

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ**

**IFCE *CAMPUS* *CANINDÉ***

**SUBSEQUENTE EM INFORMÁTICA**

**THIAGO LEITÃO LOPES**

**USABILIDADE**

**CANINDÉ/CE**

**2022**THIAGO LEITÃO LOPES

USABILIDADE

Trabalho apresentado ao curso Subsequente em Informática do Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE) – *Campus Canindé*, como requisito parcial para obtenção da nota N2 da Disciplina Análise e Projeto de Sistemas II.

Professor: Alex Lacerda Ramos.

Coordenador: Prof. Allyson Bonetti Franca.

CANINDÉ/CE

2022

**LISTA DE FIGURAS**

|  |  |  |  |
| --- | --- | --- | --- |
| Figura 1 | — | Classe de coleção e hierarquia de interface........................................................ | 09 |
| Figura 2 | — | Classe de mapa e hierarquia de interface........................................................... | 10 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**LISTA DE TABELAS**

|  |  |  |  |
| --- | --- | --- | --- |
| Tabela 1 | — | Comparativo entre interfaces............................................................................... | 12 |
| Tabela 2 | — | Implementações de uso geral................................................................................ | 14 |

**LISTA DE SIGLAS**

|  |  |
| --- | --- |
| ABNT | Associação Brasileira de Normas Técnicas |
| APP | Aplicativo Móvel |
| IFCE | Instituto Federal de Educação, Ciência e Tecnologia do Ceará |
| JAVA | Linguagem de programação |
| JDK | Java Development Kit |
| N2 | Nota 2 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**SUMÁRIO**

|  |  |  |
| --- | --- | --- |
| **1** | **INTRODUÇÃO**...................................................................................................... | 07 |
| **2** | **DESENVOLVIMENTO**......................................................................................... | 07 |
| **2.1** | **Visibilidade do status do sistema**............................................................................ | 07 |
| **2.2** | **Correspondência entre o sistema e o mundo real**.................................................. | 08 |
| **2.3** | **Liberdade e controle do usuário**............................................................................. | 10 |
| **2.4** | **Consistência e padrões**............................................................................................ | 12 |
| **2.5** | **Prevenção de erros**.................................................................................................. | 14 |
| ***2.6*** | **Reconhecer ao invés de lembrar**............................................................................. | 16 |
| ***2.7*** | **Flexibilidade e Eficiência**........................................................................................ | 16 |
| ***2.8*** | **Estética e Design minimalista**................................................................................. | 16 |
| ***2.9*** | **Auxiliar usuários a reconhecer, diagnosticar e recuperar erros**.......................... | 16 |
| ***2.10*** | **Ajuda e Documentação**........................................................................................... | 16 |
| **3** | **CONCLUSÃO**...................................................................................................... | 07 |
|  |  |  |
|  | **REFERÊNCIAS**..................................................................................................... | 19 |
|  | **APÊNDICE A — RELAÇÃO DE NORMAS TÉCNICAS VIGENTES UTILIZADAS NA NORMALIZAÇÃO DE TRABALHOS ACADÊMICOS**... | 20 |
|  | **ANEXO A — RESOLUÇÃO QUE APROVA A CRIAÇÃO DO CURSO SUPERIOR DE TECNOLOGIA EM GESTÃO AMBIENTAL NO IFCE CAMPUS PARACURU**......................................................................................... | 21 |

1. **INTRODUÇÃO**

Esse trabalho consiste na realização da escolha de um site, programa ou aplicativo de sua preferência e realizar uma análise de usabilidade com base nas heurísticas de Nielsen apresentadas na aula passada e disponíveis no link <https://medium.com/aela/10-heur%C3%ADsticas-de-nielsen-dicas-para-melhorar-a-usabilidade-de-sua-interface-35ef86a7fb41>.

A ideia é você identificar e explicar quais dessas heurísticas o software escolhido implementa e quais não implementa.

O documento também deverá conter uma capa com o nome do aluno, instituição, curso, disciplina, professor e título do trabalho.

O que são as Heurísticas de Nielsen?

Em 1990, Jakob Nielsen e Rolf Molich propuseram 10 heurísticas que devem ser levadas em consideração no desenvolvimento de qualquer interface.

Nesse contexto, heurística significa uma regra geral — de bom senso — que tem como objetivo reduzir a carga cognitiva do usuário. Assim, permite-se que sua navegação, jornada e experiência sejam mais aprimoradas e menos cansativas.

