguida, o ícone do mesmo, este, que já avia sido previamente carregado na pasta do projeto, seu tamanho e a forma com a qual será apresentado.

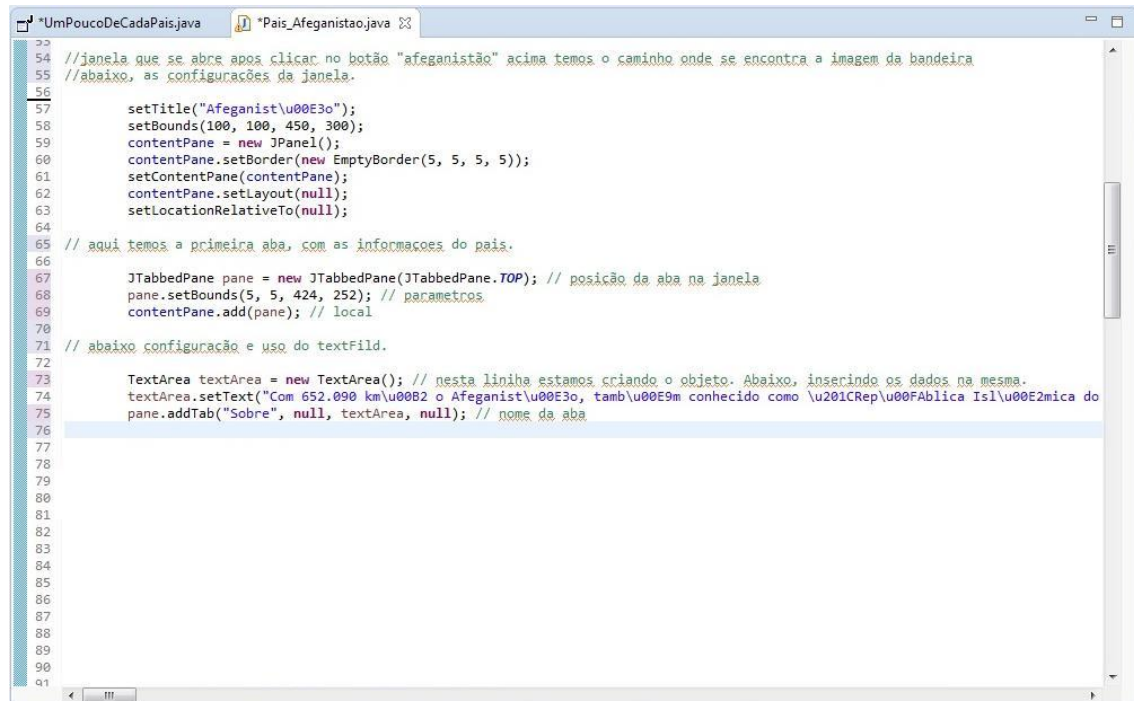
Após, logo na linha 103, criamos outro evento clique, aqui, temos uma peculiaridade, neste evento acontece muitas coisas, a primeira delas, e abrir o mapa, como podemos observar na linha 116, o caminho para o mesmo já está devidamente configurado, a segunda coisa, e a possibilidade de abrir o navegador clicando no mapa, para isso, fazemos a sobrescrita do evento click, como pode ser observado na linha 104 e logo depois, abrimos um bloco de tratamento.

Importante comentarmos que, sempre que trabalharmos com links, arquivos externos, e fluxo de dados saindo e entrando da aplicação, seremos obrigados a fazer o processo de tratamento de possíveis erros.

Dizendo isso, percebamos agora na linha 108, usamos aqui uma URL como parâmetro, e criamos um objeto da classe URI, demos um “getDesktop().browser” e passamos como parâmetro, nosso objeto link, que já tem nele, o página que desejamos que se abra.

No segundo bloco, temos uma mensagem particular caso, por algum motivo o programa não consiga abrir o link, cada botão, de cada país, tem uma mensagem específica, facilitando assim, caso venha o erro, saber com certeza onde ele ocorreu.

E por ultimo na linha 116, temos o começo do código que nos dará o nosso mapa, especifico para cada pais.



```
54 //janela que se abre apos clicar no botão "afeganistão" acima temos o caminho onde se encontra a imagem da bandeira
55 //abaixo, as configurações da janela.
56
57 setTitle("Afeganistão");
58 setBounds(100, 100, 450, 300);
59 contentPane = new JPanel();
60 contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
61 setContentPane(contentPane);
62 contentPane.setLayout(null);
63 setLocationRelativeTo(null);
64
65 // aqui temos a primeira aba, com as informações do país.
66
67 JTabbedPane pane = new JTabbedPane(JTabbedPane.TOP); // posição da aba na janela
68 pane.setBounds(5, 5, 424, 252); // parametros
69 contentPane.add(pane); // local
70
71 // abaixo configuração e uso do textFiled.
72
73 TextArea textArea = new TextArea(); // nesta linha estamos criando o objeto. Abaixo, inserindo os dados na mesma.
74 textArea.setText("Com 652.090 km² o Afeganistão, também conhecido como República Islâmica do Afeganistão");
75 pane.addTab("Sobre", null, textArea, null); // nome da aba
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
```

Figura 5: Estrutura do programa.

Fonte própria maio de 2017

Agora, na serie de imagens que seguem com as numerações 4.1, 4.2, 4.3, mostraremos e falaremos um pouco sobre as janelas abertas após clicarmos nos botões que foram citados na imagem de numero 4.

Podemos perceber que o começo é semelhante ao da nossa janela anterior já na linha 57 têm os parâmetros para essa nova janela, a diferença começa a partir da linha 67 onde se começa a configuração da primeira aba, com especificações da localização e tamanho.

Após, temos a configuração do "TextArea" um campo que usamos bastante nesta aplicação, como o nome já auto explica, podemos colocar textos extensos e formata-los da forma que achamos melhor nesta área, e é justamente isso que ocorre na linha 74, na 75 temos apenas o titulo desta guia.

```

76
77 // abaixo temos nossa segunda aba, com uma planilha.
78
79 table = new JTable();
80 table.setModel(new DefaultTableModel(
81     new Object[][] {
82         {"Posi\u00E7\u00E3o", "Cidade", "Provincia / estado", "Popula\u00E7\u00E3o"},
83         {null, null, null, null},
84         {"1", "Kabul", "Prov\u00EDncia de Kabul", "3 289 000"},
85         {"2", "Candaar", "Prov\u00EDncia de Kandahar", "491 500"},
86         {"3", "Herat", "Herat", "436 300"},
87         {"4", "Mazar e Sharif", "Prov\u00EDncia de Balkh", "368 100"},
88         {"5", "Jalalabad", "prov\u00EDncia de Kunduz", "304 600"},
89         {"6", "Lashkar Gah", "prov\u00EDncia de Takhar", "219 000"},
90         {"7", "Talocan", "prov\u00EDncia de Nangarhar", "206 500"},
91         {"8", "Khost", "prov\u00EDncia de Baghlan", "203 600"},
92         {"9", "Sheberghan", "Parwan", "171 200"},
93         {"10", "Ghazni", "\tjowzjan", "161 700"},
94     },
95     new String[] {
96         "New column", "New column", "New column", "New column" // nome das colunas
97     }
98 ) {
99     /**
100      *
101      */
102     private static final long serialVersionUID = 1L;
103     boolean[] columnEditables = new boolean[] {
104         false, false, false, false
105     };
106     public boolean isCellEditable(int row, int column) {
107         return columnEditables[column];
108     }
109 });
110
111
112
113

```

Figura 6: Estrutura do programa

Fonte própria maio de 2017

Na imagem 4.2 observamos a criação da nossa segunda aba, diferente da primeira, esta não se trata de uma “TextArea” e sim de uma tabela, tabela esta que carrega os dados das principais cidades daquele país em questão .

Como pode ser observado, nosso array tem 5 – 10 posições.

Já na pagina 81 podemos observar a criação de um array de objetos, o mesmo, carregara as nossas Strings, que vão da linha 81 ate a 93 podemos, observar que na linha 96 só aparece o nome “new column” o motivo disso, e que a coluna não esta efetivamente nomeada na estrutura interna do programa só na visual.

```
118 // aqui temos nossa terceira aba, com seus botões
119
121 JButton btnMazar_e_sharif = new JButton("Mazar-e sharif"); // primeiro botão e seu nome
122 btnMazar_e_sharif.addActionListener(new ActionListener() {
123     public void actionPerformed(ActionEvent e) {
124
125         new Cidade_Mazar_e_sharif().setVisible(true); // evento, ao clicar no botão, a nova janela aparecerá
126     }
127 });
128 panel.add(btnMazar_e_sharif); // novo painel
129
131 JPopupMenu popupMenu = new JPopupMenu();
132 addPopup(btnMazar_e_sharif, popupMenu);
133
134 JTextPane txtPnACidadeDe = new JTextPane(); // abaixo as informações sobre a cidade
135 txtPnACidadeDe.setText("A cidade de Mazar'e Sharif esta na 9ª posição das cidades mais poluídas do mundo.");
136 txtPnACidadeDe.setEditable(false); // aqui o texto é bloqueado para edição.
137 popupMenu.add(txtPnACidadeDe);
138
139
141
142 JButton btnCabul = new JButton("Cabul");
143 btnCabul.addActionListener(new ActionListener() {
144     public void actionPerformed(ActionEvent e) {
145
146         new Cidade_Cabul().setVisible(true);
147     }
148 });
149 panel.add(btnCabul);
150
152 JPopupMenu popupMenu_1 = new JPopupMenu();
153 addPopup(btnCabul, popupMenu_1);
154
155
```

Figura 7: Estrutura do programa.

Fonte própria maio de 2017

Aqui temos nossa ultima parte especial, a imagem 4.3 mostra a nossa ultima aba, aqui, temos mais botões, porem não trataremos deles, somente faremos manões, como a forma de estruturar e a mesma já mostrada na imagem 4, diremos o básico.

Aqui temos dois botões sendo configurados, um na linha 121 e o outro na linha 142

1. 121 - Aqui damos inicio ao botão que levava a cidade de Mazar'e Sharif, já nesta mesma linha, podemos ver o objeto sendo criado, e devidamente nomeado.

