UNIVERSIDADE FEDERAL DE VIÇOSA

LUCAS CARVALHO DE OLIVEIRA

DESENVOLVIMENTO DE UMA APLICAÇÃO PARA AUXILIAR NO PROCESSO DO ENSINO-APRENDIZAGEM DE TEORIA DOS GRAFOS

FLORESTAL – MINAS GERAIS

2019

LUCAS CARVALHO DE OLIVEIRA

DESENVOLVIMENTO DE UMA APLICAÇÃO PARA AUXILIAR NO PROCESSO DO ENSINO-APRENDIZAGEM DE TEORIA DOS GRAFOS

Monografia, apresentada ao Curso de Ciência da Computação da Universidade Federal de Viçosa como requisito para obtenção de título de bacharel em Ciência da Computação.

Orientador: Marcus H. S. Mendes

FLORESTAL – MINAS GERAIS

2019

LUCAS CARVALHO DE OLIVEIRA

DESENVOLVIMENTO DE UMA APLICAÇÃO PARA AUXILIAR NO PROCESSO DO ENSINO-APRENDIZAGEM DE TEORIA DOS GRAFOS

Monografia, apresentada ao Curso de Ciência da Computação da Universidade Federal de Viçosa como requisito para obtenção de título de bacharel em Ciência da Computação.

APROVADO:

_____

Profª.: Maria Amélia Lopes Silva

(UFV)

_____

Prof.: Ronan Dutra Mendonça

(UFV)

_____

Prof.: Marcus Henrique Soares Mendes

(Orientador)

(UFV)

A Deus, meus pais e familiares, meus professores e a todas amizades que nesta instituição fiz.

Meu muito obrigado.

RESUMO

Este trabalho apresenta o processo de desenvolvimento de uma aplicação para auxílio no ensino-aprendizado da disciplina Teoria dos Grafos, com o intuito de incentivar e facilitar a forma do aprendizado, utilizando uma interface gráfica de usuário que respeite os padrões de UI/UX e possa ser executada em múltiplas plataformas. A ideia é que o estudante possa utilizá-lo para fixar o conteúdo aprendido através de sala de aula e consiga entender o funcionamento de algoritmos sendo executados sobre grafos desenhados na aplicação. Ademais, a aplicação ainda mostra propriedades relevantes sobre o grafo que podem transparecer informações que ajudam o estudante a identificar uma determinada característica de um dado grafo.

Palavras-Chave: Teoria dos Grafos, Software, Aplicação, Ensino, Aprendizado, PAAD-Grafos.

ABSTRACT

This paper presents the process of developing an application to aid in the teaching and learning of Graph Theory, to encourage and facilitate learning, using a graphical user interface that respects UI / UX standards and can run on multiple platforms. An idea that the student can use the resource to reinforce the content learned through the classroom and see how the algorithm works on graphs drawn in the application. In addition, an application even shows the relevant characteristics about the graph, which can display transparent information, which can be used to identify a specific characteristic of a given graph.

Keywords: Graph Theory, Software, Application, Teaching, Learning, PAAD-Grafos.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVEATURAS E SIGLAS

WEB (McPherson, 2009) – *World Wide Web* – Sistema de informação baseado em documentos, que podem se inter-relacionar através de laços e podem ser acessados através da Internet.

NP (Garey e Johnson, 1979) – *Non-Deterministic Polynomial Time* – Classe de problemas não resolvíveis em tempo polinomial e muito provavelmente não fazem parte da classe de problemas resolvíveis em tempo polinomial.

API (Zanette, 2019) – *Application Program Interface* – Sub-rotinas de utilização de aplicação WEB.

PHP (PHP Group, 2019) – *PHP: Hypertext Processor* – Processador de Hypertextos PHP. Linguagem de programação para criação de aplicações web cliente-servidor que atua do lado do servidor.

JS (Ecma International, 2011) – *JavaScript* – Abreviação para a linguagem de programação JavaScript que atua junto ao lado do cliente a partir de um navegador.

NPM (NPM Inc, 2019) – *Node Packages Manager* – Gerenciador de pacotes do NodeJS.

PaaS (Gonçalves, 2019) – *Plataform as a Service* – Plataforma como serviço.

URL (Significados, 2019) – *Uniform Resource Locator* – Localizador padrão de recursos. *Link* ou endereço de um sítio na Internet.

UX/UI (Arty, 2018) – *User eXperience / User Interface* – Interface com o usuário e a experiência do usuário.

JSON (Ecma International, 1999) - *JavaScript Object Notation* – Notação de Objeto do *JavaScript*.

MVC (Pressman, 2006) – *Model View Control* – Padrão de projeto de construção de aplicações Modelo Visão e Controle.

SUMÁRIO

1. Introdução

Uma aplicação de apoio ao ensino-aprendizagem para a disciplina de Teoria de Grafos (Goldbarg, 2012, et. al) é uma ferramenta computacional que pode ajudar no reforço da teoria aprendida em sala de aula. A necessidade de tal ferramenta se faz por parte do auxílio lúdico ao aprendiz, que tem, por vezes, que acompanhar algoritmos complexos sendo executados sobre suas estruturas de dados e diferentes formas de representações computacionais.

Partir-se-á da definição clara dos objetivos gerais e específicos que devem ser atendidos para a construção da aplicação *Web* (McPherson, 2009), caminhando pelo referencial teórico adotado até chegar na metodologia de desenvolvimento.

A construção da aplicação é feita utilizando-se um modelo de desenvolvimento de *software Ad Hoc* (Junior, 2019), mas está bem organizado e estruturado, como será discutido no item 4 deste trabalho.

Aqui ainda se observa que já existem algumas ferramentas para visualização de grafos, contudo nenhuma com o enfoque desta: ser de fácil acesso a qualquer momento, em qualquer lugar do mundo e rodar em plataformas *mobile* e *desktop* e que ajude o estudante na fixação do conteúdo da disciplina de Teoria de Grafos.

Existem bibliotecas de visualização como o Cytoscape.js (Donnelly Centre, 2019), mais próximas de ferramentas de mercado, que não tem por objetivo tratar da disciplina, mas que pode ser inserida em qualquer aplicação *web*. Por outro lado, existem aplicações web prontas, como o VisuAlgo (Halim, 2011), que tem o enfoque do ensino, porém não é responsivo para dispositivos mobile.

Diversos trabalhos parecidos com este mostram aplicações *desktop* puras que necessitam de instalação num computador clássico e não estão disponíveis na internet. Outros trabalhos como este serão discutidos na seção 3.3 que trata dos trabalhos relacionados.

Por fim, os resultados obtidos e conclusão, além de trabalhos futuros podem ser vistos nas seções 5 e 6 deste trabalho.

Para concluir esta introdução não podemos deixar de falar sobre o que dizem as afirmações sobre informática na educação para que se possa reiterar a construção desta aplicação.

É notório que a Informática na educação vem cada dia alcançando mais brasileiros em sala de aula, seja pela facilidade de acesso ao recurso tecnológico ou a utilização de ferramentas que incrementem a experiência do estudante. Esta área já se mostra tão importante que a Conferência Acadêmica Internacional de Educação, Ensino e Aprendizado, realizado pela *WestEastInstitute*, reuni informações com padrões de projeto e é referência para sistemas de apoio a educação construídos pelo mundo.

Sem mais delongas, comecemos a destrinchar a aplicação construída.

2. Objetivo Geral

O objetivo geral deste trabalho é construir uma ferramenta computacional que auxilie no processo de ensino-aprendizado de Teoria dos Grafos.

2.1 Objetivos Específicos

Para atingir o objetivo geral da construção de uma aplicação tal como essa se faz necessário:

- Estudar Teoria dos Grafos;
- Estudar a composição de *Frameworks Front-End*;
- Estudar a composição de uma biblioteca de visualização;
- Projetar uma interface de usuário que ajude na experiência do aprendizado;
- Implementar uma base da aplicação aonde o usuário consiga “desenhar” um grafo;
- Implementar área da aplicação que mostre informações sobre um determinado grafo desenhado;
- Implementar formas de se gerar novos grafos baseados no grafo desenhado;
- Implementar formas de visualização de algoritmos sendo executados sobre a estrutura de dados.

3. Referencial Teórico

Como apontado em Goldbarg, 2012, Teoria dos Grafos, é uma área da matemática que estuda as relações entre itens de um determinado conjunto valendo-se de uma estrutura chamada Grafo.

O Grafo - $G(V,E)$ – é um arranjo de um conjunto não vazio de vértices V, que são os elementos do conjunto, e um conjunto de relação entre estes itens denominado de arestas E – do inglês *edges* – que são pares não ordenados de vértices.

Dada necessidade da aplicação as arestas podem ter uma direção, ou ainda orientação, representada graficamente por uma seta em uma das pontas da aresta e uma ponderação, associando pesos numéricos às relações.

A natureza da representação de problemas de relevância prática por grafos pode ser encontrada em diversos locais e a criação de algoritmos para operar sobre grafos é matéria de estudo importante para a Ciência da Computação.

Um exemplo clássico de utilização de grafos para o próprio curso de graduação é a utilização de grafos direcionados para a representação de máquinas de estado finito ou autômato finito (Vieira, 2006).

