UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO TECNOLÓGICO

COLEGIADO DO CURSO DE ENGENHARIA DE COMPUTAÇÃO

Elias Pereira Gomes Junior

Desenvolvimento de uma ferramenta para visualização de dados do Programa de Assistência Cirúrgica e Dermatológica da UFES

Vitória, ES

2023

Elias Pereira Gomes Junior

Desenvolvimento de uma ferramenta para visualização de dados do Programa de Assistência Cirúrgica e Dermatológica da UFES

Monografia apresentada ao Colegiado do Curso de Engenharia de Computação do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Engenharia de Computação.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Colegiado do Curso de Engenharia de Computação

Orientador: Prof. Dr. André Georghton Cardoso Pacheco

Vitória, ES

2023

Elias Pereira Gomes Junior

Desenvolvimento de uma ferramenta para visualização de dados do Programa de Assistência Cirúrgica e Dermatológica da UFES

Monografia apresentada ao Colegiado do Curso de Engenharia de Computação do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Engenharia de Computação.

Trabalho aprovado. Vitória, ES, 16 de julho de 2023:

Prof. Dr. André Georghton Cardoso Pacheco
Orientador

Giovanni Ventorim Comarela
Universidade Federal do Espírito Santo

Luis Antonio de Souza Júnior
Universidade Federal do Espírito Santo

Vitória, ES
2023

Agradeço a Deus que me deu forças para concluir este trabalho de forma satisfatória. Agradeço a todos que contribuíram de forma direta e indiretamente ao longo da minha jornada acadêmica.

Agradecimentos

Em primeiro lugar, agradeço a Deus, por estar comigo em todos os momentos. Por ser meu amigo, meu refúgio e minha fortaleza, me dando força e saúde para alcançar meus objetivos e seguir adiante com determinação.

Agradeço aos meus pais, Elias e Maria, e às minhas irmãs Meiry e Mirella, por estarem ao meu lado nos momentos difíceis, muitas vezes renunciando ao próprio conforto para que eu pudesse ter as melhores oportunidades de estudo. Além disso, me ensinaram valores fundamentais que moldaram quem eu sou. Reconheço que as melhores partes de mim vêm de vocês. Agradeço por sempre me motivarem a perseguir meus sonhos e a nunca deixar de almejar o alto. Sem o apoio e o amor de vocês, nada do que alcancei seria possível.

Quero expressar minha gratidão ao meu orientador, André Georghton Cardoso Pacheco, pelo apoio contínuo durante a realização deste trabalho. Suas ideias e sugestões foram fundamentais para o resultado obtido.

Também gostaria de agradecer ao grupo PET Engenharia de Computação, por todo o desenvolvimento pessoal e profissional que adveio da minha participação no grupo. Obrigado por todos os projetos, oportunidades, eventos e pessoas incríveis com quem tive oportunidade de trabalhar.

Gostaria de expressar meu agradecimento à banca por sua atenção, tempo e conhecimento dedicados à avaliação deste trabalho.

Agradeço aos meus amigos da CRU e do Haja Luz, encontrar vocês foi uma brisa fresca em um dia quente. Obrigado por sempre estarem comigo me proporcionando momentos tão singulares e bons durante minha caminhada universitária. Sem vocês, meu período como estudante da UFES perderia um pouco a cor.

Por fim, gostaria de agradecer a todos os meus companheiros de curso, sem a parceira de vocês nas aulas, trabalhos e provas, tudo seria mais difícil.

"Eu descobri em mim mesmo desejos os quais nada nesta Terra pode satisfazer.
A única explicação lógica é que eu fui feito para outro mundo."
C.S. Lewis

Resumo

A visualização de dados desempenha um papel importante na tomada de decisões e na compreensão das informações de qualquer organização. Isso acontece, pois ela torna os dados mais compreensíveis, úteis e acessíveis, capacitando as pessoas a extrair significado e percepções valiosas a partir de conjuntos complexos de informações. Neste sentido, este trabalho propõe a criação de um módulo de visualização de dados integrado ao Software de Análise Dermatológica (SADE).

O SADE é o software que realiza o gerenciamento do atendimento no Programa de Assistência Cirúrgica e Dermatológica da UFES (PAD-UFES). Assim, a adição do módulo de visualização de dados no sistema representa um avanço significativo à tomada de decisão dos dermatologistas, oferendo uma abordagem mais acessível e esclarecedora para analisar e interpretar os dados dos pacientes.

A implementação do módulo foi centrada na compreensão dos princípios de percepção humana e diretrizes de design. Explorou-se a integração desses princípios na construção das visualizações, visando aprimorar a interpretação dos dados dermatológicos. Isso permite que dermatologistas acessem e interpretem os dados coletados pelo programa de assistência dermatológica de forma mais eficiente. Ao empregar esse recurso, os profissionais conseguem extrair percepções relevantes, como a distribuição de tipos de pele mais frequentes, áreas corporais mais suscetíveis a lesões de pele e o comportamento etário dos pacientes

Palavras-chave: Visualização de dados. SADE. PAD-UFES. Princípios de percepção.

Lista de ilustrações

Lista de abreviaturas e siglas

INCA	Instituto Nacional de Câncer
UFES	Universidade Federal do Espírito Santo
PAD-UFES	Programa de Assistência Cirúrgica e Dermatológica da UFES
SADE	Software de Análise Dermatológica
FAPES	Fundação de Amparo à Pesquisa e Inovação em Saúde
ICEPi	Instituto Capixaba de Ensino, Pesquisa e Inovação em Saúde
API	*Application Programming Interface*
HTML	*Hyper Text Markup Language*
CSS	*Cascading Style Sheets*
UI	*User Interface*
JPA	*Java Persistence API*
CSV	*Comma Separeted Values*
CID	*Classificação Internacional de Doenças e Problemas Relacionados com a Saúde*

Sumário

1 Introdução

Nos últimos anos, a visualização de dados têm desempenhado um papel fundamental na tomada de decisões, sendo explorada em distintas áreas do conhecimento e recebendo cada vez mais atenção em estudos e pesquisas acadêmicas e profissionais (RODRIGUES; DIAS, 2017). A crescente disponibilidade de dados e a capacidade de compreender e comunicar eficazmente percepções obtidas a partir deles, têm revolucionado a forma que abordamos questões complexas e desafios de pesquisa. No contexto específico de dermatologia e tratamento de lesões de pele, a necessidade de compreender e aprimorar os serviços oferecidos é cada vez mais relevante, pois conforme o Ministério da Saúde (2023), o câncer de pele é o tipo mais comum de câncer no Brasil e no mundo. Segundo dados do Instituto Nacional de Câncer (INCA), são esperados 704 mil casos novos de câncer no Brasil para cada ano do triênio 2023-2025. Dentre os quais, o mais comum, correspondendo a 31,3% dos casos, são tumores malignos de pele do tipo não melanoma (INCA, 2022).

Motivados pelo número de casos de câncer de pele observados, em povos Pomeranos localizados no interior do estado do Espírito Santo, em 1987, um grupo de professores da Universidade Federal do Espírito Santo (UFES) criaram o Programa de Assistência Dermatológica a Lavradores Pomeranos no Espírito Santo (PAD-UFES)[1]. O programa presta assistência gratuita e completa (desde triagem, cirurgia e biópsia, se necessário) para qualquer pessoa, não apenas os Pomeranos, com algum tipo de lesão de pele. O PAD-UFES atua em 12 municípios capixabas, sendo a maioria em áreas rurais e remotas, e atende cerca de 6 mil pacientes anualmente, atuando uma vez por mês em cada um dos municípios. Uma grande parcela destes pacientes tem o PAD-UFES como a única forma de obter algum tipo de assistência dermatológica. Nesse sentido, o programa presta um serviço fundamental para milhares de capixabas, principalmente, para aqueles mais vulneráveis socialmente que não são capazes de arcar com um tratamento de saúde privado (PACHECO et al., 2023).

Durante mais de três décadas, todo o fluxo de atendimentos e documentação de dados no âmbito do PAD-UFES foram realizados de maneira manual e registrados por meio da escrita. Dessa forma, de maneira não planejada, o registro das atividades do programa foi sistematicamente desconsiderado, o que impactou negativamente na produção de publicações científicas, na disseminação do programa junto à comunidade acadêmica e ao público da UFES, na formação de parcerias com o setor privado e, acima de tudo, no aperfeiçoamento do atendimento prestado à população. No ano de 2018, uma colaboração entre laboratórios, docentes e estudantes dos cursos de Computação e Medicina da UFES resultou na concepção do Software de Análise Dermatológica (SADE) [2]. O SADE consiste

[1] Projeto de extensão n° 478. Saiba mais em: <http://pad.ufes.br/>

[2] Disponível em: <http://sade.pad.ufes.br> porém, é necessário login e senha para utilizar o sistema

em um conjunto de softwares, baseados nas plataformas web e mobile, com o propósito de gerenciar o atendimento no PAD-UFES, coletar dados de forma organizada e permitir a triagem de lesões na pele por profissionais da área de saúde. Para manter e evoluir o sistema, foi criado o PadTech, o núcleo de tecnologia vinculado ao PAD-UFES, apoiado pela Fundação de Amparo à Pesquisa e Inovação do Espírito Santo (FAPES) e o Instituto Capixaba de Ensino, Pesquisa e Inovação em Saúde (ICEPi) (PACHECO et al., 2023), a tela de login da plataforma é ilustrada na Figura 1. Apesar desse benefício, o SADE não possui mecanismos para visualizar graficamente os dados coletados, o que é um gargalo no mapeamento de algumas informações relevantes dos atendimentos, como, por exemplo, o comportamento das lesões de pele no estado do Espírito Santo.

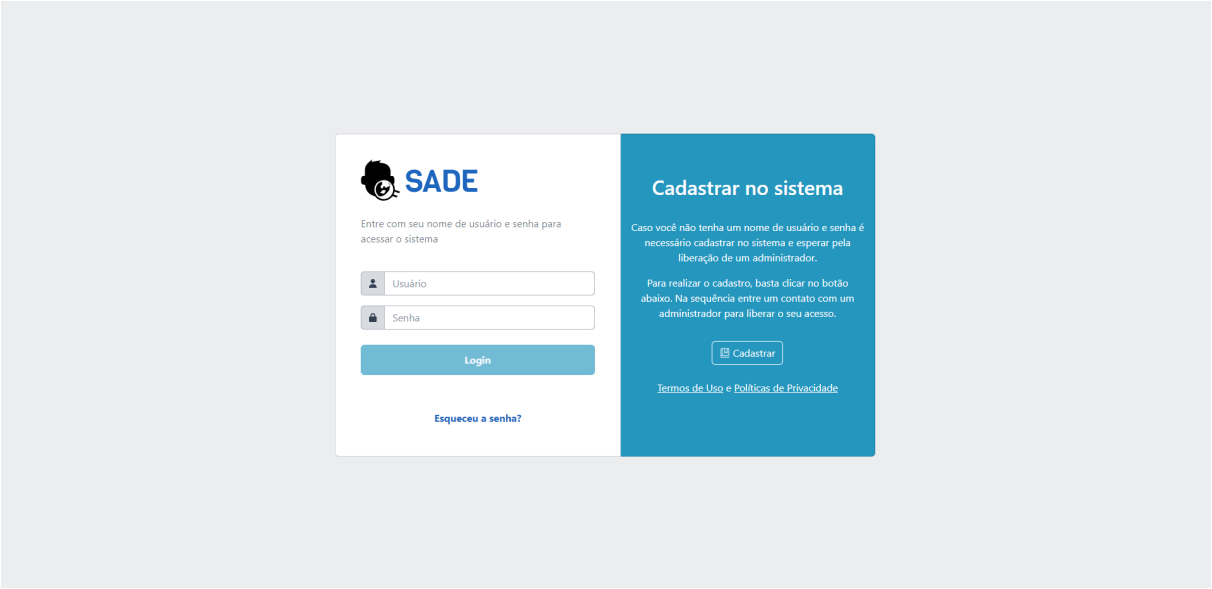

Figura 1 – Página de login da plataforma web do SADE

O principal objetivo da visualização de dados é facilitar a identificação de padrões, tendências e valores discrepantes em grandes conjuntos de dados. Ela fornece uma maneira rápida e eficaz de comunicar informações de maneira universal, usando informações visuais (BRUSH; BURNS, 2022). Uma única imagem pode conter uma riqueza de informações, e pode ser processada muito mais rápida quando comparada com páginas cheias de palavras. Isso ocorre porque a interpretação da imagem é realizada em paralelo pelo processo sequencial de leitura (WARD; KEIM; GRINSTEIN, 2015). Assim, é essencial que o SADE tenha uma ferramenta de visualização de dados, pois o uso dela possibilita aos profissionais da área da saúde do PAD-UFES uma compreensão mais clara das informações e assim, extrair percepções relevantes e alcançar conclusões acerca dos dados.

1.1 Motivação e justificativa

Diante dos benefícios que o PAD-UFES proporciona a sociedade, como, por exemplo, o oferecimento gratuito de serviços dermatológicos à população necessitada dos municípios atendidos. Torna-se clara a importância que o sistema SADE desempenha para o programa. Ele facilita o gerenciamento dos atendimentos do PAD-UFES, bem como a coleta e armazenamento dos dados dos pacientes. Contudo, como mencionado, o SADE não possui uma ferramenta de visualização de dados, o que dificulta uma compreensão rápida e precisa dos dados coletados.

Com o intuito de preencher essa lacuna, o objetivo principal deste projeto é desenvolver um módulo de visualização de dados integrado ao sistema SADE. Dessa forma, ao término do trabalho, o referido módulo irá apresentar gráficos que descrevem os dados obtidos pelo PAD-UFES, sendo gerados consoante as informações mais relevantes para os profissionais da área da saúde, como por exemplo, a distribuição dos tipos de pele mais frequentes, áreas corporais mais suscetíveis a lesões de pele e o comportamento etário dos pacientes. Possuir o conhecimento visual dos dados coletados auxiliará não apenas o entendimento da proliferação da doença no estado, mas também na dimensão do impacto que o PAD-UFES causa na sociedade. Isso auxiliará na divulgação dos resultados a comunidade e por conseguinte, na angariação de recursos para o programa junto ao poder público.

