UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

GABRIEL ALVES DAS NEVES

ARCHON-VIZ : PLATAFORMA PARA EXPLORAÇÃO E MANIPULAÇÃO DE DADOS HIERÁRQUICOS

FORTALEZA - CE

2021

GABRIEL ALVES DAS NEVES

ARCHON-VIZ : PLATAFORMA PARA EXPLORAÇÃO E MANIPULAÇÃO DE DADOS HIERÁRQUICOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientadora: Profa. Dra. Emanuele Marques Rodrigues Santos

FORTALEZA - CE

2021

GABRIEL ALVES DAS NEVES

ARCHON-VIZ : PLATAFORMA PARA EXPLORAÇÃO E MANIPULAÇÃO DE DADOS HIERÁRQUICOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em:

BANCA EXAMINADORA

Profa. Dra. Emanuele Marques Rodrigues Santos (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Antonio José Melo Leite Júnior
Universidade Federal do Ceará (UFC)

Prof. Dr. George Allan Menezes Gomes
Universidade Federal do Ceará (UFC)

À minha família, por terem abdicado seu tempo e seus projetos pessoais para que eu tivesse a oportunidade de estudar e de ter uma boa formação profissional e pessoal. Agradeço pelo apoio e pela motivação. Eu dedico este título a vocês. Sem vocês, nada disso seria possível.

AGRADECIMENTOS

Aos meus pais, Francisco Carlos Nascimento das Neves e Ana Karla Silva Alves das Neves, que formaram o homem que sou hoje, que abriram mão dos seus sonhos para que eu pudesse realizar os meus e que sempre me ensinam e me apoiam. Aos meus irmãos, Yasmim Alves das Neves e Rafael Alves das Neves, que me inspiram com suas lutas, vocês são meus maiores tesouros.

A todos que viram esse sonho nascer, crescer e se tornar realidade, em especial: Thiago Beppe, Erick Barroso, Mateus Cruz, Artur Bruno, Mariana de Lima, Rebeca Sousa, Arthur Cordeiro, Gabriel Lima, Bianca Barbazán, Ricardo Pietro, Davi Barreto, Eduardo Bezerra, Paulo Lui, Velasco, Pedro Monteiro, Guilherme Alves, Hygison Brandão, Alana Queiroz, Adller Lucena, Pedro Benevides, Lucas Noleto, entre outros. Que durante a minha jornada presenciaram minhas conquistas e minhas derrotas, que dividiram lágrimas e sorrisos; mesmo se ficarmos distantes, há uma morada pertencente a cada um em meu coração.

À minha professora orientadora Emanuele Marques Rodrigues Santos, por toda sua paciência, atenção e cuidado. Por ser um exemplo de profissionalismo para mim, e pelas horas despendidas nas reuniões. Muito obrigado por acreditar em mim.

A todos que modelaram minha vida profissional por meio de experiências extra curriculares, em especial: toda equipe do LESC, destacando Prof. Jarbas Silveira e Prof Jardel Silveira, Felipe Gaspar, Gabriel Rocha, Pedro Falcão, Leonel Maia, Adahil Muniz, Madson Ivens, Michelly e Rodrigo; à GTi Jr, à AIESEC em Fayoum e, especialmente, à Virginie Fresse, pela oportunidade de aprendizado no *Laboratoire Hubert Curien*.

Aos meus heróis Wilson Fernandes do Nascimento e Ubirajara Gonçalves das Neves. Cresci escutando e me inspirando em suas histórias e tenho certeza que, observando da eternidade, se orgulham de ver como suas famílias estão indo longe. Vou continuar me esforçando para levar nosso nome com honra.

Agradeço a todos os meus mestres professores por terem me transmitido conhecimento técnico, conselhos, experiências e inspirações. Por todas as horas dedicadas para que eu aprendesse e, acima de tudo, por terem me ensinado como aprender.

“Se você pode sonhá-lo, você pode fazê-lo.”

(Walt Disney)

RESUMO

Devido à sua vasta aplicabilidade, os serviços web têm ganhado cada vez mais popularidade. Isso significa que mais dados hierárquicos são gerados, principalmente nos formatos JSON e XML. Ainda assim, muitos usuários continuam utilizando formatos tabulares, como planilhas Excel e arquivos CSV, mesmo quando os dados nesses arquivos também são dados hierárquicos. Sem uma ferramenta específica, visualizar e compreender por completo a hierarquia que envolve os dados torna-se um trabalho difícil, necessitando até mesmo de habilidades com programação. Visando atender aos usuários que lidam com dados hierárquicos em formatos tabulares, neste trabalho, é apresentada uma ferramenta web que permite carregar, manipular e compreender dados hierárquicos de arquivos tabulares. A ferramenta foi desenvolvida utilizando tecnologias web e princípios de visualização de dados para representar os dados hierárquicos em uma *Radial Tree*. A ferramenta e suas funcionalidades são demonstradas neste trabalho por meio de um estudo de caso de imagens de câncer de pele, onde os usuários podem investigar diferentes hierarquias, alterar dados e exportá-los como arquivos JSON ou mantê-los em formato tabular.

Palavras-chave: Hierarquia. Visualização de dados. Radial Tree.

ABSTRACT

Due to its large applicability, web services have been gaining more popularity. This means that more hierarchical data are generated with hierarchical, mainly in JSON or XML formatted files. However, many users still use tabular files, such as Excel spreadsheets and CSV files. Without a specific tool, visualizing and understanding fully the hierarchy surrounding the data becomes a hard task, sometimes requiring programming skills. In order to support the users leading with hierarchical data in tabular format, in this work, we present a web-based tool that allows users to load, manipulate and understand hierarchical data from tabular data. The tool was developed using web technologies and data visualization principles to represent hierarchical data through a radial tree. The tool and its functionalities are demonstrated through a case study of skin cancer images, where users can investigate different hierarchies, manipulate and export data in JSON format or keep them as tabular data.

Keywords: Hierarchy. Data visualization. Radial Tree.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Imersos na era do *Big Data*, diariamente geramos uma grande quantidade de dados que serão utilizados como matéria prima para diversas áreas do conhecimento. Uma grande parcela desses dados são gerados na internet e estão organizados em uma estrutura hierárquica, como o formato JSON. No entanto, as estruturas mais familiares ainda são tabulares, mas essas não são apropriadas para manipular dados hierárquicos. Portanto, compreender a organização dos dados se torna uma tarefa difícil e restrita àqueles que possuem habilidades em programação.

Quando os dados estão organizados em estruturas hierárquicas, como no formato JSON, é difícil perceber como eles se relacionam, como podem ser agrupados e, muito menos como manipulá-los. A Figura 1 mostra um exemplo de como é estruturado um arquivo JSON.

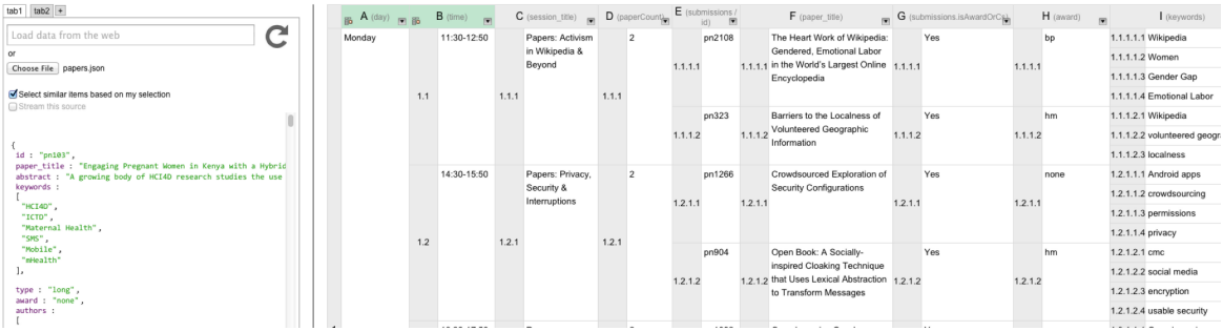
Figura 1 – Exemplo demonstrativo da estrutura de um arquivo JSON onde podemos ver que não é tão intuitivo perceber uma hierarquia entre os dados.

Fonte: Elaborada pelo autor.

O trabalho de Chang e Myers (2016) propõe uma abordagem para lidar com hie-

rarquias diretamente no formato de planilha. Na Figura 2, é exibida a interface do trabalho de Chang e Myers (2016) que propõe uma abordagem para lidar com hierarquias diretamente no formato de planilha. A figura mostra a importação de um arquivo JSON (painel à esquerda) onde, arrastando os elementos, gera-se uma visualização tabular para explorar a hierarquia. No entanto, a representação visual dos dados pode ser essencial para gerar *insights* sobre o conjunto de dados (KIRK, 2019), mas não está presente na solução de Chang e Myers (2016). Além disso, a abordagem tabular tradicional já não é suficiente e nem todas as abordagens visuais serão efetivas. Logo, fica evidente a necessidade de ferramentas que auxiliem na visualização e no manuseio desse tipo de estrutura.

Figura 2 – Uma captura de tela da ferramenta desenvolvida por Chang mostrando a interface em um cenário de uso.



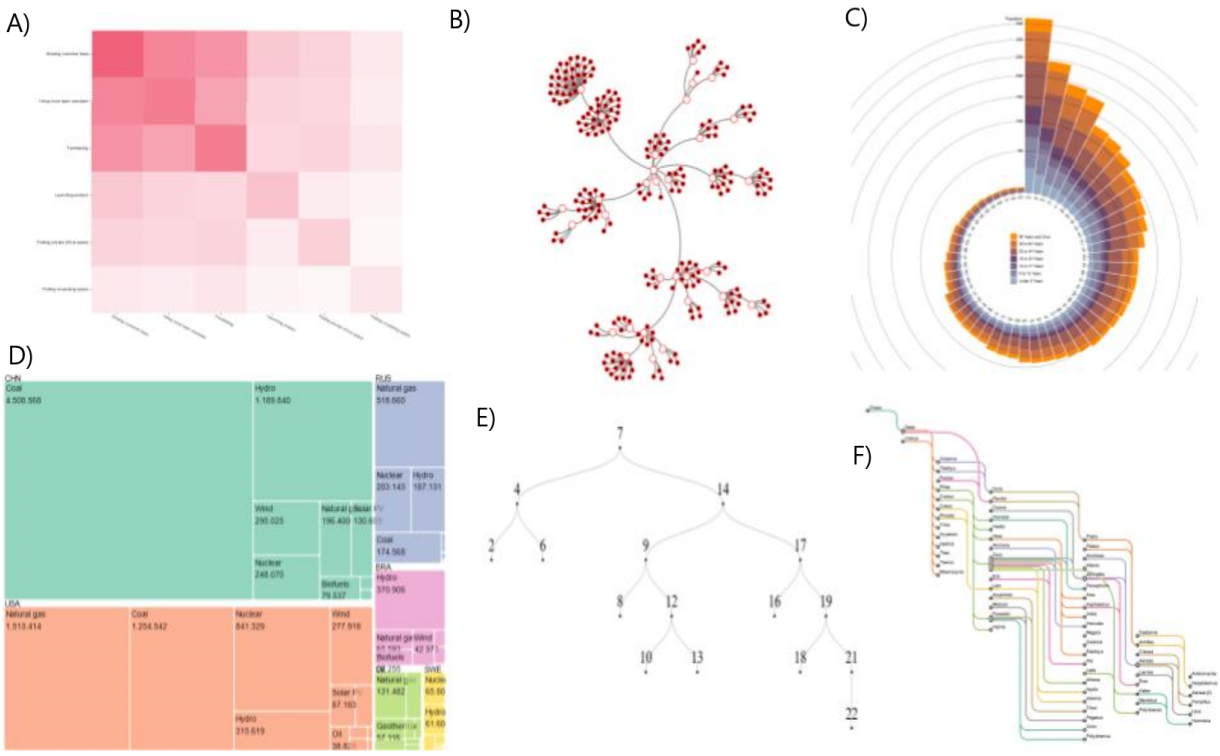
Fonte: Chang e Myers (2016).

Por outro lado, a visualização é uma área dedicada à geração de imagens que auxiliam seus usuários na compreensão de dados e processos (MUNZNER, 2014). Utilizando-se de elementos visuais (e.g.: nós e arestas), torna-se mais acessível a identificação de relações entre os dados, sendo, portanto, uma ferramenta poderosa para analisar dados com organizações complexas (WANG *et al.*, 2015). No mercado há diversas ferramentas e bibliotecas de visualização de dados, como Gephi e outras que apresentaremos no decorrer deste trabalho, contudo, essas ferramentas exigem conhecimentos de programação ou formatos específicos do arquivo de entrada.

Existem diferentes técnicas para visualizar dados hierárquicos. Por exemplo, há gráficos que representam dados por formas geométricas e suas interações por ligações entre eles como na Figura 3-B, E e F que foram geradas por diferentes algoritmos. Outros já utilizam áreas para representar a ordem da hierarquia como na Figura 3-A, C e D.

Cada uma dessas técnicas possui vantagens e desvantagens, dependendo do objetivo, do público alvo e do volume de dados.

Figura 3 – Compilado de técnicas de visualização. A) Matriz de Adjacência. B) *Force Directed*. C) Barras Radiais Empilhadas. D)*Treemap*. E) Nó e Aresta Triangular Vertical. F) Árvore Emaranhada.



Fonte: Adaptada de Bostock (2020).

Após conversas com patologistas que possuem bancos de casos clínicos armazenados em estruturas tabulares para serem analisados, que são usuários potenciais da aplicação, durante o planejamento do projeto, foram identificados os seguintes requisitos relacionados ao problema a ser resolvido:

- Requisito 1: Os dados explorados pela aplicação são de pouco volume e estão em um arquivo de entrada em formato tabular (ex.: .csv) fornecido pelo próprio usuário;
- Requisito 2: A aplicação deve permitir que os usuários possam modificar os valores dos dados estruturados hierarquicamente tanto como reorganizar a hierarquia desses dados;
- Requisito 3: O usuário não precisa possuir habilidade com programação;
- Requisito 4: A aplicação deve permitir exportar os dados modificados nos formatos *.csv* ou *.json*;
- Requisito 5: Os dados devem ser representados de forma visual, facilitando o entendimento da hierarquia de dados.

Portanto, visando atender os requisitos acima, neste trabalho, propomos o Archon-

viz, uma ferramenta web que permite carregar, manipular e explorar dados hierárquicos de arquivos tabulares. A origem do nome da ferramenta vem do termo grego Archon, que dá origem a Hierarquia, Monarca, entre outros, e remete a governar de um cargo mais alto, passando a ideia do topo da hierarquia.

Disponibilizada no GitHub (2021), no repositório <https://github.com/gabrieldasneves/Archon-viz>, o Archon-viz foi desenvolvido utilizando tecnologias web e princípios de visualização de dados. Foi escolhida uma *Radial tree* para representar os dados hierarquicamente e, com esta ferramenta, os usuários podem explorar diferentes hierarquias, alterar dados e exportá-los como arquivos JSON ou CSV.