Figura 1 – Representação visual do problema das Sete Pontes de Königsberg

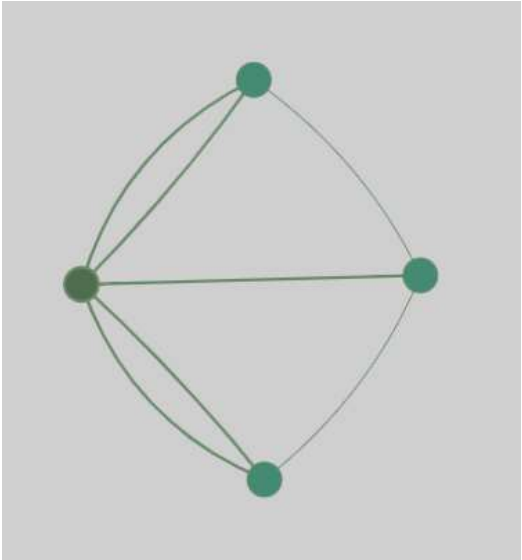
Fonte: Creative Commons

A Teoria dos Grafos nasce com o artigo de Leonhard Euler em 1736 (Ronald, 1994), a partir do problema das sete pontes de Königsberg, hoje Kaliningrado, capital

de uma província russa, e é o primeiro produto científico da teoria dos grafos. Essas pontes são exibidas na Figura 1.

Euler estudava o percorrer na topografia que representava essas 7 pontes, apenas uma vez, sem percorrer novamente uma ponte já percorrida ou pisasse em um terreno mais de uma vez, portanto ele procurava um caminho que continha todas as pontes, nas quais deveriam ser percorridas somente uma vez.

Figura 2 – Grafo que representa o problema das Sete Pontes de Königsberg.

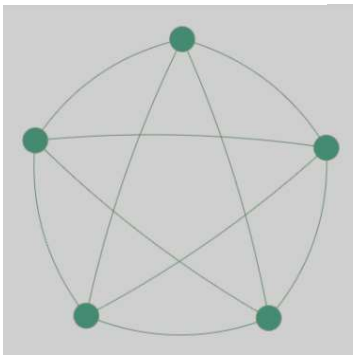


Fonte: Aplicação desenvolvida

3.1 Representação gráfica de um Grafo

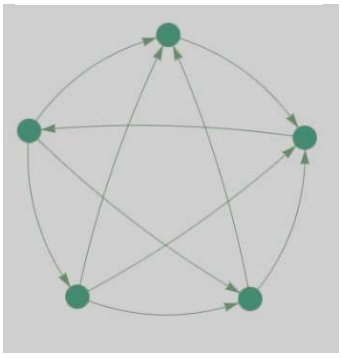
Os grafos são tipicamente desenhados utilizando-se um círculo ou quadrado para cada vértice e um segmento de reta para cada aresta conectando dois vértices. Se o grafo for direcionado, seu sentido é indicado por uma seta na aresta. As Figuras 3 e 4 mostra dois grafos de mesma topologia, porém um é não-orientado e o outro por sua vez já é orientado.

Figura 4 – Representação gráfica de um grafo não direcionado.



Fonte: Aplicação desenvolvida

Figura 3 – Representação gráfica de um grafo direcionado.



Fonte: Aplicação desenvolvida

A representação gráfica não deve ser confundida com essência abstrata do arcabouço da teoria, que não é gráfica. Os grafos apresentados nas Figuras 3 e 4 são o grafo completo de 5 vértices nas suas versões não direcionado e direcionado, respectivamente.

3.2 Problemas que envolvem grafos

Dentre os problemas que envolvem grafos estão:

1. Coloração de grafos;
2. Conjunto independente;
3. Problema do caminho mínimo;
4. Árvore de extensão mínima;
5. Problemas de roteamento (inspeção de rotas e caixeiro viajante), e;
6. Problema do fluxo máximo.

Outro problema em aberto em teoria dos grafos é o isomorfismo entre dois grafos. O problema de isomorfia em grafos procura uma função bijetora que relacione dois grafos iguais arranjados em diferentes topografias. Este é um dos problemas cuja complexidade computacional ainda está por resolver (NP-completo) (Garey e Johnson, 1979).

3.3 Aplicações e trabalhos relacionados

Existem várias aplicações e *frameworks* de visualização de grafos e algoritmos sendo executados sobre suas estruturas em diferentes linguagens de programação. Cada um deles com um enfoque ou motivações em específico.

Dentre as aplicações de apoio em teoria dos grafos existe o *Cytoscape.js* (Donnelly Centre, 2019), biblioteca feita em *JavaScript* para visualização de grafos e gráficos – análoga ao Vis.js, porém contém ferramentas mais completas para visualização de animações sobre a estrutura.

Outra aplicação conhecida é o VisuAlgo, que está hospedada na *internet* e possui um conjunto de ferramentas para visualização da execução de algoritmos sobre diversas estruturas de dados. Para a teoria de grafos, esta aplicação implementa: estruturas, conjuntos disjuntos, busca em grafos, árvore geradora mínima, caminhos mínimos, grafo bipartido e buscador de ciclos em grafos orientados.

A grande decepção do VisuAlgo vem quando tenta-se acessar a aplicação a partir de um dispositivo *mobile*. Sua estrutura não está adaptada, ou otimizada, para elementos de rede que utilizem toques na tela como forma padrão de entrada. Quando acessada via dispositivo *mobile* sua página é mostrada como na versão *desktop*.

Já nos trabalhos publicados em português há indícios de dois precedentes: TGrafos de (Silveira e Silva, 2019), cujo enfoque é o auxílio do estudo de teoria dos grafos e A-Graph de (Lozada, 2014), que também tem o mesmo propósito. Há também o *ROX Graph Theory* (Sangiorgi, 2019) que habilita criação gráfica de grafos simples e a execução de algoritmos. Ambos se diferenciam deste trabalho porque foram feitos para computadores tradicionais. Este trabalho se enfoca na mobilidade e facilidade de acesso, tendo como requisito não funcional ser acessado a qualquer momento e de qualquer lugar do mundo.

Além dessas diferenças, este trabalho foi feito como padrões *UX/UI* (Arty, 2018) modernos e mais utilizados no desenvolvimento atual de *softwares*, trazendo facilidade no uso da ferramenta e familiaridade com as operações.

4. Metodologia de desenvolvimento

O desenvolvimento da aplicação foi baseado em um ciclo iterativo e incremental de construção *Ad Hoc* (Pressman, 2006 e Dicionário Michaelis, 2019). Este estilo de desenvolvimento é um risco para a qualidade do *software* desenvolvido por não utilizar processos de desenvolvimento, não gerenciar a documentação e não executar fases previstas nos ciclos de desenvolvimentos de *software* bem estabelecidas (Junior, 2019).

Em contrapartida a metodologia de desenvolvimento *Ad Hoc* é excelente para criação de softwares a partir de experimentação prática, sendo indicada para quem está começando a aprender determinada linguagem ou está construindo uma aplicação de teste.

4.1 Ferramentas Utilizadas

Diversos *frameworks* e bibliotecas foram utilizadas para a construção da aplicação. Um *framework* se assemelha a um arcabouço conceitual que une códigos implementados um uma determinada linguagem de programação e é utilizada no auxílio ao desenvolvimento do *software*. Para aproveitar-se de suas facilidades é necessário ter o conhecimento técnico específico de como trabalhar com cada *framework* e o mesmo contribui para acelerar a implementação de código (Santos e Carvalho, 2015).

Por sua vez, uma biblioteca é uma coleção de subprogramas que beneficiam o desenvolvimento de *software*, provendo serviços a outras aplicações de forma encapsulada e modular (Zanette, 2019).

4.1.1 Framework Back-End (Laravel)

Para a construção da aplicação foi utilizado um *framework* para o desenvolvimento de aplicações *web* baseado em PHP (The PHP Group, 2019), linguagem de programação que atua do lado do servidor. O *framework* escolhido foi o Laravel – O Framework PHP para Artesões Web (Otwell, 2017).

É um *framework* que facilita a gerência do roteamento, a criação dinâmica de páginas, autenticação, autorização, bancos de dados e testes automatizados, e

considerada por muitos o melhor *Framework* PHP disponível para uso comercial gratuito. Este *framework* será especialmente interessante devido à existência do compilador de recursos do *Webpack Mix* em seu ecossistema. Esta ferramenta provê pré processadores comuns de CSS e *JavaScript* que diminuem drasticamente o tamanho dos arquivos contidos na aplicação e consequentemente tem um aumento na velocidade de reação de carregamento da página no navegador do usuário.

4.1.2 Linguagem de programação Front-End (*JavaScript*)

A maior parte desta aplicação foi desenvolvida utilizando-se a linguagem de programação que atua do lado do cliente e que roda dentro dos navegadores *JavaScript*. *JavaScript* é implementado sobre o padrão *ECMAScript Language Specification*, de identificação ECMA-262, de Junho de 2011, ou ainda ISSO/IEC 16262:2011 [15] e, de acordo com (Ecma Internetional, 2011) é baseada nas tecnologias originárias JavaScript (Netscape Corporation) e JScript (Microsfot).

4.1.3 Biblioteca de visualização de redes Vis.JS

Vis.Js (*Community edition*) é uma biblioteca para visualização de redes que foi projetada para ser fácil de usar, lidar com um número grande de dados dinâmicos e ativar manipulação e interação com os dados (Vis.Js *Community*, 2019).

Essa biblioteca é de vital importância para visualização e manipulação dos Grafos, que nela, são chamados de redes. Na documentação do componente de redes do Vis.JS pode-se observar estruturas compostas de vértices e arestas, amparadas por funções de alteração de estrutura, como, por exemplo: forma e estilo dos nós e arestas, cores, tamanhos, imagens, ferramentas do mecanismo de física e muito mais (Vis.Js *Network Documentation*, 2019).