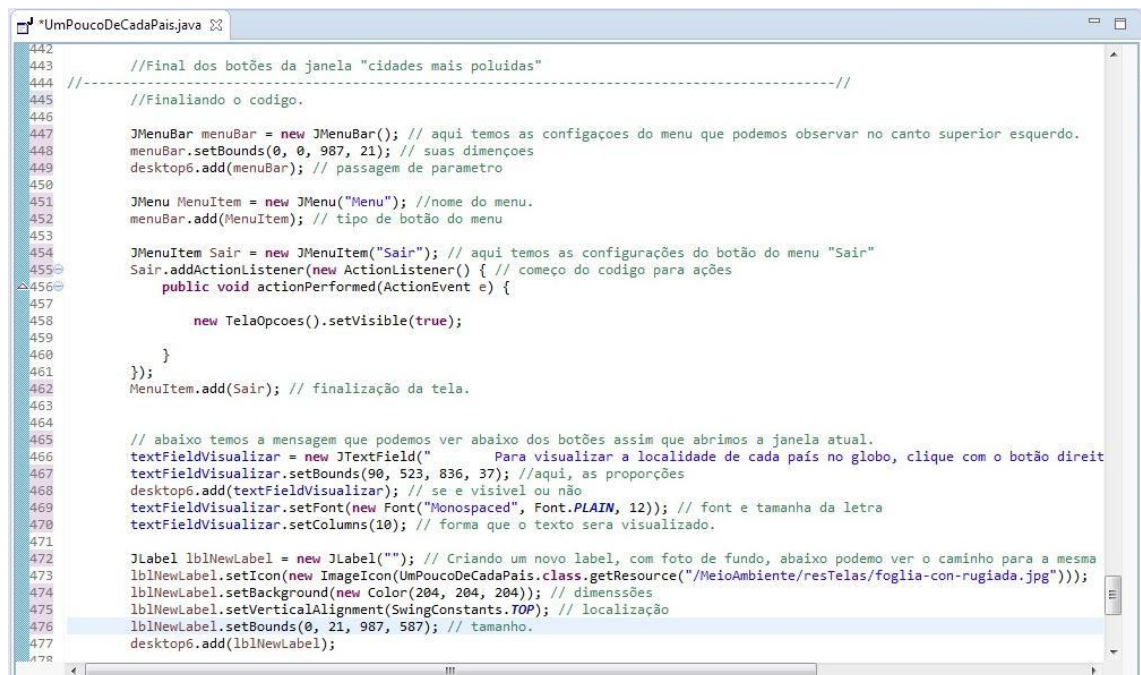
Após isso, na linha 125 temos o evento click que deixa a próxima janela visível.

E na linha 131 temos a criação de um PopUp como ClickEvent, como podemos ler, agora na 135 o texto que aparecerá no será "A cidade de Mazar'e sharif esta na 9ª posição das cidades mais poluídas do mundo", na linha baixo deixamos a função de edição do mesmo como false.

2. 142 - Aqui damos inicio ao botão que levava a cidade de Cabul, já nesta mesma linha, podemos ver o objeto sendo criado, e devidamente nomeado.

Após isso, na linha 143 temos o vento click que deixa a próxima janela visível.

Apesar de não aparecer na imagem, o segmento desta parte se desenrola de maneira igual do primeiro botão, exceto apenas, no conteúdo que será exibido no evento PopUp do seu botão.



```
442 //Final dos botões da janela "cidades mais poluídas"
443 //-----
444 //Finalizando o código.
445
446
447 JMenuBar menuBar = new JMenuBar(); // aqui temos as configurações do menu que podemos observar no canto superior esquerdo.
448 menuBar.setBounds(0, 0, 987, 21); // suas dimensões
449 desktop6.add(menuBar); // passagem de parametro
450
451 JMenu MenuItem = new JMenu("Menu"); //nome do menu.
452 menuBar.add(MenuItem); // tipo de botão do menu
453
454 JMenuItem Sair = new JMenuItem("Sair"); // aqui temos as configurações do botão do menu "Sair"
455 Sair.addActionListener(new ActionListener() { // começo do código para ações
456     public void actionPerformed(ActionEvent e) {
457         new TelaOpcoes().setVisible(true);
458     }
459 });
460 MenuItem.add(Sair); // finalização da tela.
461
462
463
464
465 // abaixo temos a mensagem que podemos ver abaixo dos botões assim que abrimos a janela atual.
466 textFieldVisualizar = new JTextField("Para visualizar a localidade de cada país no globo, clique com o botão direito
467 textFieldVisualizar.setBounds(90, 523, 836, 37); //aqui, as proporções
468 desktop6.add(textFieldVisualizar); // se e visível ou não
469 textFieldVisualizar.setFont(new Font("Monospaced", Font.PLAIN, 12)); // font e tamanho da letra
470 textFieldVisualizar.setColumns(10); // forma que o texto sera visualizado.
471
472 JLabel lblNewLabel = new JLabel(""); // Criando um novo label, com foto de fundo, abaixo podemos ver o caminho para a mesma
473 lblNewLabel.setIcon(new ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/foglia-con-rugiada.jpg")));
474 lblNewLabel.setBackground(new Color(204, 204, 204)); // dimensões
475 lblNewLabel.setVerticalAlignment(SwingConstants.TOP); // localização
476 lblNewLabel.setBounds(0, 21, 987, 587); // tamanho.
477 desktop6.add(lblNewLabel);
478
```

Figura 8: Estrutura do programa.

Fonte própria maio de 2017

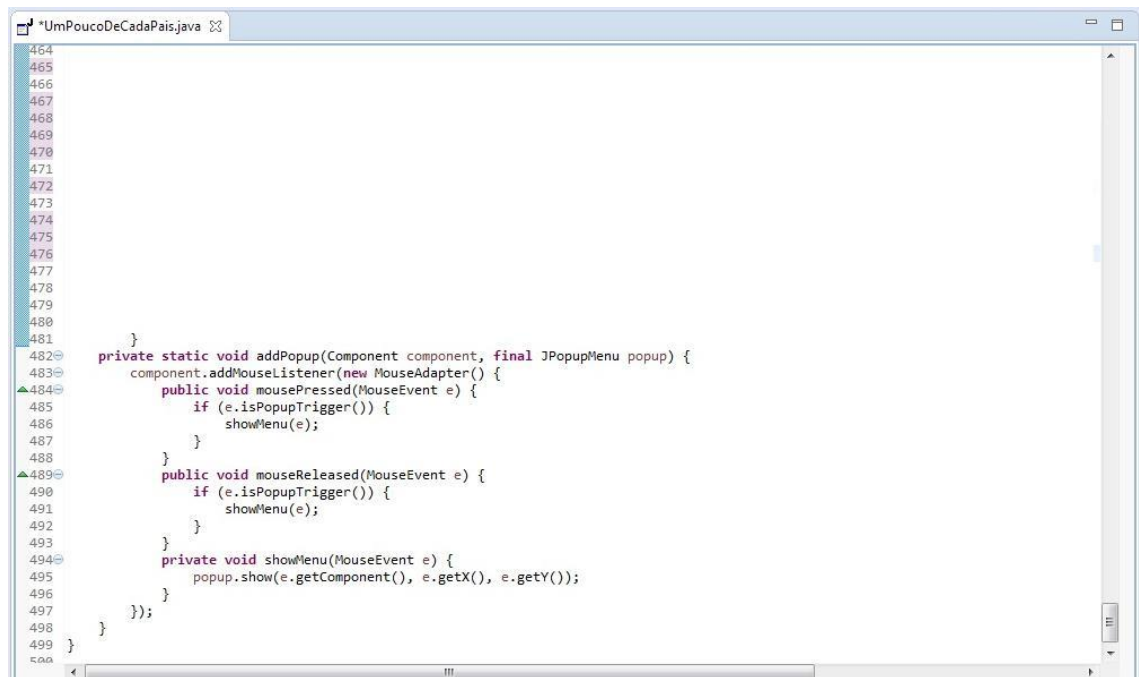
Agora voltaremos para a parte principal da nossa janela, após passarmos pelos botões, veremos seu funcionamento, um pouco da sua estrutura, e o que podemos encontrar ao clicarmos em algum, damos inicio ao encerramento do código na nossa classe main.

Podemos começar observando na imagem 5 nossa finalização começa com a configuração dos nossos menus da janela, já na linha 447, onde criamos um objeto com nome de "menuBar" da classe JMenuBar, abaixo como de padrão, seguem suas dimensões, e o local onde ele será exibido, aqui, passamos o próprio objeto menu como parâmetro.

Na linha 451 temos nosso primeiro botão do menu, que carrega o mesmo nome, curiosamente esse primeiro botão, faz juntamente a abertura do menu, naquele formato cascata que estamos acostumados a observar em quase todos os programas.

Na linha 454 temos a criação da única opção que temos neste menu, a opção de sair, e voltar para a pagina inicial do programa, na logo abaixo começamos as configurações de parâmetros, criando um evento clique, onde deixaremos a tela principal visível (com uso do parâmetro TRUE), assim ela sobrescrevera a nossa janela atual, deixando “invisível” na linha 462, a finalização do comando de menu.

A partir da linha 466 ate a 477 temos uma lista de parâmetros, configurações e atributos que são os principais responsáveis por tudo que o usuário efetivamente vê, enquanto navegam nesta janela, mensagens estáticas, plano de fundo, fonte e coloração das letras, cor da janela, oque pode estar visível, e oque não podera ser alterado por terceiros.



```
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482     }
483     private static void addPopup(Component component, final JPopupMenu popup) {
484         component.addMouseListener(new MouseAdapter() {
485             public void mousePressed(MouseEvent e) {
486                 if (e.isPopupTrigger()) {
487                     showMenu(e);
488                 }
489             }
490             public void mouseReleased(MouseEvent e) {
491                 if (e.isPopupTrigger()) {
492                     showMenu(e);
493                 }
494             }
495             private void showMenu(MouseEvent e) {
496                 popup.show(e.getComponent(), e.getX(), e.getY());
497             }
498         });
499     }
```

Figura 9: Estrutura do programa.