O restante deste trabalho está organizado da seguinte forma. O Capítulo 2 contém o referencial teórico necessário para compreender os conceitos utilizados ao longo do trabalho. O Capítulo 3 discute as soluções existentes, suas vantagens e desvantagens e, no final, apresenta um comparativo entre essas abordagens e a solução proposta. No Capítulo 4, será apresentada a ferramenta Archon-viz, que permite a exploração, visualização e edição de dados hierárquicos. Será explicada a estratégia para elaboração da ferramenta desde a interface até técnicas de programação utilizadas. No Capítulo 5, é ilustrada a utilização do Archon-viz através de um estudo de caso para lidar com um conjunto de dados de imagens de câncer de pele. Por fim, no Capítulo 6, conclui-se este documento, enumerando-se possibilidades de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Uma boa visualização não é apenas um gráfico bonito ou cheio de informações (CAIRO, 2016). Visualizações eficientes contam uma história a partir dos dados usando apenas as informações relevantes (CLEVELAND, 1985). Isso permitirá que conclusões sejam alcançadas e decisões sejam tomadas mais precisamente.

Para construir uma visualização bem sucedida, conceitos de diversas áreas como a Computação, a Psicologia e a Estatística são utilizados. Desde o pré-processamento dos dados, passando pela escolha de matizes e intensidades de cor, e até mesmo os regionalismos são minuciosamente trabalhados.

Muitos aplicativos, como Tableau (2021), foram desenvolvidos para auxiliar na comunicação usuário-dado, criando painéis com visualizações interativas a partir de múltiplas bases de dados. Da mesma forma, várias bibliotecas que auxiliam no desenvolvimento de visualizações foram criadas em diversas linguagens de programação. Além disso, há visualizações que são mais eficientes para certos tipos de dados, dependendo, por exemplo, da relação entre eles ou do algoritmo que foi utilizado em seu pré-processamento (ARCE-OROZCO *et al.*, 2017).

Este capítulo se dedica a abordar, de forma resumida, os conceitos que foram utilizados no desenvolvimento do projeto sendo, portanto, essenciais para compreensão dos capítulos seguintes. Aqui discutiremos a Visualização de Hierarquias (Seção 2.1) que foi dividida em dois grandes grupos: Diagramas Nós e Arestas (Seção 2.1.1), e Outros (Seção 2.1.2).

2.1 Visualizações de Hierarquias

Segundo o dicionário Houaiss (2001), uma hierarquia é uma "organização fundada sobre uma ordem de prioridade entre os elementos de um conjunto ou sobre relações de subordinação entre os membros de um grupo." Uma hierarquia é uma especialização de uma estrutura mais genérica chamada de rede ou grafo. Em uma rede, um elemento da hierarquia normalmente é chamado de nó ou vértice, e uma relação entre dois nós é chamada de vínculo ou aresta (WILLS, 2009). Em se tratando de conjuntos de dados hierárquicos, os nós são vinculados em relacionamentos pai-filho, cada um desses nós filhos tem apenas um pai, mas cada pai pode ter vários filhos. O primeiro nó, no topo da hierarquia, é chamado de nó raiz. Existem diferentes técnicas de visualização de dados hierárquicos. No contexto deste trabalho, as técnicas de

visualização de dados hierárquicos serão separadas em dois grupos: o grupo Digramas Nós e Arestas, e outros.

2.1.1 Diagramas Nós e Arestas

Figura 4 – Exemplo demonstrativo do layout *Force Directed* em que os dados são músicas que têm relações de gênero entre si.

Fonte: Yau (2016).

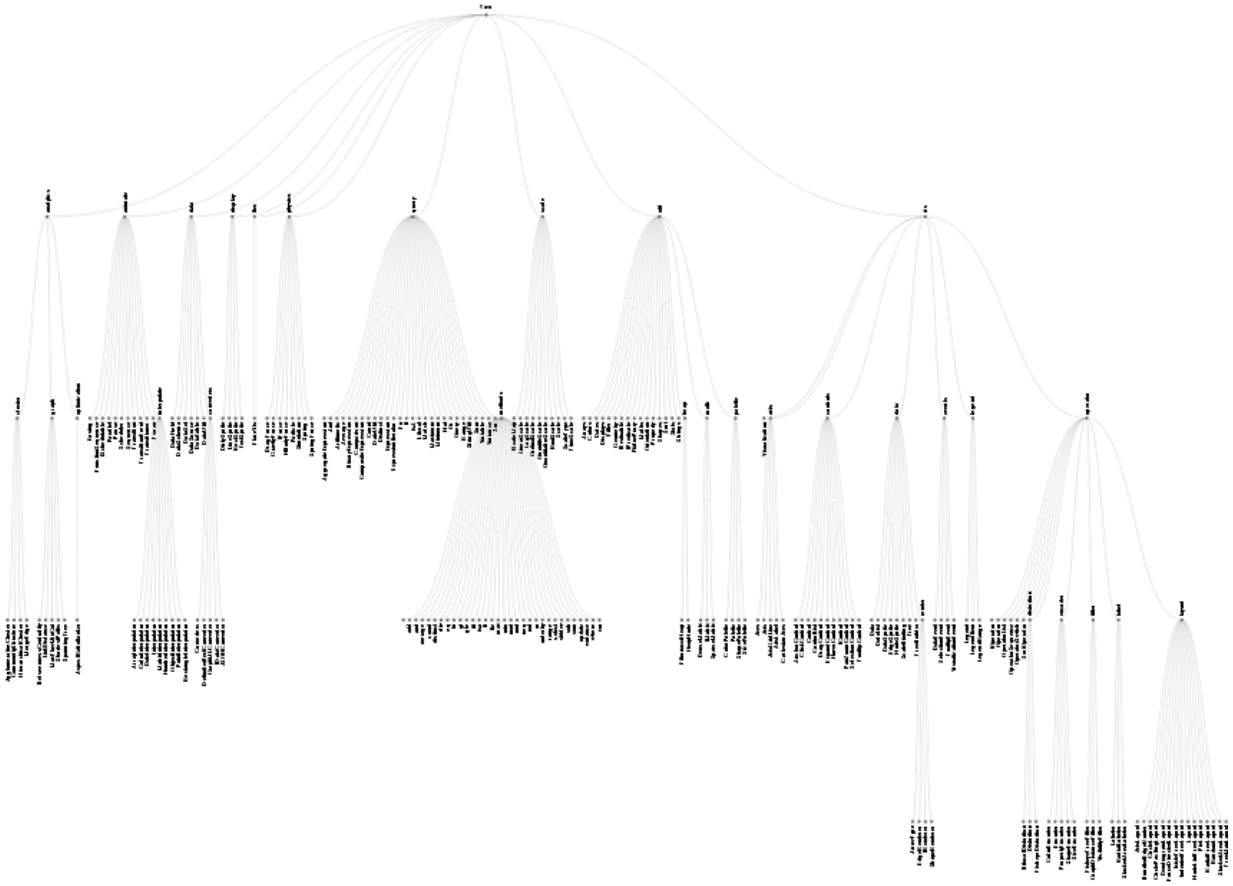
Nesse tipo de representação, os nós são representados como primitivas geométricas (e.g.: um círculo) e as arestas como linhas. É uma abordagem usada para analisar a topologia da hierarquia. Os objetos, relacionados de forma hierárquica, são dispostos como uma série de nós conectados por segmentos, chamados de arestas, com o mínimo de cruzamento entre si.

Existem diversos algoritmos para fazer o layout dos diagramas nós e arestas, dentre eles, os mais conhecidos são: triangular vertical ou horizontal, retangular vertical ou horizontal, *force-directed* e layout radial (*Radial Tree*).

No layout triangular vertical (Figura 5), os nós são posicionados em linhas equidistantes entre si. A localização horizontal do nó raiz de cada conjunto pai-filhos é o centro do espaço horizontal do layout desse conjunto (BURCH *et al.*, 2011). Dessa forma, as camadas de hierarquia começam, por exemplo, no topo do layout até embaixo com a mesma profundidade entre nós do mesmo nível de hierarquia.

Já no layout triangular horizontal, exibido na Figura 6, utiliza-se o mesmo algoritmo de construção, porém em vez de dispor os nós em linhas horizontais, faz-se em linhas verticais.

Figura 5 – Exemplo demonstrativo do layout triangular vertical em que ilustra-se a dificuldade visual gerada quando o conjunto de dados é grande.



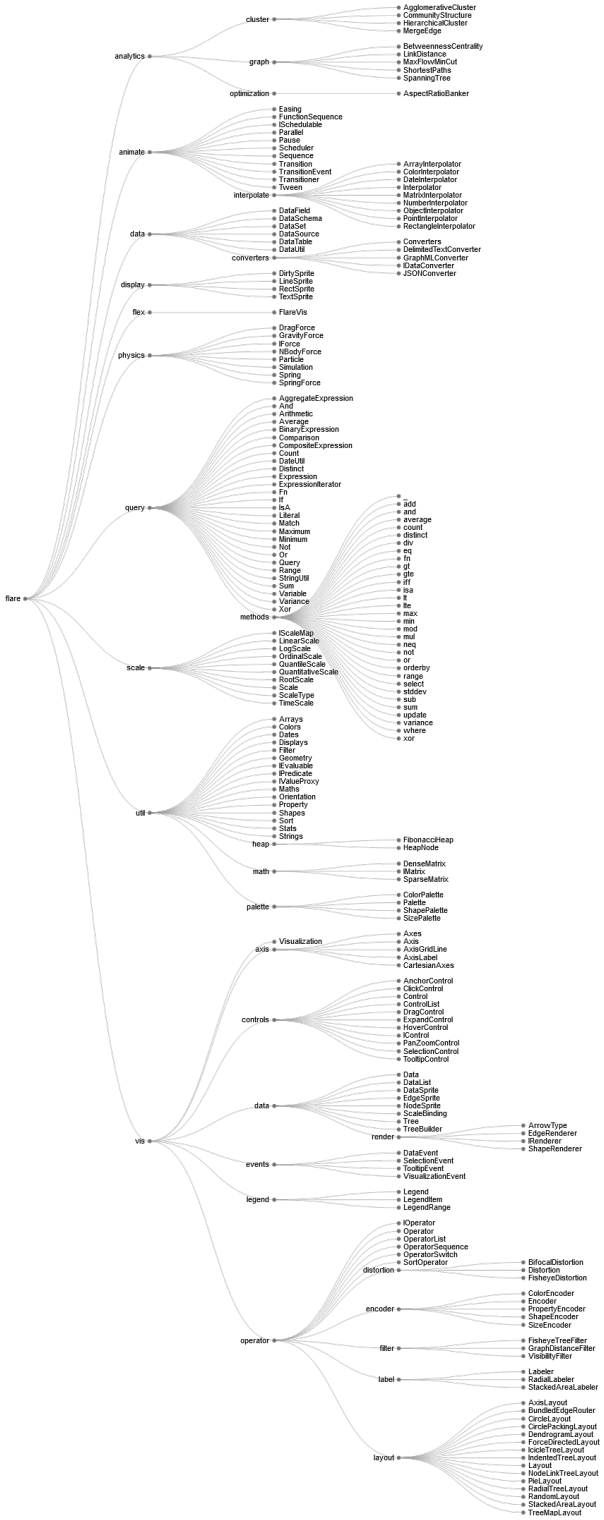
Fonte: Miles (2021).

Na categoria *Force-directed*, os algoritmos utilizados atribuem forças a nós e arestas, simulando um sistema físico. A simulação dessas forças nos conjuntos nós-arestas é utilizada para posicionar os elementos na área de visualização (HUA *et al.*, 2012). Existem variações na forma como é implementada a simulação, mas normalmente, nós se repelem entre si, enquanto arestas atuam como molas que atraem os nós em suas extremidades (MUNZNER, 2014). A Figura 4 ilustra um exemplo de layout *Force-directed*.

Esse tipo de construção permite a implementação de interações, reposicionando os nós quando há alguma alteração de posição pelo usuário, além de ser uma visualização intuitiva, em que os agrupamentos são notados facilmente.

Uma outra alternativa de layout para dados hierárquicos é o layout radial, mais comumente representado pelas *Radial-Trees*. Nesta representação, os dados são dispostos de forma circular, onde o centro do círculo é a origem, e o ângulo e o raio são usados no lugar das coordenadas x e y para o posicionamento dos elementos. Normalmente nesta representação, a raiz da hierarquia fica localizada na origem e os demais níveis são posicionados circularmente,

Figura 6 – Exemplo demonstrativo do layout triangular horizontal.



Fonte: Miles (2021).

afastando-se da origem. Nós em um mesmo nível da hierarquia estão localizados à mesma distância da raiz (MUNZNER, 2014).

Em alguns cenários, as representações lineares são mais apropriadas do que as representações radiais (BURCH; WEISKOPF, 2014). Por exemplo, é mais fácil seguir um

caminho por meio de um diagrama de árvore linear (vertical ou horizontal) porque as linhas horizontais e verticais guiam o usuário nessa tarefa. No entanto, conforme o volume de dados e os níveis de hierarquia aumentam, essa abordagem passa a ocupar bastante espaço de tela, dificultando a interação do usuário com a visualização. É aqui onde as *Radial-Trees* agregam valor pois o número de nós aumenta exponencialmente com a distância do primeiro nó, enquanto o diâmetro da circunferência aumenta linearmente. A Figura 7 ilustra essa visualização, que contém os mesmos dados apresentados no layout triangular vertical da Figura 5 e no layout triangular horizontal da Figura 6. Comparando-as podemos ver o aproveitamento de espaço obtido na abordagem radial.