Dizemos que as heurísticas de Nielsen são regras gerais porque não determinam diretrizes específicas de usabilidade ou do desenvolvimento de interfaces. Nesse sentido, as heurísticas estão mais associadas às observações e conhecimentos adquiridos pelos seus autores durante seus anos de experiência.

1. **DESENVOLVIMENTO**
   1. **Visibilidade do status do sistema:**

A estrutura de coleções Java é um conjunto de classes e interfaces que implementam estruturas de dados de coleção comumente reutilizáveis, ainda que seja referido como uma estrutura, ele funciona como uma biblioteca.

Em resumo é um conjunto bem definido de interfaces e classes para representar e tratar grupos de dados como uma única unidade.

A estrutura de coleções fornece as interfaces que definem várias coleções e classes que as implementam.

A Collections Framework contém os seguintes elementos:

*Interfaces*: tipos abstratos que representam as coleções. Permitem que coleções sejam manipuladas tendo como base o conceito “Programar para interfaces e não para implementações”, desde que o acesso aos objetos se restrinja apenas ao uso de métodos definidos nas interfaces;

*Implementações*: são as implementações concretas das interfaces;

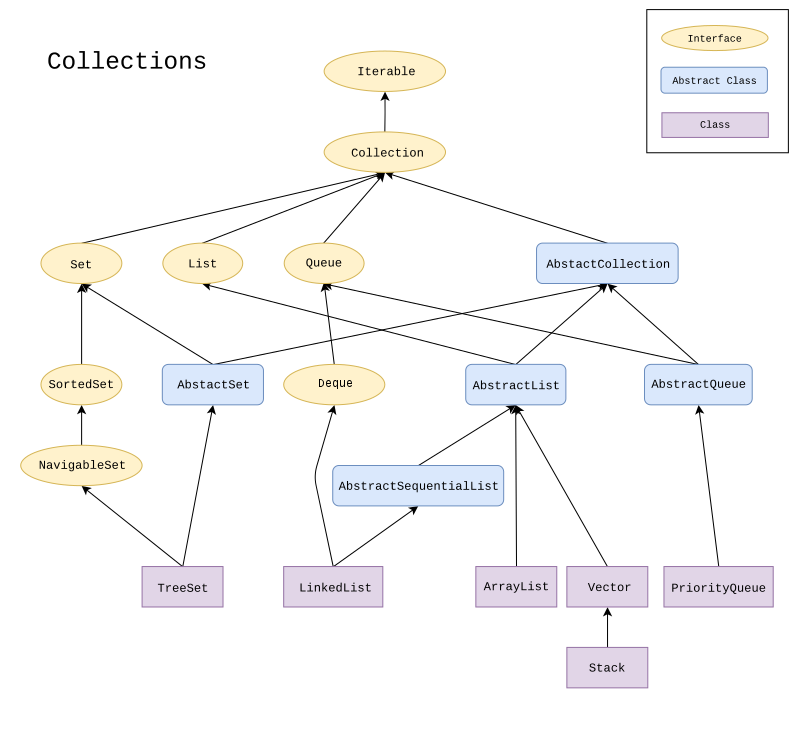
*Algoritmos*: são os métodos que realizam as operações sobre os objetos das coleções, tais como busca e ordenação.

* 1. **Correspondência entre o sistema e o mundo real:**

Desde as primeiras versões, o Java dispõe das estruturas de arrays e as classes Vector e Hashtable. No entanto, além da dificuldade em implementar estruturas de dados utilizando arrays, os desenvolvedores sentiam falta de classes que implementassem estruturas como listas ligadas e tabelas de espalhamento(hash), além da dificuldade que os usuários enfrentam para lembrar de todos os diferentes métodos, sintaxe e construtores presentes em cada classe de coleção.