1.2 Metodologia

Para desenvolver a ferramenta de visualização de dados para o SADE, foi adotada a seguinte abordagem metodológica:

- **Revisão da literatura:** Busca e estudo de pesquisas feitas na área de visualização de dados e de percepção humana. Procurando entender não apenas os princípios da visualização de dados, mas também como as pessoas estruturam e dão significado às informações sensoriais, e como constroem uma compreensão significativa do ambiente que as cerca.
- **Definição dos requisitos:** Definição do escopo do trabalho, onde foi discutido quais dados eram de interesse dos profissionais da área de saúde, que seriam usuários do sistema, e quais eram as melhores formas de criar visualizações com esses dados.
- **Compreensão das principais ferramentas e tecnologias utilizadas no SADE:** Esta fase, consistiu em estudar as tecnologias utilizadas pelo SADE, para que assim fosse possível construir o módulo de visualização dentro do sistema. Dessa forma, foram estudadas as tecnologias tanto do *front-end* (Angular Js) como do *back-end*

(Spring boot).

- **Desenvolvimento da ferramenta:** Criação dos *endpoints* necessários no *back-end*, bem como a construção das visualizações no *front-end*.
- **Validação:** Nesta etapa o módulo desenvolvido é validado pela equipe do PAD-UFES.
- **Documentação do projeto:** Escrita da documentação do módulo desenvolvido.

1.3 Estrutura do trabalho

O restante deste trabalho é organizado da seguinte forma: no segundo capítulo, serão apresentados em detalhes os princípios de percepção e de design estudados. Em seguida, no terceiro capítulo, serão fornecidas informações sobre o SADE, e das tecnologias utilizadas para a implementação do módulo de visualização proposto neste trabalho. O quarto capítulo terá como objetivo mostrar não apenas como o módulo foi organizado, desenvolvido e integrado ao sistema SADE, mas também, explicitar e discutir os resultados obtidos. Por fim, no quinto capítulo, serão apresentadas as considerações finais deste projeto.

2 Fundamentos teóricos de visualização

Neste capítulo, introduziremos os conceitos básicos de visualização de dados. Mostraremos como os princípios da percepção humana constituem uma base sólida para o desenvolvimento de diretrizes de design eficazes. Em seguida, discutiremos como a maneira com que as pessoas interpretam informações visuais e organizam elementos, influenciam a criação de designs mais intuitivos, atraentes e funcionais. Por fim, veremos como considerar a psicologia cognitiva visando a criação de interfaces que reduzem a carga cognitiva do usuário, e com isso melhora a experiência de interação como o sistema.

2.1 Princípios da percepção humana

O físico dinamarquês Tor Norretranders criou um diagrama chamado de "Largura de banda dos nossos sentidos", ilustrado na Figura 2. Ele descreve, em termos computacionais, a quantidade de informação que cada sentido humano consegue receber por segundo. Assim, o diagrama mostra o poder da visão em comparação com nossos outros sentidos. Ele demonstra por que as visualizações são tão eficazes para transmitir grandes quantidades de informações em questão de segundos. A propósito, o ponto branco no canto inferior esquerdo representa quando nos tornamos conscientes do que acabamos de perceber (MANLIHEROVA, 2020).

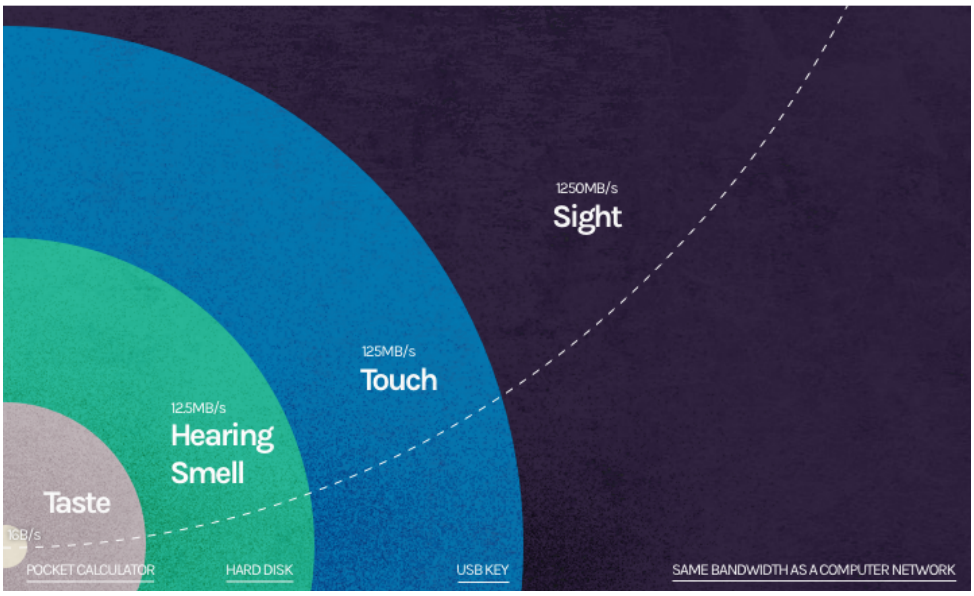

Figura 2 – A largura de banda dos nossos sentidos de acordo com Norretranders. Como pode ser observado, entre todos os sentidos humanos, a visão possui a maior capacidade de absorver informação. Fonte: (RICHARD, 2015)

Para projetar gráficos visuais eficazes, é fundamental ter uma compreensão da percepção visual e do processamento de informações pelo cérebro humano. Nesse sentido, é importante destacar o papel da memória nesse processo. Segundo Junior e Faria (2015), a memória é uma função que o cérebro humano tem de adquirir, armazenar e evocar informações. Segundo a psicologia cognitiva, existem três tipos de memória: *sensorial*, de *curto prazo* e de *longo prazo* (STERNBERG, 2000).

Logo após percebermos a informação, ela é automaticamente armazenada em nossa memória sensorial. Essa memória sensorial é uma impressão dos dados que percebemos, uma imagem mental. É uma ação independente e não requer nossa atenção (TREISMAN, 1985). Percebemos as informações muito antes de começarmos a processá-las. Esse processo é chamado de processamento pré-atentivo.

Após o processamento pré-atentivo, parte dos dados é transferida para a memória de curto prazo para um processamento adicional. Como o nome sugere, a capacidade da memória de curto prazo é limitada. As informações permanecem lá por um curto período. As informações na memória de curto prazo podem ser transferidas para a memória de longo prazo, onde podem permanecer por toda a vida mediante revisões periódicas ou da criação de associações significativas (JUNIOR; FARIA, 2015).

As características desses tipos de memória são cruciais para o design de gráficos de visualização de informações. O processamento pré-atentivo é particularmente importante porque os atributos pré-atentivos são percebidos de forma inconsciente. Também é importante lembrar que a memória de curto prazo é limitada. Portanto, não é aconselhável distribuir uma visualização em telas separadas, por exemplo.

Os atributos pré-atentivos são características visuais das formas percebidas durante o processamento pré-atentivo. Ware (2004) classificou esses atributos em quatro grupos: cor, forma, posição espacial e movimento. Um exemplo de atributo de posição espacial é a localização dos pontos de dados em um gráfico de dispersão, que representa o valor desses pontos de dados. A Figura 3 ilustra os atributos mais frequentemente utilizados dentro dessas categorias.

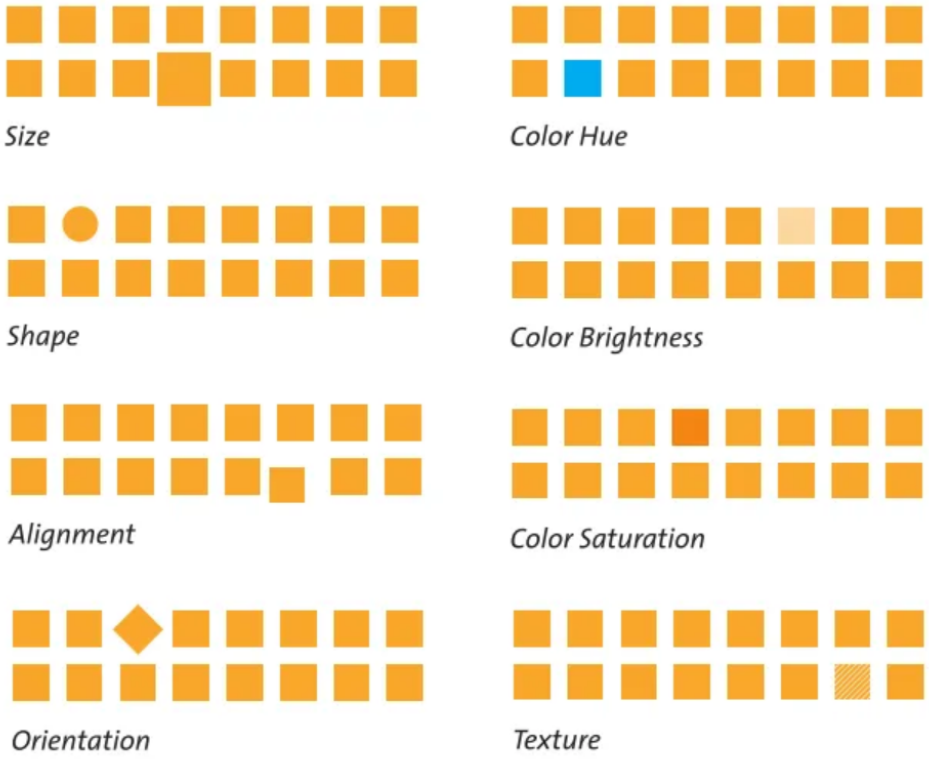

Figura 3 – Análise dos atributos pré-atentivos mais comuns. Essas características são amplamente utilizadas na representação visual de informações. Isso permite uma identificação rápida dos elementos distintos em relação ao conjunto restante. Fonte: (WARE, 2004)

Também é importante mencionar que existem atributos mais adequados para dados quantitativos ou categóricos. Esses termos estão relacionados aos tipos de dados estatísticos. Quantitativo significa que ele carrega valores numéricos. Categórico indica que ele abrange um número limitado de categorias (BUSSAB; MORETTIN, 2010). Por exemplo, a matriz da cor não é adequada para dados quantitativos. Nesse caso, o atributo de brilho deve ser aplicado.

2.2 Princípios da Gestalt

Originando-se no trabalho de Wertheimer (1923), a Gestalt é uma escola de psicologia que se concentra no estudo da percepção humana e como as pessoas organizam e interpretam as informações sensoriais para formar uma experiência significativa do mundo ao seu redor. Tendo como princípio orientador o fato psicológico de que o todo é maior do que a soma das suas partes. Ou seja, considera-se que um conjunto de informações é mais relevante do que um trecho.

A crença de que o conjunto é mais do que a simples soma das partes, levou aos estudiosos da Gestalt, como Koffka (1935) e Wertheimer (1923) a explorar uma série

de outros fenômenos perceptivos. Além disso, os princípios da Gestalt contribuíram para desenvolver a concepção de que a percepção humana não se limita apenas a observar o que está presente no mundo ao nosso redor. Ela também é influenciada por uma variedade de motivações e expectativas.

Os princípios de Gestalt são ferramentas valiosas durante o design de visualização, pois elas caracterizam nossa percepção de grupos e formas. Eles podem ser utilizados para criar interfaces que priorizem uma boa experiência do usuário. A seguir, são apresentados os principais princípios e uma breve descrição de cada um deles.

2.2.1 Princípio da proximidade

O princípio da proximidade estabelece que os objetos que estão próximos uns dos outros parecem estar mais relacionados do que se estivessem distantes. Essa relação visual é algo que encontramos com frequência em nosso cotidiano. Na Figura 4 ambas as imagens possuem os mesmo elementos, mas a mudança na proximidade faz com que os grupos e relações sejam percebidos de forma diferente (CHANG; NESBITT, 2006).

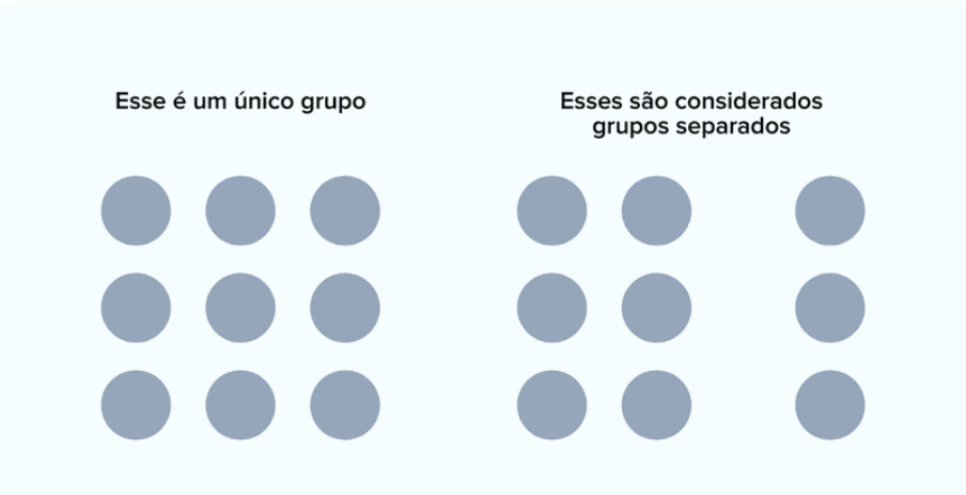

Figura 4 – Princípio da proximidade. Nesta figura podemos ver como a alteração da proximidade entre os elementos da direita faz com que percebemos dois grupos ao invés de um. Fonte: (AELA, 2020)

2.2.2 Princípio da similaridade

O princípio da similaridade diz que elementos tendem a ser agrupados se seus atributos forem percebidos como relacionados. Na Figura 5, podemos observar um conjunto de elementos que, devido à variação nas cores, são percebidos como pertencentes a um grupo distinto (CHANG; NESBITT, 2006).