Fonte própria maio de 2017

E exatamente na linha 499^a linha, temos a finalização do da nossa classe, responsável por manter nossa janela sobre poluição e cidades funcionando, ate o final, ainda temos conceitos e aplicações importantes de JAVA funcionando, e apesar de tudo que vimos, ainda a muitos outros conceitos inexplorados, muitas outras formas de se fazer as mesmas coisas que fizemos apenas nessas 499 linhas, inclusive muitas formas as vezes ate mais eficiente. A programação é uma só, mas cada programador tem a sua forma de utiliza-la, de conversar com ela, e faze-la dobrar-se a ele, assim cada programador acaba tendo sua própria forma de programar, interpretar se expressar e mostrar sua forma de ver as coisas, por meio dela.

5.8 Relatório com as linhas de código do programa.

- **Pagina inicial do programa.**

```
public class TelaPrincipal extends JFrame {

    /**
     *
     */

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;

    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run(){

                try{
                    for (LookAndFeelInfo info : UIManager.getInstalledLookAndFeels()) {
                        if ("Nimbus".equals(info.getName())) {
                            UIManager.setLookAndFeel(info.getClassName());
                            break;
                        }
                    }
                } catch (Exception e) {
                    // If Nimbus is not available, you can set the GUI to another look
and feel.

                }

                try{

                    TelaPrincipal frame = new TelaPrincipal();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }

            }
        });
    }
}
```

```

    }

    /**
     * Create the frame.
     */
    public TelaPrincipal() {
        setResizable(false);

        setIconImage(Toolkit.getDefaultToolkit().getImage(TelaPrincipal.class.getResource("/MeioAmbiente/resTelas/planeta terra.jpg")));
        setModalExclusionType(ModalExclusionType.TOOLKIT_EXCLUDE);

        contentPane = new JPanel();
        setContentPane(contentPane);
        contentPane.setLayout(null);
        setTitle("Meio Ambiente");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(1010, 660);
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setLocationRelativeTo(null);

        JDesktopPane desktop1 = new JDesktopPane();
        desktop1.setLocation(0, 0);
        desktop1.setSize(1089, 660);
        contentPane.add(desktop1);
        desktop1.setLayout(null);

        JButton btnCadastro = new JButton("CADASTRO");
        btnCadastro.setToolTipText("Clique para se cadastrar, e ter acesso a comunidade!\r\n");
        btnCadastro.setBounds(588, 310, 181, 52);
        desktop1.add(btnCadastro);

        JButton btnLogin = new JButton("LOGIN");
        btnLogin.setToolTipText("Clique para fazer login na comunidade!\r\n");
        btnLogin.setBounds(779, 310, 181, 52);
        btnLogin.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

```

```

        TelaLogin tl = new TelaLogin();

        desktop1.add(tl);

        tl.setVisible(true);

    }

});

desktop1.add(btnLogin);

JLabel lblMainpage = new JLabel("MainPage");
lblMainpage.setIcon(new
Imagelcon(TelaPrincipal.class.getResource("/MeioAmbiente/resTelas/TelaPrincipal (2).png")));
lblMainpage.setBounds(0, 32, 999, 477);
desktop1.add(lblMainpage);

JMenuBar menuBar = new JMenuBar();
menuBar.setBounds(0, 0, 1009, 34);
desktop1.add(menuBar);

JMenu Menu = new JMenu("Menu");
Menu.setIcon(new
Imagelcon(TelaPrincipal.class.getResource("/MeioAmbiente/resTelas/Home.png")));
menuBar.add(Menu);

JMenuItem HomePage = new JMenuItem("Concientiza\u00E7\u00E3o");
HomePage.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        //JOptionPane.showMessageDialog(null, "Bem Vindo !");

        TelaHome hp = new TelaHome();
        desktop1.add(hp);

        hp.setVisible(true);

    }
}

```

```

});
Menu.add(HomePage);

JMenuItem UmPoucoDe = new JMenuItem("Países e seus problemas");
UmPoucoDe.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        //JOptionPane.showMessageDialog(null, "Bem Vindo !");

        UmPoucoDeCadaPais pais = new UmPoucoDeCadaPais();
        desktop1.add(pais);

        pais.setVisible(true);

    }
});
Menu.add(UmPoucoDe);

JMenuItem Sair = new JMenuItem("Sair");
Sair.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        new TelaOpcoes().setVisible(true);

    }
});
Menu.add(Sair);

JMenu Informacoes = new JMenu("Informa\u00E7\u00F5es");
Informacoes.setIcon(new
ImageIcon(TelaPrincipal.class.getResource("/MeioAmbiente/resTelas/Sobre.jpg")));
menuBar.add(Informacoes);

JMenuItem Sobre = new JMenuItem("Sobre");
Sobre.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        new TelaSobre().setVisible(true);

    }
}

```

```

    });
    Informacoes.add(Sobre);

    JMenuItem InformacoesDoPrograma = new JMenuItem("Informa\u00E7\u00F5es
do Programa");
    InformacoesDoPrograma.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {

            new TelaInformacoes().setVisible(true);

        }
    });
    Informacoes.add(InformacoesDoPrograma);

    setVisible(true);

    btnCadastro.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {

            JOptionPane.showMessageDialog(null, "Bem Vindo !");
            TelaCadastro tc = new TelaCadastro();
            desktop1.add(tc);

            tc.setVisible(true);

        }
    });

}
}

```

- **A respeito do meio ambiente**

```

public class TelaMeioAmbiente extends JInternalFrame {
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private JTextField txtFotoSuppryreproduo;
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    TelaMeioAmbiente frame = new TelaMeioAmbiente();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     * @throws PropertyVetoException
     */
    public TelaMeioAmbiente(){

        setTitle("Melhorar Nosso Meio Ambiente\r\n");
        setClosable(true);
        setSize(993, 516);
        getContentPane().setLayout(null);


        JDesktopPane desktop5 = new JDesktopPane();
        desktop5.setBounds(0, 0, 979, 474);
        getContentPane().add(desktop5);
        desktop5.setBackground(new Color(0, 206, 209));


        JMenuBar menuBar = new JMenuBar();
        menuBar.setBounds(0, 0, 979, 21);
    }
}

```



```
desktop5.add(menuBar);
```

```
JMenu Menu = new JMenu("Menu");
```

```
menuBar.add(Menu);
```

```
JMenuItem Sair = new JMenuItem("Sair");
```

```
Sair.addActionListener(new ActionListener() {
```

```
    public void actionPerformed(ActionEvent e) {
```

```
        new TelaOpcoes().setVisible(true);
```

```
    }
```

```
});
```

```
Menu.add(Sair);
```

```
JLabel lblDicasSobreComo = new JLabel("  Dicas Sobre Como Podemos Melhorar  
Nosso Meio Ambiente");
```

```
lblDicasSobreComo.setForeground(new Color(255, 255, 255));
```

```
lblDicasSobreComo.setFont(new Font("Monospaced", Font.BOLD, 25));
```

```
lblDicasSobreComo.setBounds(42, 40, 899, 55);
```

```
desktop5.add(lblDicasSobreComo);
```

```
JButton btnDesmatamento = new JButton("Desmatamento");
```

```
btnDesmatamento.addActionListener(new ActionListener() {
```

```
    public void actionPerformed(ActionEvent e) {
```

```
        new TelaDesmatamento().setVisible(true);
```

```
    }
```

```
});
```

```
btnDesmatamento.setForeground(new Color(0, 128, 0));
```

```
btnDesmatamento.setFont(new Font("Arial Black", Font.PLAIN, 12));
```

```
btnDesmatamento.setBounds(10, 106, 200, 55);
```

```
desktop5.add(btnDesmatamento);
```

```
JPopupMenu popupMenu = new JPopupMenu();
```

```
addPopup(btnDesmatamento, popupMenu);
```

```
JLabel lblDesmatamento = new JLabel("Desmatamento");
```

```
popupMenu.add(lblDesmatamento);
```

```
 JButton btnSustentabilidade = new JButton("Sustentabilidade");  
btnSustentabilidade.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
  
        new TelaSustentabilidade().setVisible(true);  
  
    }  
});  
btnSustentabilidade.setForeground(new Color(0, 128, 0));  
btnSustentabilidade.setFont(new Font("Arial Black", Font.PLAIN, 12));  
btnSustentabilidade.setBounds(44, 394, 200, 55);  
desktop5.add(btnSustentabilidade);
```

```
JPopupMenu popup2 = new JPopupMenu();  
addPopup(btnSustentabilidade, popup2);
```

```
JLabel lblGarrafasdevidro = new JLabel("Sustentabilidade");  
popup2.add(lblGarrafasdevidro);
```

```
 JButton btnAlimentos = new JButton("Alimentos");  
btnAlimentos.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
  
        new TelaAlimentos().setVisible(true);  
  
    }  
});  
btnAlimentos.setForeground(new Color(0, 128, 0));  
btnAlimentos.setFont(new Font("Arial Black", Font.PLAIN, 12));  
desktop5.add(btnAlimentos);
```

```
btnAlimentos.setBounds(364, 106, 200, 55);
```

```
JPopupMenu popup3 = new JPopupMenu();  
addPopup(btnAlimentos, popup3);
```

```
JLabel lblAlimentos = new JLabel("Alimentos");  
popup3.add(lblAlimentos);
```

```
JButton btnEnergia = new JButton("Energia");  
btnEnergia.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
  
        new TelaEnergia().setVisible(true);  
  
    }  
});  
btnEnergia.setForeground(new Color(0, 128, 0));  
btnEnergia.setFont(new Font("Arial Black", Font.PLAIN, 11));  
desktop5.add(btnEnergia);  
btnEnergia.setBounds(403, 394, 200, 55);
```

```
JPopupMenu popup4 = new JPopupMenu();  
addPopup(btnEnergia, popup4);
```

```
JLabel lblEnergia = new JLabel("Energia");  
popup4.add(lblEnergia);
```

```
JButton btnCarregadores = new JButton("Agua");  
btnCarregadores.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
  
        new TelaAgua().setVisible(true);
```

```

        }
    });
    btnCarregadores.setForeground(new Color(0, 128, 0));
    btnCarregadores.setFont(new Font("Arial Black", Font.PLAIN, 12));
    desktop5.add(btnCarregadores);
    btnCarregadores.setBounds(625, 304, 200, 55);

    JPopupMenu popup5 = new JPopupMenu();
    addPopup(btnCarregadores, popup5);

    JLabel lblCarregadores = new JLabel("Agua");
    popup5.add(lblCarregadores);

    JButton btnRoupasEBrinquedosEOutros = new JButton("Consumo Verde");
    btnRoupasEBrinquedosEOutros.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {

            new TelaConsumo().setVisible(true);

        }
    });
    btnRoupasEBrinquedosEOutros.setForeground(new Color(0, 128, 0));
    desktop5.add(btnRoupasEBrinquedosEOutros);
    btnRoupasEBrinquedosEOutros.setFont(new Font("Arial Black", Font.PLAIN, 12));
    btnRoupasEBrinquedosEOutros.setBounds(137, 306, 290, 50);

    JPopupMenu popup6 = new JPopupMenu();
    addPopup(btnRoupasEBrinquedosEOutros, popup6);

    JLabel lblProdutosEOutros = new JLabel("Produtos");
    popup6.add(lblProdutosEOutros);

    JButton btnTransportes = new JButton("Transportes");
    btnTransportes.addActionListener(new ActionListener() {