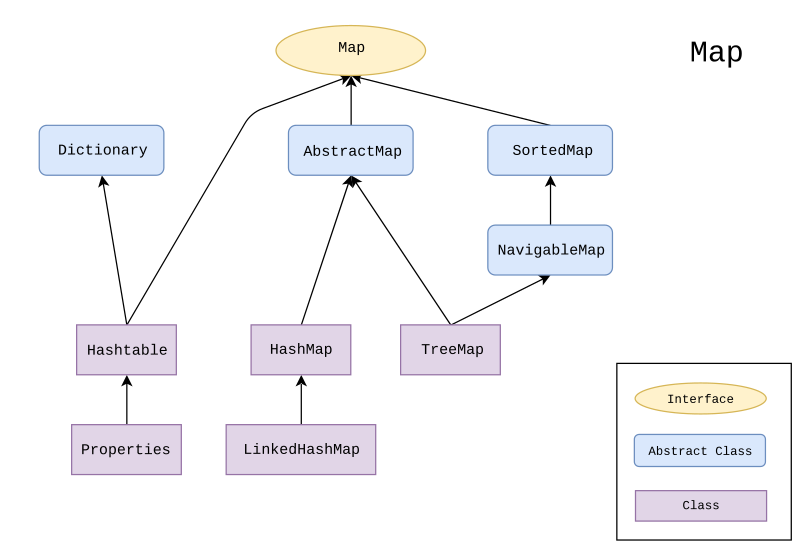
Todas essas coleções não tinham uma interface comum, embora o objetivo principal de todas as coleções seja o mesmo, a implementação de todas essas coleções foi definida de forma independente e não teve correlação entre elas.

Figura 1 — Classe de coleção e hierarquia de interface.



Fonte: Internet - <https://en.wikipedia.org/>, 2022.

Figura 2 — Classe de mapa e hierarquia de interface



Fonte: Internet - <https://en.wikipedia.org/>, 2022.

* 1. **Liberdade e controle do usuário**

Temos quatro grandes tipos de coleções: Set (conjunto), List (lista), Queue (fila) e Map (mapa), a partir dessas interfaces, temos muitas subclasses concretas que implementam várias formas diferentes de se trabalhar com cada coleção.

Todas as interfaces e classes são encontradas dentro do pacote (package) java.util, embora a interface Map não ser filha direta da interface Collection ela também é considerada uma coleção devido a sua função.

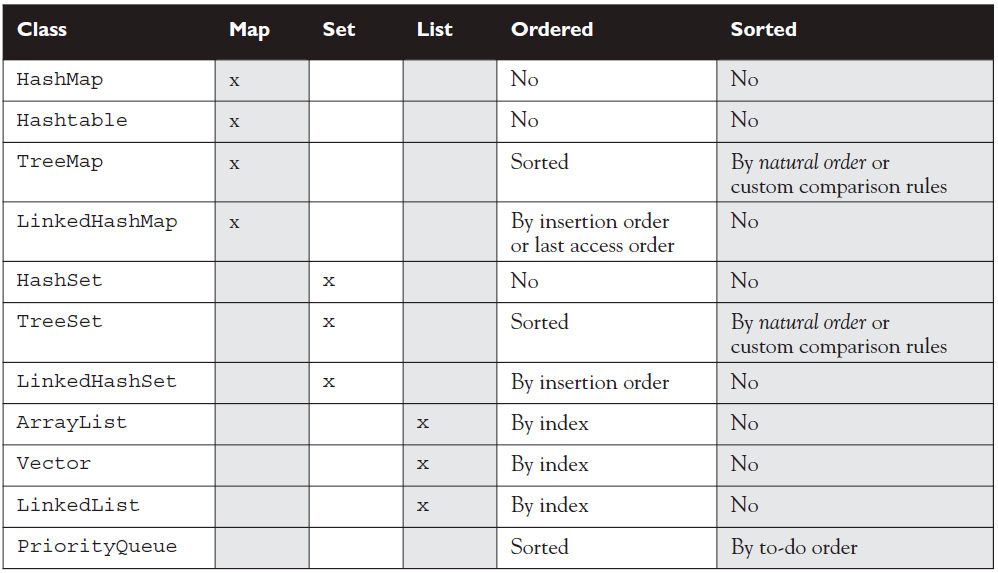
* + 1. *Set(Conjunto)* - é uma coleção do tipo conjunto de elementos. As características principais deste tipo de coleção são: os elementos não possuem uma ordem de inserção e não é possível ter dois objetos iguais.
    2. *List(Lista)* - é uma coleção do tipo lista, em que a ordem dos elementos é dado através de sua inserção dentro da lista.
    3. *Queue* - é uma coleção do tipo fila. As principais características deste tipo de coleção são: a ordem que os elementos entram na fila é a mesma ordem que os elementos saem da fila (FIFO - First In First Out), podemos também criar filas com prioridades.
    4. *Map(Mapa)* - é uma coleção do tipo mapa. As principais características deste tipo de coleção são: os objetos são armazenados na forma de chave / valor, não pode haver chaves duplicadas dentro do mapa.