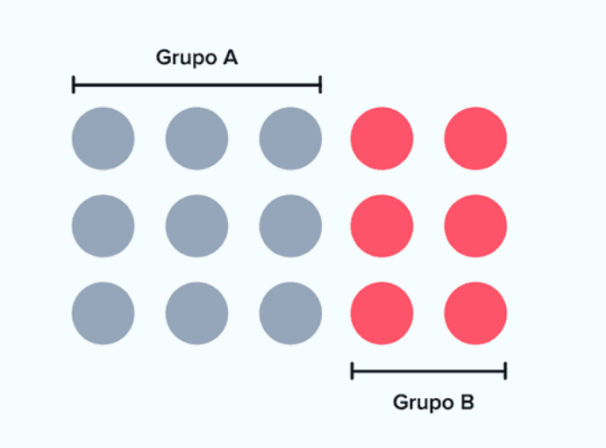

Figura 5 – Princípio da proximidade. Na imagem, é possível reparar como a cor diferencia os itens em dois grupos diferentes. Fonte: (AELA, 2020)

2.2.3 Princípio da continuidade

O princípio da continuidade, por outro lado, descreve a tendência que as pessoas têm de perceber um contorno suave e contínuo entre pontos, em vez de linhas com mudanças bruscas ou irregulares na direção. Assim, os elementos serão agrupados se um padrão contínuo puder ser interpretado e este padrão será assumido como contínuo, mesmo que algumas partes estejam ocultas (CHANG; NESBITT, 2006). Na Figura 6, apesar da distinção de cores sugerir um agrupamento por cor, o princípio da continuidade prevalece. Tendemos a perceber os círculos como duas linhas que se cruzam.

Figura 6 – Princípio da continuidade. Como pode ser observado, o princípio da continuidade prevalece sobre a tendência de agrupar os itens por cor. Fonte: (HASANOV, 2015)

2.2.4 Princípio do fechamento

O princípio do fechamento descreve nossa tendência natural de preencher informações ausentes quando um sinal é percebido. Para completar formas incompletas, as pessoas ignoram lacunas, preenchendo as partes faltantes com um padrão familiar para completar a forma (CHANG; NESBITT, 2006). Um bom exemplo do princípio do fechamento pode ser visto na figura 7. Embora o panda na imagem esteja incompleto, é apenas o suficiente para reconhecermos um panda.

Figura 7 – Princípio do fechamento no logotipo WWF *for Nature*. Podemos observar que embora a figura não esteja completamente fechada, tendemos a perceber plenamente a figura de um panda. Fonte: (HASANOV, 2015)

2.2.5 Princípio da figura de fundo

É natural para os seres humanos distinguirem entre o primeiro plano e o plano de fundo quando recebem informações. O princípio da figura de fundo afirma que nossa percepção instintivamente categoriza objetos como estando no primeiro plano ou no plano de fundo. Como seres humanos, não somos capazes de focar simultaneamente no primeiro plano e no plano de fundo, precisamos selecionar apenas um deles (CHANG; NESBITT, 2006). A Figura 8 retrata este princípio.

Figura 8 – Princípio da figura de fundo. Nesta figura é possível perceber duas imagens distintas, em um plano podemos identificar uma taça e no outro o rosto de duas pessoas. Fonte: (AELA, 2020)

2.2.6 Princípio do ponto focal

O princípio do Ponto Focal refere-se a elementos em uma visualização que captam a atenção de uma pessoa. As pessoas perceberão os elementos como pontos focais se os atributos desses elementos forem significativamente diferentes dos outros. (CHANG; NESBITT, 2006). A Figura 9 retrata este princípio.

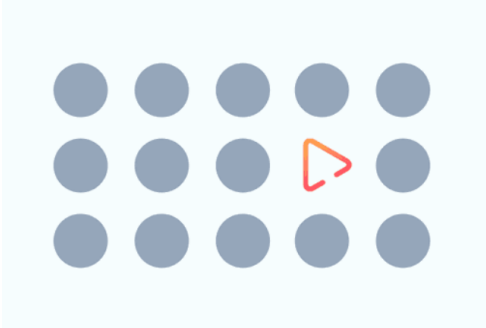

Figura 9 – Princípio do ponto focal. Nesta figura é possível reparar que nossa atenção é direcionada para o elemento diferente do conjunto. Fonte: (AELA, 2020)

2.2.7 Princípio da região comum

O princípio da região comum na Gestalt refere-se à nossa tendência de perceber objetos que estão dentro da mesma região ou área fechada como parte de um grupo coeso. Essa percepção ocorre porque nosso cérebro tende a agrupar elementos que compartilham características espaciais próximas, como proximidade e continuidade. (CHANG; NESBITT, 2006). A Figura 10 retrata este princípio.

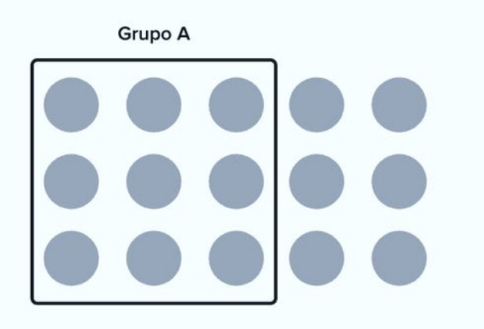

Figura 10 – Princípio da região comum. Como pode ser observado, os elementos externos ao quadrado não são percebidos como parte do grupo. Fonte: (AELA, 2020)

2.3 Princípios de visualização

Os princípios de percepção, mostrados na sessão anterior, oferecem uma base sólida para o desenvolvimento de diretrizes de design eficazes. Ao compreendermos como nosso sistema visual processa as informações, somos capazes de criar experiências visuais que são intuitivas e agradáveis. Esta sessão ressaltará a importância de incorporar os princípios da percepção, na prática do design.

2.3.1 Princípio da tinta de dados

O princípio da tinta de dados definido por Tufte (2001) refere-se ao princípio de minimizar o uso de elementos visuais supérfluos ou não essenciais em uma visualização de dados, a fim de dedicar a maior quantidade possível de espaço à representação direta dos dados. Ele argumenta que muitas visualizações de dados são poluídas por elementos decorativos, como bordas excessivas, sombras grades e efeitos visuais desnecessários. Esses elementos visuais consomem espaços desnecessários e atrai a atenção dos usuários. Por isso é importante deixar a visualização o mais simples possível e retirar elementos que não representem nenhum dado.

Na Figura 11, é possível ver um exemplo de quando o princípio da tinta de dados é violado. No gráfico da esquerda, vemos que a inclusão da terceira dimensão não acrescenta nenhuma informação relevante aos dados. Pelo contrário, essa adição desnecessária pode distrair e até mesmo transmitir informações incorretas aos usuários. Portanto, é recomendado remover esse terceira dimensão, como exemplificado no gráfico da esquerda da Figura 11.

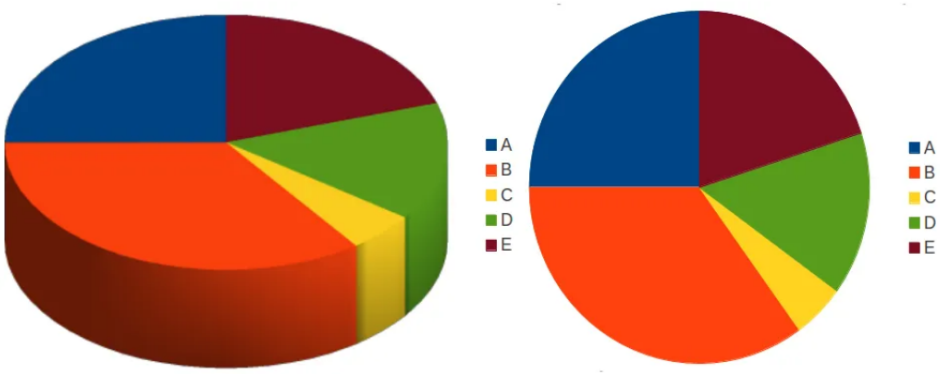

Figura 11 – O princípio da tinha de dados, violado (esquerda) e cumprido (direita). Fonte: (HASANOV, 2015)

2.3.2 Tornar a visualização inclusiva e esteticamente agradável

Os seres humanos, em geral, apreciam um design esteticamente agradável. Um design atrativo contribui para a retenção de informações e é capaz de diferenciar um gráfico em um ambiente de informações concorrentes. Neste contexto, as cores desempenham um papel significativo na visualização de dados, elas são usadas para realçar dados de relevância e representar conexões entre diferentes tipos de informações. Além disso, elas desempenham um papel crucial na orientação visual do observador e têm a capacidade de evocar emoções com base nos princípios da psicologia das cores (CRAVIT, 2022).

Fazer escolhas apropriadas em relação às cores auxilia a construção de visualizações não apenas agradáveis, mas também inclusivas. Isso por que é preciso considerar que existem pessoas com limitações visuais, como o daltonismo. O daltonismo é uma condição visual que afeta a capacidade de distinguir cores com precisão. Ela afeta aproximadamente 1 em 12 homens (8%) e 1 em 200 mulheres (0.5%). Estima-se que existam cerca de 300 milhões de pessoas com daltonismo em todo o mundo (Colour Blind Awareness, 2023). O daltonismo ocorre quando o tecido sensível à luz na parte posterior do olho - a retina - não responde adequadamente às variações nos comprimentos de onda de luz que permitem às pessoas verem cores diferentes.

Com isso, a utilização de paletas de cores, que incluem - com certa limitação - pessoas daltônicas na visualização de dados, desempenha um papel fundamental na acessibilidade e na eficácia das representações visuais. Embora muitos possam acreditar que isso limite a criatividade e a estética das visualizações, na verdade, não há motivo para não podermos criar representações visualmente agradáveis, mesmo levando em consideração a necessidade de inclusão. Existem algumas estratégias que permitem combinar inclusão e estética. Uma delas é usar paletas de cores com alto contraste entre os tons ou escolher um conjunto de cores que seja inequívoco tanto para daltônicos quanto para não daltônicos (FERREIRA, 2017), o que não apenas ajuda as pessoas com daltonismo, mas também pode criar um visual

dinâmico e atraente. Além disso, podemos adicionar informações extras às visualizações. Podemos, por exemplo, passar informações utilizando símbolos, linhas pontilhadas e diversos tipos de hachura. No exemplo mostrado na Figura 12 podemos ver a aplicação de algumas dessas diretrizes.

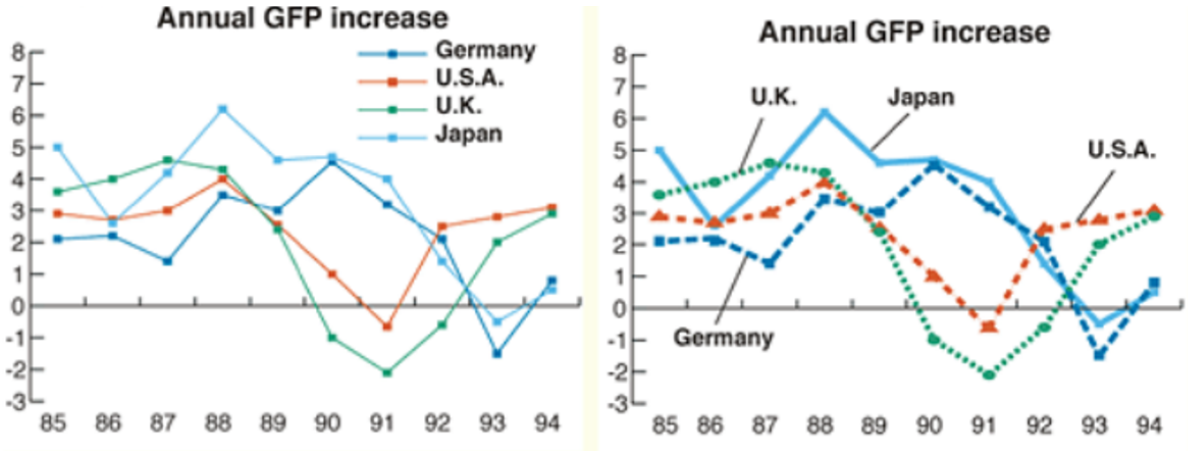

Figura 12 – Exemplo de aplicação de estratégias de inclusão para pessoas daltônicas: no lado esquerdo é necessário associar o nome da cor à legenda, isso é prejudicial para uma pessoa daltônica, pois ela pode não ver a cor exata ou não conseguir distingui-la entre as outras cores. No lado direito, para cada linha, é associado um padrão e a legenda aponta diretamente para a linha. Isso faz com que o gráfico seja mais facilmente discernível por todos. Fonte: (FERREIRA, 2017)

3 O Software de Análise Dermatológica (SADE)

Neste capítulo, serão apresentados todos os conceitos que foram necessários serem aprendidos e desenvolvidos para a criação do módulo de visualização proposto. Em resumo, será descrito como o SADE funciona, mostrando sua arquitetura e as tecnologias utilizadas em sua construção. Em seguida, será abordado a construção e a integração do módulo de visualização.

3.1 Arquitetura e funcionalidades

O SADE é um sistema web desenvolvido para servir como alicerce de suporte ao PAD-UFES. Sua abrangência vai além do ambiente acadêmico, encontrando aplicação tangível no mundo real, onde profissionais da saúde ligados à universidade e agentes estaduais e municipais o utilizam amplamente. Destaca-se na gestão e armazenamento de dados, servindo como repositório seguro para informações vitais de milhares de pacientes, sendo um aliado crucial na tomada de decisões ágeis e embasadas na área da saúde. O sistema é mantido pelo PadTech, que é o núcleo de tecnologia vinculado ao PAD-UFES, que conta com uma equipe diversificada de programadores engajados na melhoria contínua do sistema. O código do SADE está disponibilizado na organização do GitHub do projeto, cujo acesso só é permitido para membros. O código não é aberto, pois lida diretamente com dados sensíveis dos pacientes atendidos.

O sistema possui dois componentes essenciais: uma para o gerenciamento e coleta de dados, e outra para a triagem de lesões. A primeira parte é utilizada durante os atendimentos realizados pelos profissionais de saúde vinculados ao PAD-UFES, tornando o processo 100% informatizado. As principais funcionalidades desta seção englobam: o cadastro e gerenciamento de pacientes e profissionais de saúde, agendamento de consultas, gerenciamento de consultas, administração de cirurgias, geração automática de relatórios, criação de prontuários médicos, prescrição de receitas, preenchimento de formulários histopatológicos, gestão da fila de impressão de documentos gerados durante consultas e/ou cirurgias, e análise exploratória de dados. A Figura 13 ilustra uma página do software de gerenciamento e coleta de dados, onde o módulo de visualização de dados foi incluído.