```

```

        public void actionPerformed(ActionEvent e) {

            new TelaTransportes().setVisible(true);

        }
    });
    btnTransportes.setForeground(new Color(0, 128, 0));
    desktop5.add(btnTransportes);
    btnTransportes.setFont(new Font("Arial Black", Font.PLAIN, 12));
    btnTransportes.setBounds(565, 206, 200, 55);

    JPopupMenu popup7 = new JPopupMenu();
    addPopup(btnTransportes, popup7);

    JLabel lblLiquidos = new JLabel("Transporte");
    popup7.add(lblLiquidos);

    JButton btnEletronicos = new JButton("Lixo Eletronico");
    btnEletronicos.setBackground(new Color(255, 255, 255));
    btnEletronicos.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {

            new TelaLixoEletronico().setVisible(true);

        }
    });
    btnEletronicos.setForeground(new Color(0, 128, 0));
    btnEletronicos.setFont(new Font("Arial Black", Font.PLAIN, 12));
    btnEletronicos.setBounds(181, 206, 200, 55);
    desktop5.add(btnEletronicos);

    JPopupMenu popup8 = new JPopupMenu();
    addPopup(btnEletronicos, popup8);

    JLabel lblAgua = new JLabel("Eletronicos");

```

```

popup8.add(lblAgua);

txtFotoSuppryreproduo = new JTextField();
txtFotoSuppryreproduo.setEnabled(false);
txtFotoSuppryreproduo.setEditable(false);
txtFotoSuppryreproduo.setHorizontalAlignment(SwingConstants.CENTER);
txtFotoSuppryreproduo.setText("Foto: Suppry/Reprodu\u00E7\u00E3o");
txtFotoSuppryreproduo.setBounds(804, 442, 152, 32);
desktop5.add(txtFotoSuppryreproduo);
txtFotoSuppryreproduo.setColumns(10);

JLabel lblNewLabel = new JLabel("New label");
lblNewLabel.setIcon(new
Imagelcon(TelaMeioAmbiente.class.getResource("/MeioAmbiente/resTelas/ambiente-2.jpeg")));
lblNewLabel.setBounds(-95, 11, 1074, 487);
desktop5.add(lblNewLabel);

}

private static void addPopup(Component component, final JPopupMenu popup) {
    component.addMouseListener(new MouseAdapter() {
        public void mousePressed(MouseEvent e) {
            if (e.isPopupTrigger()) {
                showMenu(e);
            }
        }
        public void mouseReleased(MouseEvent e) {
            if (e.isPopupTrigger()) {
                showMenu(e);
            }
        }
        private void showMenu(MouseEvent e) {
            popup.show(e.getComponent(), e.getX(), e.getY());
        }
    });
}
}

```

- **Sobre cidades mais poluídas e seu países.**

```

public class UmPoucoDeCadaPais extends JFrame {
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    //private JPanel contentPane;
    //private JTextField txtParaVisualizarA;
    private JTextField textFieldVisualizar;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    UmPoucoDeCadaPais login = new UmPoucoDeCadaPais();
                    login.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
    public UmPoucoDeCadaPais() {

        setClosable(true);
        setTitle("Cidades mais poluidas");
        setSize(1006, 639);
        setResizable(false);
        getContentPane().setLayout(null);
    }
}

```

```

JDesktopPane desktop6 = new JDesktopPane();
desktop6.setBackground(Color.GRAY);
desktop6.setBounds(0, 0, 990, 610);
getContentPane().add(desktop6);
desktop6.setLayout(null);

JButton btnAfeganistao = new JButton("Afeganistao");
btnAfeganistao.setBackground(new Color(153, 153, 153));
btnAfeganistao.setToolTipText("Clique para obter informa\u00E7\u00F5es sobre a
situa\u00E7\u00E3o do pais e alguns de seus estados.\r\n\r\n");
btnAfeganistao.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        new Pais_Afeganistao().setVisible(true);

    }
});

btnAfeganistao.setIcon(new
ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/Afeganistao.png")));
btnAfeganistao.setBounds(41, 304, 215, 105);
desktop6.add(btnAfeganistao);

JPopupMenu popup1 = new JPopupMenu();
addPopup(btnAfeganistao, popup1);

JLabel lblAfeganistao = new JLabel("");
lblAfeganistao.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {

        try {

            URI link = new URI("https://binged.it/2r7ZzG0");
            Desktop.getDesktop().browse(link);

        }catch(Exception erro){
            System.out.println("Deu ruim isis!");

```



```

    }
}

});

lblAfeganistao.setIcon(new
ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/Afghanistan.globo.png")));

popup1.add(lblAfeganistao);

```

```

 JButton btnArabia = new JButton("Arabia");
btnArabia.setBackground(new Color(153, 153, 153));
btnArabia.setToolTipText("Clique para obter informa\u00E7\u00F5es sobre a
situa\u00E7\u00E3o do pais e alguns de seus estados.");
btnArabia.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {

        new Pais_Arabia().setVisible(true);

    }
});

btnArabia.setIcon(new
ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/Arabia.png")));
btnArabia.setBounds(656, 45, 215, 105);
desktop6.add(btnArabia);

```

```

 JPopupMenu popup2 = new JPopupMenu();
addPopup(btnArabia, popup2);

```

```

 JLabel lblArabia = new JLabel("");
lblArabia.addMouseListener(new MouseAdapter() {
     @Override
     public void mouseClicked(MouseEvent e) {

```

```

    try {

```

```

         URI link = new URI("https://binged.it/2r7SSnb");
         Desktop.getDesktop().browse(link);

```

```

    }catch(Exception erro){

```

```

        System.out.println("Deu ruim zé, AlakAgbar!");

    }

}

});

lblArabia.setIcon(new
Imagelcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/Saudi_Arabia.globo.
png")));

popup2.add(lblArabia);


JButton btnBahrein = new JButton("Bahrein");
btnBahrein.setBackground(new Color(153, 153, 153));
btnBahrein.setToolTipText("Clique para obter informa\u00E7\u00F5es sobre a
situa\u00E7\u00E3o do país e alguns de seus estados.");
btnBahrein.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        new Pais_Bahrein().setVisible(true);

    }

});

btnBahrein.setIcon(new
Imagelcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/Behien.png")));
btnBahrein.setBounds(354, 45, 215, 105);
desktop6.add(btnBahrein);


JPopupMenu popup3 = new JPopupMenu();
addPopup(btnBahrein, popup3);


JLabel lblBahrein = new JLabel("");
lblBahrein.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {

        try {

            URI link = new URI("https://binged.it/2r7PVDc");

```

```

        Desktop.getDesktop().browse(link);

    }catch(Exception erro){
        System.out.println("Deu ruim, com o país com nome de
cerveja");

    }

}

});

lblBahrein.setIcon(new
ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/Bahrain.globo.png"))
);

popup3.add(lblBahrein);

```

```

JButton btnBrasil = new JButton("Brasil");
btnBrasil.setBackground(new Color(153, 153, 153));
btnBrasil.setToolTipText("Clique para obter informa\u00E7\u00F5es sobre a
situa\u00E7\u00E3o do país e alguns de seus estados.");
btnBrasil.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        new Pais_Brasil().setVisible(true);

    }
});

btnBrasil.setIcon(new
ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/Brazil.png")));
btnBrasil.setBounds(656, 304, 215, 105);
desktop6.add(btnBrasil);

JPopupMenu popup4 = new JPopupMenu();
addPopup(btnBrasil, popup4);

JLabel lblBrasil = new JLabel("");
lblBrasil.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {

```

```

        try {

            URI link = new URI("https://binged.it/2r7L598");
            Desktop.getDesktop().browse(link);

        }catch(Exception erro){
            System.out.println("ta ligado, algo de certo esta errado!");
        }

    }

});

lblBrasil.setIcon(new
ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/Brasil.globo.png")));
popup4.add(lblBrasil);

JButton btnChina = new JButton("China");
btnChina.setBackground(new Color(153, 153, 153));
btnChina.setToolTipText("Clique para obter informa\u00E7\u00F5es sobre a
situa\u00E7\u00E3o do pais e alguns de seus estados.");
btnChina.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        new Pais_China().setVisible(true);

    }

});

btnChina.setIcon(new
ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/China.png")));
btnChina.setBounds(656, 174, 215, 105);
desktop6.add(btnChina);

JPopupMenu popup5 = new JPopupMenu();
addPopup(btnChina, popup5);

JLabel lblChina = new JLabel("");
lblChina.addMouseListener(new MouseAdapter() {

```

```

        @Override
        public void mouseClicked(MouseEvent e) {

            try {

                URI link = new URI("https://binged.it/2r7NEYu");
                Desktop.getDesktop().browse(link);

                }catch(Exception erro){
                    System.out.println("Deu ruim zé-chan!");
                }

            }
        });

        lblChina.setIcon(new
        ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/China.globo.png")));
        popup5.add(lblChina);

        JButton btnIndia = new JButton("India");
        btnIndia.setForeground(new Color(0, 0, 0));
        btnIndia.setBackground(new Color(153, 153, 153));
        btnIndia.setToolTipText("Clique para obter informa\u00E7\u00F5es sobre a
        situa\u00E7\u00E3o do pais e alguns de seus estados.");
        btnIndia.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

                new Pais_India().setVisible(true);

            }
        });

        btnIndia.setIcon(new
        ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/India.png")));
        btnIndia.setBounds(41, 45, 215, 105);
        desktop6.add(btnIndia);

        JPopupMenu popup6 = new JPopupMenu();
        addPopup(btnIndia, popup6);

