Sistema de Navegação com Pilhas

André Kilesse

27 de março de 2025

1 Introdução

Este trabalho implementa um sistema que simula o histórico de navegação de um browser, utilizando estruturas de dados do tipo pilha (*First In Last Out - FILO*). O programa permite:

- Navegar entre páginas
- Voltar para páginas anteriores
- Avançar para páginas visitadas

A solução foi desenvolvida em JavaScript/Node.js com três componentes principais: uma classe `Navegador` que gerencia o estado e duas classes (`Una` e `Node`) que implementam a estrutura de pilha.

A ideia é de se utilizar duas pilhas, sendo uma com os avanços (*Forward*) e outra para os retornos (*History*), sendo assim quando usuário solicitar, por exemplo, o retorno, o topo da pilha *History*, passará a ser o topo da pilha *Forward*, e vice-versa.

2 Implementação

2.1 Estrutura de Dados

> "A pilha [...] é bem mais simples. Quando você insere um item, ele é colocado no topo da pilha. Quando você lê um item, lê apenas o item do topo da pilha e ele é retirado da lista. Logo, sua lista contém apenas duas ações: push (inserir) e pop (remover)."*

* Adaptado de BHARGAVA, Aditya. *Entenda Algoritmos.* São Paulo: Novatec, 2017. p. 61. Original: "lista de afazeres".

Foi utilizado uma pilha (`Una`) com:

- Operações básicas: *push()* e *pop()*
- Armazenamento via nós encadeados (`Node`)

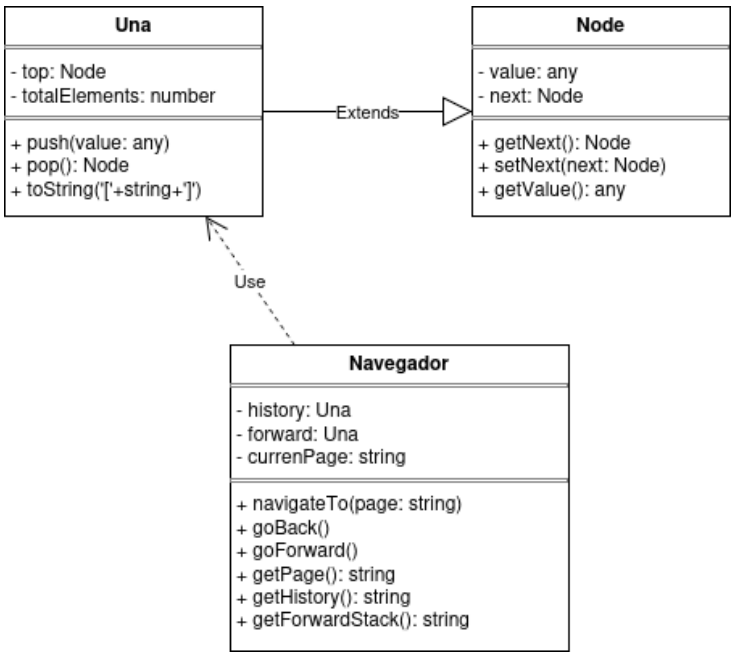

Figura 1: Diagrama de Classes da estrutura

2.2 Funcionamento

O sistema mantém:

- *`history`*: pilha de páginas visitadas
- *`forward`*: pilha de páginas para avançar
- *`currentPage`*: página atual

Sendo o Navegador responsável por tranferir os nós de uma pilha para outra conforme necessidades do usuário.