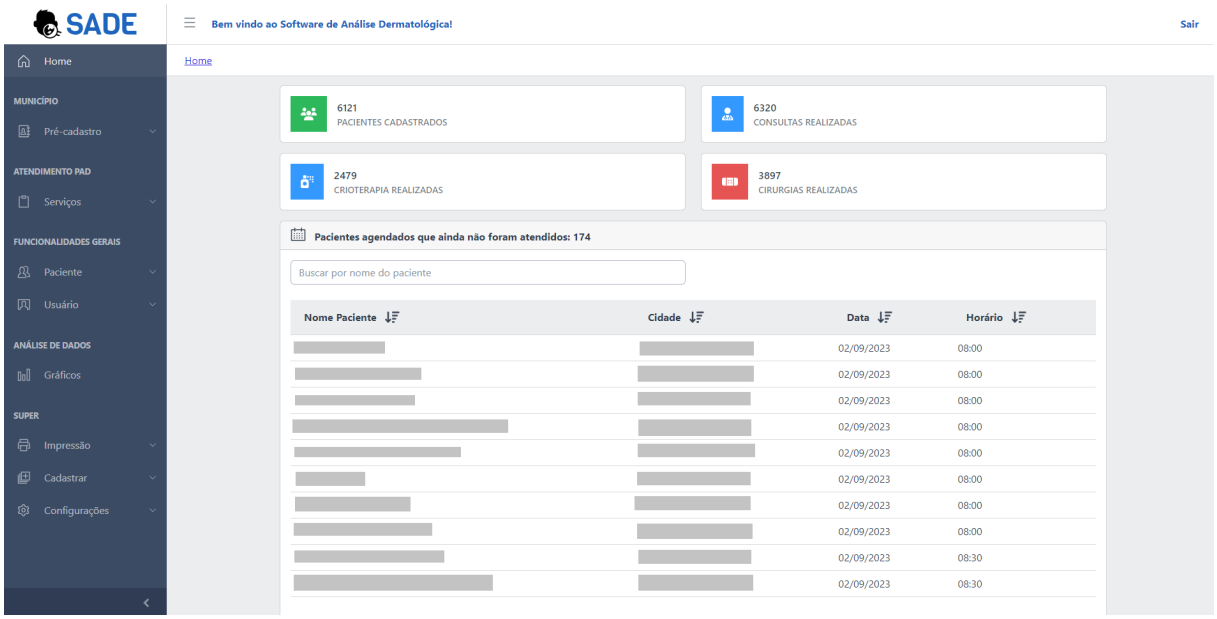

Figura 13 – Página inicial da plataforma web do SADE, os nomes e cidades dos pacientes foram ocultos nesta imagem para manter a privacidade dos pacientes.

A função de coleta do SADE está disponível para os profissionais de saúde por meio de uma plataforma web. Isso permite que o sistema seja agnóstico ao sistema operacional utilizado. Grande parte do sistema é desenvolvido utilizando dois frameworks: Angular[1] e Spring Boot[2]. O Angular é um framework de código aberto desenvolvido pelo Google para a criação de aplicativos da web dinâmicos e robustos. Ele utiliza a linguagem de programação TypeScript e oferece uma estrutura para o desenvolvimento de interfaces de usuário interativas e componentes reutilizáveis. O Spring Boot é uma extensão do framework Spring que visa simplificar o desenvolvimento de aplicativos Java, com foco em criar APIs de forma eficiente. Ele fornece configurações padrão e uma série de funcionalidades predefinidas para acelerar o processo de desenvolvimento.

Como banco de dados é utilizado o MySQL[3], que é um sistema de gerenciamento de banco de dados relacional de código aberto amplamente utilizado pela indústria. Além disso, a aplicação é orquestrada utilizando *containers* Docker[4]. Com ele é possível empacotar aplicativos e suas dependências, o que torna a implantação da aplicação consistente e escalonável em ambientes diferentes. Por fim, a aplicação é servida utilizando Nginx[5], que é um servidor web de código aberto conhecido por sua eficiência e desempenho.

1 <https://angular.io/>
2 <https://spring.io/projects/spring-boot>
3 <https://www.mysql.com/>
4 <https://www.docker.com/>
5 <https://www.nginx.com/>

3.1.1 Front-end

O *front-end* se refere à parte visível e interativa de um aplicativo ou site, com a qual os usuários interagem diretamente. Inclui elementos como a interface do usuário, design, layout, botões e elementos gráficos. Na maioria dos sistemas Web, existe uma aplicação *back-end* que lida com várias funções complexas, às quais o usuário não tem acesso direto. Nesse contexto, o *front-end* atua como intermediário, permitindo que o usuário interaja com os serviços oferecidos pelo sistema. Para a construção do *front-end* o SADE utiliza o Angular, que segue uma arquitetura conhecida como *Component-based*. Nesta arquitetura, a aplicação é dividida em componentes reutilizáveis, cada um com sua própria lógica, modelo e *view*. Segue abaixo uma breve descrição de como a arquitetura do Angular funciona:

- **Componentes:** Os componentes são as unidades fundamentais de construção de um aplicativo Angular. Cada componente é responsável por uma parte específica da interface do usuário e inclui uma classe TypeScript, um modelo HTML e estilos CSS associados.
- ***Model:*** O *model* ou modelo, contém os dados que um componente precisa exibir e manipular. Geralmente, o modelo é uma representação dos dados que o componente está gerenciando.
- ***View:*** A *view* é a representação visual do componente. No caso do Angular, a *view* é geralmente definida usando um modelo HTML.
- **Controller:** Em vez de um *controller* separado, o Angular utiliza a classe do componente para controlar o comportamento do componente. A lógica de controle é incorporada na classe do componente, e as ações do usuário são tratadas por meio de métodos nessa classe.

A Figura 14, ilustra a estrutura principal de pastas da plataforma web do SADE. No diretório "*views*", encontram-se todos os componentes visuais da aplicação, como o login, o *dasboard* e toda a parde de coleta e gerenciamento das informações coletadas pelo PAD-UFES. Em "*models*", estão armazenadas todas as classes que definem os modelos de dados ligados a entidades no *back-end*. Além disso, também podemos destacar o diretório "*services*", que contém toda a lógica de interação com o *back-end* e o arquivo **app-routing.module.ts**, que define todas as rotas do sistema.

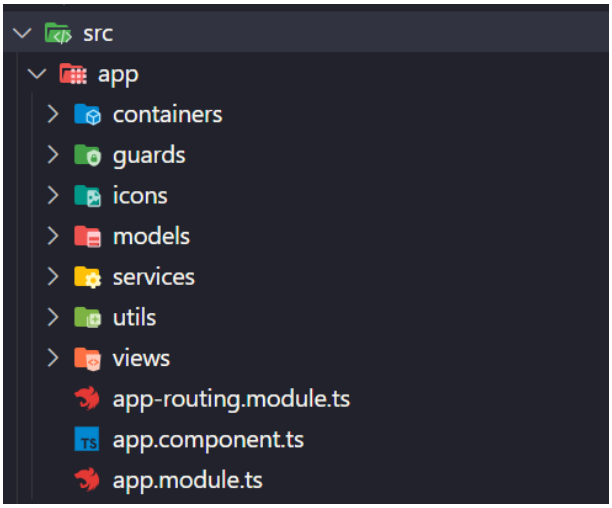

Figura 14 – Organização de pastas do *front-end* do SADE. Na imagem, podemos observar que a aplicação é dividida em várias pastas, como a de "models", que contém as classes da aplicação, e a de "views" que contém todos os componentes.

Para facilitar o desenvolvimento do *front-end*, o SADE utiliza o CoreUi for Angular[6], um modelo de interface do usuário (UI) e um kit de ferramentas de administração específico para o *framework* Angular. Essa ferramenta não apenas fornece componentes e estilos predefinidos que aceleram a construção de interfaces, mas também oferece suporte para a integração com algumas bibliotecas de criação de gráficos como a ApexCharts[7].

3.1.2 Back-end

O *back-end* é a parte não visível de um sistema, que lida com a lógica de negócios, o processamento de dados e a comunicação com o banco de dados. Ele funciona como o "cérebro" do sistema, processando solicitações do *front-end*, executando a lógica do aplicativo e retornando os resultados. A interação entre essas duas partes da aplicação, é normalmente realizada via API's, que permitem a realização de requisições para o sistema. Para a construção do *back-end*, o SADE utiliza o Spring Boot. Segue a baixo um resumo da arquitetura básica de uma aplicação Spring Boot.

- **Spring Boot Starter:** O desenvolvimento em Spring Boot começa geralmente com a inclusão de dependências específicas, chamadas de *starters*. Os *starters* são pacotes pré-configurados que fornecem funcionalidades comuns, como persistência de dados, segurança, web, etc.
- **Configuração Automática:** O Spring Boot utiliza o conceito de configuração automática para fornecer configurações padrão baseadas nas dependências presentes no

[6] <https://coreui.io/angular>
[7] <https://apexcharts.com/>

classpath. Isso significa que muitas configurações podem ser feitas automaticamente, reduzindo a necessidade de configuração manual.

- **Contêiner Embutido:** Spring Boot inclui um contêiner embutido (como Tomcat, Jetty ou Undertow) que permite executar a aplicação sem a necessidade de um servidor externo.
- **Convenções de Estrutura de Diretórios:** O Spring Boot segue convenções de estrutura de diretórios. Por exemplo, classes controladoras, modelos, e componentes estão geralmente organizados em pacotes específicos.
- **Injeção de Dependência:** O Spring Boot utiliza fortemente o conceito de Injeção de Dependência, facilitando a configuração e gerenciamento de componentes.
- **Propriedades de Configuração:** As configurações da aplicação podem ser definidas em arquivos **application.properties** ou **application.yml**. Essas propriedades podem ser usadas para ajustar o comportamento da aplicação.

Essa é uma visão geral da arquitetura de uma aplicação Spring Boot. É importante mencionar que o Sprint Boot é altamente configurável e pode ser estendido para atender a uma variedade de requisitos de aplicativos. A Figura 15 ilustra a estrutura principal de pastas do *back-end* do SADE. É possível ver todas as partes mais importantes da aplicação, como a pasta "*controller*", que atua como um ponto de entrada para as requisições, sendo responsável por definir os *end-points* da API. Os controladores interagem com a camada de serviço, representada pela pasta "*service*", para delegar a lógica de negócio. A pasta "*repository*" é responsável por interagir com o banco de dados, é nela que as pesquisas são realizadas. Por fim, também é possível observar que a aplicação possui partes específicas responsáveis pelas configurações, segurança e filtros.

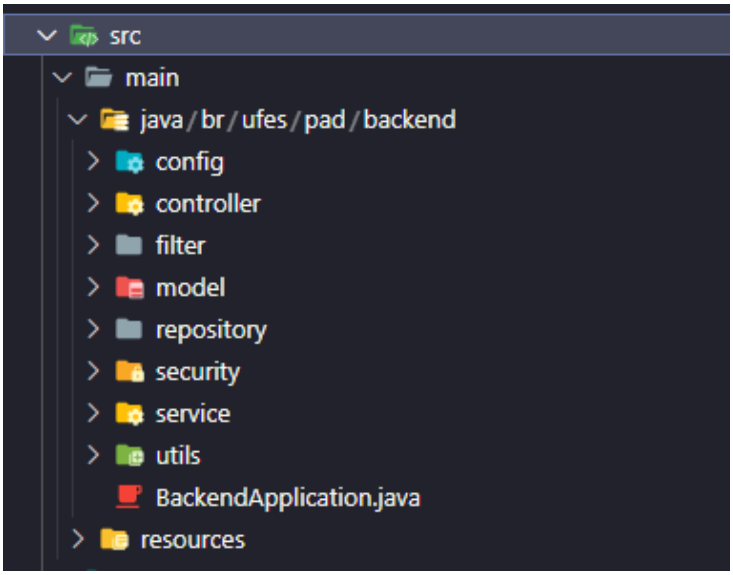

Figura 15 – Organização de pastas do *back-end* do SADE. Na imagem, podemos observar as principais partes da aplicação, como o diretório o "*model*" que define as entidades existentes no banco de dados.

4 Módulo de visualização de dados do SADE

Neste capítulo, será apresentada uma visão detalhada da plataforma de visualização de dados desenvolvida neste trabalho, mostrando não apenas a organização do *front-end* e do *back-end*, mas também os gráficos criados. Cada visualização será acompanhada por uma descrição detalhada dos dados que estão sendo apresentados e das conclusões que podem ser extraídas a partir deles.

4.1 Organização do módulo no *front-end*

Para dar início ao desenvolvimento do módulo de visualização proposto, tornou-se imperativo criar um componente dedicado a ele na seção de "*views*". Chamamos este componente de "data-analysis". Como um módulo de visualização é constituído por diversos elementos de interface, tais como gráficos e opções de configuração para os mesmos, foi necessário segmentar a aplicação em diferentes componentes. A pasta "*components*", mostrada na Figura 16, abriga todos esses elementos da interface, enquanto a pasta "*interfaces*" inclui todas as interfaces, os quais são formas de definir a estrutura dos objetos de dados com as quais a aplicação interage.

A pasta "*utils*", por sua vez, engloba todas as funções auxiliares necessárias para a criação dos gráficos, incluindo o arquivo de definição das paletas de cores do projeto. Os arquivos **data-analysis.component.html** e **data-analysis.component.ts**, possuem respectivamente o esquema geral do HTML e da lógica da aplicação; os outros dois são responsáveis pelo roteamento e pela importação de todas as dependências necessárias para o funcionamento do componente.

Cada gráfico criado dentro da pasta "*components*" possui sua própria lógica e seu próprio HTML, podendo ser individualmente customizado para melhor atender as necessidades dos usuários. Alguns gráficos possuem funções auxiliares - armazenadas na pasta "*utils*" - necessárias para sua criação. Em alguns casos, essas funções são importantes para diminuir a complexidade da lógica contida no arquivo principal de cada gráfico, aumentando assim a leitura do código e sua manutenibilidade.