```

```

JLabel lblIndia = new JLabel("A parte em verde claro representa o territ\u00F3rio
disputado da Caxemira (que inclui reivindic\u00E7\u00F5es de Paquist\u00E3o e China)");
lblIndia.addMouseListener(new MouseAdapter() {

    @Override

    public void mouseClicked(MouseEvent e) {

        try {

            URI link = new URI("https://binged.it/2r7ALxO");
            Desktop.getDesktop().browse(link);

        }catch(Exception erro){
            System.out.println("Deu ruim, Esquece a India!");
        }

    }

});
lblIndia.setIcon(new
ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/India.globo.png")));
popup6.add(lblIndia);

JButton btnIran = new JButton("Iran");
btnIran.setBackground(new Color(153, 153, 153));
btnIran.setToolTipText("Clique para obter informa\u00E7\u00F5es sobre a
situa\u00E7\u00E3o do pa\u00eds e alguns de seus estados.\r\n");
btnIran.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        new Pais_Iran().setVisible(true);

    }

});
btnIran.setIcon(new
ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/Iran.png")));
btnIran.setBounds(41, 174, 215, 105);
desktop6.add(btnIran);

```

```

JPopupMenu popup7 = new JPopupMenu();
addPopup(btnIran, popup7);

JLabel lblIran = new JLabel("");
lblIran.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {

        try {

            URI link = new URI("https://binged.it/2q8V2Wi");
            Desktop.getDesktop().browse(link);

        }catch(Exception erro){
            System.out.println("Deu ruim zézinho sheik, !");
        }

        }
    });
    lblIran.setIcon(new
    ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/Iran.globo.png")));
    popup7.add(lblIran);

JButton btnNigeria = new JButton("Nigeria");
btnNigeria.setBackground(new Color(153, 153, 153));
btnNigeria.setToolTipText("Clique para obter informa\u00E7\u00F5es sobre a
situa\u00E7\u00E3o do pais e alguns de seus estados.");
btnNigeria.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        new Pais_Nigeria().setVisible(true);

    }
    });
    btnNigeria.setIcon(new
    ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/Nigeria.png")));

```

```

btnNigeria.setBounds(354, 304, 215, 105);
desktop6.add(btnNigeria);

JPopupMenu popup8 = new JPopupMenu();
addPopup(btnNigeria, popup8);

JLabel lblNigeria = new JLabel("");
lblNigeria.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {

        try {

            URI link = new URI("https://binged.it/2q9fxCI");
            Desktop.getDesktop().browse(link);

            catch(Exception erro){
                System.out.println("Deu ruim... hum...!");
            }

        }
    });
    lblNigeria.setIcon(new
    ImageIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/Nigeria.globo.png"))))
    ;

    popup8.add(lblNigeria);

JButton btnPaquistao = new JButton("Paquistão");
btnPaquistao.setBackground(new Color(153, 153, 153));
btnPaquistao.setToolTipText("Clique para obter informa\u00E7\u00F5es sobre a
situa\u00E7\u00E3o do pais e alguns de seus estados.");
btnPaquistao.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        new Pais_Paquistao().setVisible(true);

    }

```



```

});
    btnPaquistao.setIcon(new
ImagemIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/Paquistao.png")));
    btnPaquistao.setBounds(354, 174, 215, 105);
    desktop6.add(btnPaquistao);

    JPopupMenu popup9 = new JPopupMenu();
    addPopup(btnPaquistao, popup9);

    JLabel lblPaquistao = new JLabel("");
    lblPaquistao.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {

            try {

                URI link = new URI("https://binged.it/2q9eNgp");
                Desktop.getDesktop().browse(link);

            }catch(Exception erro){
                System.out.println("Deu ruim zé das areias!");
            }

        }

    });
    lblPaquistao.setIcon(new
ImagemIcon(UmPoucoDeCadaPais.class.getResource("/MeioAmbiente/resTelas/Paquistao.globo.png
")));
    popup9.add(lblPaquistao);

    JMenuBar menuBar = new JMenuBar();
    menuBar.setBounds(0, 0, 987, 21);
    desktop6.add(menuBar);

    JMenu MenuItem = new JMenu("Menu");
    menuBar.add(MenuItem);

    JMenuItem Sair = new JMenuItem("Sair");
    Sair.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {

```

```

        new TelaOpcoes().setVisible(true);

    }

});

MenuItem.add(Sair);

        textFieldVisualizar = new JTextField("        Para visualizar a localidade de cada país
no globo, clique com o botão direito sobre o bandeira.");
        textFieldVisualizar.setBounds(90, 523, 836, 37);
        desktop6.add(textFieldVisualizar);
        textFieldVisualizar.setFont(new Font("Monospaced", Font.PLAIN, 12));
        textFieldVisualizar.setColumns(10);

        JLabel lblNewLabel = new JLabel("");
        lblNewLabel.setIcon(new
        ImageIcon(UmPoucoDeCadaPaís.class.getResource("/MeioAmbiente/resTelas/foglia-con-
rugiada.jpg")));
        lblNewLabel.setBackground(new Color(204, 204, 204));
        lblNewLabel.setVerticalAlignment(SwingConstants.TOP);
        lblNewLabel.setBounds(0, 21, 987, 587);
        desktop6.add(lblNewLabel);

    }

    private static void addPopup(Component component, final JPopupMenu popup) {
        component.addMouseListener(new MouseAdapter() {
            public void mousePressed(MouseEvent e) {
                if (e.isPopupTrigger()) {
                    showMenu(e);
                }
            }
            public void mouseReleased(MouseEvent e) {
                if (e.isPopupTrigger()) {
                    showMenu(e);
                }
            }
        });
        private void showMenu(MouseEvent e) {

```

```
        popup.show(e.getComponent(), e.getX(), e.getY());
    }
    });
}
```

5.9 Apresentação do programa funcionando em um computador.

O programa a seguir foi desenvolvido no Eclipse que é uma IDE para desenvolvimento de aplicações tanto em JAVA, quanto PHP, C/C++, PYTHON e etc.

Foi usado também, e principalmente um plug-in do mesmo chamado *Windowbuilder*, ele serve para auxiliar e agilizar o desenvolvimento com ambientes gráficos, e faz parte das ferramentas de interface gráfica no Eclipse.

O foco, foi abordar o meio ambiente e a poluição, com paginas diretas, cheias de conteúdos e significância, abordamos esse assunto tão essencial e importante hoje em dia, caminhando de simples cuidados e ações diárias que nos mesmo podemos desempenhar, a graves problemas ambientais, que governos e cidades precisariam trabalhar arduamente por anos para conseguirem resolver.

- **Tela inicial .**

Para começar, temos a tela inicial, ainda sem muitos botoes ou informações, para mantermos um primeiro contato amigavel da aplicação com

o usuário, aqui temos dois botões que são a forma de se acessar as duas principais partes do programa.

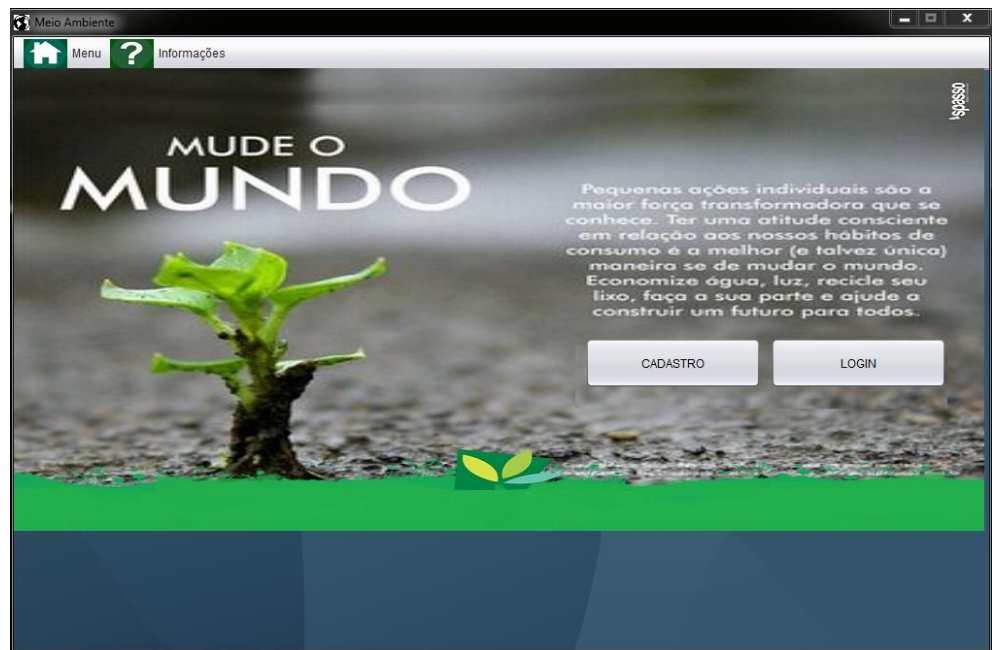


Figura 10: Tela inicial.

Fonte própria maio de 2017

- **Tela inicial – Menu.**

Neste primeiro menu em cascata, podemos observar que o programa oferece ao usuário dois caminhos um é área de conscientização, onde

abordaremos o meio ambiente, e o outro, a área de países e seus problemas, onde abordaremos a poluição.

I



Figura 11: Tela inicial – menu.

Fonte própria maio de 2017

- **Tela inicial - Informações.**

No segundo menu de cascata, temos duas opções simples, diferentes do primeiro, o foco deste e o prprograma em si, com a area 'sobre' e a 'informações do programa'.