2.1.2 Outros

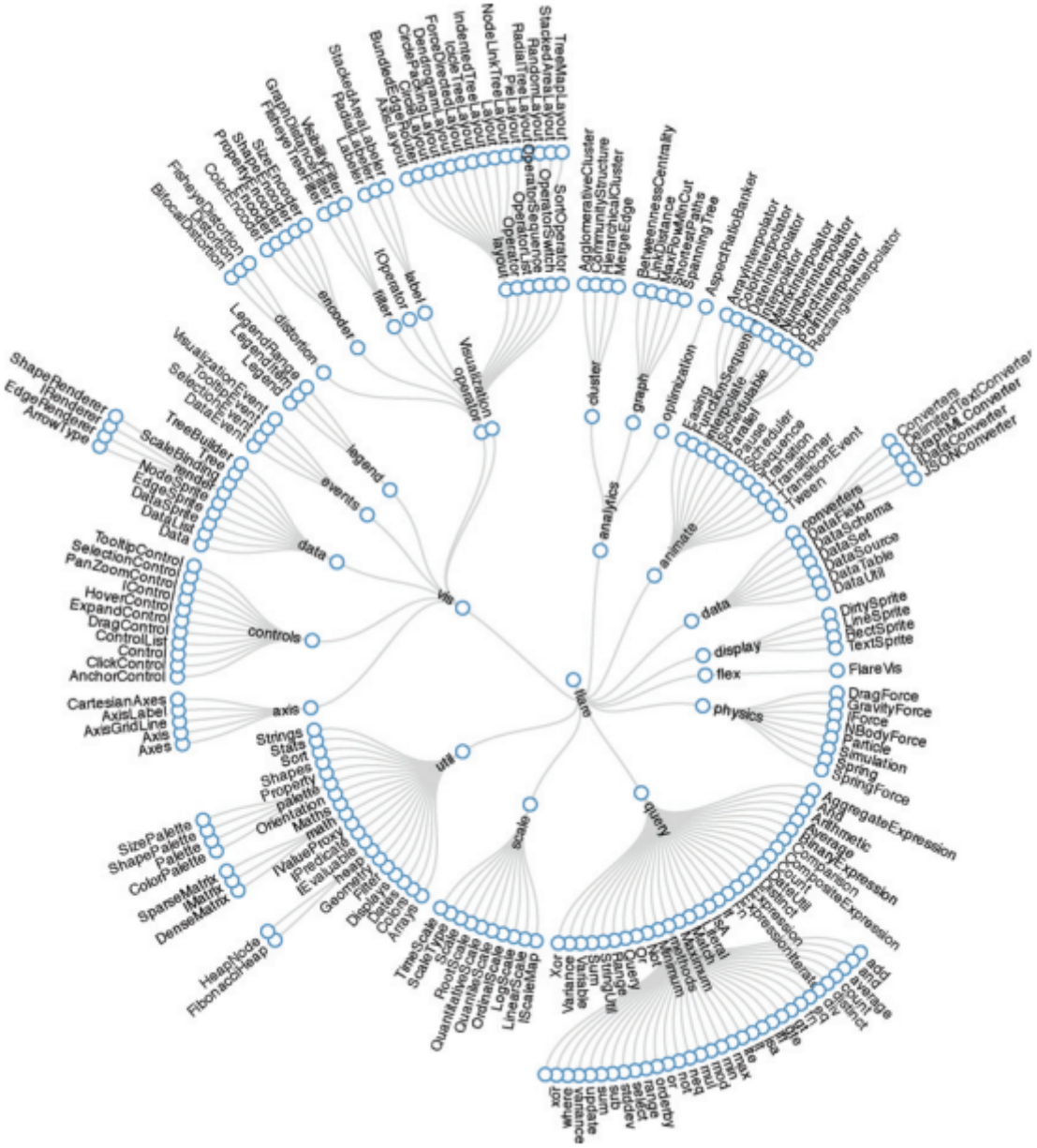
Além dos diagramas nós e arestas, existem outras técnicas de visualização de hierarquias. Podem ser utilizadas matrizes de adjacência e técnicas baseadas no preenchimento do espaço.

No caso de matrizes de adjacência, todos os elementos da hierarquia são posicionados ao longo dos lados horizontal e vertical de uma região quadrada e os vínculos entre dois nós são indicados por uma marcação na célula da matriz que é a interseção entre a linha e a coluna dos elementos. Informações adicionais da hierarquia podem ser codificadas usando diferentes marcas e cores nas células da matriz. A Figura 8 mostra a visão de matriz de adjacência e o diagrama nós e arestas para a mesma hierarquia.

A maior vantagem das matrizes de adjacência é possuir uma alta densidade visual de nós e arestas, ou seja é possível visualizar muitos nós e muitas arestas em um pequeno espaço de tela. Teoricamente, seria possível usar até o tamanho de um pixel para representar uma célula da matriz e, usando técnicas de reordenação de nós, é possível detectar facilmente agrupamentos dos nós (GHONIEM *et al.*, 2005). No caso particular da Figura 8, as colunas e linhas foram reordenadas para facilitar a visualização das comunidades dos personagens. No entanto, como desvantagens podemos citar: é uma técnica não familiar para a maioria dos usuários e não possui suporte para investigação da estrutura topológica da hierarquia, sendo difícil a identificação de caminhos entre nós. Essas desvantagens foram o fator predominante para a não adoção dessa técnica na ferramenta proposta.

Dentre as técnicas de preenchimento de espaço, a mais comum é o *Treemap* (JOHNSON; SHNEIDERMAN, 1991). O *treemap* é uma técnica de visualização que mapeia as

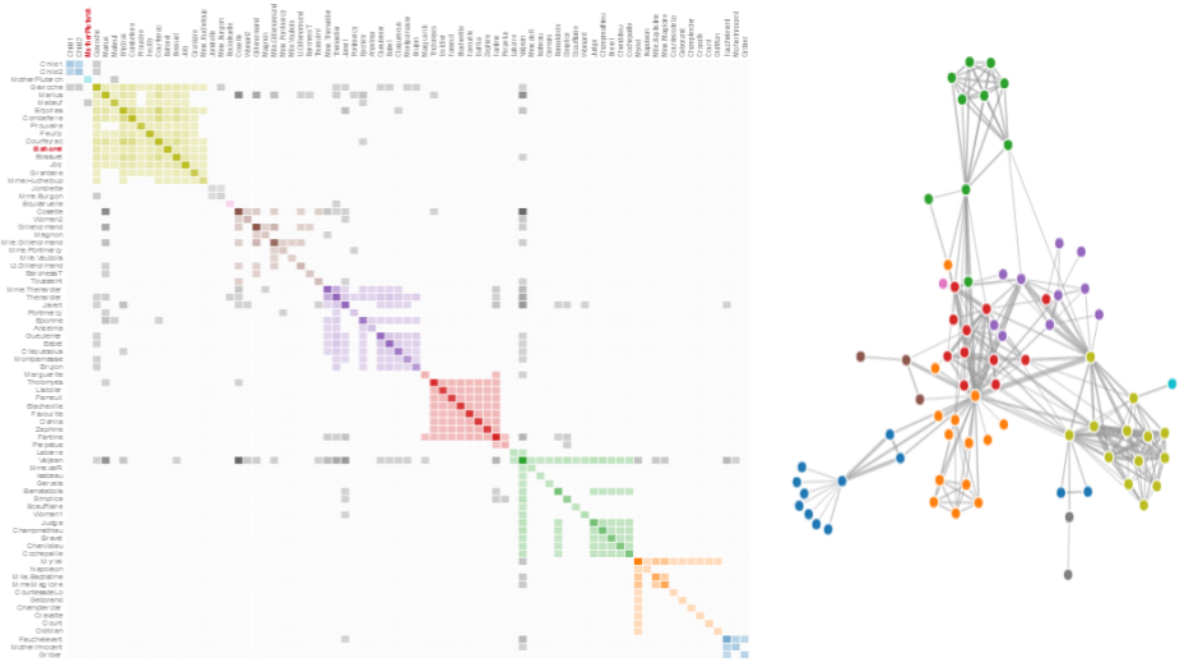
Figura 7 – Exemplo demonstrativo do layout *Radial-Tree* onde usamos o mesmo conjunto de dados dos dois últimos layouts e obtemos um maior aproveitamento de espaço.



Fonte: Bostock (2021).

informações em retângulos de forma que preencham totalmente o espaço dedicado à visualização sendo, portanto, uma boa opção quando temos uma grande quantidade de dados hierárquicos e um espaço visual limitado. Nessa visualização, espaço visual é recursivamente preenchido por retângulos que são dimensionados e ordenados de acordo com alguma variável quantitativa (proporção de exportações no caso da Figura 9). Também podemos usar cores para diferenciar as sub-hierarquias ou ainda mapear uma outra variável quantitativa. Dessa forma, obtemos retângulos dentro de retângulos, como mostra a Figura 9.

Figura 8 – Matriz de adjacências e o diagrama de nós e arestas correspondente, ambos mostrando a interação entre personagens da obra *Les Misérables*.



Fonte: Staats (2021).

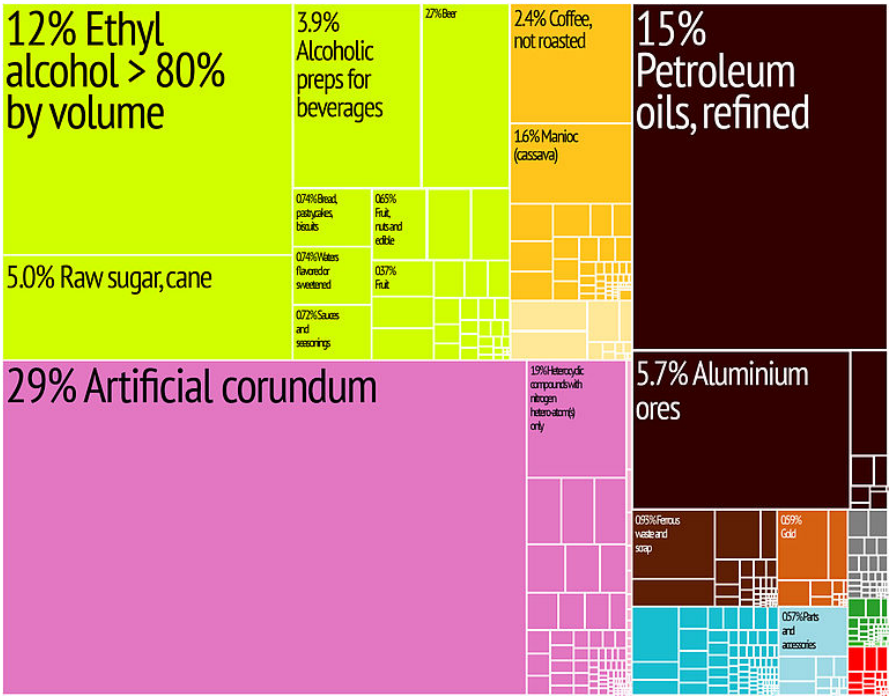
Outra variante de técnicas de preenchimento de espaço são os círculos aninhados (Figura 10). No entanto, *treemaps* retangulares são mais eficientes no aproveitamento do espaço (note os espaços vazios no interior dos círculos da Figura 10). Apesar dessa eficiência, com uma grande quantidade de dados, ou com hierarquias profundas, chegaremos a um ponto onde a visibilidade dos dados é comprometida, o que poderia ser compensado com a implementação técnicas de interação, como zoom, por exemplo.

2.2 Considerações Finais

Para este trabalho, foi adotada a técnica de diagrama radial para representar visualmente os dados hierárquicos. Pensando em casos com muita profundidade em hierarquia, podemos investir em interatividade (e.g.: zoom) para melhorar a experiência dos usuários (YEE *et al.*, 2001).

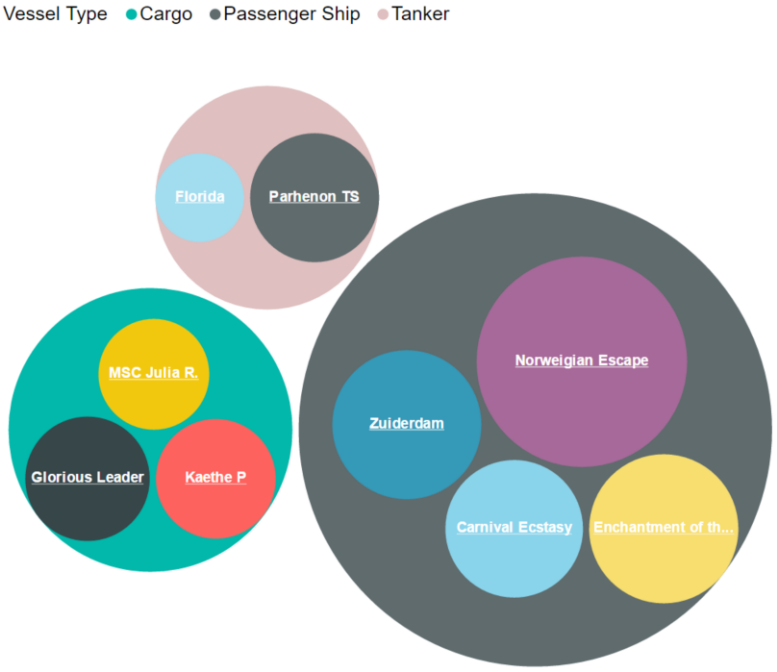
Esta abordagem visual consegue capturar a atenção do usuário pois, instintivamente, somos atraímos por formas circulares (GAMBINO, 2013; LIMA, 2017). Além disso, devido à sua simetria, nesta abordagem, a informação é apresentada de forma mais agradável.

Figura 9 – Visualização de exportações da Jamaica em 2012 por setor utilizando um *treemap*.



Fonte: Haussmann Cesar Hidalgo (2012).

Figura 10 – Diferentes embarcações oceânicas que foram agrupadas por seu tipo de embarcação representadas em um layout de círculos aninhados.



Fonte: Devinknight (2018).

3 TRABALHOS RELACIONADOS

Devido aos benefícios que uma boa visualização de dados pode trazer para um projeto, muitas ferramentas, desde aplicações completas a bibliotecas, foram desenvolvidas em diversas linguagens a fim de simplificar os dados para inspirar decisões. Neste capítulo, são discutidos os trabalhos que estão relacionados à solução proposta. Na Seção 3.1, são discutidas as bibliotecas gráficas, na Seção 3.2, são discutidas as aplicações voltadas para visualização de hierarquias, e por fim, na Seção 3.3, é apresentado um comparativo geral dos trabalhos relacionados.

3.1 Bibliotecas

3.1.1 Biblioteca D3.js

Desenvolvida em JavaScript, a biblioteca D3 (BOSTOCK *et al.*, 2011), que é utilizada neste projeto, oferta poderosos componentes de visualização e uma abordagem orientada a dados, consistindo em uma das mais eficientes bibliotecas para manipulação, visualização e interação de grandes conjuntos de informações (GASPAR *et al.*, 2014).