4.1.4 Bibliotecas e Ferramentas auxiliares

O desenvolvimento desta aplicação contou com diversas bibliotecas auxiliares para que o tempo de desenvolvimento decaísse em função da utilização de componentes reutilizáveis presentes nestes conjuntos de códigos. Dentre essas bibliotecas se destacam o *jQuery*, *jQuery UI* e o *Font Awesome*.

jQuery é uma biblioteca *JavaScript* concisa, de alta responsividade e carregada com diversos recursos (jQuery *Foundation*, 2019). Ela possui uma API (Zanette, 2019) de fácil utilização e funciona dentre uma variedade de navegadores. Essa biblioteca

facilita a implementação de aplicações *JavaScript* e permite que programadores ganhem tempo de desenvolvimento e qualidade de construção de código.

jQuery UI é um conjunto de interfaces de usuário contendo interações, efeitos e temas construído sobre a biblioteca *jQuery*. Essa biblioteca é extremamente útil para construção de movimentações *drag-and-drop* e animações por sobre estruturas de dados diversas. *jQuery UI* implementa uma biblioteca de iteração com o usuário a partir de dispositivos móveis que facilita o desenvolvimento aonde movimentos mais associados à dispositivos *touch screen* são utilizados (jQuery *Foundation*, 2019).

Font Awesome é um conjunto de ícones vetorizados que se autodenomina ser o *kit* de ferramentas e ícones mais popular da web (Fonticons Inc, 2019). São mais de 1500 ícones gratuitos profissionalmente desenhados de acordo com guias e padrões forjados a partir de anos de experiência de ilustração e projeto de ícones.

Além dessas três bibliotecas a aplicação utilizou também de um *framework front-end* de utilização em larga escala no mercado: o *Bootstrap*. *Bootstrap* te permite criar projetos de aplicações web com foco em *mobile* e se diz ser o componente de biblioteca mais popular do mundo (*The Bootstrap Team and Contributors,* 2019).

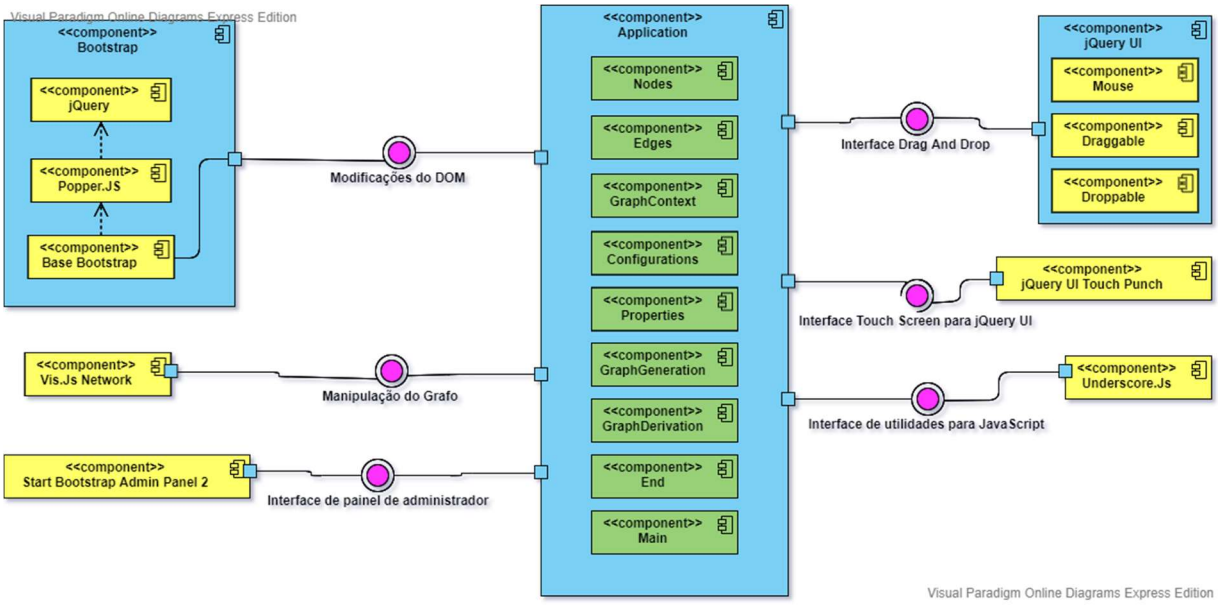
Outras bibliotecas, de menos relevância, foram utilizadas para auxiliar na construção da aplicação, mas que podem ser destacadas aqui:

- *Add-To-Homescreen*: Biblioteca para adicionar a aplicação à telas iniciais de *smartphones* e *tablets*.
- *Deep-equal:* Biblioteca pré-requisito para o *jQuery Touch Punch*.
- *Underscore*: Biblioteca de auxílio à programação *JavaScript*.
- *Json-stable-stringify*: Biblioteca de auxílio à validação de arquivos JSON.

4.2 Os componentes *JavaScript*

A utilização de pacotes *JavaScript* pré disponibilizados na internet facilita a reutilização do código e evita a “reinvenção da roda”. Todos esses pacotes são instalados através do gerenciador de pacotes NPM do *NodeJS* (*Node Inc*, 2019).

Figura 5 – Diagrama de componentes da aplicação.



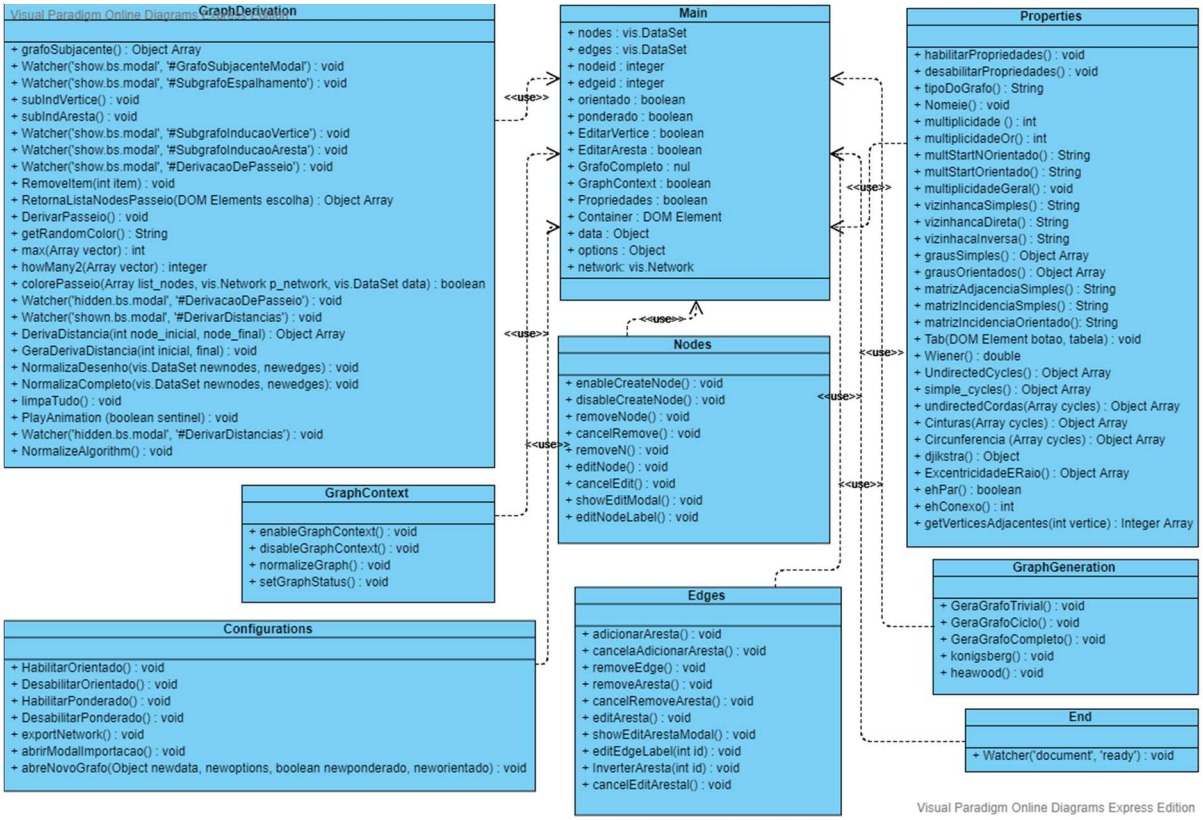
Fonte: VisualParadigm Online

Uma vez importados, os pacotes NPM podem ser utilizados dentro do compilador de recursos do *Webpack Mix* de forma bem intuitiva. Na Figura 5 pode ser encontrado um diagrama de componentes que mostra como eles estão organizados no escopo do ECMA-262.

4.2.1 Classes envolvidas

Para facilitar o desenvolvimento e modularizar as responsabilidades de cada componente da aplicação foram criados módulos *JavaScript* independentes que comunicam entre si ativando os diferentes componentes presentes na aplicação.

Figura 6 – Diagrama de classes front-end.



Fonte: VisualParadigm Online

Um diagrama de classes completo da aplicação pode ser encontrado na Figura 5. Lembrando que, como a aplicação foi construída toda basicamente em *front-end*, esse diagrama de classes reflete apenas a lógica implementada na linguagem de programação *JavaScript*.