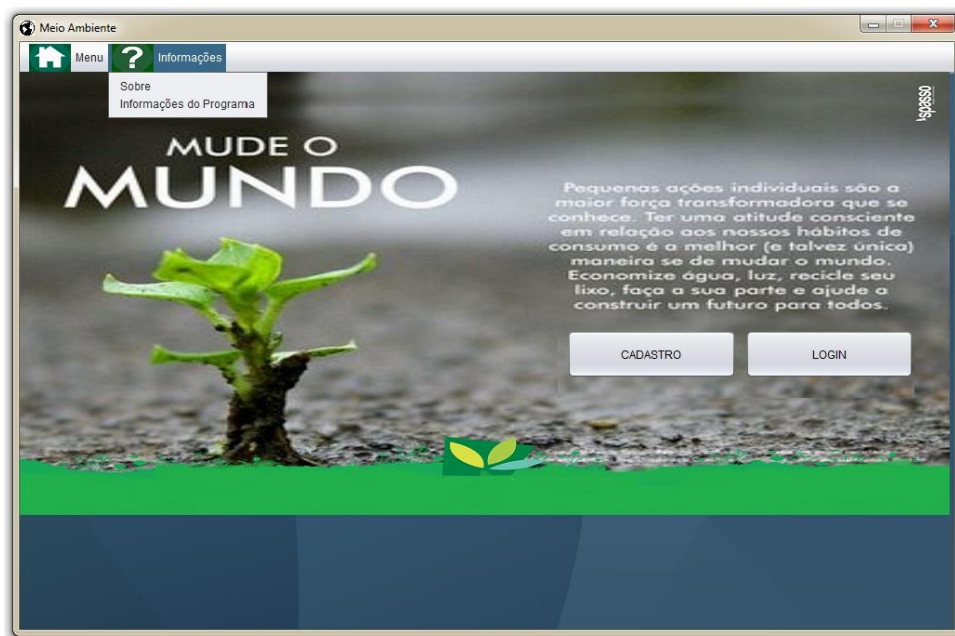


Figura 12: Tela inicial – Informações.

Fonte própria maio de 2017

- **Tela inicial - Informações – Sobre.**

Na primeira janela que apresentamos, estão os dados dos alunos que trabalharam nesta APS, a equipe de desenvolvimento, e a equipe de pesquisas.

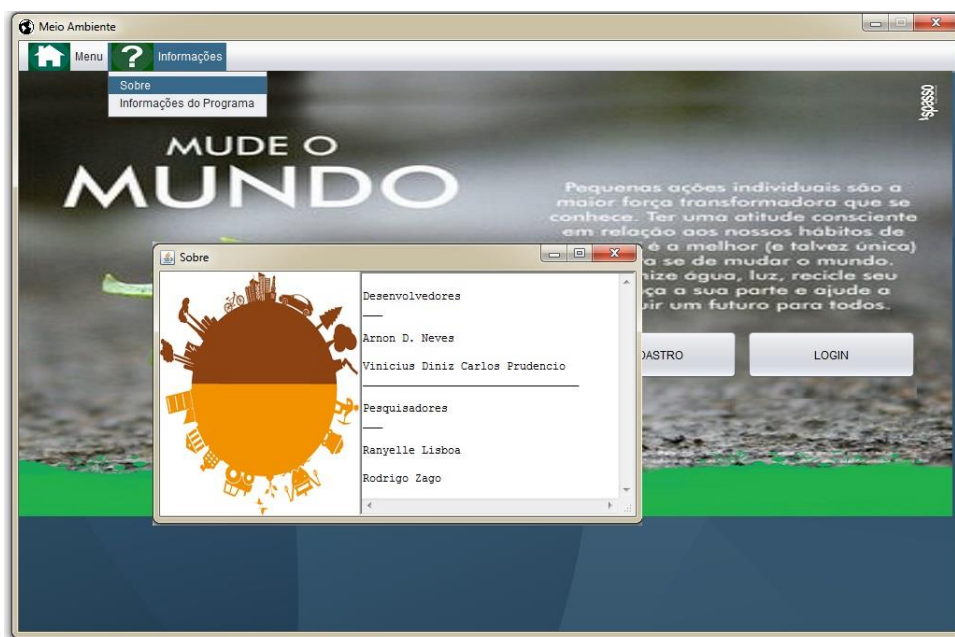


Figura 13: Informações – Sobre.

Fonte própria maio de 2017

- **Tela inicial - Informações – informações sobre o programa.**

Na segunda janela do menu informações, temos um informativo sobre o programa, com descrição detalhada de cada janela e um resumo compacto do conteúdo que pode ser encontrado nela.

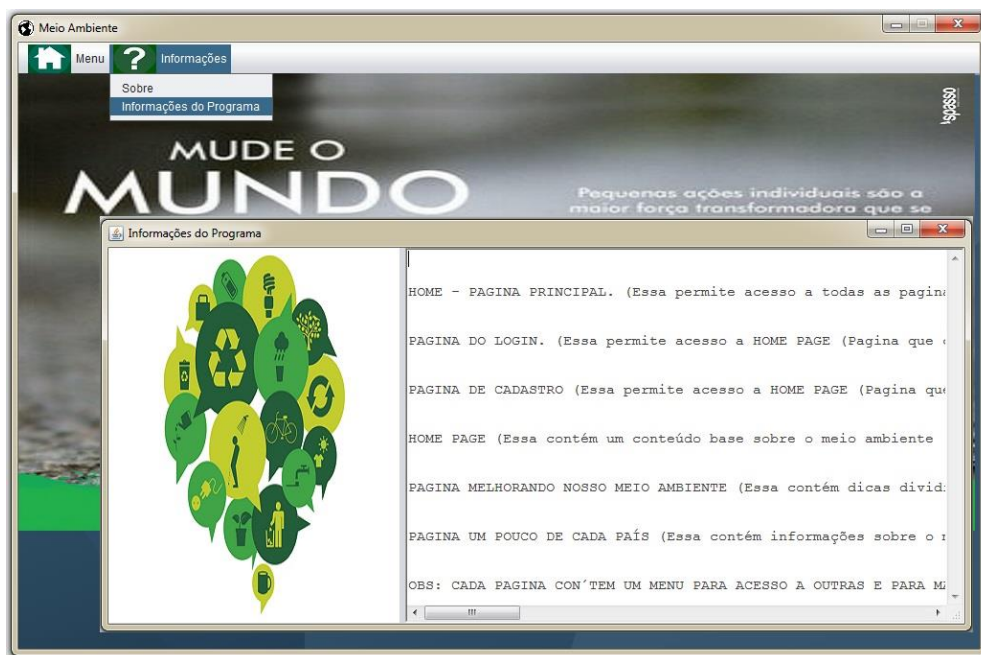


Figura 14: Tala inicial – Informações sobre o programa.

Fonte própria maio de 2017

- **Meio ambiente.**

A primeira das duas partes principais do programa, aqui, somos apresentados a um ambiente cuidadosamente planejado, com um texto de introdução sobre o meio ambiente e uma foto de fundo que remete a união.

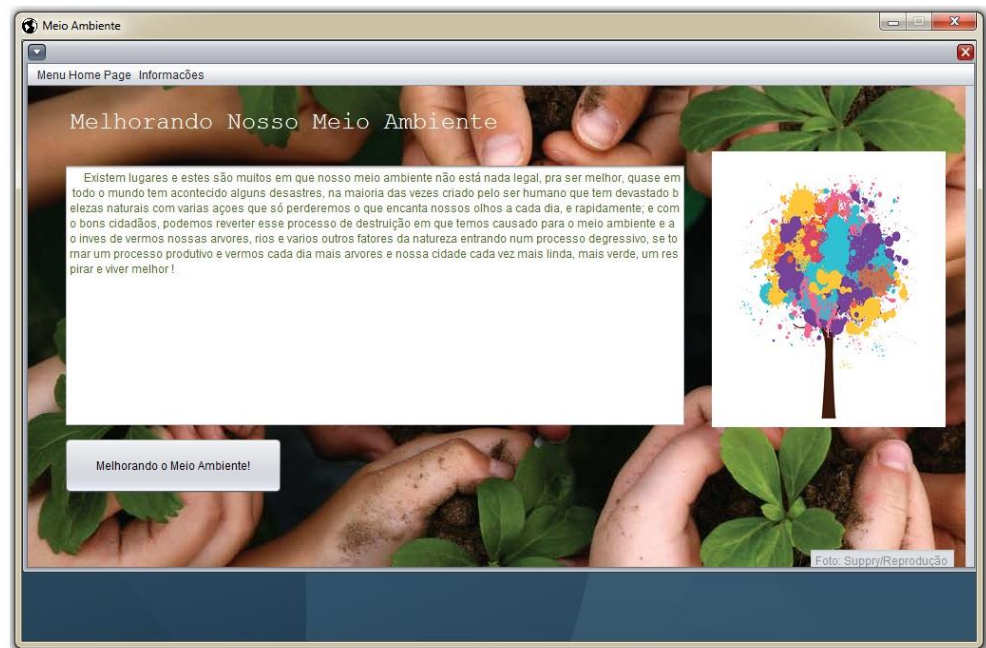


Figura 15: Meio ambiente.

Fonte própria maio de 2017

- **Meio ambiente – Melhorando nosso meio ambiente.**

Na próxima janela, que se abre ao clicarmos em ‘melhorando o meio ambiente’ o contexto de união ganha um reforço com a imagem podendo ser observada quase livremente, aqui cada botão nos leva para um assunto específico que gira em torno da preservação, cuidado, atividades sustentáveis e colaborativas para um meio ambiente saudável.

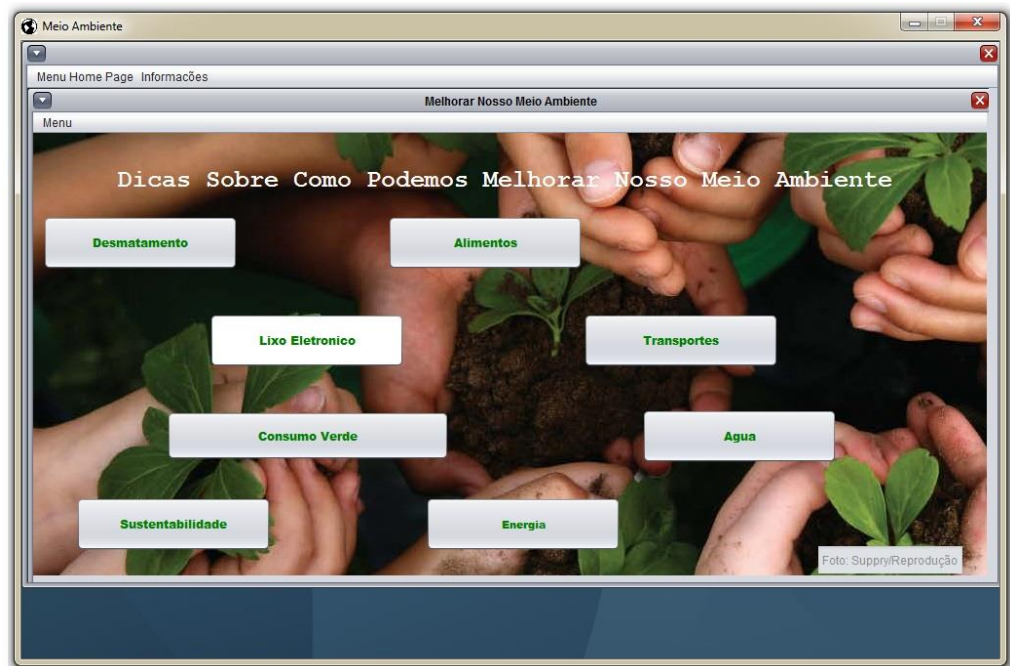


Figura 16: Meio ambiente – Melhorando nosso meio ambiente 1.