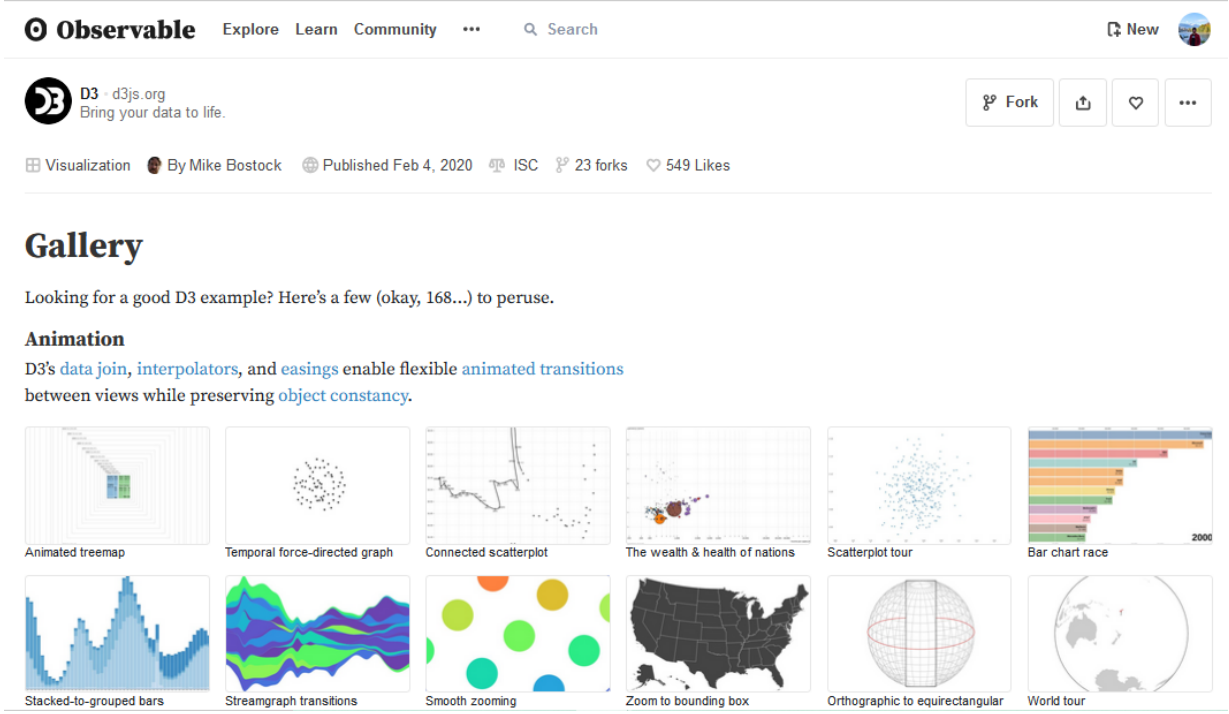
Em poucas palavras, a D3 atua convertendo dados em elementos visuais para aplicações Web. Isso implica em um alto grau de interatividade e abre uma gama de possibilidades de interação, pois os gráficos podem interagir entre si e com outros elementos em HTML.

A relevância da D3 se deve à sua flexibilidade. Funcionando perfeitamente com as tecnologias da web existentes e podendo manipular qualquer parte do DOM (*Document Object Model*), utilizá-la é tão flexível quanto usar o clássico grupo de tecnologias da web (HTML, CSS, SVG).

Uma das mais notáveis vantagens da D3 é a grande comunidade colaborativa que existe em torno da biblioteca. Isso gera uma grande quantidade de exemplos que auxiliam em desenvolvimentos futuros. Uma demonstração dessa comunidade é a plataforma *Observable* (BOSTOCK, 2020), um ambiente colaborativo para desenvolvimento e aprendizado de técnicas de visualização de dados na linguagem JavaScript (Figura 11).

No entanto, mesmo com a existência da comunidade ativa, a biblioteca D3, por ainda ser uma biblioteca, exige profundos conhecimentos de programação de forma que, mesmo para pessoas que já têm experiência com a linguagem JavaScript, não é intuitivo gerar visualizações

Figura 11 – Demonstração ilustrativa da galeria da plataforma Observable.



Fonte: Bostock (2020).

com a D3. Assim, essa abordagem não atende ao Requisito 3 - O usuário não precisa possuir habilidade com programação.

3.1.2 Biblioteca Matplotlib

Matplotlib é uma biblioteca abrangente para a criação de visualizações estáticas, animadas e interativas em Python (MATPLOTLIB, 2021). Os recursos são voltados para que os desenvolvedores tirem proveito de outras bibliotecas em Python. É aqui onde esta biblioteca se destaca, na possível integração com outros pacotes baseados em Python como o NumPy e o SciPy.

A integração com NumPy se deve ao suporte que esta biblioteca tem para estruturas de dados matemáticos como matrizes, além de uma imensa coleção de funções matemáticas. Já o SciPy, complementa a anterior com outras rotinas para resolver equações diferenciais, por exemplo.

Além de depender de outros pacotes, como o NumPy no processo de preparação dos dados para gerar a visualização, uma outra desvantagem seria a necessidade de profundos conhecimentos de programação para realizar a codificação visual dos elementos da hierarquia, sendo mais baixo nível do que o próximo exemplo, a biblioteca NetworkX. Assim, essa abordagem também falha em satisfazer o Requisito 3.

3.1.3 Biblioteca NetworkX

O NetworkX (HAGBERG *et al.*, 2008) é um pacote Python para a criação, manipulação de redes complexas. A Figura 12 mostra um exemplo simples de código e a visualização que ele gera. A partir desse exemplo, é possível perceber que conhecimentos básicos em programação já são suficientes para gerar visualizações simples. No entanto, o usuário ainda precisa lidar com muitas configurações explicitamente, tais como posicionamento dos nós, e outros atributos de desenho. Essa controle manual faz com que a complexidade do uso da biblioteca aumente consideravelmente com o aumento do tamanho da rede. Além disso, outra limitação dessa biblioteca é que as visualizações não são interativas nativamente, requerendo um grande esforço de programação para torná-las interativas.

Figura 12 – Demonstração de código e a visualização gerada por ele utilizando NetworkX.

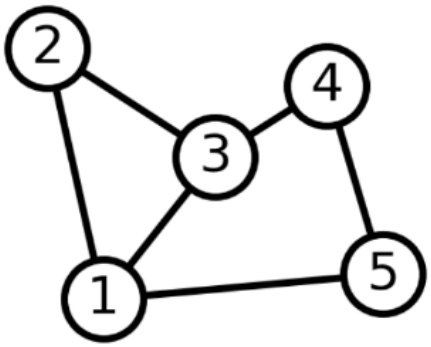
```python
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_edge(1, 2)
G.add_edge(1, 3)
G.add_edge(1, 5)
G.add_edge(2, 3)
G.add_edge(3, 4)
G.add_edge(4, 5)

# explicitly set positions
pos = {1: (0, 0), 2: (-1, 0.3), 3: (2, 0.17), 4: (4, 0.255), 5: (5, 0.03)}

options = {
    "font_size": 36,
    "node_size": 3000,
    "node_color": "white",
    "edgecolors": "black",
    "linewidths": 5,
    "width": 5,
}
nx.draw_networkx(G, pos, **options)

# Set margins for the axes so that nodes aren't clipped
ax = plt.gca()
ax.margins(0.20)
plt.axis("off")
plt.show()
```



Fonte: NetworkX Developers (2021).

3.2 Aplicações

3.2.1 Gephi Software

Criado no ano de 2008, o Gephi é um software de código aberto que possui um mecanismo de renderização espacial para gerar gráficos em tempo real em que múltiplos algorit-

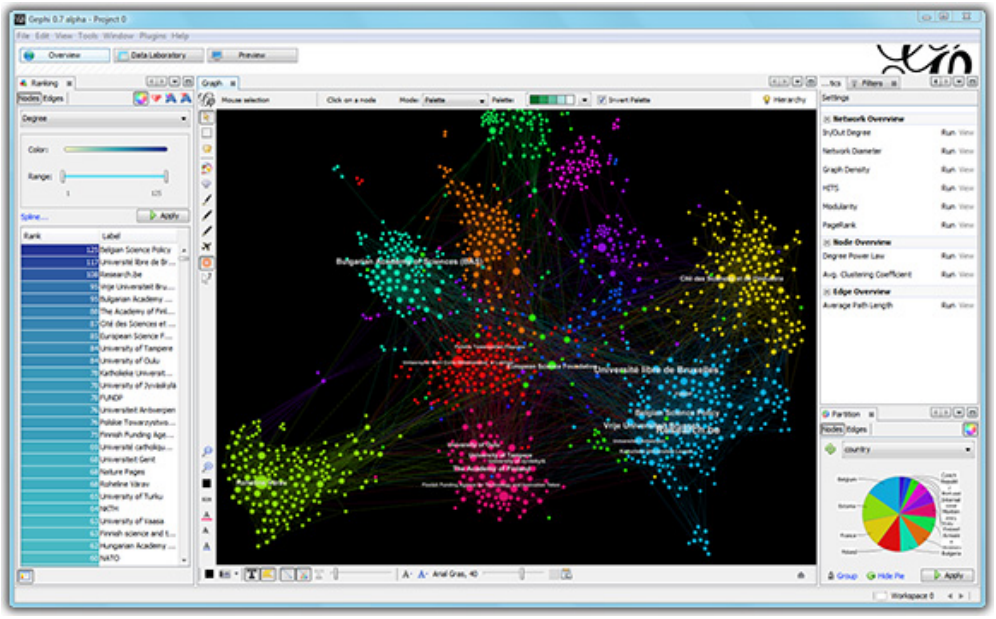
mos podem ser executados simultaneamente sem comprometer a experiência com a interface do usuário (BASTIAN *et al.*, 2009).

É um software que auxilia na elaboração de hipóteses e na descoberta de padrões que são importantes para a análise de dados hierarquizados. A Figura 13 ilustra a interface do Gephi.

Esta ferramenta é muito utilizada na visualização de relações em redes (e.g.: redes sociais). No Gephi, o usuário pode fazer alterações e interagir com estruturas de redes bem como exportar as visualizações em formatos SVG, PDF ou PNG, tendo como exportar também um arquivo no formato GEPHI com as configurações para a visualização gerada.

Outra funcionalidade seria o uso de *Plug-ins* que são desenvolvidos pela comunidade e podem auxiliar na visualização. Além disso este software organiza o trabalho dos dados em três abas para ter uma visão geral, trabalhar com os dados em si e para a visualização respectivamente. Também há diversas funcionalidades que melhoram a experiência visual como a escolha de cores, agrupamentos etc.

Figura 13 – Exemplo de visualização na interface do Gephi.



Fonte: Gephi.org (2021).

Dentre as desvantagens do Gephi está a exportação do arquivo após a edição dos dados, essa funcionalidade só permite exportações nos formatos SVG, PDF e PNG, o que dificulta caso o usuário deseje utilizar os dados editados como entrada em outro momento.

Um outro ponto negativo seria no caso de importar um arquivo hierárquico no formato CSV. Nesse caso, o Gephi só reconhece a hierarquia se no arquivo estiverem explícitas as ligações entre os nós ou se dois arquivos forem importados: um descrevendo os nós e o

outro as arestas. Por último, outra desvantagem é que, apesar de possuir a funcionalidade *Data Laboratory*, uma interface estilo Excel que permite manipular colunas, transformar e buscar os dados, não é possível modificar a hierarquia dos dados.

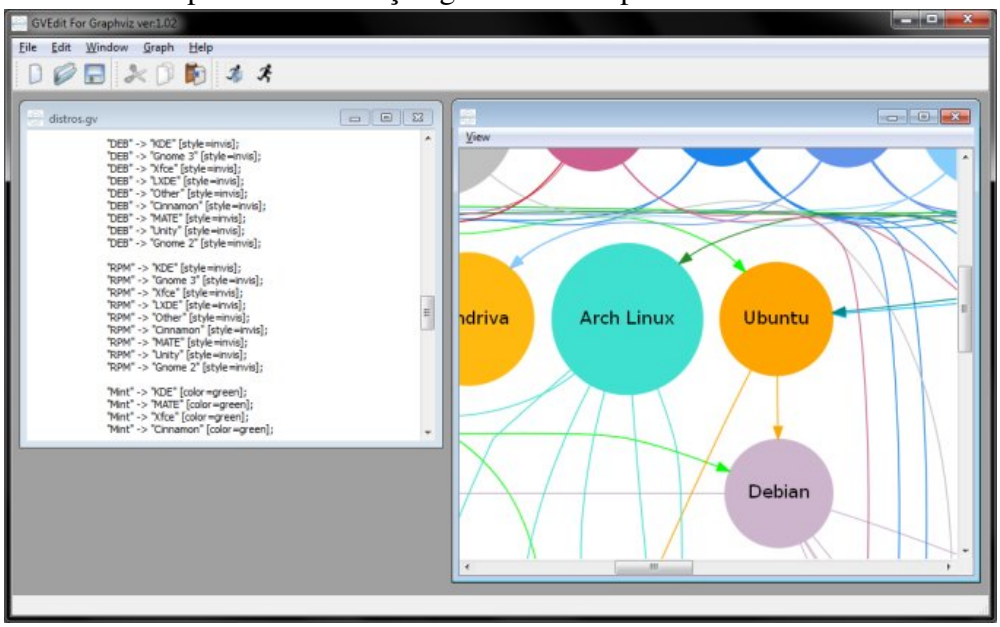
3.2.2 GraphViz Software

Graphviz.org (2021) é um conjunto de programas de código aberto usado para manipular e processar grafos e redes, logo, pode ser usado para visualizar dados hierárquicos (WANG *et al.*, 2015).

Esta ferramenta gera diagramas a partir de descrições de gráficos em uma linguagem de texto simples e faz diagramas em formatos de imagens, SVG para páginas da Web, PDF ou Postscript para inclusão em outros documentos.

As ferramentas do GraphViz executam em modo autônomo, mas interfaces, como a da Figura 14, podem ser utilizadas. Também é possível a criação de scripts *DOT* para uma comunicação com programas ou arquivos externos bem como mudar configurações no comportamento do editor. DOT é um formato de arquivo de texto para descrição de dados de uma rede (GRAPHVIZ.ORG, 2021).

Figura 14 – Exemplo de visualização gerada no GraphViz.

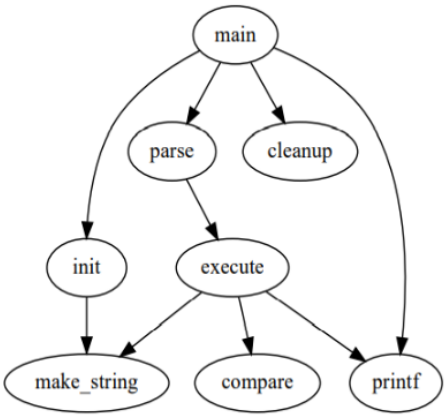


Fonte: Graphviz.org (2021).

A Figura 15 mostra um exemplo simples da sintaxe de um arquivo DOT bem como a visualização gerada por este arquivo. Podemos perceber que, até para uma rede simples, são necessários conhecimentos prévios de programação.

Figura 15 – Exemplo da sintaxe de um arquivo DOT e seu resultado.

```
1:  digraph G {
2:      main -> parse -> execute;
3:      main -> init;
4:      main -> cleanup;
5:      execute -> make_string;
6:      execute -> printf
7:      init -> make_string;
8:      main -> printf;
9:      execute -> compare;
10: }
```



Fonte: Graphviz.org (2021).

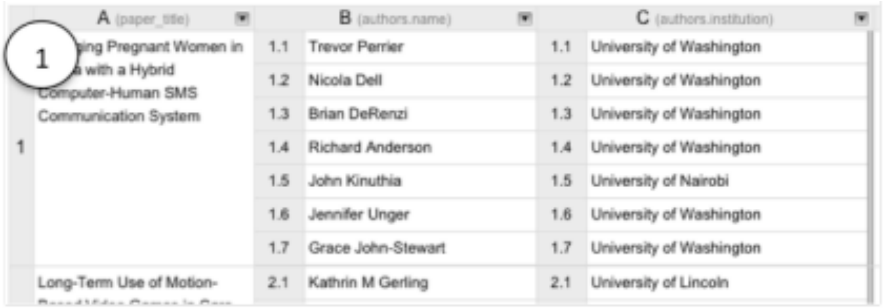
Aceitar arquivos DOT como entrada faz com que o usuário tenha que estruturar os dados em um formato não intuitivo e, além disso, o usuário precisa instalar localmente o software e o suporte para a interface de edição. Esses dois pontos são notáveis desvantagens desta ferramenta.