Como citado na seção 3.1, o SADE não possui seu código aberto, pois o sistema lida com dados sensíveis dos pacientes do programa. Assim, para obter uma compreensão mais profunda da implementação do *front-end* do módulo, basta consultar os apêndices deste trabalho. Lá é possível encontrar uma seção que detalha essa implementação.

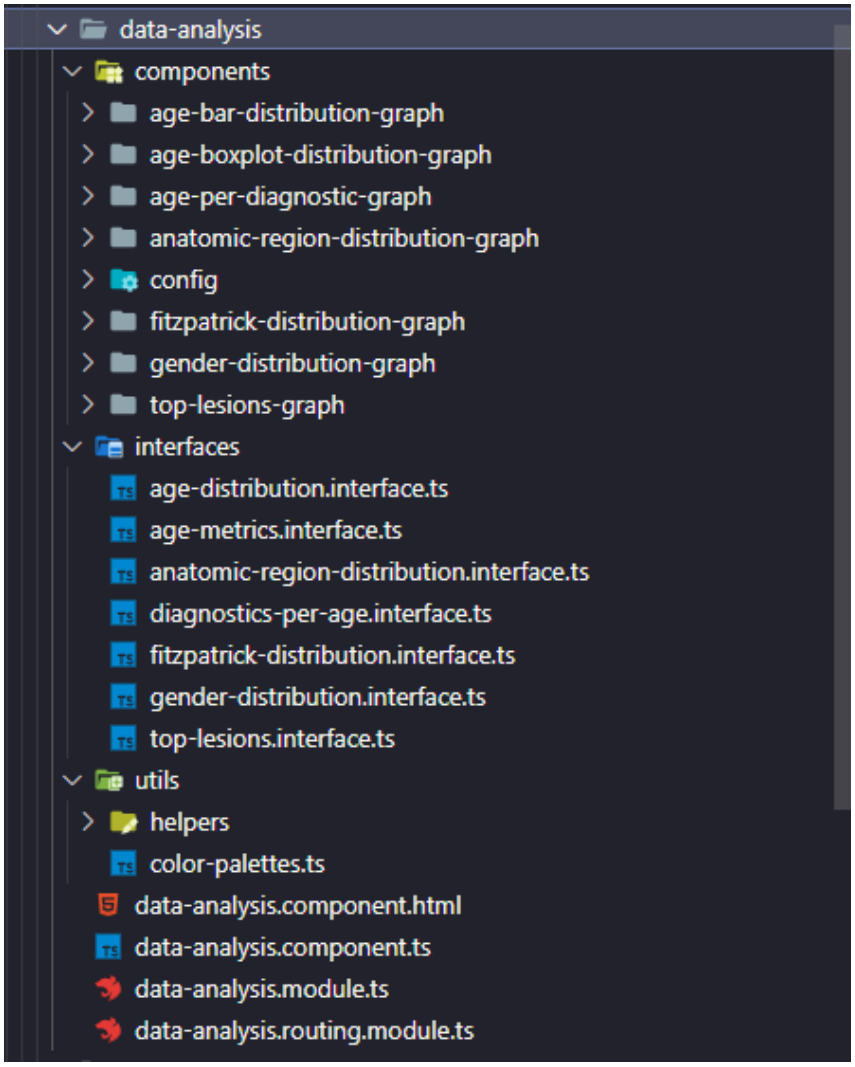

Figura 16 – Estrutura do componente "data-analysis", ele contém todo o código do módulo de visualização.

4.2 Organização do módulo no *back-end*

A Figura 15 mostra como está configurado a estrutura de pastas do *back-end* do SADE. É possível reparar que a arquitetura do Spring Boot é diferente da arquitetura vista no Angular. Existem pastas específicas para cada camada da aplicação. Nesse contexto, para que o desenvolvimento dos recursos necessários para que a construção dos gráficos no *front-end* fosse possível, foi preciso criar os arquivos correspondentes nas camadas do *back-end*.

Conforme o discutido na seção 3.1.2, a camada dos controladores é responsável pela gerência de tráfego de entrada e saída da API. Dessa forma, é indispensável criar um controlador, dentro da pasta "controller", para gerenciar os *endpoints* necessários para a construção dos gráficos. Para isso criamos o arquivo **DataAnalysisController.java** que possui os seguintes *endpoints* do tipo **GET**:

- **/data-analysis/gender-distribution:** Responsável por entregar quantidade total de pessoas cadastradas por gênero. O banco de dados possui informações de pessoas com gênero masculino, feminino e outros. Todas essas informações são trazidas pelo endpoint.
- **/data-analysis/age-distribution:** Traz a distribuição etária dos pacientes atendidos pelo programa.
- **/data-analysis/diagnostics-per-age:** Este *endpoint* traz uma lista de valores contendo o diagnóstico clínico, a idade do paciente, e quantos pacientes daquela idade tiveram este diagnóscito.
- **/data-analysis/fitzpatrick-distribution:** Responsável por devolver a distribuição de tipos de pele existentes no banco de dados.
- **/data-analysis/top-lesions:** Traz a quantidade dos casos registrados no banco de dados dos tipos de lesões de pele mais comuns atendidas pelo SADE. São elas: Melanoma, Carcinoma Espinocelular, Ceratose Actinica, Ceratose Seborreica e Nevo. Essas lesões foram mapeadas como as lesões mais comuns pela equipe do SADE.
- **/data-analysis/anatomic-region-distribution:** Este *endpoint* retorna a distribuição de lesões por região anatômica. As regiões retornadas são: Braço, face, perna, tronco, costas e ombros.

Para que esses *end-points* fossem criados, foi necessário criar uma pesquisa SQL específica para cada um deles. As pesquisas foram criadas nos arquivos **LessionRepository.java** e **PatientRepository.java**, que são *repositories* das tabelas que armazenam, respectivamente, os dados de lesões e de pacientes no banco de dados. Analogamente ao que foi feito para o *front-end*, nos apêndices do trabalho é possível encontrar uma seção que detalha a implementação do *back-end*.

4.3 Justificativa dos gráficos

Os gráficos criados neste trabalho foram desenvolvidos em estreita colaboração com os profissionais da saúde do PAD-UFES e representam um marco significativo para o SADE. Cada gráfico foi meticulosamente gerado com base nos interesses e necessidades desses especialistas, enfatizando as informações mais pertinentes e cruciais para o seu contexto profissional. Essa abordagem colaborativa assegura que os gráficos sejam ferramentas úteis para a análise e tomada de decisões. Foram criados os seguintes gráficos:

- Gráfico de distribuição de gênero.

- Gráfico de distribuição etária - boxplot.
- Gráfico de distribuição etária - barras.
- Gráfico de lesões mais comuns.
- Gráfico de distribuição de lesões por região anatômica.
- Gráfico de distribuição de etária por diagnóstico.

Cada um desses gráficos é explicado na próxima seção.

4.4 Visualização da Plataforma

A Figura 17 apresenta a integração do módulo de visualização de dados no sistema SADE. Para acessar essa nova funcionalidade no site, foi incluída a opção "Gráficos" no menu lateral esquerdo da aplicação. Ao selecioná-la, inicialmente não se exibe nenhum gráfico, apenas as opções de configuração. Estas consistem em escolher as visualizações desejadas e a paleta de cores a ser utilizada. Após definir as configurações desejadas e clicar em "Salvar", os gráficos são exibidos no restante da página.

No *box* de configurações é possível perceber como alguns dos princípios de percepção, discutidos no capítulo 2, são aplicados. Podemos ver como os atributos pré-atentivos, como a cor e o formato dos elementos - aplicados no *card* de informação e no botão de salvar, por exemplo - permitem uma rápida distinção dos elementos em relação ao conjunto restante. Vemos também como o princípio da proximidade (vide seção 2.2.1) e da similaridade (vide seção 2.2.2) nos faz interpretar os itens da listagem de gráficos como relacionados entre si. As opções das paletas de cores também são percebidas como parte de um conjunto pelos mesmos princípios. A disponibilidade de diferentes paletas de cores, que incluem - com certa limitação - pessoas daltônicas, possibilita o cumprimento do princípio discutido da seção 2.3.2. Essas paletas desempenham um papel essencial na acessibilidade e na eficácia das representações visuais.

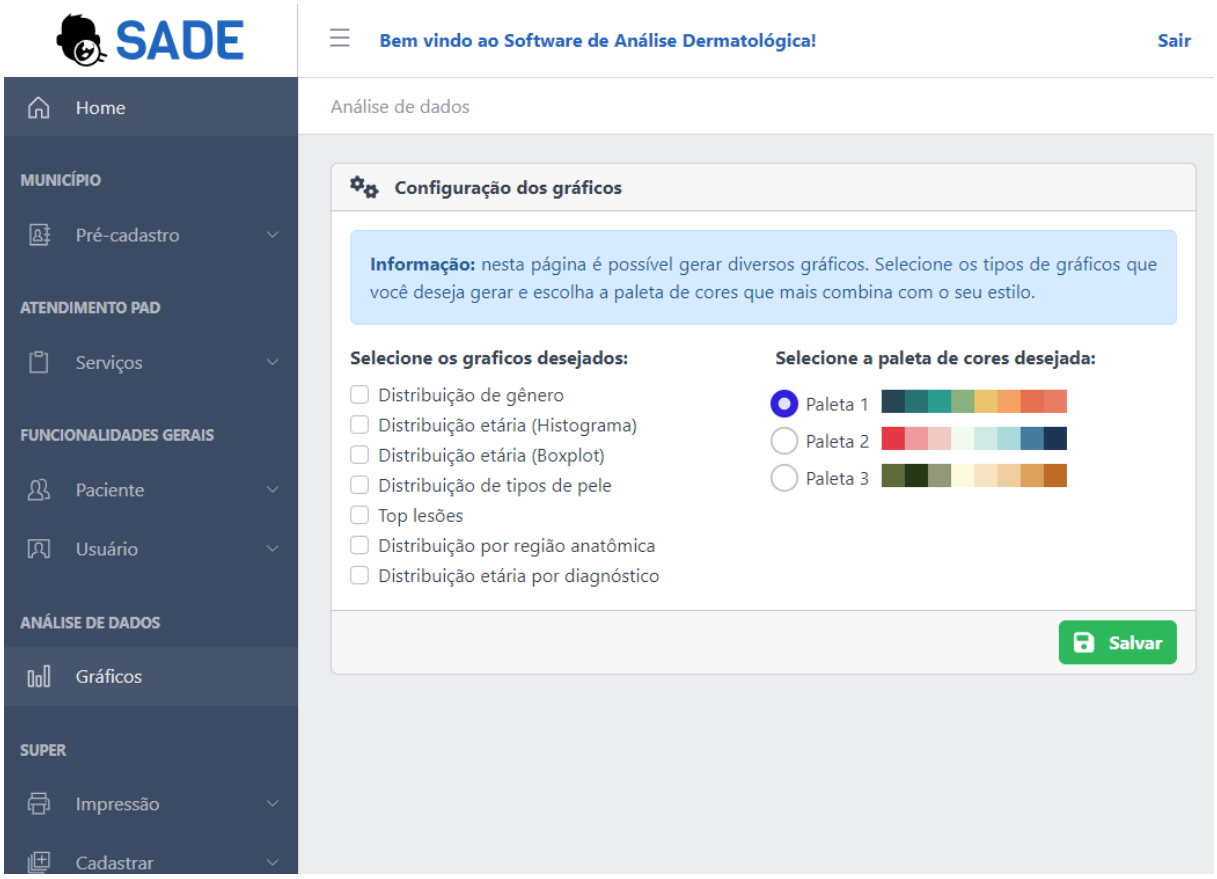

Figura 17 – Página do módulo de visualização construído. Nesta imagem podemos observar o componente de configurações dos gráficos. À esquerda do componente é possível selecionar os gráficos desejados, enquanto na direita, podemos escolher qual paleta de cor gostaríamos.

Ao selecionar uma configuração inicial para a página, os gráficos são carregados. Na parte superior do componente de visualização, estão localizados os filtros do gráfico, bem como os botões de aplicação e limpeza dos filtros. Todo componente de gráfico criado nesta página é alimentado pelo próprio *endpoint*, que possui uma *query* especificamente criada para ele (para mais informações veja o apêndice no fim do trabalho). Eles possuem a mesma estrutura principal: na parte de cima ficam os filtros, que variam dependendo do gráfico visualizado, e na parte de baixo, fica a visualização em si. Observe que cada gráfico oferece a opção de download nos formatos SVG, PNG e CSV, como é possível visualizar na Figura 18. Para acessar essa funcionalidade, basta clicar no ícone de menu localizado logo abaixo do botão de filtrar resultados. Isso oferece uma flexibilidade para o usuário caso ele pretenda realizar alguma customização no gráfico ou inserir os dados em alguma outra plataforma. Além disso, ao passar o ponteiro do *mouse* sobre algum elemento do gráfico surge uma descrição mais precisa sobre a grandeza representada por ele, como é possível ver na Figura 18, onde o elemento central é detalhado.

4.4.1 Gráfico de distribuição de gênero

No gráfico de distribuição de gênero, mostrado na Figura 18, vemos a quantidade de pessoas de cada gênero atendidas pelo programa PAD-UFES, percebe-se que o público feminino é o tipo mais atendido. Podemos afirmar que nesta visualização o princípio da tinta de dados (vide seção 2.3.1) é preservado, pois é dedicado a maior quantidade possível de espaço à representação direta dos dados e é evitado o uso de elementos visuais não essenciais.