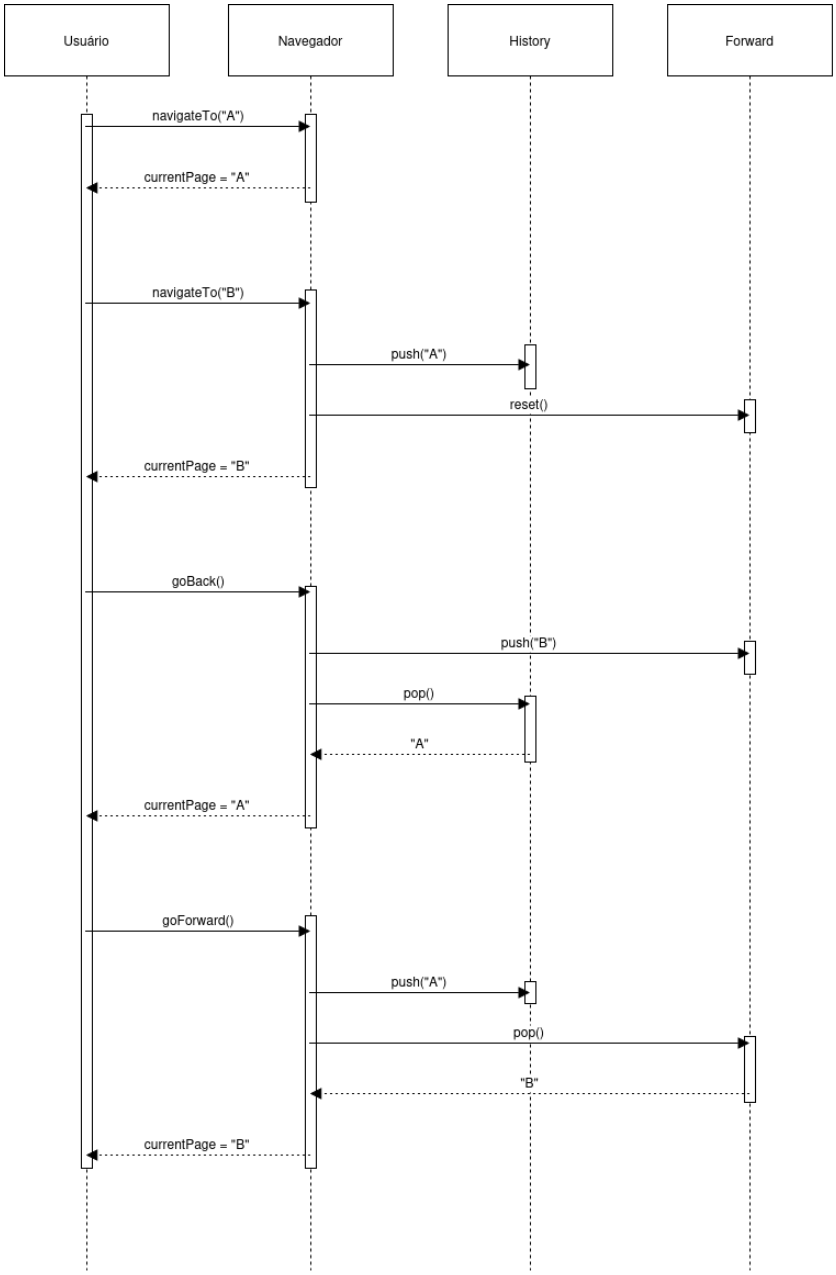

Figura 2: Fluxo das operações back/forward

"Esta estrutura de dados é chamada de pilha. A pilha é uma estrutra de dados simples. Você a tem usado esse tempo todo sem perceber!" *

* BHARGAVA, Aditya. *Entenda Algoritmos*. São Paulo: Novatec, 2017. p. 62.

2.3 Principais Funções

- `navigateTo(page)`: Adiciona página ao histórico
- `goBack()`: Move página atual para *`forward`* e recupera do *`history`*
- `goForward()`: Operação inversa ao `goBack()`

3 Testes Executados

Foram realizados testes manuais via console:

```
> navigateTo("A")
> navigateTo("B")
> goBack() // Retorna para A
> goForward() // Volta para B
```

O sistema comportou-se conforme esperado, mantendo o estado consistente das pilhas, conseguindo alterar os nós de uma pilha, foram também realizados testes adicionando mais nós a partir de laços de repetições, que também comportou como esperado.

4 Conclusão

Portanto, pode-se concluir que as pilhas são de grande importância e utilidade em estrutura de dados, sendo uma estrutura utilizada em sistemas onde o último dado a entrar deve ser o primeiro a sair (*FILO*). As principais dificuldades encontradas foram:

- Gerenciar corretamente a limpeza da pilha `forward` ao navegar
- Garantir a consistência entre as pilhas em operações consecutivas

O trabalho demonstrou na prática a aplicação de pilhas em um cenário real.

5 Bibliografia

- BHARGAVA, Aditya. *Entenda Algoritmos.* Novatec, 2017.