Além disso, não existem interfaces disponíveis para GraphViz que atendam todos os requisitos levantados na introdução deste documento, e a única maneira de atender aos requisitos seria construir uma interface e usar os aplicativos no modo biblioteca.

3.2.3 Abordagem de Chang e Myers (2016)

O Trabalho de Chang e Myers (2016) foi a inspiração para este projeto. Na ferramenta de Chang, os algoritmos desenvolvidos permitem manipular e visualizar dados hierárquicos em uma planilha. A Figura 16 mostra como os dados são dispostos na interface dessa ferramenta.

Figura 16 – Uma captura de tela da ferramenta desenvolvido por Chang mostrando a interface em um cenário de uso.



Fonte: Chang e Myers (2016).

A ferramenta desenvolvida por Chang permite que o usuário faça a seleção, o agrupamento, a junção, a classificação e a filtragem de dados hierárquicos em planilhas sem

escrever linha de código alguma. Também é possível modificar a hierarquia alterando a ordem das colunas, mas essa ferramenta não gera visualizações. Em vez disso os dados são ainda organizados em formato tabular, não apresentando os benefícios de uma codificação visual.

3.3 Comparativo geral

De forma geral, as ferramentas abordadas neste capítulo, apesar de bem consolidadas e largamente utilizadas, não atendem completamente os requisitos enunciados na introdução que são:

- Requisito 1: Os dados explorados pela aplicação são de pouco volume e estão em um arquivo de entrada em formato tabular (ex.: .csv) fornecido pelo próprio usuário;
- Requisito 2: A aplicação deve permitir que os usuários possam modificar os valores dos dados estruturados hierarquicamente tanto como reorganizar a hierarquia desses dados;
- Requisito 3: O usuário não precisa possuir habilidade com programação;
- Requisito 4: A aplicação deve permitir exportar os dados modificados nos formatos *.csv* ou *.json*;
- Requisito 5: Os dados devem ser representados de forma visual, facilitando a compreensão da hierarquia dos dados.

A Tabela 1 apresenta os requisitos atendidos por cada abordagem, possibilitando fazer a comparação entre elas.

Tabela 1 – Comparativo dos trabalhos relacionados em relação aos requisitos da solução proposta (Archon-viz).

	D3	Matplotlib / NetworkX	Gephi	GraphViz	Chang e Myers (2016)	**Archon-viz**
Requisito 1	✓	✓	✓		✓	✓
Requisito 2	✓	✓		✓	✓	✓
Requisito 3			✓		✓	✓
Requisito 4	✓	✓			✓	✓
Requisito 5	✓	✓	✓	✓		✓

Fonte: elaborada pelo autor.

4 SOLUÇÃO PROPOSTA: ARCHON-VIZ

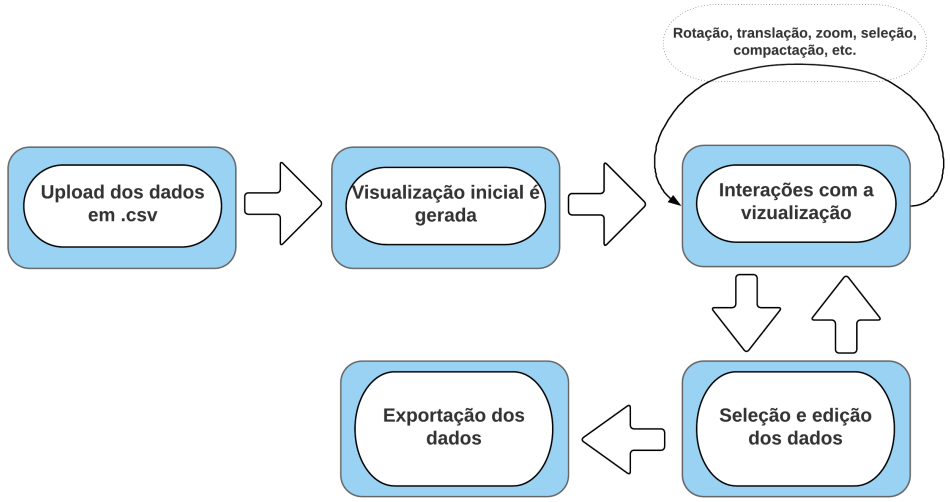
Neste capítulo, é apresentado o Archon-viz, a solução proposta para atender os requisitos apresentados na introdução (Capítulo 1) e que serão discutidos em mais detalhe nas próximas seções.

Na Seção 4.1, é apresentada uma visão geral da solução e da jornada do usuário da aplicação. A Seção 4.2 descreve a interface gráfica do Archon-viz, a Seção 4.3 detalha os algoritmos e a estrutura de dados utilizados e a Seção 4.4 contém os detalhes de implementação da solução.

4.1 Visão Geral

O digrama da Figura 17 ilustra a jornada do usuário ao utilizar o Archon-viz. A jornada consiste em sessões que iniciam na importação dos dados para a ferramenta, depois a realização de operações com os dados e finaliza com a exportação dos dados editados.

Figura 17 – Fluxo da jornada a ser seguida pelo usuário ao utilizar o Archon-viz.



Fonte: Elaborada pelo autor.

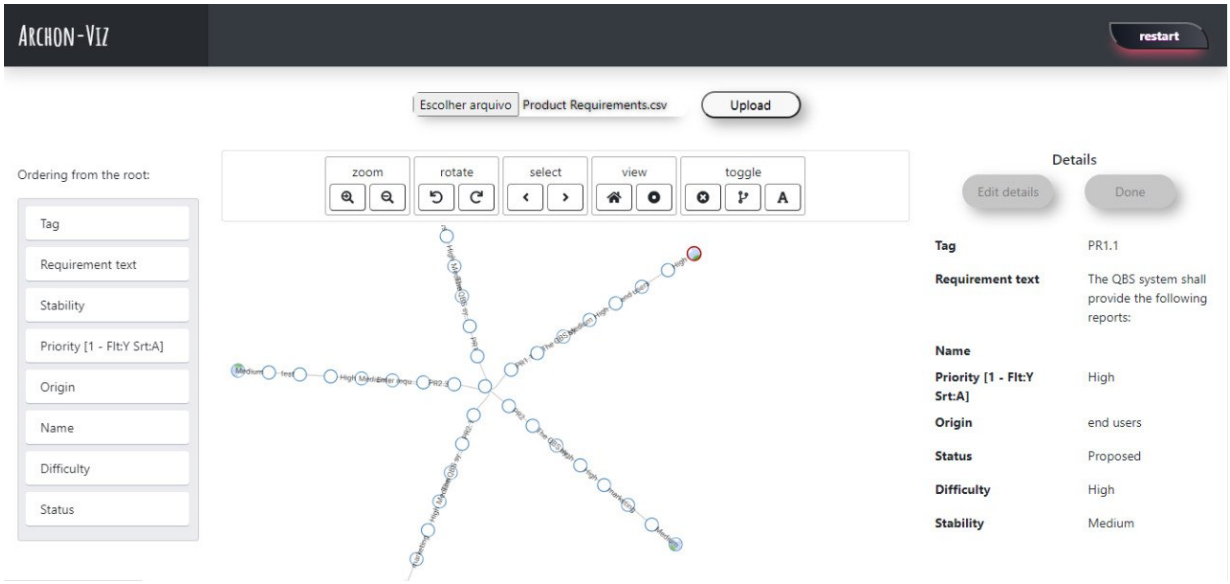
A ferramenta de código aberto foi desenvolvida em JavaScript e objetiva atender aos requisitos do público alvo. Foi utilizada a biblioteca D3 pois ela permite fazer a integração desses dados usando: HTML, CSS e SVG. Além disso, a biblioteca escolhida permite gerar as visualizações de nós e arestas, implementar interações com o usuário e efeitos de transição, faz uso de programação declarativa, facilitando a programação, e conta com uma comunidade vasta graças a sua documentação bem estruturada.

Em meio à versatilidade proporcionada pelo D3, escolhemos utilizar a técnica *Radial Tree* para a visualização dos dados devido ao seu aproveitamento de espaço e à facilidade de relacionar, visualmente, os níveis de hierarquia com as camadas circulares.

4.2 Interface Gráfica do Archon-viz

Tendo em vista a versatilidade da biblioteca escolhida, D3, desenvolvemos uma única interface, como mostra a Figura 18, onde cada etapa da jornada de usuário é realizada.

Figura 18 – Interface do Archon-vis.



Fonte: Elaborada pelo autor.

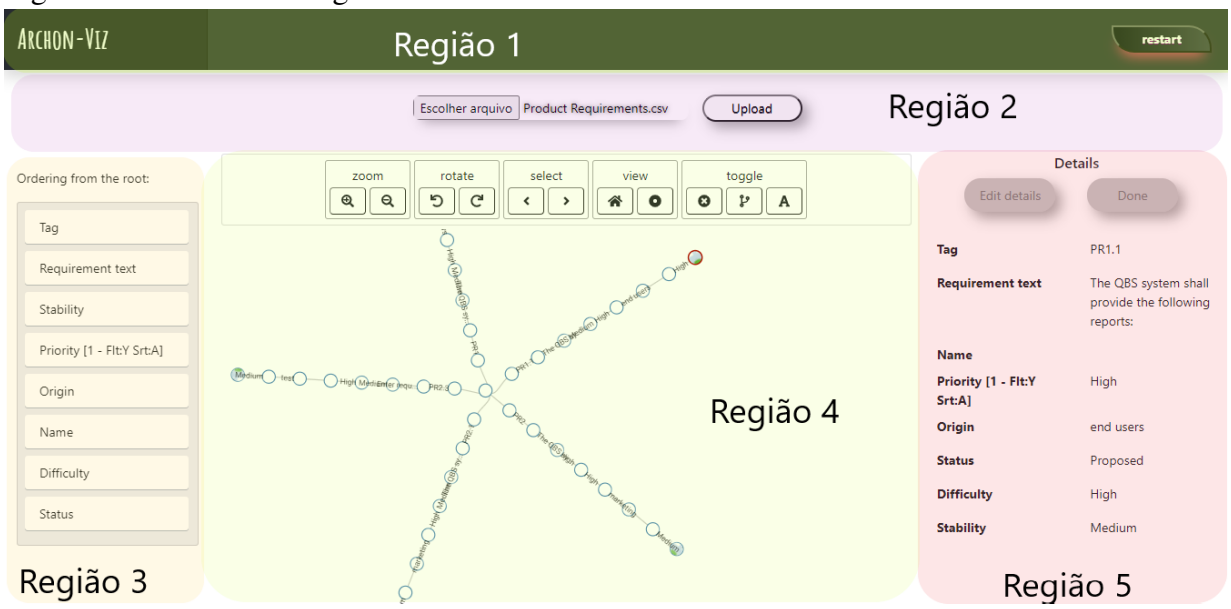
A interface é organizada em 5 regiões (Figura 19): Região 1 para a área de restart, Região 2 para fazer o Upload dos dados, Região 3 para alterar a ordem da hierarquia, Região 4 onde é gerada a visualização e onde ficam os botões auxiliares e, finalmente, a Região 5 onde vemos as informações dos nós e onde fazemos o donwload dos dados editados.

Na primeira região temos o cabeçalho bem como o botão Restart. Este botão deve ser pressionado depois do donwload dos dados editados pois a sessão voltará ao início da jornada do usuário.

Na Região 2 fazemos o Upload dos dados hierárquicos que estão em formato tabular (Requisito 1). Nessa região só serão aceitos documentos em formato *CSV*. Depois de escolher o arquivo que estará armazenado localmente o usuário deve apertar o botão *Upload* que carregará os dados e exibirá uma representação gráfica inicial do conjunto de dados.

Na Região 3, listamos, de cima para baixo, a ordem das categorias desde a raiz até as

Figura 19 – Diferentes regiões de trabalho da interface.



Fonte: Elaborada pelo autor.

folhas. As categorias são extraídas a partir dos nomes dos campos do arquivo CSV. Nesse arquivo CSV, a hierarquia está implícita. Pensando nisso, nesta região foi implementada a funcionalidade de alterar a ordem da hierarquia, através da funcionalidade de arrastar e soltar (Requisito 2).

A funcionalidade de arrastar e soltar permite que o usuário movimente os *cards* das categorias mudando a ordem em tempo de execução, e então uma nova visualização com a nova ordem de categorias é gerada. Essa funcionalidade permite explorar as diversas ordens de hierarquia e supre a necessidade de manipulações externas manuais ou por meio de programação (Requisito 3).

Baseando-se nas técnicas de visualização de dados que foram apresentadas no Capítulo 2 deste documento, o Archon-viz representa os dados hierárquicos usando o layout *Radial Tree* (Requisito 5), que será gerado na Região 4.

A árvore gerada possui nós circulares com bordas azuis. Quando um nó é selecionado a cor da borda deste fica vermelha. Quando um nó é colapsado, o nó restante da operação é preenchido com o matiz azul claro. Já as arestas são construídas com a cor cinza e usam o método de curvas Bézier (BAYDAS, 2019) onde são formadas pela interpolação linear entre pontos.

Na Região 4, estão disponíveis botões para navegação e visualização da hierarquia, como ilustra a Figura 20. Estão incluídas funções de zoom, rotação, seleção, posição da raiz no centro da tela, compactação e exibição/ocultação de labels. Essas funções foram adicionadas para contornar o caso em que a hierarquia contém muitos nós, o que dificultaria a visualização.

Os botões de zoom amplificam e reduzem o tamanho da árvore dentro da janela de visão, caso a imagem fique maior que esta área, a árvore pode ser arrastada na tela com o mouse. Já a funcionalidade de rotação (*rotate*) rotaciona a árvore entorno da raiz no sentido horário ou anti-horário.

Os botões de select selecionam o próximo nó irmão à direita ou à esquerda. Há também dois botões de *view* onde o primeiro retorna a árvore para a posição inicial e, o segundo, coloca o nó selecionado no centro da janela de visão.

Por fim, na região de *toggle* há três botões: o primeiro oculta os filhos do nó selecionado, fazendo com que ele adquira uma coloração azul clara; o segundo faz o processo reverso, ou seja, expande os filhos que estão ocultos; e o terceiro retira ou insere os rótulos da visualização.

Figura 20 – Botões auxiliares.

Fonte: Elaborada pelo autor.

Finalmente, na Região 5, exibimos as informações de cada folha selecionada. Aqui, as informações são os valores de cada categoria desde a raiz até chegar à folha. Além disso, há dois botões: *Edit details* e *Done*. Uma vez selecionada a folha, o primeiro botão habilita a edição dos dados de cada nó que compõe o caminho raiz-folha, cumprindo assim o Requisito 2; o segundo botão grava esta edição e então, na Região 4, uma nova visualização é gerada com os dados que foram editados. Também são habilitados ambos os botões para extrair os dados editados em *csv* e em *json* (Requisito 4).