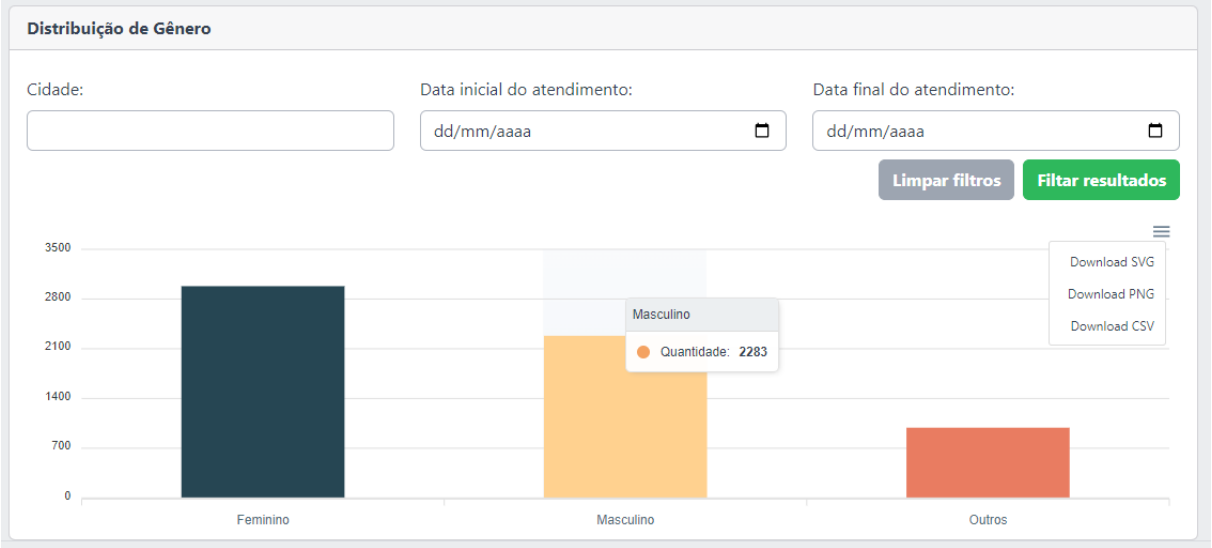

Figura 18 – Gráfico de distribuição de gênero. Nesta imagem podemos observar que existe uma clara distinção entre as cores e que existem três tipos de gênero cadastrados no banco de dados, sendo que o maior deles é o feminino. No eixo das abscissas estão os tipos de gênero, enquanto no eixo das ordenadas está a quantidade de pessoas.

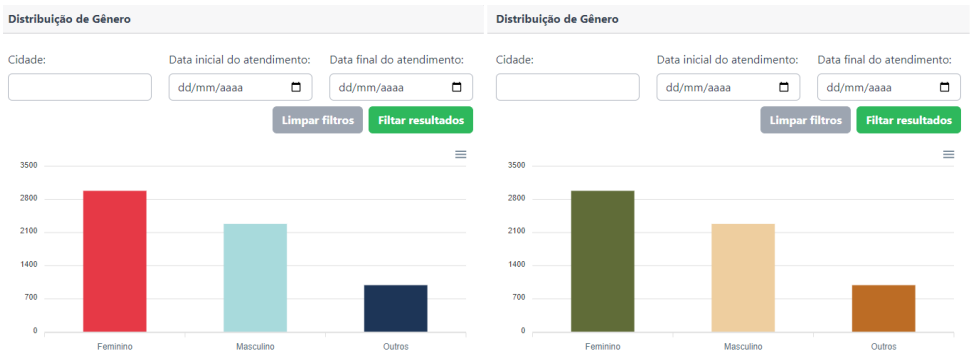

Figura 19 – Gráfico de distribuição de gênero em duas versões. A paleta 2 foi escolhida no gráfico da esquerda, enquanto no da direita a paleta 3 foi escolhida.

4.4.2 Gráfico de distribuição etária - boxplot

O gráfico de distribuição etária em no formato de boxplot, mostrado na Figura 20, fornece várias informações úteis sobre o público atendido pelo PAD-UFES. O boxplot exibe

a mediana, quartis, e a presença de *outliers* concisamente. Isso permite uma compreensão rápida da dispersão e centralidade dos dados. Vemos que o princípio da continuidade (vide seção 2.2.3) no boxplot é violado pela aparição do painel flutuante com os dados estatísticos. O painel utiliza do princípio da figura de fundo (vide seção 2.2.5), para se destacar na percepção do usuário. Um painel com dados complementares a visualização aparece em todos os gráficos desenvolvidos neste trabalho e serve para complementar a informação passada pela visualização. A Figura 21 mostra como fica a aparência do gráfico quando as outras paletas são escolhidas ao invés da primeira.

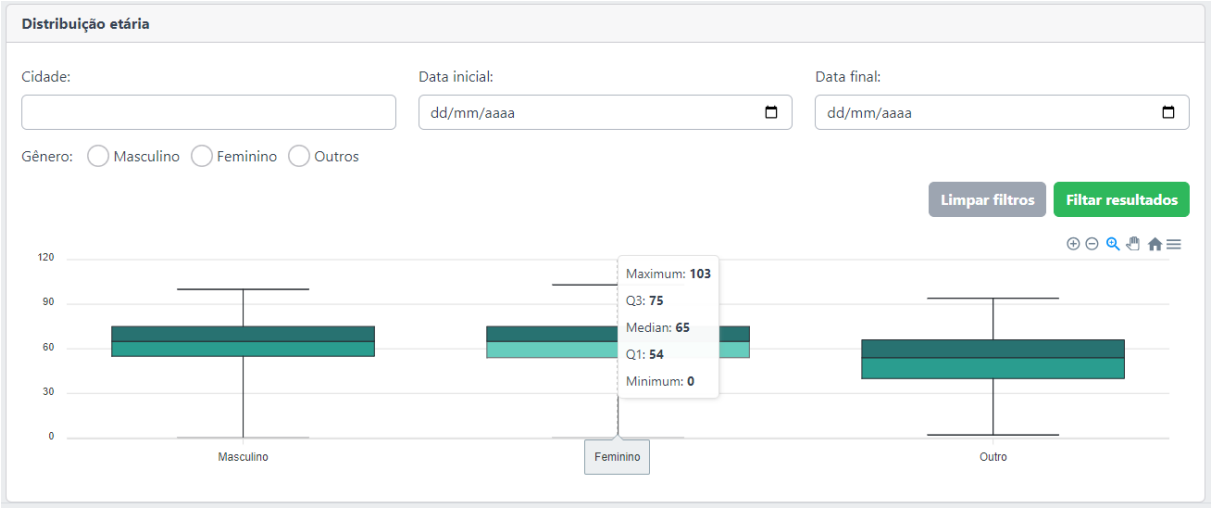

Figura 20 – Gráfico boxplot de distribuição etária. Nesta imagem, vemos alguns dados estatísticos sobre a distribuição etária do público atendido pelo PAD-UFES. No eixo das abscissas estão os tipos de gênero, enquanto no eixo das ordenadas está a idade do paciente.

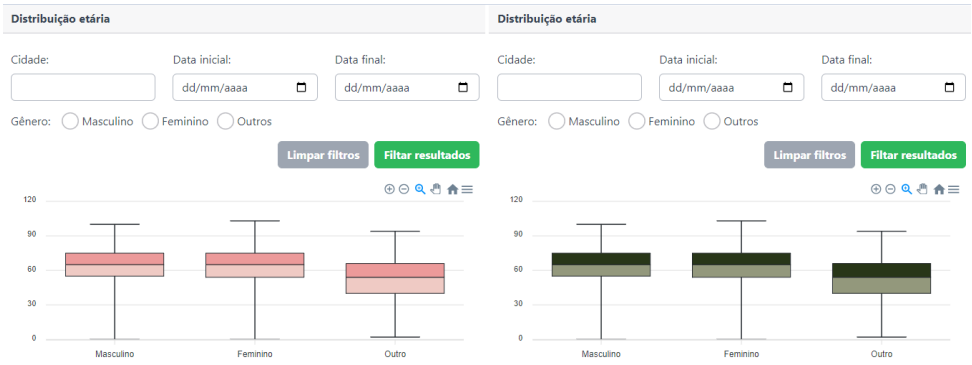

Figura 21 – Gráfico de distribuição etária em duas versões. A paleta 2 foi escolhida no gráfico da esquerda, e a paleta 3 no da direita.

4.4.3 Gráfico de distribuição etária - barras

Através do gráfico em barras de distribuição etária, é possível obter o conhecimento mais detalhado da quantidade de pessoas atendidas que possuem determinada idade, isso para saber exatamente a quantidade de pessoas por grupo etário. Na Figura 22, é possível

ver que o princípio da similaridade (vide seção 2.2.2) e da proximidade (vide seção 2.2.1) fazem com que percebamos as barras como parte do mesmo conjunto. Na Figura 23 vemos como seria aparência do gráfico caso as paletas 2 ou 3 fossem escolhidas.

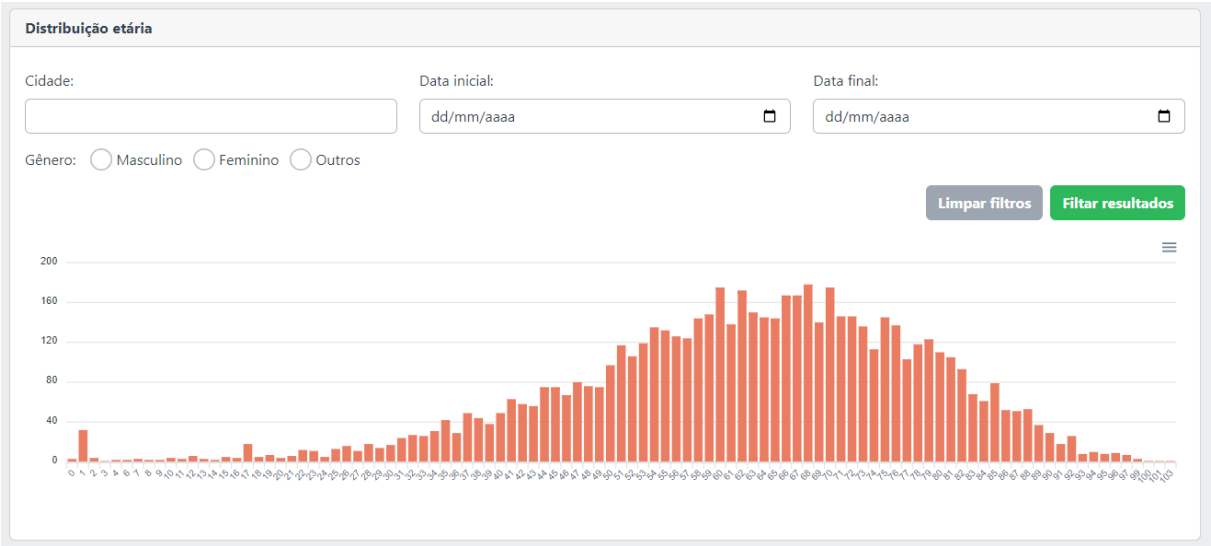

Figura 22 – Gráfico em barras de distribuição etária. No eixo das abscissas está a idade do paciente e no eixo das ordenadas está a quantidade de pacientes.

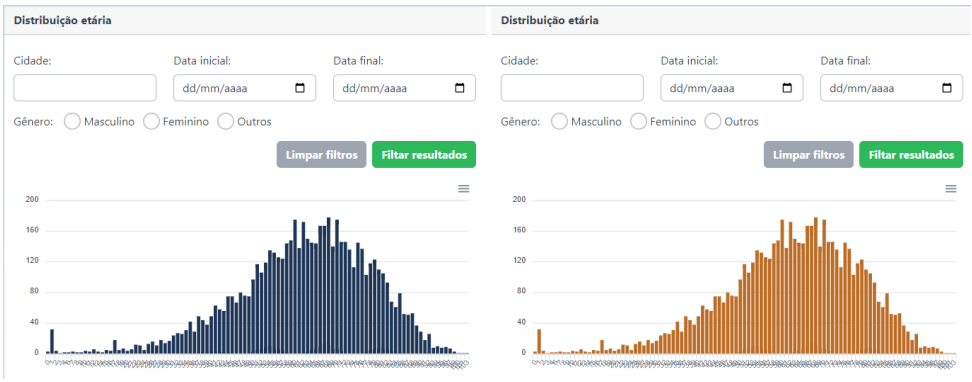

Figura 23 – Gráfico de distribuição de idade em duas versões. A paleta 2 foi escolhida no gráfico da esquerda, enquanto no da direita a paleta 3 foi escolhida.

4.4.4 Gráfico de distribuição de tipos de pele

No gráfico da Figura 24, é possível ver seis tipos de pele diferentes divididos conforme a distribuição de *Fitzpatrick* (MOTA; BARJA, 2006). Essa distribuição categoriza o tipo de pele com base na resposta dela à exposição solar. Essa classificação considera a tendência à queimação, bronzeamento e sensibilidade ao sol. Na listagem abaixo segue uma breve explicação dos seis tipos de pele.

- **Tipo 1:** Pele muito clara, sempre queima, nunca bronzeia.
- **Tipo 2:** Pele clara, queima facilmente, bronzeia minimamente.

- **Tipo 3:** Pele clara a moderada, queima ocasionalmente, bronzeia gradualmente.
- **Tipo 4:** Pele moderada, queima raramente, bronzeia facilmente.
- **Tipo 5:** Pele morena, raramente queima, bronzeia com facilidade.
- **Tipo 6:** Pele escura, nunca queima, bronzeia muito facilmente.

É possível observar que a maioria do público atendido pelo PAD-UFES possui o tipo de pele clara, concentrada basicamente nos dois primeiros tipos de pele. Esses tipos, são mais propensos a queimaduras solares e têm uma maior tendência a desenvolver câncer de pele em comparação com os tipos de pele mais escuros, como os tipos 5 e 6. A Figura 25 mostra as outras opções, disponíveis no sistema, para coloração do gráfico.