4.3 Controle de versão e hospedagem

O controle de versão para esta aplicação foi executado utilizando-se a ferramenta Git em comunhão com o GitHub. O código fonte da aplicação pode ser encontrado no sítio eletrônico < https://github.com/EspetoRx/paad-grafos >. Lá pode-

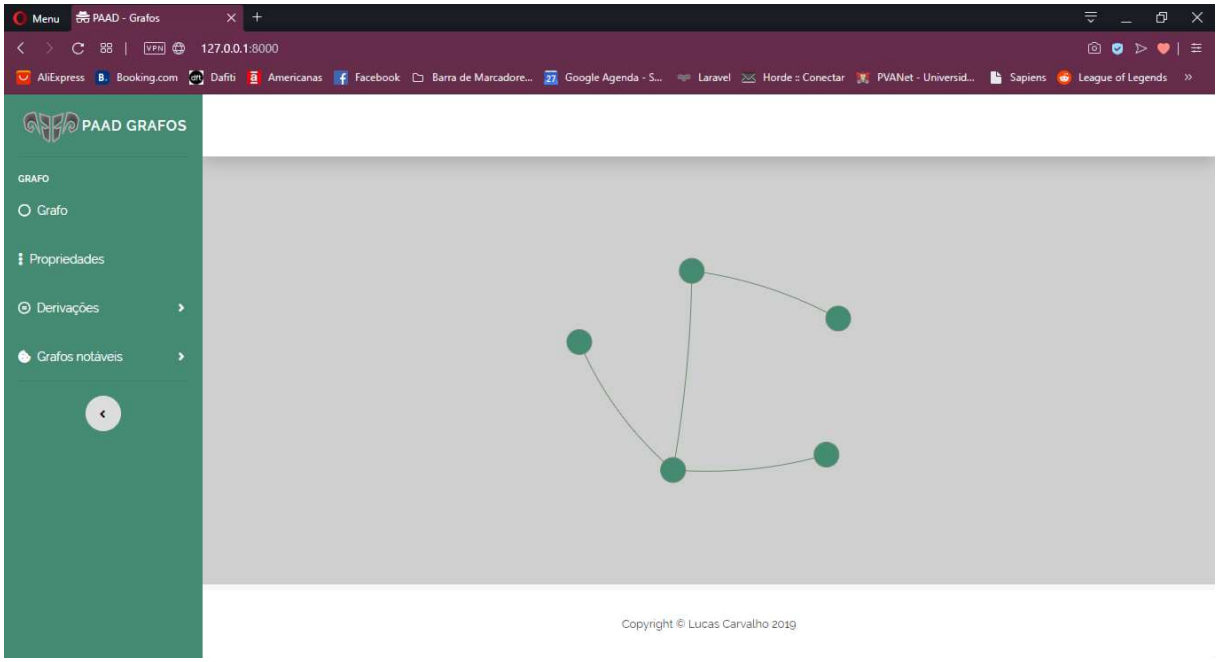
se verificar que hove 115 *commits* no ramo principal do repositório, identificando diversas versões *Alpha* com o passar do tempo.

A hospedagem da aplicação foi realizada no servidor gratuito para testes *Heroku*. O *Heroku* (Salesforce, 2019) é uma PaaS (Gonçalves, 2019) da computação em nuvem e disponibiliza a sua aplicação para testes em uma URL (Significados, 2019) disponível na internet, no caso deste trabalho < https://paad-grafos.herokuapp.com >, carregando no URL o nome de como ele surgiu.

5. Resultados Obtidos

Nessa seção será abordada a aplicação que foi desenvolvida. Neste caso partir-se-á da tela inicial. Já nesta tela, Figura 7, se pode ver um grafo desenhado, gravitando em torno do eixo central e uma barra lateral de menu que pode ser aberta ao utilizar o ícone de menu de “três barras”.

Figura 7 – Tela inicial com o grafo e o menu lateral.



Fonte: Aplicação desenvolvida

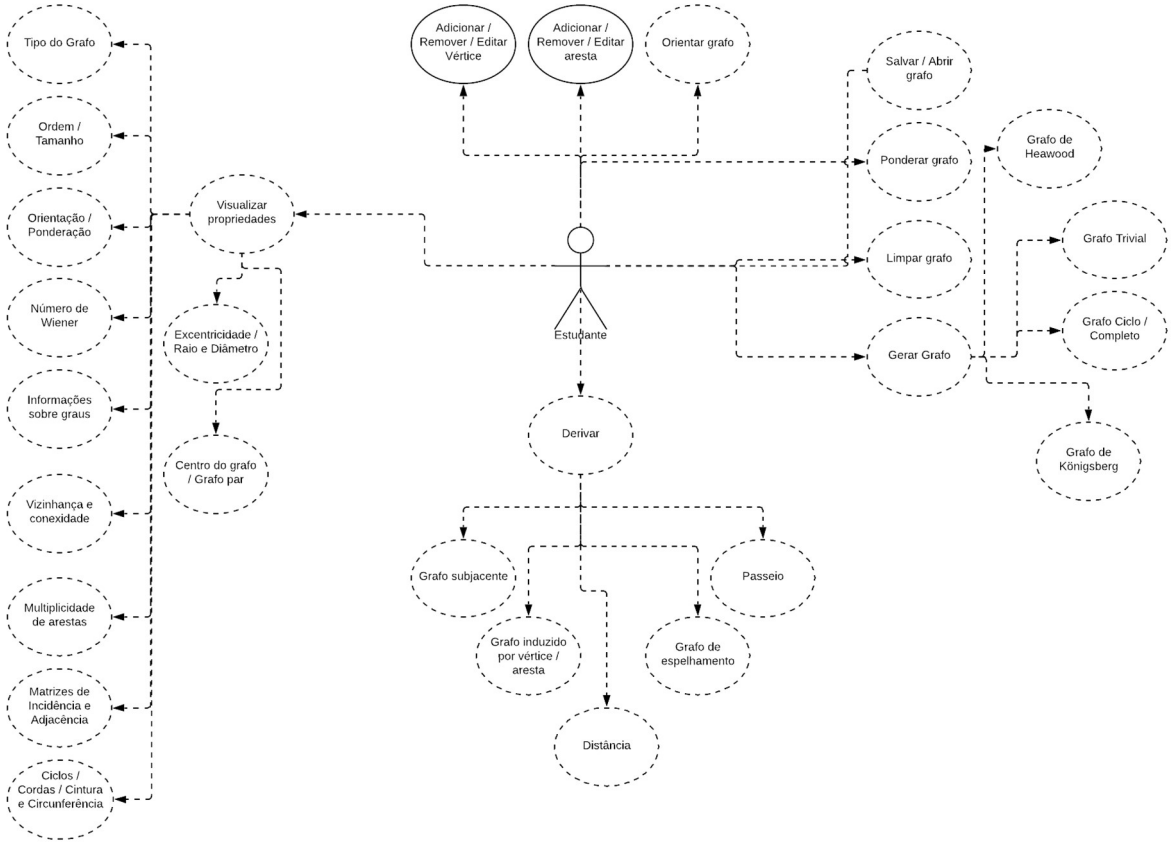
5.1 Diagrama de Caso de Uso

Antes de continuar pelo fluxo de telas que será apresentado é interessante olhar para os casos de uso presentes nesta aplicação.

O entendimento dos casos de uso de uma determinada aplicação revela informações importantes sobre o que aquela aplicação faz e quais são as ações necessárias para que o usuário consiga realizar uma operação.

Na Figura 8 é apresentado o Diagrama de Casos de Uso (Pressman, 2006) para esta aplicação. Observe que devido ao uso do modelo de desenvolvimento *Ad Hoc*, o diagrama de casos de uso não foi construído antes do desenvolvimento da aplicação e sim depois. Com este modelo de desenvolvimento, só é possível dizer o que a aplicação faz uma vez que ela já esteja pronta.

Figura 8 – Diagrama de casos de uso da aplicação



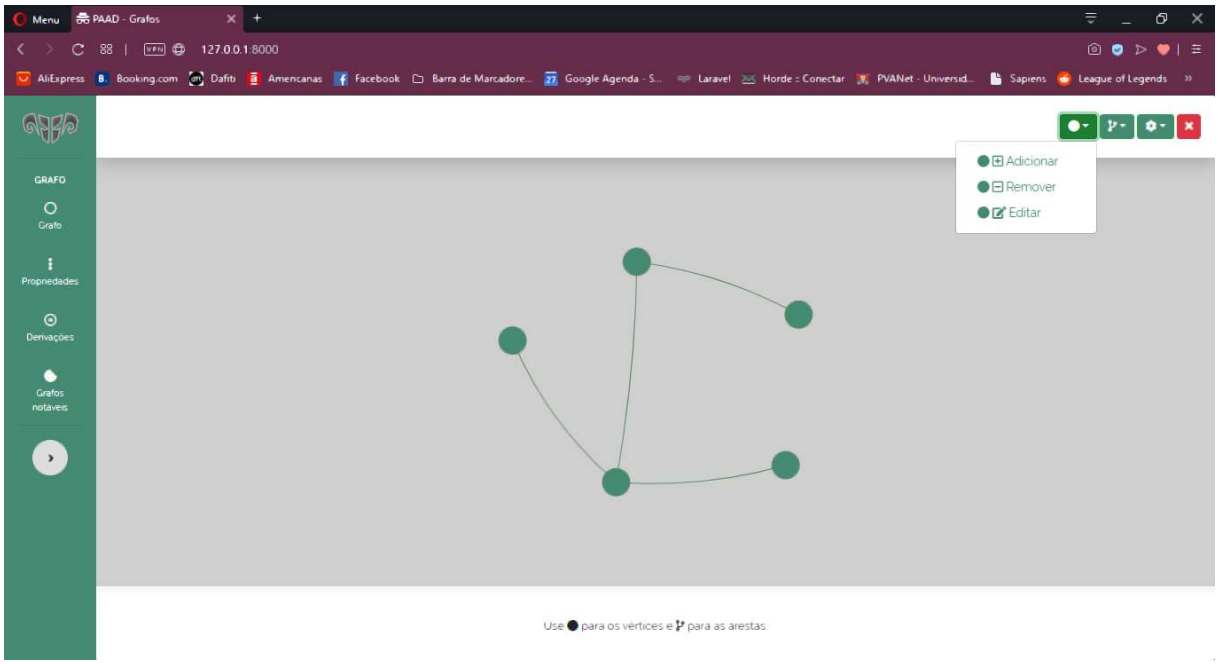
Fonte: VisualParadigm Online

5.2 Tela de alteração do Grafo

As figuras 10, 11 e 12, mostram a tela de alteração do grafo em 3 diferentes modos: primeiramente as modificações que podem ser feitas nos vértices dos grafos, depois as alterações sobre uma aresta e finalmente as alterações sobre a estrutura do grafo no geral como, por exemplo, orientação e ponderação.