4.3 Estrutura de Dados e Algoritmos

Abordando uma visão mais microscópica da estratégia adotada neste projeto, a utilização de algumas estruturas de dados e uma linha lógica foram priorizadas.

Primeiramente, no carregamento dos dados, utilizamos o método *csvParse* da biblioteca D3 para transformar os dados em um vetor de objetos onde cada folha é um objeto do vetor. Depois, a partir dos nomes dos campos do arquivo são construídos os nós internos da hierarquia, da raiz até cada folha, usando os valores dos atributos de cada objeto. Em seguida chamamos a

função que desenhará a Radial-Tree passando esse dado como argumento e salvamos os dados iniciais na memória do navegador.

Com os dados na memória do navegador, podemos construir a hierarquia que será mostrada pelos cards. No desenho da árvore, o posicionamento dos nós é feito usando coordenadas polares em que a coordenada x é o ângulo e a y é o tamanho do raio a partir da origem.

Para demonstrar o algoritmo de formação da hierarquia a partir de dados tabulares vamos utilizar o conjunto de dados-exemplo meramente ilustrativo da Figura 21.

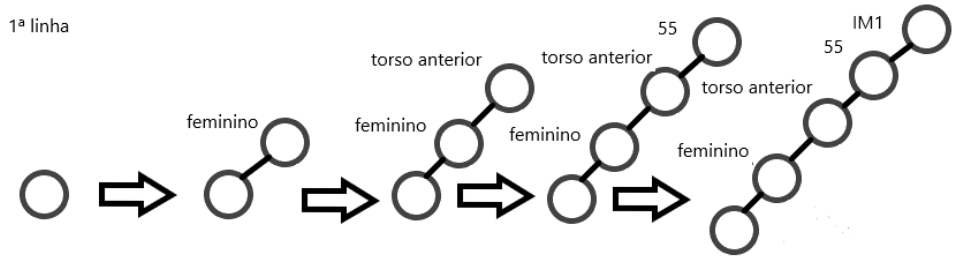
Figura 21 – Conjunto de dados-exemplo para construir a hierarquia.

sexo	Local anatômico	Idade	Imagem
feminino	torso anterior	55	IM1
feminino	torso anterior	30	IM2
feminino	Pernas	60	IM3
masculino	Pernas	30	IM4
masculino	torso posterior	80	IM5

Fonte: Elaborada pelo autor.

Primeiro criamos o nó raiz, em seguida lemos cada valor dentro da primeira linha fazendo: Existe um nó com o valor "feminino"ligado ao nó atual? Não. Então criamos um nó com esse valor e fazemos a ligação com o nó atual que, no caso, é a raiz. Esse novo nó é então selecionado como nó atual. Seguimos o mesmo raciocínio para os próximos atributos. Esse algoritmo segue até o fim da linha e o resultado da 1ª linha pode ser visto na Figura 22.

Figura 22 – Resultado do algoritmo aplicado à primeira linha da amostra.

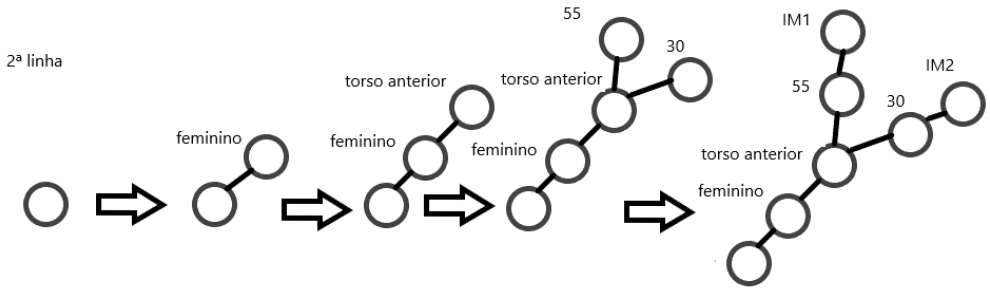


Fonte: Elaborada pelo autor.

Para a segunda linha retornamos à raiz da árvore e fazemos a pergunta novamente: Há algum nó conectado há raiz com o valor "feminino"? Sim, então não criamos um elemento e o novo nó atual será esse nó que já existe. Para o próximo valor da segunda linha: Há algum nó com o valor "torso anterior"ligado ao nó atual? Sim. esse nó se torna o nó atual. Para o próximo valor: Há algum nó com o valor 30 ligado ao atual? Não, então criamos um nó com esse valor

e fazemos a ligação entre ele e o nó com o valor atual que, no caso, é o nó com valor "torso anterior". O resultado até a segunda linha pode ser visto na Figura 23.

Figura 23 – Resultado do algoritmo aplicado até a segunda linha da amostra.



Fonte: Elaborada pelo autor.

Assim é executado o algoritmo, lendo o arquivo de entrada linha por linha, se não houver um nó seguinte com o dado dessa linha, criamos um novo nó com esse valor conectado ao anterior, caso haja um nó com o valor, apenas percorremos o caminho na árvore e seguimos para o próximo valor da linha. O pseudocódigo para este algoritmo está representado na Figura 24 e o resultado para a amostra de dados está ilustrado na Figura 25.

Figura 24 – Pseudo código da lógica de construção da Radial Tree.

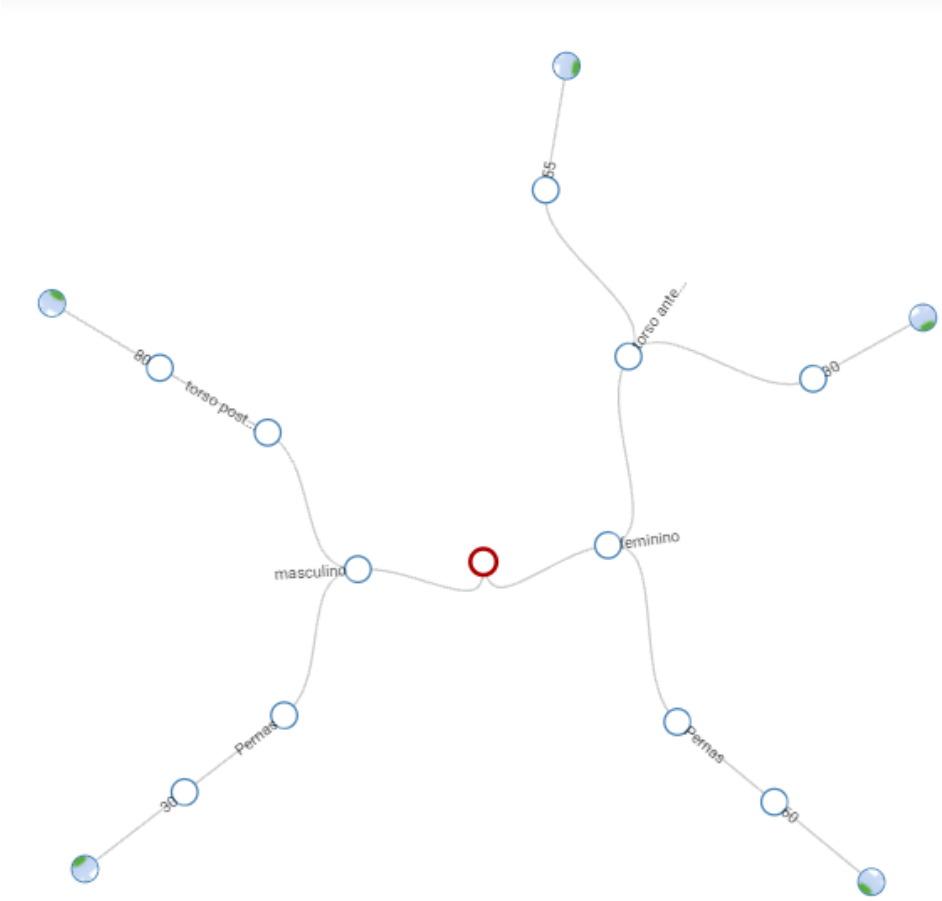
```
funcao cria_hierarquia(dados) {

    cria nó raiz vazio
    para cada elemento d em dados{
       no_atual = raiz
       para cada atributo d.a{
           se( d.a em no_atual.filhos){
               no_atual = no_atual.filhos[d.a]
           }
           senão{
               no_atual.cria_filho(d.a)
               no_atual = no_atual.filhos[d.a]
           }
       }
    }

}
```

Fonte: Elaborada pelo autor.

Figura 25 – Resultado final do algoritmo aplicado à amostra de dados.



Fonte: Elaborada pelo autor.

4.4 Implementação

A organização do código foi feita em 3 arquivos principais: index.html, onde organizamos a nossa interface; style.css onde cuidamos das questões de cores e estilos da página; e tree.js onde desenvolvemos as funções em JavaScript.

Para armazenar valores a serem usados ou alterados, foi utilizado o *LocalStorage*, um recurso nativo no navegador de fácil implementação que não necessita de bibliotecas externas. Com ele, podemos armazenar dados como chave e valor. Essa abordagem permitiu a manipulação dos dados fazendo a chave ser a posição do dado referente ao nó selecionado e o valor sendo o próprio valor do dado.

A fim de facilitar a implementação, utilizamos *JQuery*, uma biblioteca de código aberto em JavaScript que facilita na interação entre elementos HTML e funções em JavaScript. Esta biblioteca foi utilizada para atualização dos dados exibidos nos elementos HTML bem como na obtenção dos dados já exibidos para armazenamento de estados.

5 ESTUDO DE CASO: EXPLORANDO CONJUNTO DE DADOS DE LESÕES CUTÂNEAS.

Neste capítulo, ilustramos a utilização do Archon-viz através de um estudo de caso. Na Seção 5.1, são descritos os dados, bem como a sua proveniência e sua estrutura. Na Seção 5.2 usamos as funcionalidades desenvolvidas para explorar os dados e demonstrar como utilizar esta ferramenta. Por fim, na Seção 5.3, discutimos as limitações presentes na solução proposta.

5.1 Descrição dos Dados

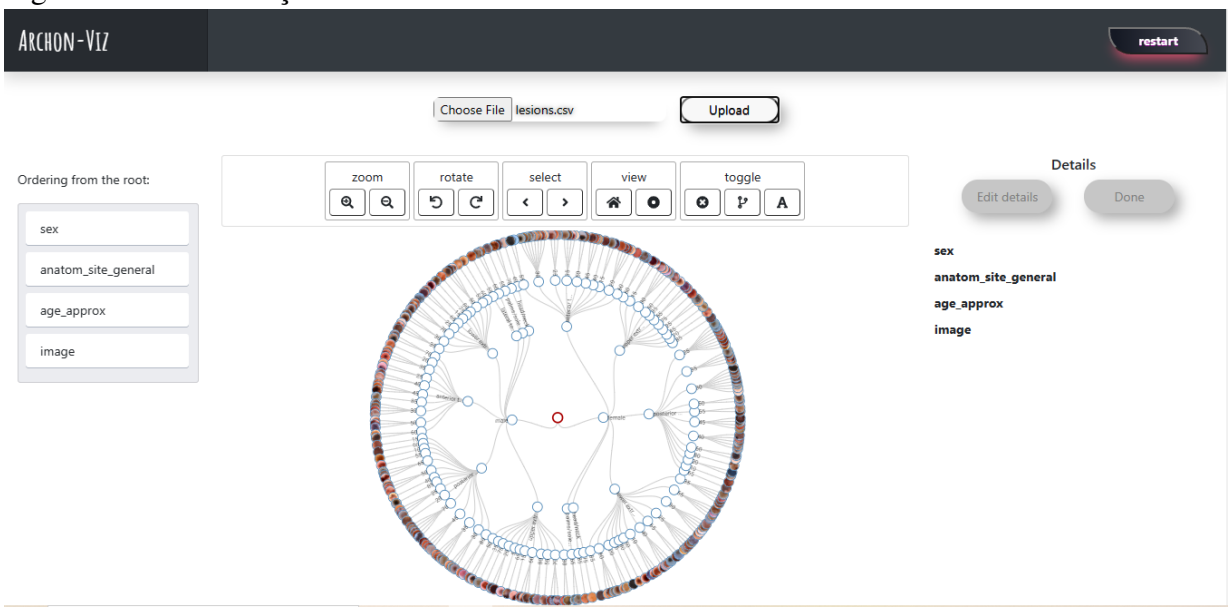
Os dados são obtidos da *International Skin Imaging Collaboration* (ISIC, 2021). O conjunto utilizado é um subconjunto das imagens dermatoscópicas de lesões cutâneas constituído por 232 itens, uma quantidade pequena mas suficiente para simular uma biblioteca de imagens construída por um médico a partir dos seus pacientes. Nessa biblioteca cada item possui o sexo do indivíduo (*sex*), a localização anatômica da lesão (*anatom_site_general*), sua idade (*age_approx*), e o endereço da imagem da lesão (*image*). Inicialmente o arquivo de dados está no formato CSV, como mostra a Figura 26. Podemos observar que existe uma hierarquia implícita nos dados, mas que nesse formato é difícil explorá-la e visualizar como os dados estão divididos em cada nível.

Figura 26 – Uma amostra do conjunto de dados de lesões cutâneas utilizado no estudo de caso.

sex	anatom_site_general	age_approx	image
female	anterior torso	55	images/ISIC_0000000.jpg
female	anterior torso	30	images/ISIC_0000001.jpg
female	upper extremity	60	images/ISIC_0000002.jpg
male	upper extremity	30	images/ISIC_0000003.jpg
male	posterior torso	80	images/ISIC_0000004.jpg
female	posterior torso	25	images/ISIC_0000006.jpg
female	posterior torso	25	images/ISIC_0000007.jpg
female	anterior torso	30	images/ISIC_0000008.jpg
female	anterior torso	30	images/ISIC_0000009.jpg
female	posterior torso	35	images/ISIC_0000010.jpg
female	lower extremity	35	images/ISIC_0000011.jpg
male	posterior torso	30	images/ISIC_0000012.jpg
female	posterior torso	30	images/ISIC_0000013.jpg
male	posterior torso	35	images/ISIC_0000014.jpg
male	posterior torso	35	images/ISIC_0000015.jpg
female	anterior torso	55	images/ISIC_0000016.jpg
female	posterior torso	50	images/ISIC_0000017_downsampled.jpg
male	posterior torso	30	images/ISIC_0000018_downsampled.jpg
female	posterior torso	30	images/ISIC_0000019_downsampled.jpg
female	anterior torso	25	images/ISIC_0000020_downsampled.jpg
female	posterior torso	55	images/ISIC_0000021_downsampled.jpg
female	lower extremity	55	images/ISIC_0000022_downsampled.jpg

Fonte: Elaborada pelo autor.

Figura 27 – Visualização inicial no Archon-viz.



Fonte: Elaborada pelo autor.