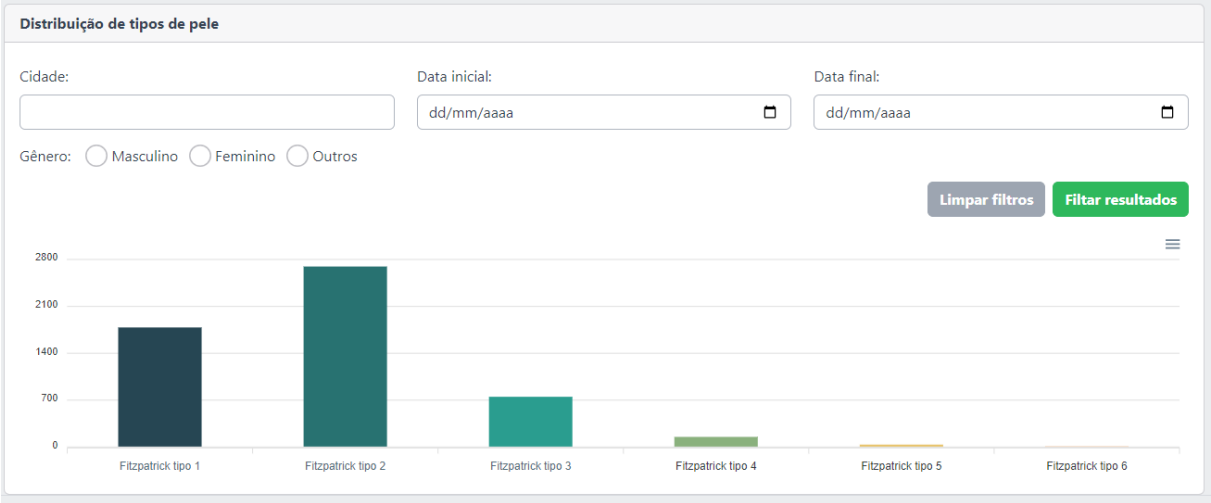

Figura 24 – Gráfico em barras de distribuição de tipos de pele. No eixo das abscissas estão os tipos de pele segundo a distribuição de Fitzpatrick, enquanto no eixo das ordenadas está a quantidade de pessoas atendidas.

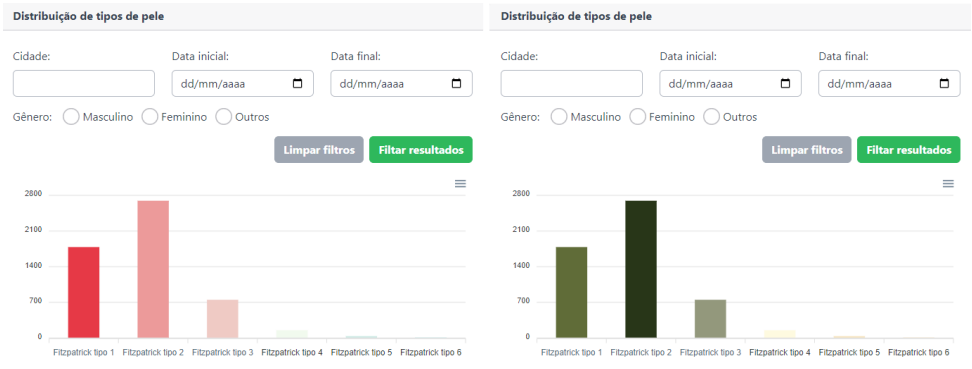

Figura 25 – Gráfico de distribuição de tipos de pele em duas versões. A paleta 2 foi escolhida no gráfico da esquerda, enquanto no da direita a paleta 3 foi escolhida.

4.4.5 Gráfico de lesões mais comuns

A Figura 26 mostra como estão distribuídos os números de casos em cada uma das principais lesões registradas na base de dados do SADE. No gráfico, é possível perceber através do princípio da similaridade (vide seção 2.2.2) que existem dois grupos de lesões diferentes, divididos pelas cores e pelo espaço. Esses dois grupos representam duas escalas diferentes, a escala do grupo de lesões da esquerda é maior que o da direita. Isso foi necessário, pois devido à distância de números de casos entre algumas dessas lesões, as lesões à direita não eram visualizadas. Assim, para conseguir expressar todos os dados, foi necessário a criação de dois grupos distintos. Os grupos são divididos conforme o número de casos da lesão. Caso a lesão possua mais de 500 casos registrados, ela vai para o grupo da esquerda. Caso contrário, ela vai para o grupo da direita. Na Figura 27 é possível ver as outras possíveis configurações de cor deste gráfico.

As principais lesões mostrados no gráfico são:

- C43 - Melanoma maligno da pele.
- D03 - Lentigo maligno.
- C44 - Carcinoma Espinocelular.
- D04 - Doença de Bowen.
- C80 - Carcinoma Basocelular.
- L57 - Ceratose Actínica.
- D22 - Nevo.
- L82 - Ceratose Seborreica.

Esta lista de lesões representa os principais diagnósticos encontrados na base de dados do SADE. Essa variedade reflete a diversidade de diagnósticos e preocupações dermatológicas que um programa de assistência desse tipo enfrenta. O código que aparece antes do nome da descrição da lesão (letra seguida de dois números), é utilizado para identificar e categorizar diferentes condições médicas ou, diagnósticos e corresponde à Classificação Internacional de Doenças e Problemas Relacionados com a Saúde(CID).

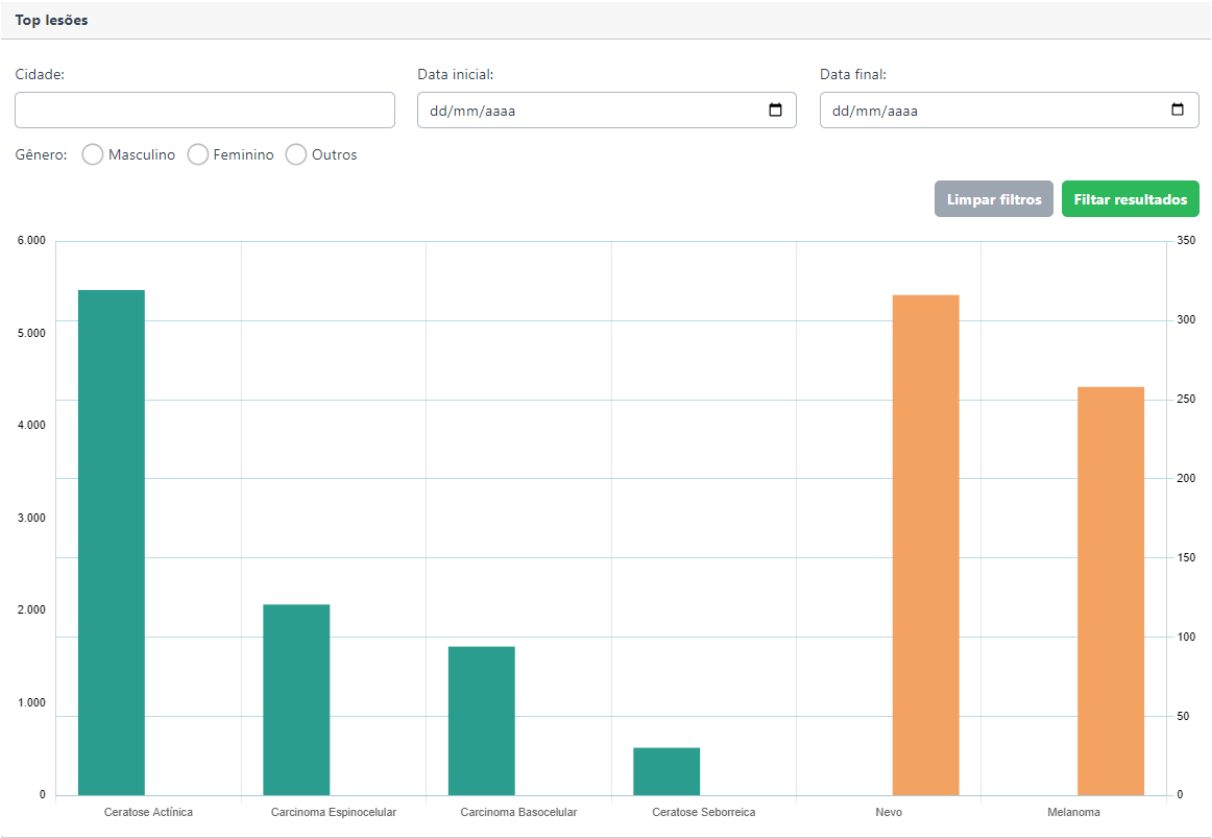

Figura 26 – Gráfico em barras de distribuição do número de casos entre as principais lesões. No eixo das abscissas estão as principais lesões de pele e no eixo das ordenadas estão a quantidade de lesões

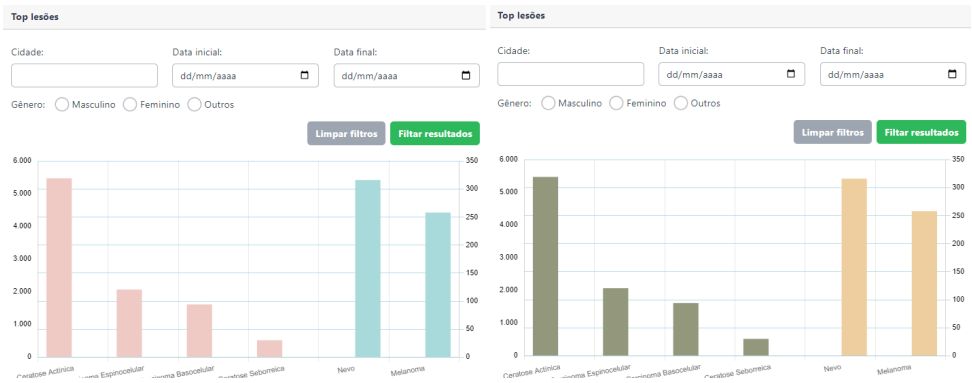

Figura 27 – Gráfico de distribuição das principais lesões em duas versões. A paleta 2 foi escolhida no gráfico da esquerda, enquanto no da direita a paleta 3 foi escolhida.

4.4.6 Gráfico de distribuição de lesões por região anatômica

O gráfico representado na Figura 28, mostra a distribuição de lesões de pele nas áreas do corpo. Isso facilita a identificação de padrões e áreas de maior incidência, permitindo uma análise rápida e eficaz. É possível perceber que as áreas mais recorrentes são a face e os braços, isso faz sentido visto que essas são regiões do corpo muito expostas

à luz solar. Vemos também que regiões que geralmente são cobertas por roupas, como os ombros e as costas, possuem um baixo número de lesões quando comparadas as áreas mais expostas. A Figura 29 mostra as demais opções de coloração deste gráfico.

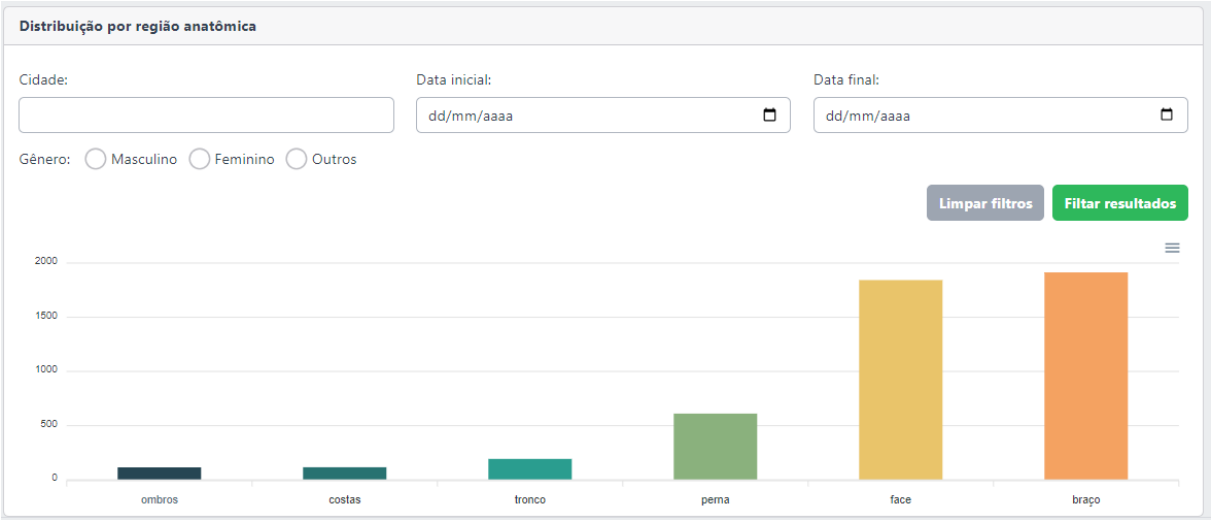

Figura 28 – Gráfico de distribuição de lesões por áreas do corpo No eixo das abscissas estão as áreas do corpo e no eixo das ordenadas estão a quantidade de lesões.

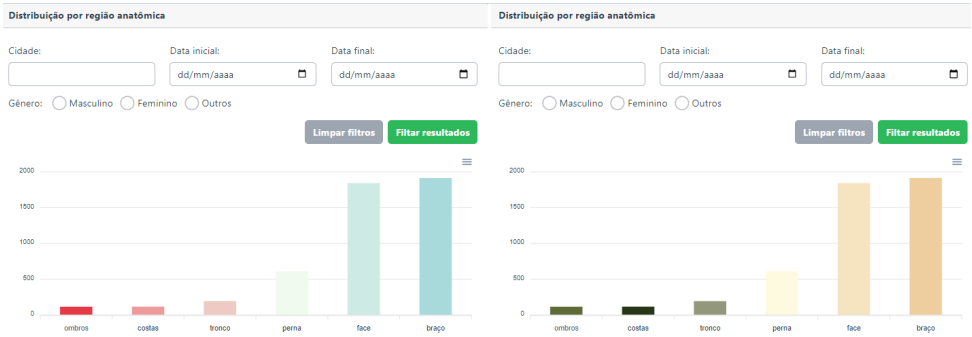

Figura 29 – Gráfico de quantidade de lesões por área do corpo. A paleta 2 foi escolhida no gráfico da esquerda, enquanto no da direita a paleta 3 foi escolhida.