Para ter acesso a essa tela, basta que o usuário clique ou toque em Grafo no menu lateral.

Figura 9 – Menu de alteração dos vértices

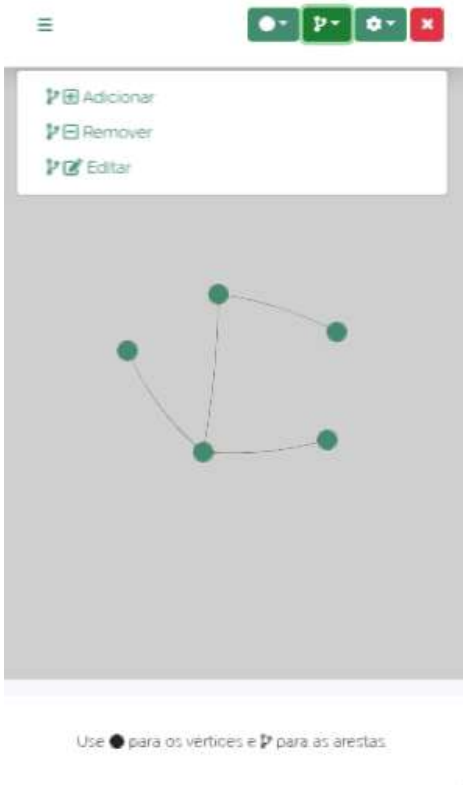


Fonte: Aplicação desenvolvida

Os nós do grafo desenhado podem ser alterados através do menu de modificação dos vértices, encontrado no canto superior direito da tela de modificação de grafo. Uma vez selecionada a opção desejada, você pode fazer alterações dentro da estrutura de vértices do grafo através de cliques ou toques por sobre a área cinza de desenho do grafo.

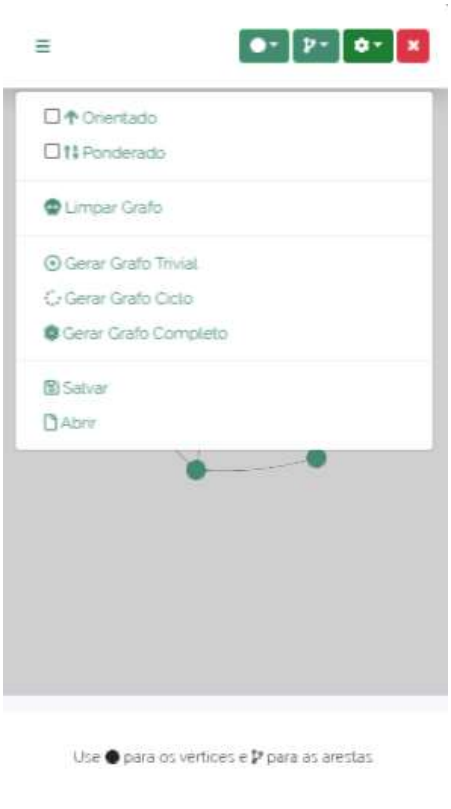
A opção de edição dos vértices lhe deixa nomeá-lo como quiser, seja com números ou com letras e a remoção de algum nó requer dupla confirmação, aonde o usuário deve clicar no vértice escolhido para remoção e no botão “ - ” que surge na barra de informações na parte inferior. Para cancelar qualquer uma destas opções basta clicar no “ X “ na mesma barra, ou selecionar outra opção no menu lateral.

Figura 11 – Menu de edição de arestas



Fonte: Aplicação desenvolvida

Figura 10 – Menu de configuração do grafo.



Fonte: Aplicação desenvolvida

Como exibido na Figura 10 o menu de alteração de arestas/arcos tem certa semelhança com o de vértices, mantendo o padrão de construção do menu *dropdown*.

As ações requeridas para modificar arestas são as mesmas que para os vértices, com exceção que, para a adição de arestas é utilizado um sistema de arrastar e soltar, do vértice inicial ao final, criando assim a elucidação de um elo entre os nós. Também é possível criar uma aresta sobre um vértice único – laço.

A opção editar permite ao usuário modificar o peso associado àquela aresta e inverter a ordem de conexão entre os nós ligados.

Uma vez criado, algumas configurações do grafo podem ser alteradas rapidamente. Essas configurações são orientação e ponderação do grafo.

Ainda em relação à alteração da estrutura dos grafos pode-se: Limpar a área de desenho, gerar grafos triviais, ciclo e completo, salvar um grafo no formato Vis.js DataSet em JSON e abrí-lo.

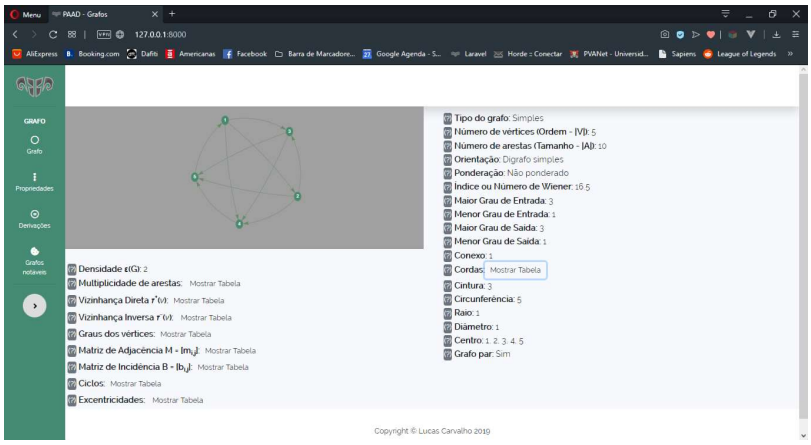
Todas essa operações foram alocadas aqui porque alteram a estrutura do grafo presente na área de desenho e tem funções importantes para serem desempenhadas de forma mais rápida.

Uma vez visualizadas todas estas opções que nos permite alterar a estrutura do grafo em GUI, segue-se para o próximo item do menu lateral.

5.3 Propriedades do Grafo

As propriedades de um determinado grafo são um conjunto de informações pertinentes acerca do grafo sob a área de desenho. A captura de tela na Figura 12 mostra como as informações das propriedades estão dispostas em computadores pessoais.

Figura 12 – Captura de tela das propriedades de um grafo.



Fonte: Aplicação desenvolvida.

Toda a aplicação conta com preparação para exibição em dispositivos mobile. Desta forma as tabelas escondidas sobre os botões “Mostrar tabelas” são responsivas.

Ao lado de cada propriedade há um ícone/botão de ajuda, para que, quando clicado, mostre alguma informação pertinente àquela propriedade. Isso é mostrado na Figura 13.

Figura 13 – Tabelas expansíveis.

Fonte: Aplicação desenvolvida

Devida quantidade de propriedades pertencentes a esta tela que não aparecem em 2. Referencial Teórico, segue um breve resumo sobre o que são estas propriedades e qual o

significado semântico associado a cada uma delas na Tabela 1.

Propriedade	Descrição
Tipo do Grafo	Se simples não possui arestas laço nem arestas paralelas entre dois vértices adjacentes, se pseudografo não possui arestas paralelas, contudo possui laço, ou se multigrafo contém tanto arestas paralelas quanto laços.
Ordem	Número de vértices de um grafo.
Tamanho	Número de arestas de um grafo.
Orientação	Se orientado.
Ponderação	Se ponderado.
Índice ou Número de Wiener	Metade da soma das distâncias de cada vértice aos demais.
Menor e maior grau	Maior e maior quantidade de arestas/arcos que incidem em determinado vértice.
Se conexo	Se há apenas uma componente conexa no grafo.
Sua densidade	Razão entre o número de vértices e o número de arestas pertencentes ao grafo.
Multiplicidade de arestas	[Tabela] Quantas arestas ou arcos existem em cada par ordenado de vértices.
Vizinhança	[Tabela] Mostra os vizinhos de cada vértice
Graus dos vértices	[Tabela] Mostra a quantidade de arcos incidentes em cada vértice.
Matriz de adjacência	[Tabela] 1 se houver uma aresta que ligue o nó *a* pertencente a linha em um nó *b* pertencente à coluna, e 0 caso contrário.
Matriz de incidência do grafo	[Tabela] -1 se houver uma aresta que ligue o no *a* pertencente a uma linha em uma aresta *b* pertencente à coluna e se *a* for origem. 1 se houver uma aresta que ligue o no *a* pertencente a uma linha em uma aresta *b* pertencente à coluna e se *a* for destino. 0 caso não houver aresta com este nó.
Ciclos	[Tabela] Mostra os ciclos presente no grafo

Excentricidades	[Tabela] Mostra a excentricidade de cada vértice, que é a maior distancia entre dois vértices v e w pertencentes ao grafo.
Cordas	[Tabela] Mostra as arestas que são cordas pertencentes aos ciclos.
Cintura	O tamanho do ciclo de menor comprimento no grafo.
Circunferência	O tamanho do ciclo de maior comprimento no grafo.
Raio	Menor valor dentre as excentricidades.
Diâmetro	Maior valor entre as excentricidades.
Centro	Mostra os vértices que compõe o centro do grafo.
Grafo par	Denota se o grafo é par ou não.