Fonte própria maio de 2017

- **Meio ambiente – Melhorando nosso meio ambiente – desmatamento / sustentabilidade.**

Ao clicar nos botoes, janelas com caixas de texto menores apareceram, nelas poderam ser encontrados textos e dicas com informações importantes sobre desmatamento, materias e dados educativos, e a possibilidade de uma interação com o usuario por meio de links que podem ser abertos no navegador do mesmo com um simples clique.

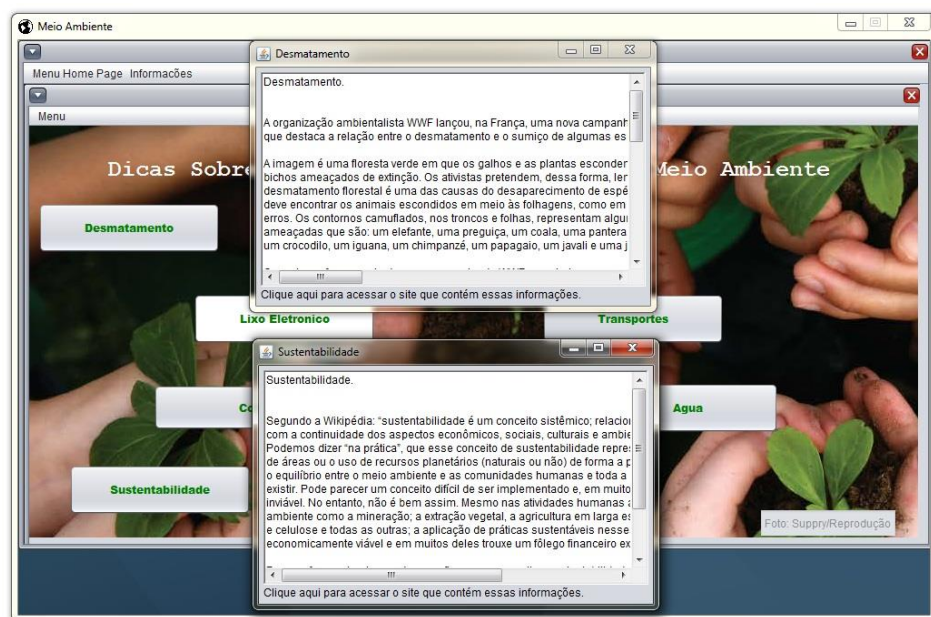


Figura 17: Meio ambiente- Melhorando nosso meio ambiente 2.

Fonte própria maio de 2017

- **Cidades mais poluídas.**

Aqui temos a segunda parte mais importante, onde abordaremos os problemas de poluição, com base na última lista da OMS sobre as cidades mais poluídas do mundo, resolvemos pegar as vinte piores e separá-las por países, isso deixou evidente a relação estreita que a degradação do meio ambiente tem com o nível de desenvolvimento e fortuna do país.



Figura 18: Cidade mais poluídas.

Fonte própria maio de 2017

- **Cidades mais poluídas – Mapa interativo.**

Clicando com o botão direito na bandeira do país desejada, temos a primeira interação com o usuário desta janela, com um mapa interativo que, além de mostrar o local do país no nosso globo, ainda pode abrir um mapa online que o usuário tenha mais informações sobre o mesmo.



Figura 19- Cidades mais poluídas – Mapa interativo.

Fonte própria maio de 2017

- **Cidades mais poluídas – País – Sobre.**

Com esquadro, temos uma nova janela dividida em abas. Na primeira, as principais informações sobre o país em questão, tal como tamanho, população, PIB, principais problemas e estratégias de soluções.

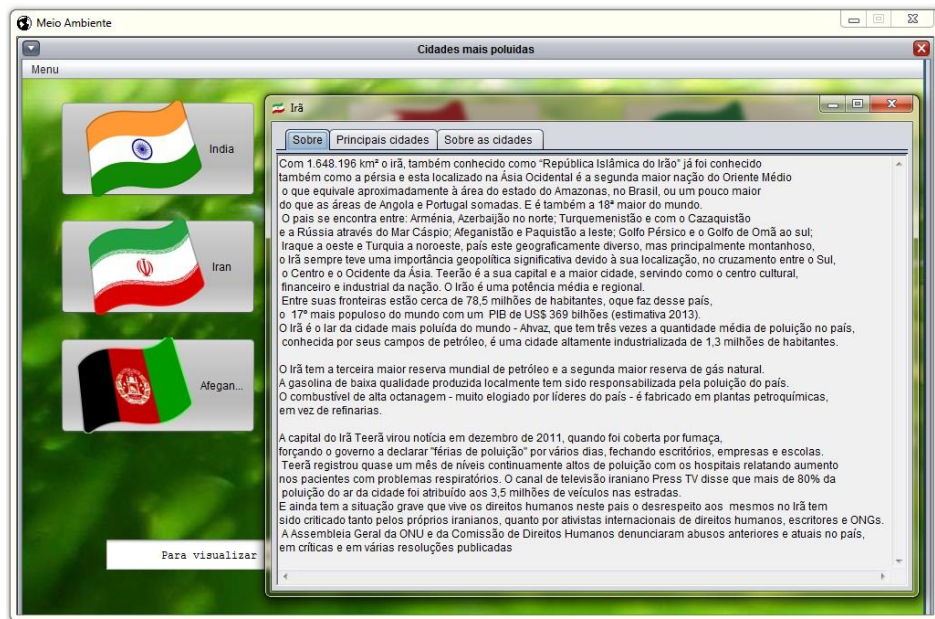
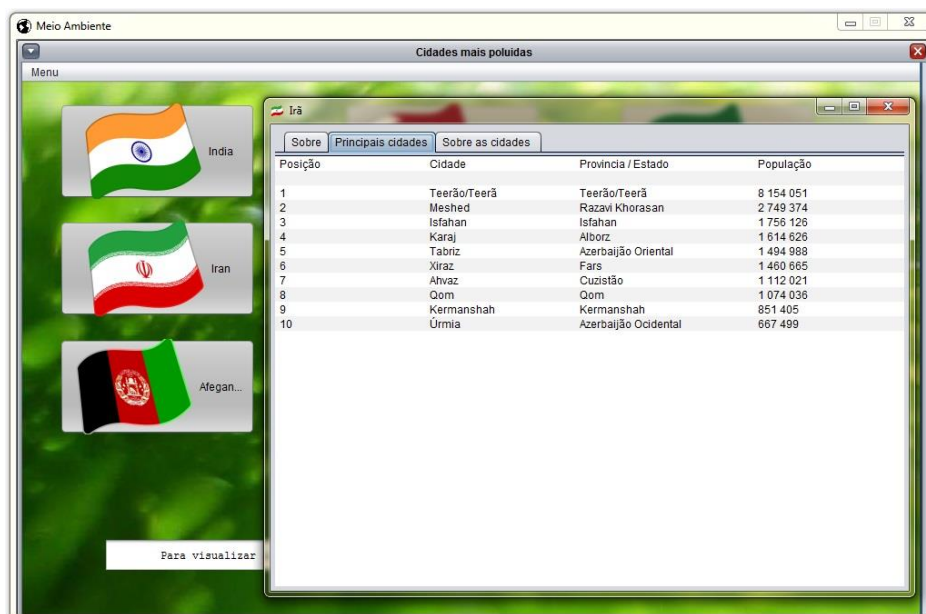


Figura 20: Cidades mais poluídas – país - Sobre.

Fonte própria maio de 2017

- **Cidades mais poluídas – País – Principais cidades.**

Na segunda aba, temos uma pequena tabela destacando as dez principais cidades do país, com suas respectivas províncias ou estados, população, e posição.



Posição	Cidade	Província / Estado	População
1	Teerão/Teerã	Teerão/Teerã	8 154 051
2	Meshed	Razavi Khorasan	2 749 374
3	Isfahan	Isfahan	1 756 126
4	Karaj	Alborz	1 614 826
5	Tabriz	Azerbaijão Oriental	1 494 988
6	Xiraz	Fars	1 460 665
7	Ahvaz	Cuzistão	1 112 021
8	Qom	Qom	1 074 036
9	Kermanshah	Kermanshah	851 405
10	Úrmia	Azerbaijão Ocidental	667 499

Figura 21: Cidades mais poluídas – País - Principais.

Fonte própria maio de 2017

- **Cidades mais poluídas – País – Sobre as cidades.**

Na terceira e ultima aba desta janela, estão destacadas as cidades mais poluídas do país, que estão na lista da OMS como uma das vinte mais poluídas do mundo.

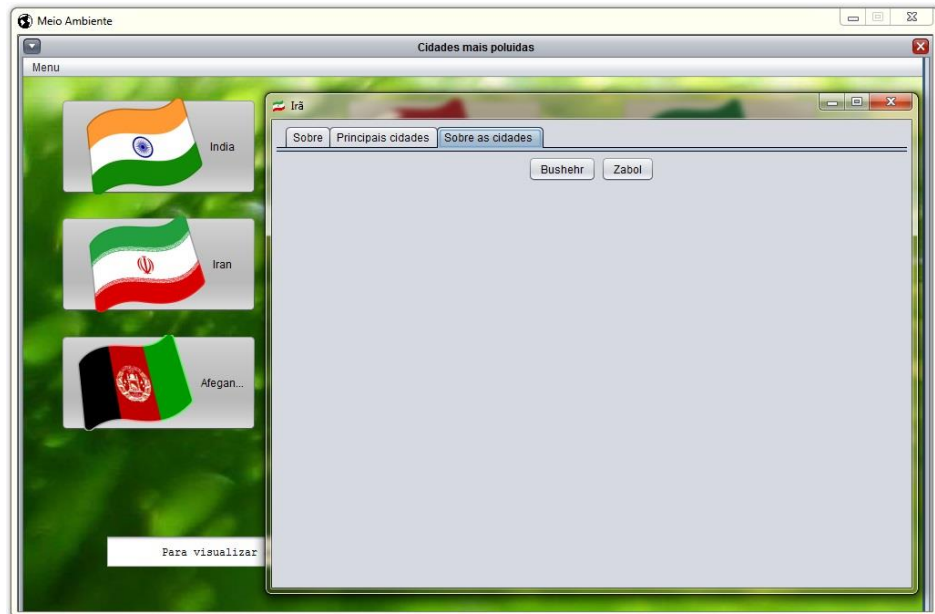


Figura 22: Cidades mais poluídas – País - Sobre.