A Código Fonte

Arquivo node.js

```
/**
 * Classe que representa um n   da pilha
 */
class Node {
    constructor(value) {
        this.value = value;  // Valor armazenado no n
        this.next = null;    // Refer ncia para o pr ximo n
    }

    // Retorna o pr ximo n
    getNext() {
        return this.next;
    }

    // Define o pr ximo n
    setNext(next) {
        this.next = next;
    }

    // Retorna o valor do n
    getValue() {
        return this.value;
    }

    // Representa  o em string do n
    toString() {
        return this.value.toString();
    }
}

module.exports = Node;
```

Arquivo una.js

```
const Node = require('./node');

/**
 * Implementa  o de uma pilha (LIFO) usando n s encadeados
 */
class Una {
    constructor() {
        this.top = null;          // Refer ncia para o topo da pilha
        this.totalElements = 0;   // Contador de elementos
    }
```

```
    /**
     * Adiciona um valor no topo da pilha
     * @param {any} value Valor a ser armazenado
     */
    push(value) {
        const newNode = new Node(value);
        newNode.setNext(this.top); // O novo n  aponta para o antigo topo
        this.top = newNode;        // Atualiza o topo
        this.totalElements++;
    }

    /**
     * Remove e retorna o elemento do topo
     * @returns {Node} N  removido
     */
    pop() {
        if (this.top === null) {
            throw new Error("Pilha vazia");
        }
        const aux = this.top;
        this.top = this.top.getNext(); // Atualiza o topo
        this.totalElements--;
        return aux;
    }

    /**
     * Retorna representa  o em string da pilha
     * @returns {string}
     */
    toString() {
        if (this.totalElements === 0) return "[ ]";

        let currentNode = this.top;
        let builder = "[";

        for (let i = 0; i < this.totalElements; i++) {
            builder += currentNode.getValue();
            if (i < this.totalElements - 1) builder += ", ";
            currentNode = currentNode.getNext();
        }

        return builder + "]";
    }
}

module.exports = Una;
```

Arquivo global.js

```
const Una = require('./una');

class Navegador {
    constructor() {
        this.history = new Una();  // Pilha de hist rico (back)
        this.forward = new Una();  // Pilha de avan o (forward)
        this.currentPage = '';     // P gina atual
    }

    /**
     * Navega para uma nova p gina
     * @param {string} page - URL da p gina
     */
    navigateTo(page) {
        if (this.currentPage !== '') {
            this.history.push(this.currentPage);
        }
        // Reseta a pilha de avan o
        while (this.forward.totalElements > 0) {
            this.forward.pop();
        }
        this.currentPage = page;
    }

    /**
     * Volta para a p gina anterior
     */
    goBack() {
        if (this.history.totalElements > 0) {
            this.forward.push(this.currentPage);
            this.currentPage = this.history.pop().getValue();
        }
    }

    /**
     * Avan a para a pr xima p gina
     */
    goForward() {
        if (this.forward.totalElements > 0) {
            this.history.push(this.currentPage);
            this.currentPage = this.forward.pop().getValue();
        }
    }

    // Getters
    getPage() {
        return this.currentPage;
    }
```

```
    getHistory() {
        return this.history.toString();
    }

    getForwardStack() {
        return this.forward.toString();
    }
}

module.exports = Navegador;
```

Arquivo main.js

```
const Navegador = require('./global');
const prompt = require('prompt-sync')();
const navegador = new Navegador();

exibirMenu = () => {
  console.log('\n====== MENU ======');
  console.log('1. Navegar para nova p gina');
  console.log('2. Voltar');
  console.log('3. Avan ar');
  console.log('4. Exibir estado atual');
  console.log('5. Sair');
}

let opcao;
do {
  exibirMenu();
  opcao = prompt('Escolha uma op   o: ');

  switch(opcao) {
    case '1':
      const pagina = prompt('Digite a URL da p gina: ');
      navegador.navigateTo(pagina);
      break;
    case '2':
      navegador.goBack();
      break;
    case '3':
      navegador.goForward();
      break;
    case '4':
      console.log('\n=== ESTADO ATUAL ===');
      console.log('P gina atual:', navegador.getPage());
      console.log('Hist rico:', navegador.getHistory());
      console.log('Avan o:', navegador.getForwardStack());
      break;
    case '5':
```

```
      console.log('Saindo...');
      break;
    default:
      console.log('Op  o inv lida!');
  }
} while (opcao !== '5');
```