Tabela 1 – Propriedades presentes na tela de propriedades

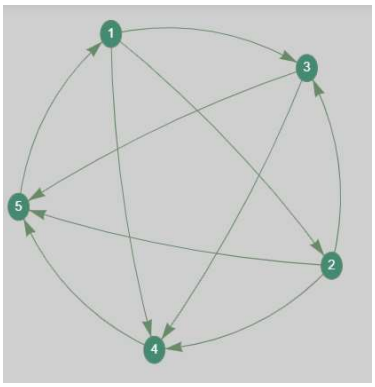
Dadas algumas propriedades do grafo exibidades, existem algumas operações, ou ainda como escrito no menu “Derivações” que nos permitem executar algumas ações sobre o grafo na área de desenho.

5.4 Derivações do Grafo

Outras informações que são melhor visualizadas graficamente foram disponibilizadas através da opção de Derivações do menu lateral. Estes se encaixam melhor em um caixa flutuante e podem ser acessados a qualquer momento sem ter que recarregar a área de desenho ou tela de propriedades, poupando tempo na execução dos algoritmos necessários.

5.4.1 Grafo subjacente

Figura 14 – Grafo completo tamanho 5.

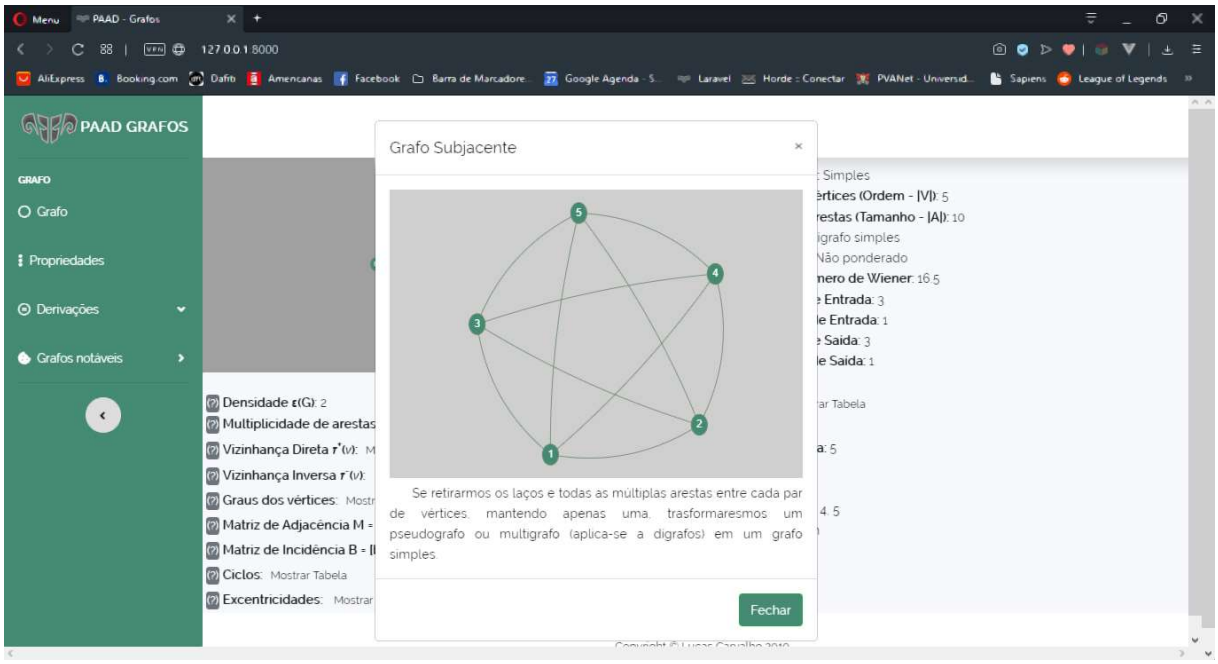


Fonte: Aplicação desenvolvida.

Cria-se um grafo subjacente ao da área de desenho, removendo todas as suas arestas múltiplas entre dois nós adjacentes, removendo os laços sobre nós únicos e removendo sua orientação. A ideia desta ferramenta é transformar o grafo que está na área de desenho em grafo simples se não o for.

A figura 15 mostra como o grafo completo de tamanho 5, orientado, presente na Figura 14, se transforma em um grafo subjacente. Este grafo é completo pois há uma aresta para cada par de vértices.

Figura 15 – Grafo subjacente ao grafo presente na figura 14.



Fonte: Aplicação desenvolvida.

Figura 16 – Sub-dígrafo de espelhamento.

Fonte: Aplicação desenvolvida.

5.4.2 Sub-grafo ou sub-dígrafo de espelhamento

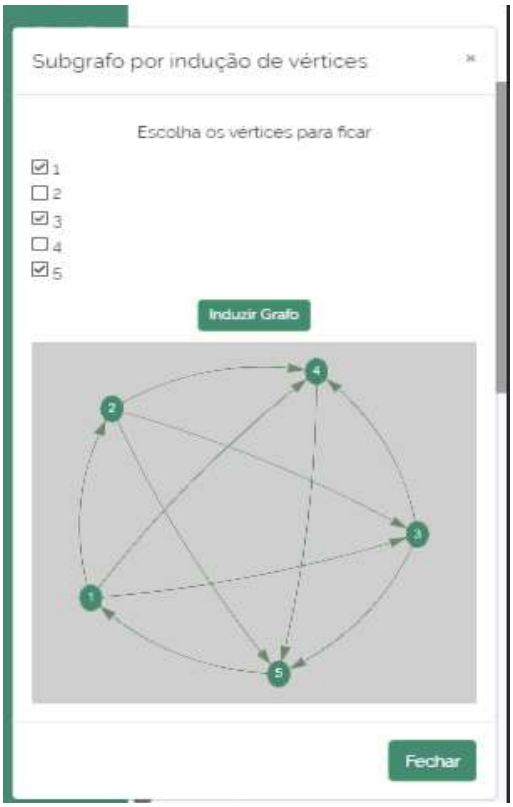
Um sub-grafo ou sub-dígrafo de espelhamento é um grafo aleatório gerado a partir da área de desenho. Ele tem 50% de ter sido removido ou um vértice ou uma aresta do grafo de desenho. Esta derivação contém um botão para que outro grafo aleatório possa ser gerado novamente.

O exemplo de sub-dígrafo de espelhamento na Figura 16, mostra o mesmo grafo completo de tamanho 5 da Figura 14. Contudo, repare que desta vez o grafo gerado pode não ser simples e sua orientação é levada em considereção. Se o grafo é direcionado gera-se um sub-dígrafo, caso contrário um sub-grafo.

5.4.3 Sub-grafo por indução de vértice ou aresta

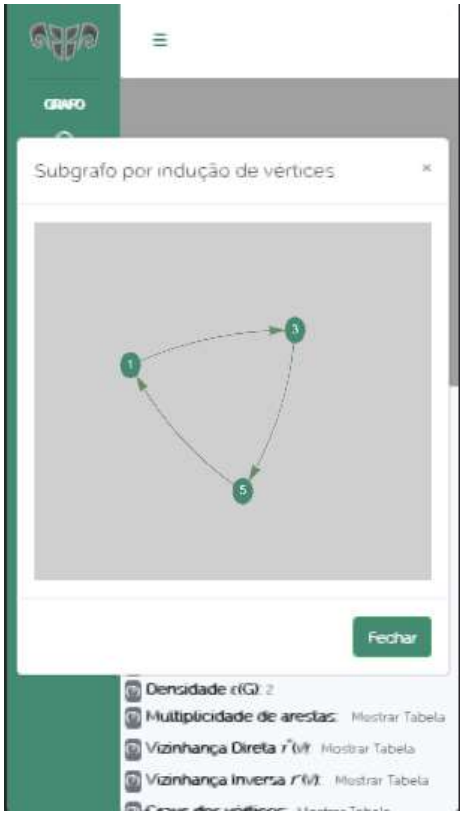
Um sub-grafo gerado por indução de vértices ou arestas é aquele onde o usuário escolhe as arestas e os vértices que vão ficar. Nesta aplicação suas derivações são independentes, ou seja, há possibilidade de fazer uma indução por aresta ou uma por vértice, porém, não ambas.

Figura 18 – Indução por vértices



Fonte: Aplicação desenvolvida.