Neste estudo de caso, o objetivo do usuário é verificar como as lesões se relacionam com os demais atributos dos dados, explorando diferentes possibilidades de hierarquias. Para isso, é simulada uma jornada de usuário, usando o Archon-viz, com esse conjunto de dados.

5.2 Jornada de Usuário

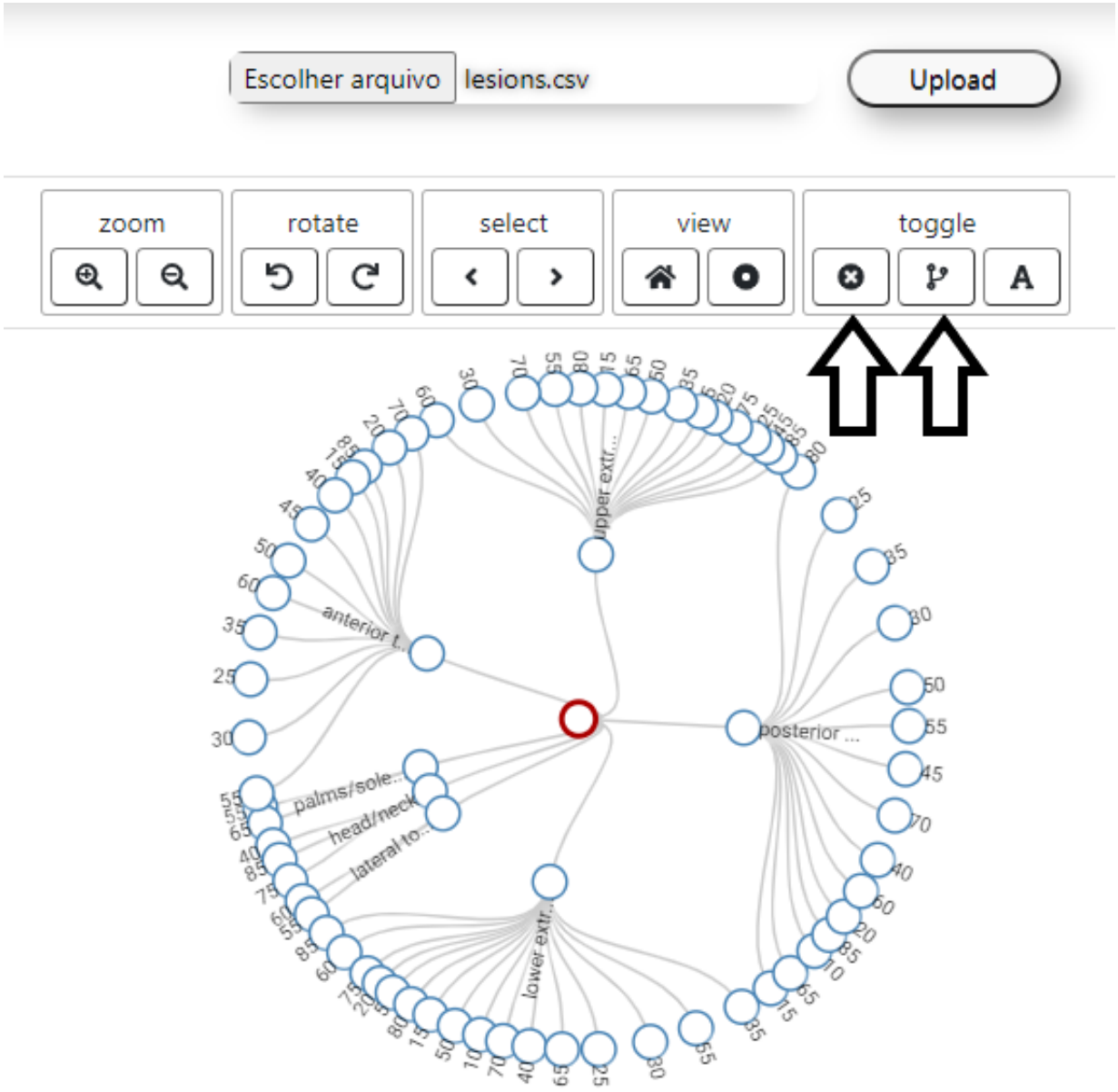
Após carregar os dados tabulares através do botão Upload, é gerada uma visualização inicial de acordo com a Figura 27. Aqui já se pode observar que o usuário tem uma visão completa dos dados no espaço de uma única tela, evitando rolagens de página.

A visualização consiste em uma *Radial-Tree*, que foi abordada no Capítulo 2. Ela é composta por um nó central, também chamado de Raiz, que se conecta aos nós-folhas através de nós intermediários representando os valores de cada campo do arquivo CSV. Cada caminho da raiz até um nó folha corresponde a uma linha da tabela. A árvore é construída de tal maneira que cada camada corresponde a um campo da tabela, possuindo um número de nós equivalente ao número distinto de valores de campo presentes na tabela.

No caso do conjunto de dados utilizado, inicialmente desenhamos a *Radial-Tree* seguindo os níveis a partir da raiz, mostrados no painel esquerdo da Figura 27: *sex*, *anatom_site_general*, *age_approx*, e *image*. Esta última, também chamada de folha guarda a URL da imagem a ser exibida.

Podemos tentar retirar informações da visualização inicial como, por exemplo, saber

Figura 28 – *age_approx* e *anatom_site_general* à mostra na hierarquia.



Fonte: Elaborada pelo autor.

que há, aproximadamente, o mesmo numero de casos entre homens e mulheres.

O próximo passo é alterar a ordem da hierarquia para descobrir novas informações. Buscando saber a faixa etária com maior número de casos de lesões vamos agrupar as idades em grupos de locais anatômicos para isso vamos clicar e arrastar os cards alterando a ordem da hierarquia para *age_approx*, *anatom_site_general*, *sex* e *image*. Utilizando os botões de *Toggle Node* e *Toggle Level* uma e duas vezes respectivamente, abrimos a hierarquia até a segunda camada que corresponde à disposição mostrada na Figura 28, onde percebemos que o conjunto de dados possui um número levemente superior de ocorrências de lesões em indivíduos com idades de 55, 60 e 65 anos.

A fim de tirar informações a respeito dos locais anatômicos com maior numero de casos podemos ordenar os cards como: *anatom_site_general*, *age_approx*, *sex* e *image*, utilizando os botões de *toggle Node* e *Toggle Level* três vezes como fizemos anteriormente, abrimos a hierarquia até a segunda camada e podemos gerar a visualização ilustrada na Figura 29. Aqui podemos concluir que, nesse conjunto de dados, o local anatômico com maiores ocorrências foi a extremidade superior e os locais com menores ocorrências são cabeça/pescoço, palmas das mãos e solas do pé e lateral do torso.

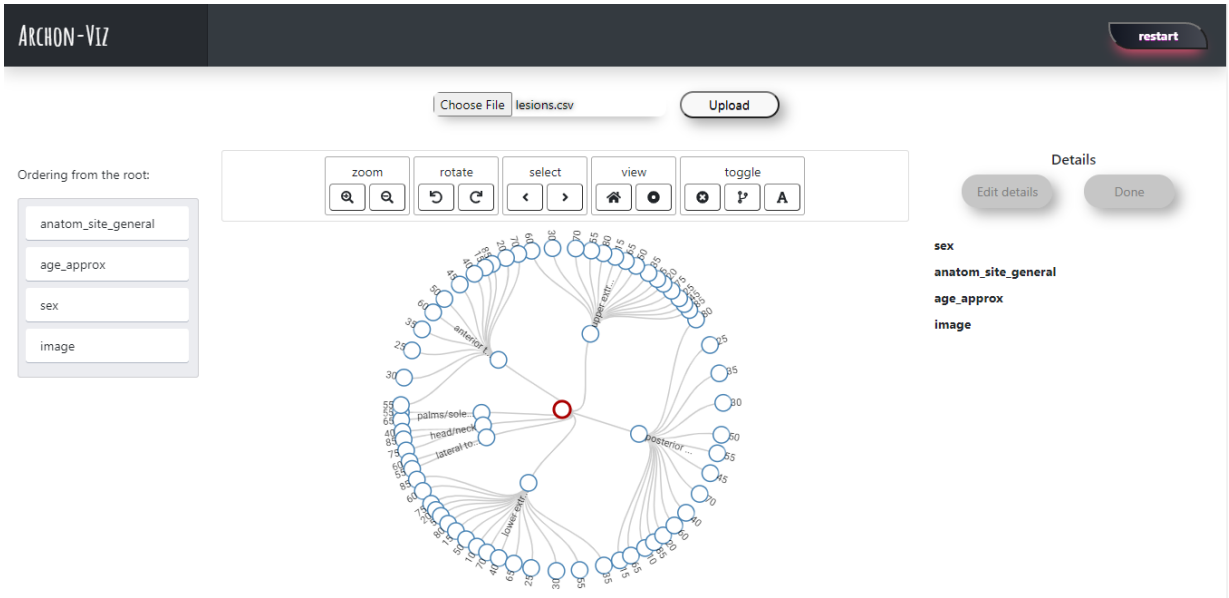
Para isolar apenas os atributos de interesse, podemos selecionar um nó que não seja de interesse no nível da hierarquia que queremos analisar e, logo em seguida, utilizando o primeiro botão da região *Toggle* podemos contrair os filhos do nó selecionado. A Figura 30 ilustra a disposição final, nela deixamos apenas as três idades com mais ocorrências expandidas.

Para expandir um nó que foi contraído, basta clicar duas vezes nele para que o próximo nível da hierarquia apareça para ele.

Se a visualização tiver uma quantidade de dados tal que o tamanho da tela não seja suficiente para uma boa exploração, podemos utilizar o botão de zoom, por exemplo. Ao selecionar uma extremidade da visualização, podemos ver os valores dos atributos e as imagens de cada caso. Além disso, podemos também habilitar a edição desses atributos, como mostra a Figura 31.

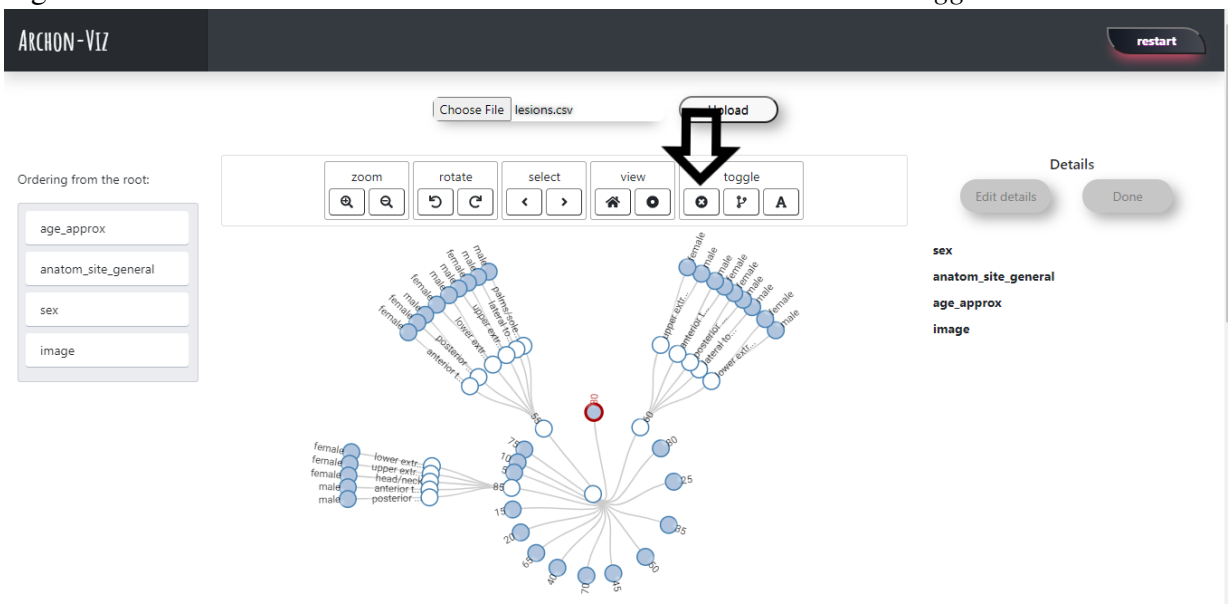
Logo após a edição, pressionando o botão *Done*, serão habilitados os botões para o download dos novos dados em dois formatos: .csv e .json. Além disso, uma nova visualização

Figura 29 – Análise por *anatom_site_general*.



Fonte: Elaborada pelo autor.

Figura 30 – Obtendo uma visão mais detalhada com a funcionalidade *Toggle*.



Fonte: Elaborada pelo autor.

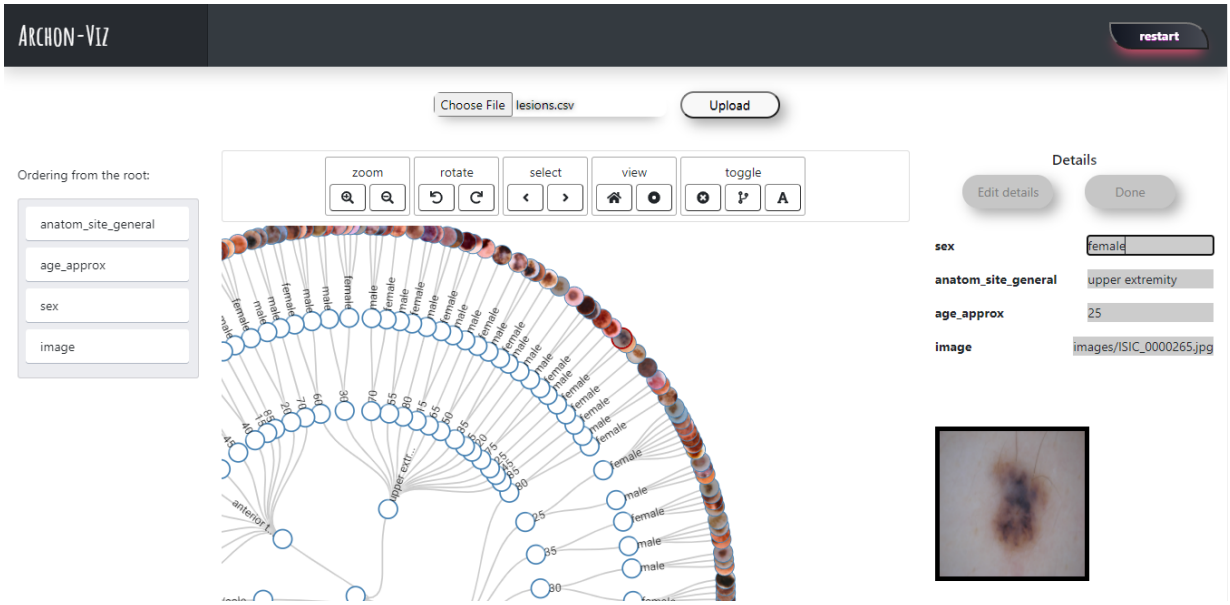
com os dados editados é gerada.