As principais diferenças dentre as interfaces citadas acima são:

* *List*: Lista de coisas;
* *Set:* Lista de coisas sem repetição;
* *Map*: Lista com chave-valor, sendo que a chave deve ser única;
* *Queue*: Fila.

Cada interface possui classes que as implementam. Como são muitas vou apenas colocar o quadro comparativo de cada uma delas, que acho que resume muito bem a funcionalidade de cada uma:

##### Tabela 01 - Comparativo entre interfaces

Fonte: Internet - <https://pt.stackoverflow.com>, 2022.

* 1. **Consistência e padrões**
     1. *Set* - O uso de um Set pode parecer desvantajoso: não armazena a ordem, e não aceita elementos repetidos, não há métodos que trabalham com índices, como o get(int) que as listas possuem.

A grande vantagem do Set é que existem implementações, como a HashSet, que possui uma performance incomparável com as Lists quando usado para pesquisa (método contains por exemplo).

* + 1. *Queue* - Para pegar um elemento é muito fácil: basta pegarmos a célula em que aquele elemento se encontra e acessar o elemento de dentro dela Perceba que este método consome tempo linear. Esta é uma grande desvantagem da Lista Ligada em relação aos Vetores. Vetores possuem o chamado acesso aleatório aos elementos: qualquer posição pode ser acessada em tempo constante. Apesar dessa grande desvantagem, diversas vezes utilizamos uma Lista e não é necessário ficar acessando posições aleatórias: comumente percorremos a lista por completa.
  1. **Prevenção de erros**

As interfaces apresentadas anteriormente possuem diversas implementações que são utilizadas para armazenar as coleções, na tabela abaixo estão resumidas as implementações mais comuns:

##### Tabela 02 - Implementações de uso geral

##### 

Fonte: Internet - <https://www.devmedia.com.br>, 2022).

Abaixo seguem apresentadas algumas características das implementações que podem ajudar a decidir qual delas utilizar em uma aplicação:

* + 1. *ArrayList* –é como um array cujo tamanho pode crescer. A busca de um elemento é rápida, mas inserções e exclusões não são. Podemos afirmar que as inserções e exclusões são lineares, o tempo cresce com o aumento do tamanho da estrutura. Esta implementação é preferível quando se deseja acesso mais rápido aos elementos. Por exemplo, se você quiser criar um catálogo com os livros de sua biblioteca pessoal e cada obra inserida receber um número sequencial (que será usado para acesso) a partir de zero;
    2. *LinkedList* – implementa uma lista ligada, ou seja, cada nó contém o dado e uma referência para o próximo nó. Ao contrário do ArrayList, a busca é linear e inserções e exclusões são rápidas. Portanto, prefira LinkedList quando a aplicação exigir grande quantidade de inserções e exclusões. Um pequeno sistema para gerenciar suas compras mensais de supermercado pode ser uma boa aplicação, dada a necessidade de constantes inclusões e exclusões de produtos;
  1. **Reconhecer ao invés de lembrar**
     1. *Exemplo 01 -* java.util.List

import java.util.ArrayList;

import java.util.List;

/\*\*

\* Classe utilizada para demonstrar o uso da estrutura

\* de dados Lista.

\*/

public class ExemploLista {

public static void main(String[] args) {

List nomes = new ArrayList();

nomes.add("Zezinho");

nomes.add("Luizinho");

nomes.add("Joãozinho");

for(int cont = 0; cont < nomes.size(); cont++) {

String nome = (String) nomes.get(cont);

System.out.println(nome);

}

}

}

* 1. **Flexibilidade e Eficiência;**
  2. **Estética e Design minimalista;**
  3. **Auxiliar usuários a reconhecer, diagnosticar e recuperar erros;**
  4. **Ajuda e Documentação.**

1. **CONCLUSÃO**

**REFERÊNCIAS**

ANDRADE, M. M. **Introdução à metodologia do trabalho científico**: elaboração de trabalhos na graduação. 10. ed. São Paulo: Atlas, 2010.

\_\_\_\_\_\_. **O que são Java Collections**. Rio de Janeiro, c2022. Disponível em: < <https://en.wikipedia.org/wiki/Java_collections_framework>>. Acesso em: 11 jun. 2022.