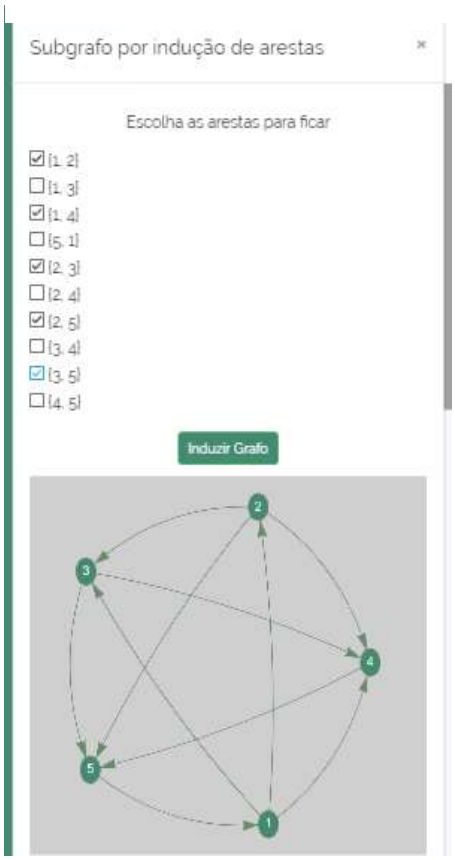
Figura 17 – Grafo induzido da figura 17.



Fonte: Aplicação desenvolvida.

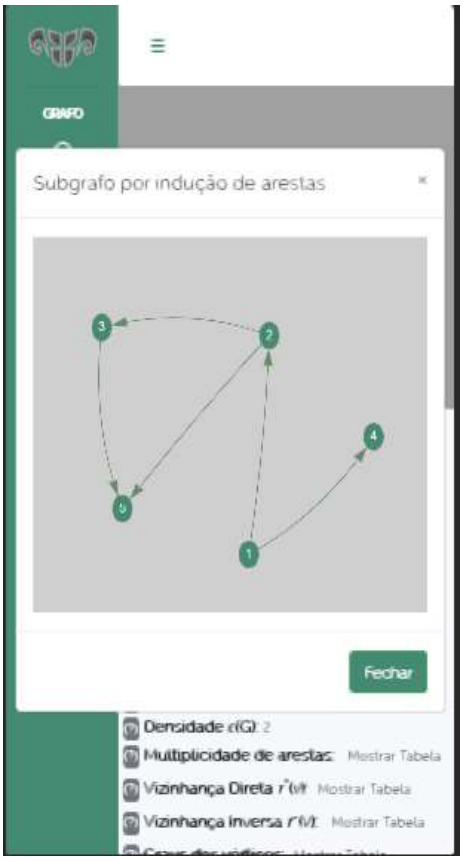
No exemplo das Figuras 18, 19, 20, 21 exemplifica-se como ocorre a indução. Primeiro seleciona-se os elementos que estarão presentes nos sub-grafos gerados e clica/toca no botão de induzir. O resultado será um novo grafo sendo exibido em outra janela flutuante, de acordo com aquilo que o usuário escolher.

Figura 19 – Indução por arestas.



Fonte: Aplicação desenvolvida.

Figura 20 – Grafo induzido por arestas da Figura 19.



Fonte: aplicação desenvolvida

5.4.4 Derivação de passeio

Figura 21 – Derivação de passeio.

Fonte: Aplicação desenvolvida

Há também uma ferramenta desenvolvida para elucidar um passeio sob determinado grafo. A ideia conceito é que o usuário vá descrevendo o caminho “puxando” nós para dentro de uma caixa de derivação. As ações disponíveis para a construção do Passeio são *drag-and-drop* – ou seja, de arrastar e soltar.

Uma vez que o usuário derivou o passeio ele pode ver por quantas vezes passou por cada aresta/vértice, se o passeio é fechado, se é uma cadeia ou trilha, se é um caminho, se é um ciclo ou circuito e o total do seu comprimento. A derivação do passeio pode ser feita utilizando a tela da Figura 21 ao arrastar os quadrados com os vértices até a caixa de derivação.

5.4.5 Derivar Distância

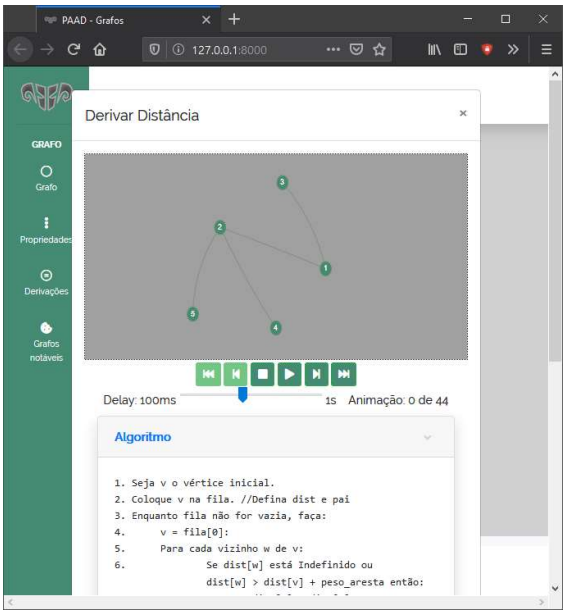
Talvez seja a parte mais interessante de toda esta aplicação: algoritmo de menor distância de um vértice a outro. Esta animação mostra como um algoritmo de busca em profundidade com uma pequena modificação para achar o menor caminho atua.

A interface se parece com um reprodutor: com tocar, parar, próximo, anterior, primeiro e último, além de uma barra para controlar o tempo de espera entre cada passo sendo executado no algoritmo. O tempo de espera de execução pode variar entre 100 milissegundos e 1 segundo. O total de efeitos, ou ainda de animações pode ser visto à direita da barra de velocidade de execução do algoritmo.

O algoritmo em pseudocódigo tem seu texto diferenciado por cores correspondentes àquelas sendo utilizadas na visualização gráfica quando executadas e pode ser minimizado para poder dar campo de visão às estruturas de dados presentes na tela: a fila de acesso, a lista dos pais e a lista das distâncias.

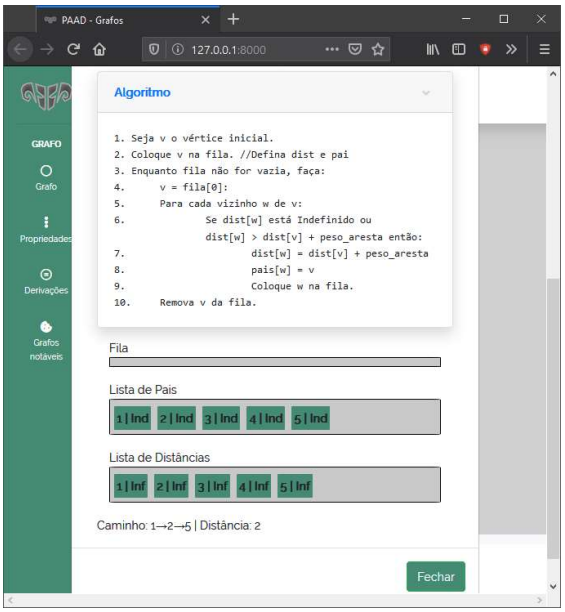
Ao fim de tudo ele mostra o caminho e a menor distância encontrados.

Figura 22 – Tela de animação da execução de um algoritmo.



Fonte: Aplicação desenvolvida.

Figura 23 – Algoritmo e estruturas de dados representadas

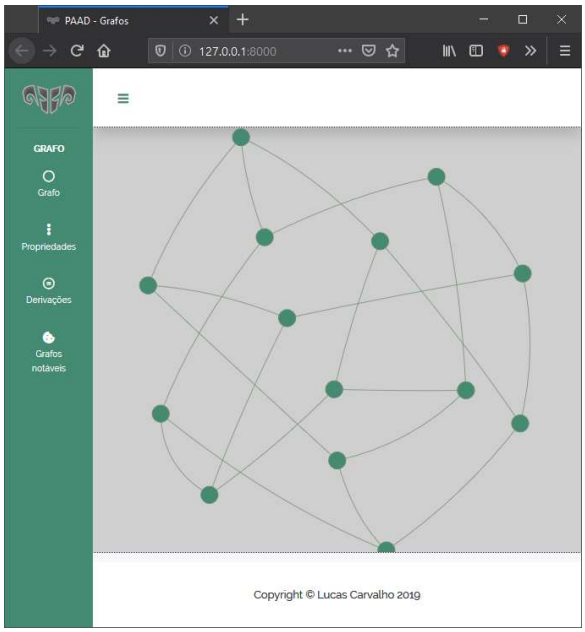


Fonte: Aplicação desenvolvida.

5.5 Grafos notáveis

Na esperança de completar a aplicação com alguns grafos notáveis estudados dentro da disciplina, tem-se a geração de alguns estudados logo no começo: o grafo das 7 pontes de Königsberg, como representado na Figura 2 em 2. Referencial Teórico, e o grafo de Heawood (Weisstein, 2019), como representado na Figura 24.

Figura *24* – Grafo de Heawood.



Fonte: Aplicação desenvolvida

5.6 Resultados referentes à responsividade

Com relação à responsividade e otimização desta aplicação *web* para execução em diversas plataformas diferentes foram feitos testes em ferramentas de *benchmark*, disponíveis *online*, que avaliam a velocidade na qual as operações são feitas em determinada aplicação pertencente à *Internet*.

Em termos computacionais esta aplicação consome poucos recursos. O tamanho total é de somente 1,24 megabytes e demora de 3 a 4 segundos para carregá-la por completo dependendo da velocidade de *download* da *internet* que está sendo utilizada para acessar a página. Abaixo tem-se algumas avaliações de ferramentas conceituadas que mostram como a aplicação foi otimizada para trabalhar com diversos dispositivos:

1. Google PageSpeed Insights (Desktop): 99/100 (Google, 2019)
2. Google PageSpeed Insights (Mobile): 91/100 (Google, 2019)
3. GTmetrix: 97%/100% (GT.net, 2019)

6. Conclusões e trabalhos futuros

Para concluir, pode-se salientar que a aplicação atendeu ao objetivo de ser uma aplicação que auxilie no processo do ensino-aprendizado em Teoria dos Grafos e fornece experiência fluida na execução, seja em dispositivos móveis ou fixos.