Fonte própria maio de 2017

- **Cidades mais poluídas – País – Sobre as cidades – Cidade – Interatividade.**

Ao escolher uma cidade, o usuário terá acesso aos dados da mesma, e a um pequeno menu, nele temos uma função parecida com a dos mapas dos países, onde o usuário pode diretamente no seu navegador, por meio de um clique, ter acesso a mapas atuais e informações adicionais.



Figura 23: Cidades mais poluídas – País – sobre – cidade – interatividade.

Fonte própria maio de 2017

- **Cidades mais poluídas – País – Sobre as cidades – Cidade.**

Alem da facilidade da interatividade, o usuario tambem pode abrir varias janelas com as principais informações das cidades de uma so vez, permitimos isso pensando na comodidade do usuario que, nessesitando de muito informação para uma pesquisa ou trabalho, não seja obrigado a escolher, qual janela ler por vez.

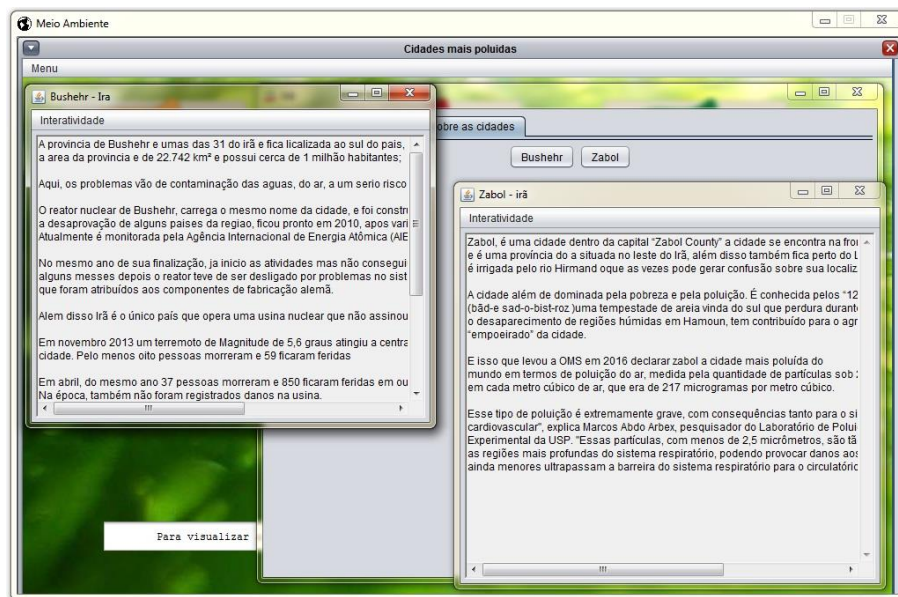



Figura 24: Cidades mais poluídas – País –Sobre – Cidades.

Fonte própria maio de 2017

- **Tela inicial - Cadastro de usuário.**

Voltando para o menu principal, temos a tela de cadastro para novos usuários interessados, logo após o cadastro, os mesmo já podem fazer login.



The screenshot shows a web browser window titled 'Meio Ambiente' with a sub-header 'Cadastro do Usuario'. The page has a dark blue background. On the left, there is a 'Menu' dropdown and a link to 'Informações'. The main form is titled 'Cadastro do Usuario' and is divided into two columns. The left column contains fields for 'Nome Completo', 'Sexo' (with a dropdown menu set to 'Masculino'), 'Data de Nascimento' (with a date picker), 'Rua', 'Cidade', and 'Estado'. The right column contains fields for 'Cria uma senha', 'Digite sua senha novamente', 'Número de celular', and 'E-mail'. A 'Cadastrar' button is located at the bottom right of the form. To the right of the form is a graphic of a tree where the leaves are represented by various green icons related to the environment, such as a recycling symbol, a lightbulb, a house, a leaf, and a water drop.

Figura 25: Cadastro de usuário.

Fonte própria maio de 2017

- **Tela inicial – Login.**

E aqui, temos nossa tela de login, para entrar, o usuário só precisara saber o apelido escolhido na hora do cadastro e a senha.

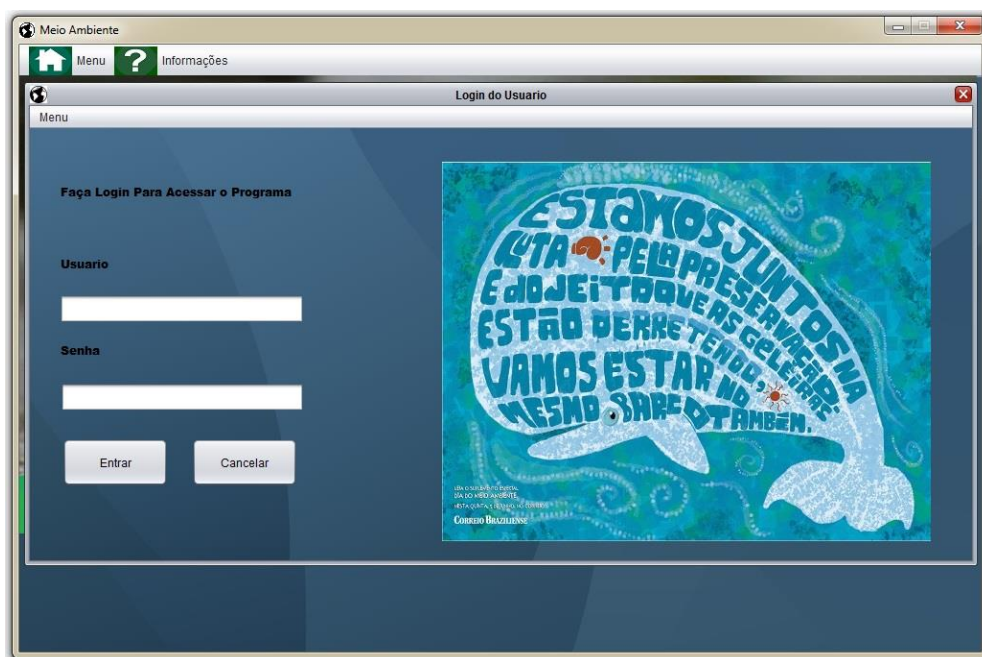


Figura 26: Login.

Fonte própria maio de 2017

6. CONCLUSÃO

Os autores do presente trabalho compreendem que cumpriram com os objetivos da proposta de Atividade Prática Supervisionada (APS) do terceiro semestre de 2017 do curso de Ciências da Computação da Universidade Paulista (UNIP).

Com isso o projeto se encerra com o entendimento de que ele só se tornou possível graças à atenção às exigências contidas nele, perante o nossos esforços, exigindo-se a compreensão da linguagem Java, para a implementação de um programa, cujo mesmo, serve para nos auxiliar e nos alertar sobre os impactos ambientais que diversos países enfrentam hoje em dia.

7. REFERÊNCIAS BIBLIOGRÁFICAS

SANTOS, Rafael. **Introdução à programação Orientada a Objetos usando Java**. 2.ed. Rio de Janeiro: Campus Editora, 2013. p. VI – 98.

UNIP. **Manual de APS**. 2017. p. 2.

BRAZ, Christian. **Introdução à Linguagem Java**. Apostila. p. 4 – 77.

HORSTMANN, Cay. **Conceitos de computação com Java**. 5.ed. Porto Alegre: Bookman Editora, 2009. p. 30 – 59; p. 226 – 479.

Banco de dados : <http://www.macoratti.net/cbmd1.htm>.

8. Fichas de Atividades Praticas Supervisionadas.

8.1 RODRIGO ZAGO DE SA GOUVEIA – N989ED-3

[illegible]

8.2 VINICIUS DINIZ CARLOS PRUDENCIO – N866CE-2

[illegible]

8.3 RANYELLE EVELIN LISBOA BARBOSA – N10274-8

1

FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

Atividades Práticas Supervisionadas (laboratórios, atividades em biblioteca, iniciação Científica, trabalhos individuais e em grupo, práticas de ensino e outras)

NOME: Ranyelle Evelin Lisboa Barbosa

RA: N10274-8

CURSO: Ciência da Computação

CAMPUS: Paraisópolis

SEMESTRE: 02/CC 1A68

TURNOS: manhã

DATA	ATIVIDADE	TOTAL DE HORAS	ASSINATURA	
			ALUNO	PROFESSOR
10/04	Pesquisa Bibliográfica - Coleta de informações	5h	Ranyelle	
11/04	Pesquisa Bibliográfica - Coleta de informações	5h	Ranyelle	
12/04	Pesquisa Bibliográfica - Coleta de informações	5h	Ranyelle	
13/04	Pesquisa Bibliográfica - Coleta de informações	5h	Ranyelle	
14/04	Dissertação	5h	Ranyelle	
11/05	Dissertação	5h	Ranyelle	
12/05	Dissertação	5h	Ranyelle	
13/05	Dissertação	5h	Ranyelle	
14/05	Desenvolvimento de Software	5h	Ranyelle	
20/05	Desenvolvimento de Software	5h	Ranyelle	
21/05	Desenvolvimento de Software	10h	Ranyelle	
22/05	Desenvolvimento de Software	10h	Ranyelle	
24/05	Revisão Geral	2h	Ranyelle	
25/05	Revisão Geral	3h	Ranyelle	
26/05	Revisão Geral	2h	Ranyelle	
27/05	Revisão Geral	3h	Ranyelle	

TOTAL DE HORAS: 80h

8.4 ARNON DAMASCENO NEVES DE SOUZA – N1047H-2

[illegible]