\_\_\_\_\_\_. **Para que são usadas**. Rio de Janeiro, c2022. Disponível em: <<https://www.devmedia.com.br/java-collections-como-utilizar-collections/18450>>. Acesso em: 11 jun. 2022.

\_\_\_\_\_\_. **Quais os principais tipos de Collections**. Rio de Janeiro, c2022. Disponível em: <<http://www.universidadejava.com.br/java/java-collection/>>. Acesso em: 11 jun. 2022.

\_\_\_\_\_\_. **Comparativo entre interfaces**. Rio de Janeiro, c2022. Disponível em: <<https://pt.stackoverflow.com/questions/34605/quais-tipos-de-cole%C3%A7%C3%B5es-e-suas-diferen%C3%A7as-em-java>>. Acesso em: 11 jun. 2022.

\_\_\_\_\_\_. **Quais as vantagens de desvantagens de cada uma**. Rio de Janeiro, c2022. Disponível em: <<https://www.alura.com.br/conteudo/java-collections>>. Acesso em: 11 jun. 2022.

\_\_\_\_\_\_. **Quando devem ser usadas**. Rio de Janeiro, c2022. Disponível em: <<https://www.devmedia.com.br/java-collections-como-utilizar-collections/18450>>. Acesso em: 11 jun. 2022.

\_\_\_\_\_\_. **Implementações de uso geral**. Rio de Janeiro, c2022. Disponível em: <<https://www.devmedia.com.br/java-collections-como-utilizar-collections/18450>>. Acesso em: 11 jun. 2022.

\_\_\_\_\_\_. **Exemplos de utilização**. Icapuí, c2022. Disponível em: <<https://acervolima.com/colecoes-em-java/>>. Acesso em: 11 jun. 2022.

\_\_\_\_\_\_. **Exemplos de utilização**. Rio de Janeiro, c2022. Disponível em: <<http://www.universidadejava.com.br/java/java-collection/> Acesso em: 11 jun. 2022.

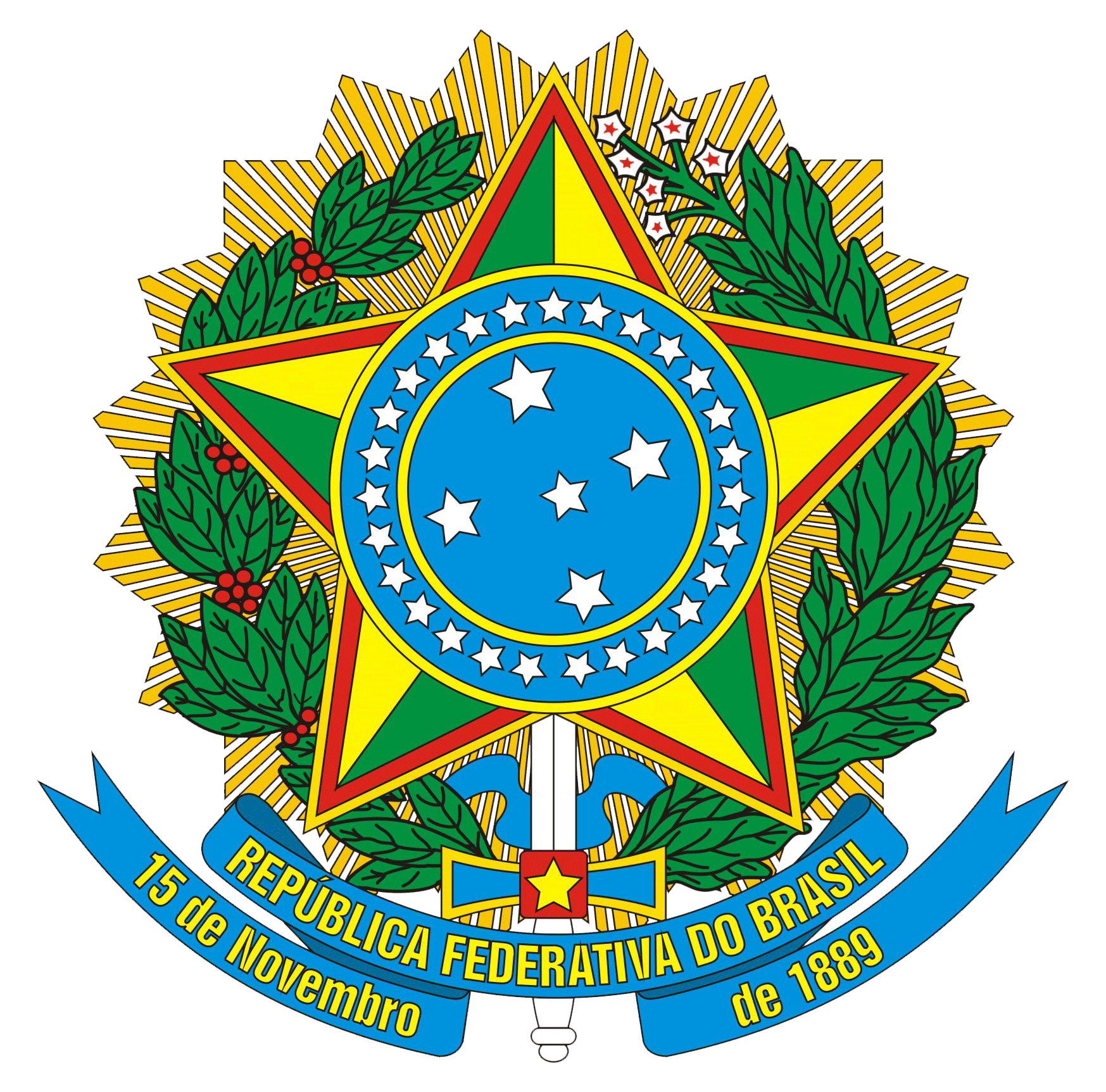
**APÊNDICE A** – **RELAÇÃO DE NORMAS TÉCNICAS VIGENTES UTILIZADAS NA NORMALIZAÇÃO DE TRABALHOS ACADÊMICOS**