As propriedades apresentadas na tela de propriedades contêm informações relevantes acerca do grafo desenhado na área cinza e pode auxiliar estudantes a confirmar a teoria aprendida em sala de aula.

O menu de derivações apresenta ferramentas interativas para que o estudante possa verificar em tempo real o funcionamento de determinados algoritmos sendo executados sobre a estrutura de dados e visualizar operações de alteração sobre o grafo.

Como trabalhos futuros para esta aplicação deve-se acrescentar mais algumas propriedades de um grafo – como planaridade, k-partições e coloração. É importante também acrescentar mais algoritmos para serem executados na plataforma, como, por exemplo: Busca em largura e o algoritmo de Djikstra.

Evoluir na construção da aplicação também é importante e por isso, pode ser que um pouco mais para frente, seja necessário adaptar esta aplicação para *frameworks front-end* JavaScript reativos, como Vue.js ou React.js. A migração da aplicação para alguma dessas duas tecnologias permite uma divisão mais clara do código e fornece organização MVC para as classes front-end.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] PRESSMAN, R. S. Engenharia de Software, 6ª edição, Editora McGraw-Hill, 2006.

[2] MCPHERSON, S. S. Tim Berners-Lee: Inventor of the World Wide Web. Twenty-First Century Books, 2009.

[3] CHAFFEE. A. "What is a web application (or "webapp")?", 2012. Disponível em < http://www.jguru.com/faq/view.jsp?EID=129328 >. Acesso em: 17 de novembro de 2019.

[4] GOLDBARG, M, GOLDBARG, E. Grafos – Conceitos, algoritmos e aplicações. Editora Campus, 2012.

[5] VIEIRA, N. J. Introdução aos fundamentos da computação: linguagens e máquinas. Pionera Thomson Learning, 2006.

[6] EULER, L. “From the Problem of the Seven Bridges of Königsberg. Em CALINGER, Ronald. S. Classics of Mathematics. Pearson, 1994.

[7] GAREY, M. R. and JOHNSON, D. S. Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, 1979.

[8] JUNIOR, H. E. Desenvolvimento de software AD HOC, a Armadilha! Disponível em: < https://www.profissionaisti.com.br/2013/07/desenvolvimento-de-software-ad-hoc-a-armadilha/ >. Acesso em: 17 de novembro de 2019.

[9] AD HOC. In: DICIONÁRIO Michaelis. Disponível em: < http://michaelis.uol.com.br/busca?r=0&f=0&t=0&palavra=ad+hoc >. Acesso em: 17 de novembro de 2019.

[10] SANTOS, A. H. CARVALHO, N. R. Frameworks e seus Benefícios no Desenvolvimento de Software. Revista Pensar Tecnologia, Janeiro de 2015.

[11] ZANETTE, A. Framework x Biblioteca x API. Entenda as diferenças! In: BeCode. Disponível em: < https://becode.com.br/framework-biblioteca-api-entenda-as-diferencas/ >. Acesso em: 17 de novembro de 2019.

[12] THE PHP GROUP. History of PHP. In: PHP. Disponível em < https://www.php.net/manual/en/history.php.php >. Acesso em: 17 de novembro de 2019.

[13] LARAVEL LLC. OTWELL, Taylor. Laravel – The PHP Framework for Web Artisans. Disponível em < https://laravel.com >. Acesso em: 17 de novembro de 2019.

[14] LARAVEL LLC. OTWELL, Taylor. Compiling Assets (Mix). Disponível em < https://laravel.com/docs/6.x/mix#introduction >. Acesso em: 17 de novembro de 2019.

[15] ECMA INTERNATIONAL. Standard ECMA-262, ECMAScript Language Specification. 5.1ª Edição, Junho de 2011. Disponível em < https://web.archive.org/web/20150412040502/http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf# >. Acesso em: 17 de novembro de 2019.

[16] VIS.JS COMMUNITY. Vis.Js Community edition. Disponível em < https://visjs.org >. Acesso em: 17 de novembro de 2019.

[17] VIS.JS COMMUNITY. Network Documentation. Disponível em < https://visjs.github.io/vis-network/docs/network/ >. Acesso em: 17 de novembro de 2019.

[18] THE jQuery FOUNDATION. jQuery – write less do more. Disponível em < https://jquery.com >. Acesso em: 17 de novembro de 2019.

[19] THE jQuery FOUNDATION. jQuery – user interface. Disponível em < https://jqueryui.com >. Acesso em: 17 de novembro de 2019.

[20] FONTICONS, INC. Font Awesome Icon Pack and Toolkit. Disponível em < https://fontawesome.com >. Acesso em: 17 de novembro de 2019.

[21] THE BOOTSTRAP TEAM AND CONTRIBUTORS. Bootstrap. Disponível em < https://getbootstrap.com >. Acesso em: 17 de novembro de 2019.

[22] NPM, INC. About npm. Disponível em < https://www.npmjs.com/about >. Acesso em 17 de novembro de 2019.

[23] GONÇALVES, A. R. O que é SaaS, IaaS e PaaS em Cloud Computing? (Conceitos básicos). Disponível em < https://antonioricardo.org/2013/03/28/o-que-e-saas-iaas-e-paas-em-cloud-computing-conceitos-basicos/ >. Acesso em 17 de novembro de 2019.

[24] URL. In: Significados. Disponível em: < https://www.significados.com.br/url/ >. Acesso em: 25 de novembro de 2019.

[25] DONNELLY CENTRE. In: University of Toronto. Citoscape.js about section. Disponível em < https://js.cytoscape.org/#introduction/about >. Acesso em 25 de novembro de 2019.

[26] SILVEIRA, E. B. de A., SILVA, M. O da. Grafos – Desenvolvimento de um aplicativo educacional para o estudo de Teoria dos Grafos. Universidade Presidente Antônio Carlos. Disponível em < http://www.pucrs.br/ciencias/viali/graduacao/po_2/literatura/grafos/monografias/Silveira_Eduardo.pdf >. Acesso em 25 de novembro de 2019.

[27] Lozada, L. A. P. A-Graph – Uma ferramenta computacional de suporte para o ensino-aprendizado da disciplina de Teoria dos Grafos e seus Algoritmos. Universidade Federal do ABC. 3º Congresso Brasileiro de Informática na Educação, CBIE 2014 – Workshops, WCBIE 2014. Disponível em < https://www.br-ie.org/pub/index.php/wcbie/article/view/3172 >. Acesso em 25 de novembro de 2019.

[28] Sangiorgi, U. B. Rox: Uma ferramenta para o auxílio no aprendizado de Teoria dos Grafos. Faculdade Ruy Barbosa. Disponível em < https://s3.amazonaws.com/academia.edu.documents/3252118/Weibase2004Artigo006.pdf?response-content-disposition=inline%3B%20filename%3DRox_Uma_Ferramenta_Para_O_Auxilio_No_Apr.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWOWYYGZ2Y53UL3A%2F20191125%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20191125T143942Z&X-Amz-

Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=94fb9312879bfea26ab55ac7b14aa4882d6f8410ecef8537534e1dc616aa6c81 >. Acesso em 25 de novembro de 2019.

[29] Arty, D. UX Design e UI Design – Qual a diferença entre eles? In: Chief Of Design. Novembro de 2018. Disponível em < https://www.chiefofdesign.com.br/ux-design-e-ui-design/ >. Acesso em 25 de novembro de 2019.

[30] HEROKU. In: SALESFORCE. About Heroku. Disponível em < https://www.heroku.com/about >. Acesso em 25 de novembro de 2019.

[31] ECMA INTERNATIONAL. Standart ECMA-404, The JSON Data Interchange Standard. 3ª Edição, dezembro de 1999. Disponível em < http://www.json.org >. Acesso em: 25 de novembro de 2019.

[32] WEISSTEIN, E. W. In: MathWorld – A Wolfram Web Resource. Heawood Graph. Disponível em < http://mathworld.wolfram.com/HeawoodGraph.html >. Acesso em 25 de novembro de 2019.

[33] GOOGLE PageSpeed Insights – Desktop. Disponível em < https://developers.google.com/speed/pagespeed/insights/?hl=pt-br&url=https%3A%2F%2Fpaad-grafos.herokuapp.com%2F&tab=desktop >. Acesso em 25 de novembro de 2019.

[34] GOOGLE PageSpeed Insights – Mobile. Disponível em < https://developers.google.com/speed/pagespeed/insights/?hl=pt-br&url=https%3A%2F%2Fpaad-grafos.herokuapp.com%2F&tab=mobile >. Acesso em 25 de novembro de 2019.

[35] GT.net GTmetrix – Latest Performance Report for: https://paad-grafos.herokuapp.com. Disponível em < https://gtmetrix.com/reports/paad-grafos.herokuapp.com/29jz6bcZ >. Acesso em 25 de novembro de 2019.

[36] YOU, E. Vue.Js – The Progressive JavaScript Framework. Disponível em < https://vuejs.org >. Acesso em 25 de novembro de 2019.

[37] FACEBOOK Inc. – Facebook Open Source – React – Uma biblioteca JavaScript para cirar interfacaes de usuário. Disponível em < https://pt-br.reactjs.org >. Acesso em 25 de novembro de 2019.

[38] HALIM, S. VisuAlgo: Visualização de Estruturas de dados e Algoritmos através de animação. 2011. Disponível em < https://visualgo.net/pt >. Acesso em 29 de novembro de 2019.