5.3 Discussão e Limitações

A ideia inicial do Archon-viz partiu de um médico patologista, especialista em nefropatologia, que estava construindo sua própria árvore de lesões nefropatológicas a partir de estudos da literatura e sua própria experiência.

Inicialmente ele estava usando um software de mapa mental para construir sua

Figura 31 – Edição de atributos.



Fonte: Elaborada pelo autor.

árvore manualmente. Após uma versão inicial da árvore, ele resolveu transpor seus dados para uma planilha para poder colaborar com cientistas da computação que estavam trabalhando na classificação automática das imagens das lesões nefropatológicas. O problema é que ele quis replicar a estrutura hierárquica do mapa mental na própria planilha, desenhando bordas na planilha representando os ramos da árvore e utilizando células em branco para manter a estrutura aparente. Ao fazer isso, a planilha se tornava inútil para qualquer aproveitamento dos dados pelas equipes de aprendizagem de máquina e de visualização. Então, a saída encontrada para esse problema foi construir uma ferramenta em que ele pudesse explorar a hierarquia dos dados, editando os atributos e as imagens das lesões, abstraindo a manipulação da planilha. O resultado é o Archon-viz. Nele o patologista interage com a árvore, pode ver as imagens organizadas por coloração, diagnóstico nosológico e outras características das imagens, e exportar os resultados em formato tabular que pode ser consumido por outras técnicas de aprendizagem de máquina e visualização. A ferramenta também foi apresentada para outros patologistas que confirmaram uma boa interação com o formato radial.

Infelizmente, não foi possível apresentar esse caso como estudo de caso neste trabalho de conclusão de curso, devido à colaboração ainda estar em andamento e os seus resultados ainda serão publicados. Assim, foi preciso utilizar um exemplo semelhante. Mesmo não sendo tão motivador como o exemplo original, o estudo de caso das lesões cutâneas demonstra todas as funcionalidades do Archon-viz.

Após interações entre o patologista e uma versão do Archon-viz, foram detectadas algumas limitações da solução. Nesta versão, não está implementada a funcionalidade para upload/troca da imagem na folha. Além disso, os valores dos atributos só são mostrados no campo *Details* quando o nó selecionado for um nó folha, ou seja, no modo contraído os detalhes não são exibidos.

Apesar dos mecanismos de navegação e interação implementados, árvores muito grandes ainda podem comprometer a experiência do usuário ao utilizar a aplicação. Mecanismos de busca e filtros para poda da árvore são fundamentais.

Outra limitação é a aplicação funcionar localmente no navegador do cliente. Uma implementação mais robusta requeriria uma aplicação cliente-servidor, onde os dados seriam guardados em uma base de dados no servidor. Isso abriria portas para permitir a colaboração entre patologistas na manipulação da hierarquia.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho apresentamos o Archon-viz, uma solução para exploração e manipulação de dados hierárquicos que, baseado em técnicas de visualização de dados, facilita o trabalho com este tipo de estrutura de dados, atendendo os requisitos apresentados anteriormente, que são:

- Requisito 1: Os dados explorados pela aplicação são de pouco volume e estão em um arquivo de entrada em formato tabular (ex.: .csv) fornecido pelo próprio usuário;
- Requisito 2: A aplicação deve permitir que os usuários possam modificar os valores dos dados estruturados hierarquicamente tanto como reorganizar a hierarquia desses dados;
- Requisito 3: O usuário não precisa possuir habilidade com programação;
- Requisito 4: A aplicação deve permitir exportar os dados modificados nos formatos *.csv* ou *.json*;
- Requisito 5: Os dados devem ser representados de forma visual, facilitando a compreensão da hierarquia dos dados.

Para tanto, foi desenvolvida uma aplicação utilizando a linguagem de programação JavaScript com ênfase na biblioteca D3-js. A solução adota o layout radial tree, que apesar de algumas limitações, atende aos requisitos enumerados. Também foi demonstrada a utilidade da solução através de um estudo de caso baseado em uma amostra de um conjunto de dados de lesões cutâneas. Dessa forma, concluímos que o objetivo deste trabalho foi alcançado com êxito.

No Capítulo 5 já foram discutidas algumas limitações da solução que trazem algumas possibilidades de trabalhos futuros. Agora, apresentaremos futuras melhorias a serem implementadas.

Com relação aos algoritmos, apesar dos usuários adorarem o formato circular, também pretendemos explorar com outros layouts, como por exemplo um layout *force-directed*. A justificativa é que o layout force-directed também pode se comportar de forma suave e orgânica, sendo mais favorável para agrupar os dados usando outras funções de similaridade, que não estejam embutidas na própria estrutura. Por exemplo, no estudo de caso das lesões cutâneas, poderíamos desenhar a árvore seguindo a estrutura dos campos mas os nós mais semelhantes, de acordo com o resultado de um algoritmo de classificação de imagens, ficariam localizados mais próximos entre si. Além disso, se o volume de dados for realmente muito grande, valeria a pena

usar uma representação de matriz de adjacência. No entanto, seriam necessários estudos com usuários para verificar as vantagens e as desvantagens dos diferentes layouts. Uma solução que permita a troca de layout em tempo de execução parece ser uma alternativa razoável.

Com relação aos dados, a solução também poderia lidar com formatos naturalmente hierárquicos, como do tipo JSON. Tipos diferentes de atributos de dados também poderiam ser melhor tratados, diferenciando dados numéricos e categóricos e exibindo filtros apropriados).

Com relação à implementação também podem ser incorporadas algumas otimizações. Durante a exploração, muitas funcionalidades respondem atualizando algum elemento da interface, essa estratégia pode ser otimizada utilizando React-js, um framework para interfaces baseadas em JavaScript, para otimizar a atualização de elementos visuais, como os valores na região *Details*. Esse framework também permitiria a atualização da árvore de forma localizada, não necessitando redesenhar a árvore por completo em cada alteração.

Como mencionado no Capítulo 5, implementar uma funcionalidade de pesquisa na interface onde podemos localizar um nó a partir de uma palavra que seria o valor de seu atributo é uma funcionalidade que pode tornar a exploração mais ágil. Além disso, a inclusão de uma ferramenta de anotação onde seria possível incluir notas na área de plotagem, seria enriquecedor pois permitiria fazer um *Story Telling* através das notas.

Por fim, uma outra possibilidade de trabalho futuro, seria permitir a colaboração entre mais de um usuário numa mesma árvore. Numa visualização interativa na Web, a colaboração em tempo real é bastante desafiadora e soluções, tais como as de Badam *et al.* (2019), de Schwab *et al.* (2021) e de Brehmer e Kosara (2021), merecem ser investigadas.

REFERÊNCIAS

ARCE-OROZCO, A.; CAMACHO-VALERIO, L.; MADRIGAL-QUESADA, S. Radial tree in bunches: Optimizing the use of space in the visualization of radial trees. In: **2017 International Conference on Information Systems and Computer Science (INCISCOS)**. [S.l.: s.n.], 2017. p. 369–374.

BADAM, S. K.; MATHISEN, A.; RäDLE, R.; KLOKMOSE, C. N.; ELMQVIST, N. Vistrates: A component model for ubiquitous analytics. **IEEE Transactions on Visualization and Computer Graphics**, v. 25, n. 1, p. 586–596, 2019.

BASTIAN, M.; HEYMANN, S.; JACOMY, M. Gephi: An open source software for exploring and manipulating networks. In: . [S.l.: s.n.], 2009.

BAYDAS, K. Defining a curve as a bezier curve. **Journal of Taibah University for Science**, 2019.

BOSTOCK, M. **Observable Gallery**. 2020. Disponível em: <https://gephi.org/users/supported-graph-formats/graphviz-dot-format/>. Acesso em: 18-09-2021.

BOSTOCK, M. **Radial tidy tree**. Estados Unidos: [s.n.], 2021. Disponível em: <https://observablehq.com/@d3/radial-tidy-tree>. Acesso em: 29-07-2021.

BOSTOCK, M.; OGIEVETSKY, V.; HEER, J. D³ Data-Driven Documents. **IEEE Transactions on Visualization and Computer Graphics**, v. 17, n. 12, p. 2301–2309, 2011.

BREHMER, M.; KOSARA, R. From jam session to recital: Synchronous communication and collaboration around data in organizations. **IEEE Transactions on Visualization and Computer Graphics**, IEEE Computer Society, Los Alamitos, CA, USA, n. 01, p. 1–1, sep 2021. ISSN 1941-0506. Preprint.

BURCH, M.; HEINRICH, J.; KONEVTSOVA, N.; HöFERLIN, M.; WEISKOPF, D. Evaluation of traditional, orthogonal, and radial tree diagrams by an eye tracking study. **IEEE transactions on visualization and computer graphics**, v. 17, p. 2440–8, 12 2011.

BURCH, M.; WEISKOPF, D. On the benefits and drawbacks of radial diagrams. In: ____. **Handbook of Human Centric Visualization**. New York, NY: Springer New York, 2014. p. 429–451. ISBN 978-1-4614-7485-2. Disponível em: <https://doi.org/10.1007/978-1-4614-7485-2_17>.

CAIRO, A. **The Truthful Art: Data, Charts, and Maps for Communication**. 1st. ed. USA: New Riders Publishing, 2016. ISBN 0321934075.

CHANG, K.; MYERS, B. Using and exploring hierarchical data in spreadsheets. In: . [S.l.: s.n.], 2016. p. 2497–2507.

CLEVELAND, R. M. W. S. Graphical perception and graphical methods for analyzing scientific data. **Science**, American Association for the Advancement of Science, v. 229, n. 4716, 1985. ISSN 00368075, 10959203. Disponível em: <http://www.jstor.org/stable/1695272>.

DEVINKNIGHT. **Power BI Custom Visuals Class (Module 86 – Bubble Chart by Akvelon)**. 2018. Disponível em: <https://devinknightsql.com/2018/01/19/power-bi-custom-visuals-class-module-86-bubble-chart-by-akvelon/>.

GAMBINO, M. **Do Our Brains Find Certain Shapes More Attractive Than Others?** Smithsonian Institution, 2013. Disponível em: <https://www.smithsonianmag.com/science-nature/do-our-brains-find-certain-shapes-more-attractive-than-others-180947692/>.

GASPAR, H.; BRETT, P.; EBRAHIMI, A.; KEANE, A. Data-driven documents (d3) applied to conceptual ship design knowledge. In: . [S.l.: s.n.], 2014.

GEPHI.ORG. **Gephi DOT file format**. 2021. Disponível em: <https://gephi.org/users/supported-graph-formats/graphviz-dot-format/>. Acesso em: 27-07-2021.

GHONIEM, M.; FEKETE, J.-D.; CASTAGLIOLA, P. On the readability of graphs using node-link and matrix-based representations: A controlled experiment and statistical analysis. **Information Visualization**, v. 4, n. 2, p. 114–135, 2005. Disponível em: <https://doi.org/10.1057/palgrave.ivs.9500092>.

GitHub. **GitHub**. 2021. Disponível em: <https://github.com/>.

GRAPHVIZ.ORG. **Graphviz file format**. 2021. Disponível em: <https://graphviz.org/doc/info/lang.html>. Acesso em: 27-07-2021.

HAGBERG, A. A.; SCHULT, D. A.; SWART, P. J. Exploring network structure, dynamics, and function using networkx. In: VAROQUAUX, G.; VAUGHT, T.; MILLMAN, J. (Ed.). **Proceedings of the 7th Python in Science Conference**. Pasadena, CA USA: [s.n.], 2008. p. 11 – 15.

HAUSSMANN CESAR HIDALGO, e. R. **Jamaica Export Treemap**. 2012. Disponível em: <https://commons.wikimedia.org/wiki/File:Jamaica_Export_Treemap.jpg>. Acesso em: 30-05-2021.

HOUAISS, A. **Dicionário Houaiss da Língua Portuguesa**. [S.l.]: Ed. Objetiva, 2001.

HUA, J.; HUANG, M. L.; HUANG, W.; WANG, J.; NGUYEN, Q. V. Force-directed graph visualization with pre-positioning - improving convergence time and quality of layout. In: **2012 16th International Conference on Information Visualisation**. [S.l.: s.n.], 2012. p. 124–129.

ISIC. **The International Skin Imaging Collaboration**. 2021. Disponível em: <https://www.isic-archive.com/#!/topWithHeader/wideContentTop/main>. Acesso em: 20-09-2021.

JOHNSON, B.; SHNEIDERMAN, B. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In: **Proceeding Visualization '91**. [S.l.: s.n.], 1991. p. 284–291.

KIRK, A. **Data Visualization: a successful design process**. 2. ed. [S.l.]: Sage Publications Ltd, 2019.

LIMA, M. **The book of circles: Visualizing spheres of knowledge**. [S.l.]: Chronicle Books, 2017.

MATPLOTLIB. **Matplotlib: Python**. 2021. Disponível em: <https://matplotlib.org/>. Acesso em: 30-07-2021.

MILES, B. **Radial tidy tree**. Inglaterra, Reino Unido: [s.n.], 2021. Disponível em: <https://observablehq.com/@bmpms/d3-tree-radial-horizontal-vertical>. Acesso em: 30-07-2021.

MUNZNER, T. **Visualization Analysis and Design**. [S.l.]: CRC Press, 2014.

NetworkX Developers. **Simple Graph**. 2021. Disponível em: <https://networkx.org/documentation/stable/auto_examples/basic/plot_simple_graph.html#sphx-glr-auto-examples-basic-plot-simple-graph-py>. Acesso em: 07-11-2021.

SCHWAB, M.; SAFFO, D.; ZHANG, Y.; SINHA, S.; NITA-ROTARU, C.; TOMPKIN, J.; DUNNE, C.; BORKIN, M. A. Visconnect: Distributed event synchronization for collaborative visualization. **IEEE Transactions on Visualization and Computer Graphics**, v. 27, n. 2, p. 347–357, 2021.

STAATS, B. **Graph Visualization Introduction**. Estados Unidos: [s.n.], 2021. Disponível em: <https://observablehq.com/@bstaats/graph-visualization-introduction>. Acesso em: 30-07-2021.

TABLEAU. **Business Intelligence and Analytics Software**. 2021. Disponível em: <https://www.tableau.com/>.

WANG, R.; PEREZ-RIVEROL, Y.; HERMJAKOB, H.; VIZCAINO, J. Open source libraries and frameworks for biological data visualisation: A guide for developers. **Proteomics**, v. 15, 02 2015.

WILLS, G. Visualizing hierarchical data. In: ____. **Encyclopedia of Database Systems**. Boston, MA: Springer US, 2009. p. 3425–3432. ISBN 978-0-387-39940-9. Disponível em: <https://doi.org/10.1007/978-0-387-39940-9_1380>.

YAU, N. **How to Make an Interactive Network Visualization**. 2016. Disponível em: <https://flowingdata.com/2012/08/02/how-to-make-an-interactive-network-visualization/>.

YEE, K.-P.; FISHER, D.; DHAMIJA, R.; HEARST, M. Animated exploration of dynamic graphs with radial layout. In: . [S.l.: s.n.], 2001. p. 43–50. ISBN 0-7695-7342-5.