Quadro 2 — Normas técnicas vigentes sobre normalização de trabalhos acadêmicos do ABNT/CB - 014

|  |  |
| --- | --- |
| **Número** | **Título** |
| 6022:2018 | Artigo em publicação periódica técnica e/ou científica - Apresentação |
| 6023:2002 | Referências - Elaboração |
| 6024:2012 | Numeração progressiva das seções de um documento - Apresentação |
| 6027:2012 | Sumário - Apresentação |
| 6028:2003 | Resumo - Apresentação |
| 6034:2004 | Índice - Apresentação |
| 10520:2002 | Citações em documentos - Apresentação |
| 10719:2015 | Relatório técnico e/ou científico - Apresentação |
| 12225:2004 | Lombada - Apresentação |
| 14724:2011 | Trabalhos acadêmicos - Apresentação |
| 15287:2011 | Projeto de pesquisa - Apresentação |
| 15437:2006 | Pôsteres técnicos e científicos - Apresentação |

Fonte: elaborado pelo autor, de acordo com o Catálogo da ABNT.

**ANEXO A — RESOLUÇÃO QUE APROVA A CRIAÇÃO DO CURSO SUPERIOR DE TECNOLOGIA EM GESTÃO AMBIENTAL NO IFCE CAMPUS PARACURU**

****

**SERVIÇO PÚBLICO FEDERAL**

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ**

**CONSELHO SUPERIOR**

**RESOLUÇÃO N° 01, DE 10 DE JANEIRO DE 2018**

Aprova *ad referendum* a criação do curso Superior de Tecnologia em Gestão Ambiental no *campus* Paracuru.

**O PRESIDENTE EM EXERCÍCIO DO CONSELHO SUPERIOR DO INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ**, no uso de suas atribuições legais e estatutárias e considerando o Memorando nº 001/2018/GDG da direção-geral do *campus* Paracuru,

**R E S O L V E:**

**Art. 1º -** Criar, *ad referendum* do Conselho Superior, o curso Superior de Tecnologia em Gestão Ambiental do *campus* Paracuru e autorizar a oferta de 35 vagas semestrais.

**Parágrafo único -** O curso será ofertado na modalidade presencial e nos turnos matutino e vespertino, conforme definido no projeto pedagógico em anexo.

**Art. 2º -** A interrupção da oferta e/ou a extinção do referido curso deverá ser submetida a este conselho para aprovação, com as devidas justificativas e a apresentação do planejamento de realocação de recursos humanos e de materiais vinculados ao curso.

José Wally Mendonça Menezes

**Presidente em exercício do Conselho Superior**