4.4.7 Gráfico de distribuição etária por diagnóstico

A Figura 30 mostra a distribuição etária das principais lesões discutidas na seção 4.4.5. Este gráfico fornece uma visualização clara da faixa etária em que essas lesões são mais prevalentes, destacando tendências e identificando grupos etários afetados. Isso fornece aos profissionais da área da saúde uma base para estudos epidemiológicos, o que possibilita um direcionamento de estratégias de prevenção e planejamento de recursos de forma mais precisa e eficaz. A Figura 31 mostra como é a aparência deste gráfico quando as outras paletas de cores são escolhidas

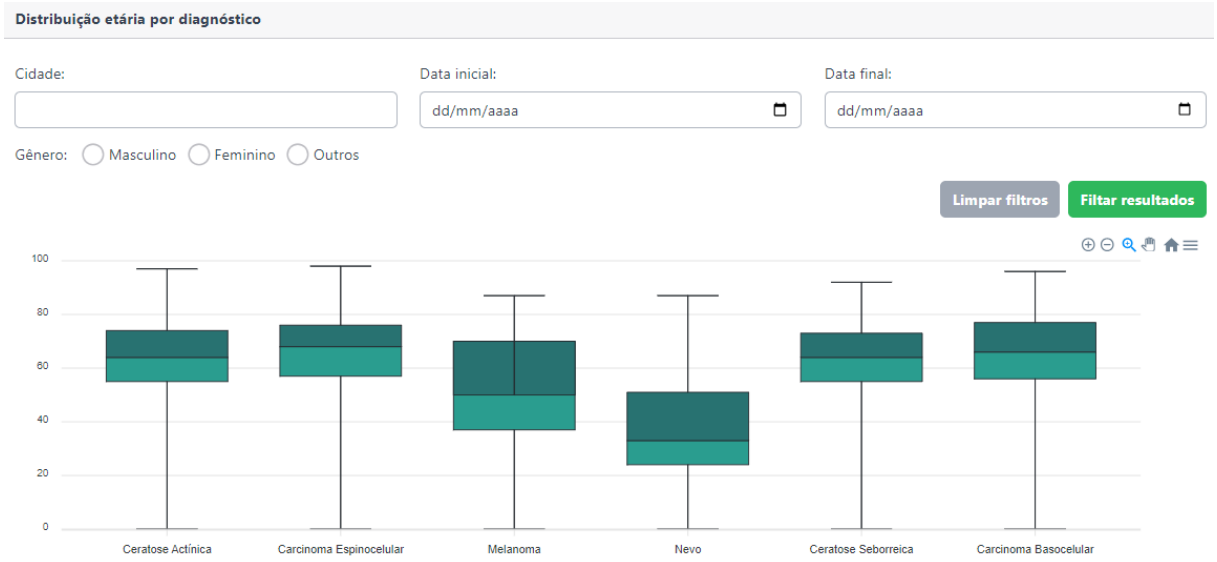

Figura 30 – Gráfico em barras de distribuição etária por diagnóstico. No eixo das abscissas estão as lesões de pele mais comuns, atendidas pelo PAD-UFES, enquanto no eixo das ordenadas está a quantidade de lesões.

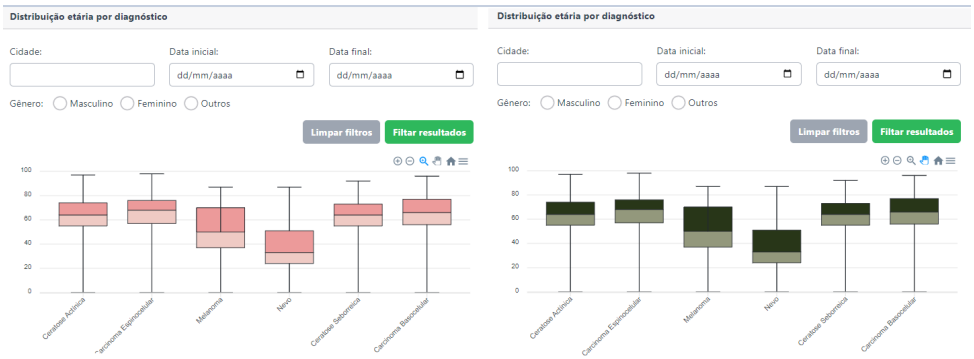

Figura 31 – Gráfico de distribuição etária por diagnóstico em duas versões. A paleta 2 foi escolhida no gráfico da esquerda, enquanto no da direita a paleta 3 foi escolhida.

4.5 Outras configurações comuns

Todos os gráficos criados neste módulo de visualização são responsivos. Ao tornar um gráfico responsivo, ele se adapta automaticamente ao espaço disponível, otimizando a apresentação dos dados. Isso é especialmente importante em um mundo onde os sistemas web podem ser acessados por meio de uma variedade de dispositivos com diferentes tamanhos de tela. Na Figura 32 é possível ver como fica a aparência do módulo quando visualizado pelo celular.

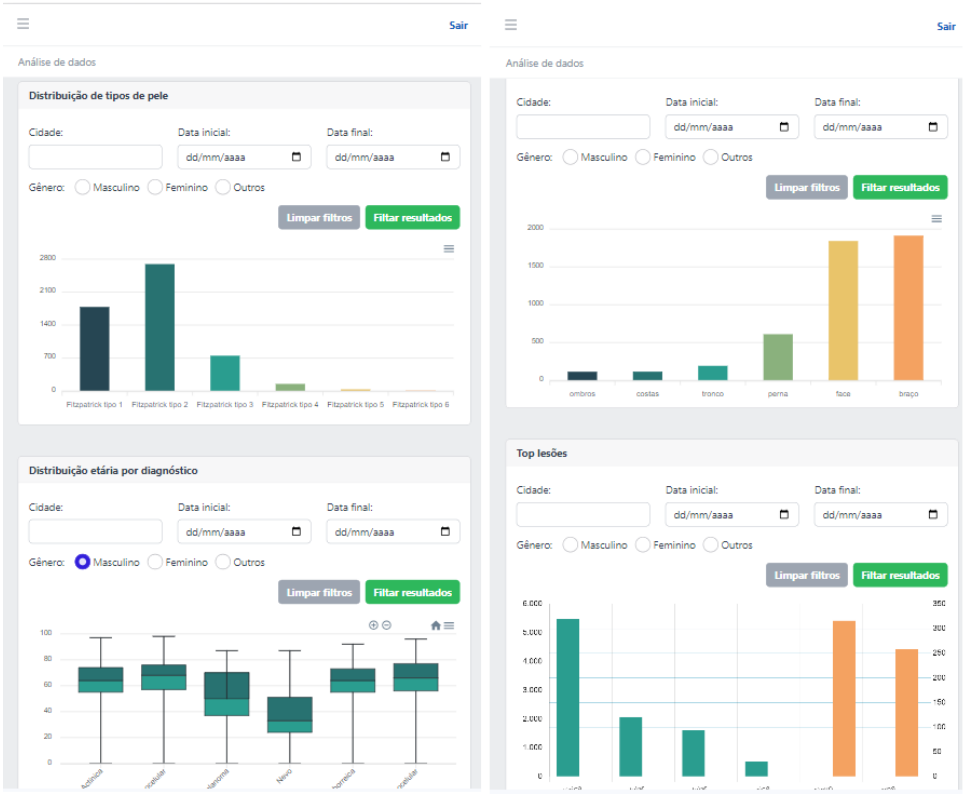

Figura 32 – Visualização *mobile* de alguns gráficos do módulo de visualização.

Além disso, todos os gráficos podem ser filtrados através das opções localizadas na parte superior do *card* do gráfico. A inclusão desses filtros é vital para oferecer aos usuários a capacidade de personalizar e explorar os dados de maneira mais profunda e relevante. Os filtros permitem uma experiência interativa, onde é possível selecionar, analisar e extrair informações específicas dos conjuntos de dados, adaptando a visualização às necessidades individuais. Na Figura 33 é possível ver um exemplo da aplicação de filtros no gráfico de distribuição de tipos de pele. Observe como a aplicação deste filtro permite concluir que existem apenas pessoas do tipo de pele 1, que possuem o gênero "Outro", cadastrado no banco de dados da aplicação.

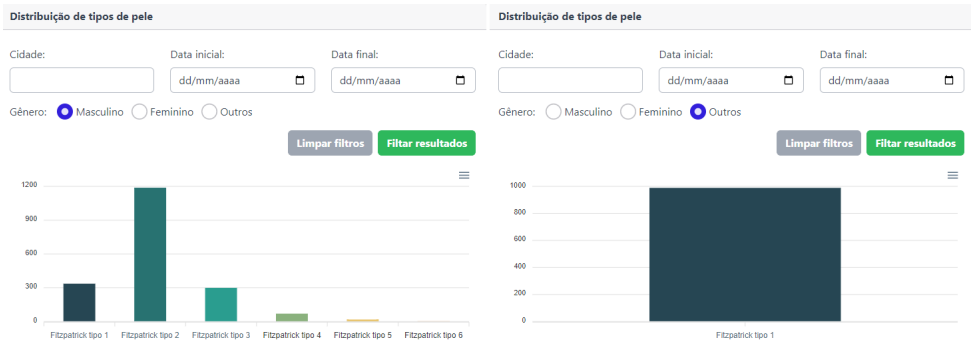

Figura 33 – Exemplo de aplicação de filtro no gráfico de distribuição de tipos de pele. A esquerda é possível ver o gráfico filtrado com resultados apenas dos pacientes com gênero masculino. Na direita é possível ver o resultado quando a opção "Outros" é escolhida no filtro de gênero.

5 Considerações Finais

Foi desenvolvido neste projeto de graduação um módulo de visualização de dados integrado no SADE, com o propósito de auxiliar os profissionais da área da saúde a compreender padrões das lesões de pele nos locais atendidos pelo PAD-UFES. O objetivo do módulo é fornecer representações gráficas das informações clínicas e estatísticas. A ferramenta possibilita a identificação de tendências e relações ocultas nos dados, o que pode fornecer percepções valiosas para tomadas de decisão e criação de novas estratégias.

Como o módulo precisava ser integrado no sistema SADE, ele foi desenvolvido com as mesmas tecnologias usadas pelo programa. Com isso, no *front-end* foi utilizado o Angular, enquanto no *back-end* foi utilizado o Spring Boot. O uso dessas tecnologias permitiu aquisição de conhecimento, visto que foi necessário aprender diversas tecnologias e conceitos que não são abordados durante a graduação, mas que são largamente utilizadas no mercado de trabalho. A aplicação foi construída seguindo as arquiteturas utilizadas, o que facilita a compreensão e manutenção do código pelos outros desenvolvedores do SADE.

O desenvolvimento do módulo foi, de certo modo, baseado em estudos sobre a percepção humana, descritos no capítulo 2. Foi observado que os princípios de percepção constituem uma base sólida para o desenvolvimento de diretrizes de design eficazes. Compreender o processo de processamento das informações visuais pelo cérebro humano é crucial para a concepção de visualizações que alcancem maior efetividade.

Foram criadas sete visualizações, que podem fornecer várias informações relevantes aos profissionais da área da saúde. É possível identificar o gênero mais comum atendido pelo PAD-UFES, a distribuição da idade dos pacientes e quais os tipos de pele mais frequentes. Adicionalmente, fornecem dados sobre as áreas do corpo mais frequentemente afetadas por lesões, a distribuição das lesões mais comuns e o padrão etário dos pacientes com essas lesões.

Existem algumas melhorias que podem ser feitas no módulo. As principais delas são: alteração do formato de saída da API para o formato JSON, que é amplamente utilizado pela indústria. Além disso, seria possível adicionar mais análises estatísticas nas visualizações, isso melhoraria a capacidade de obter informações dos dados, fornecendo informações úteis aos usuários do sistema.

Ao final, acredita-se que o desenvolvimento deste projeto de graduação tenha sido bem-sucedido. Além de proporcionar ao aluno graduando uma grande fonte de conhecimento técnico, em um campo de amplo interesse e atualmente emergente, também foi possível desenvolver uma aplicação que buscou atender uma necessidade real de um

sistema de um programa de assistência que afeta diretamente a vida de muitas pessoas no estado do Espírito Santo.

Referências

AELA. *Os 7 Princípios de Gestalt e Como Utilizá-los em Projetos de UI Design.* 2020. Disponível em: https://medium.com/aela/os-7-princípios-de-gestalt-e-como-utilizá-los-em-projetos-de-ui-design-46d6d832abf6. Citado 5 vezes nas páginas 7, 20, 21, 23 e 24.

BRUSH, K.; BURNS, E. *Data Visualization.* 2022. Disponível em: https://www.techtarget.com/searchbusinessanalytics/definition/data-visualization. Citado na página 14.

BUSSAB, W. d. O.; MORETTIN, P. A. Estatística básica. Editora Saraiva, v. 6°, 2010. Citado na página 19.

CHANG, D.; NESBITT, K. V. Developing gestalt-based design guidelines for multi-sensory displays. In: *Proceedings of the 2005 NICTA-HCSNet Multimodal User Interaction Workshop - Volume 57.* AUS: Australian Computer Society, Inc., 2006. (MMUI '05), p. 9–16. ISBN 1920682392. Citado 4 vezes nas páginas 20, 21, 22 e 23.

Colour Blind Awareness. *About Colour Blindness.* 2023. Disponível em: <https://www.colourblindawareness.org/colour-blindness/>. Citado na página 25.

CRAVIT, R. *How to Use Color Blind Friendly Palettes to Make Your Charts Accessible.* 2022. Disponível em: <https://venngage.com/blog/color-blind-friendly-palette/>. Citado na página 25.

FERREIRA, M. F. The information visualization thematic, head to head with color blindness. 2017. Disponível em: <https://www.researchgate.net/profile/Mafalda-Ferreira-10/publication/313657827_The_Information_Visualization_Thematic_Head_to_Head_with_Color_Blindness/links/58a1c19da6fdccf5e970e999/The-Information-Visualization-Thematic-Head-to-Head-with-Color-Blindness.pdf>. Citado 3 vezes nas páginas 8, 25 e 26.

HASANOV, H. *Visualization principles.* 2015. Disponível em: https://hasanoviz.medium.com/visualization-principles-a8c6e46ddc5f. Citado 4 vezes nas páginas 7, 21, 22 e 25.

INCA. *INCA estima 704 mil casos de câncer por ano no Brasil até 2025.* 2022. Disponível em: <https://www.gov.br/inca/pt-br/assuntos/noticias/2022/inca-estima-704-mil-casos-de-cancer-por-ano-no-brasil-ate-2025>. Citado na página 13.

JUNIOR, C. A. M.; FARIA, N. C. Memória. *Psicologia: Reflexão e Crítica*, Curso de Pós-Graduação em Psicologia da Universidade Federal do Rio Grande do Sul, v. 28, n. 4, p. 780–788, Oct 2015. ISSN 0102-7972. Disponível em: <https://doi.org/10.1590/1678-7153.201528416>. Citado na página 18.

KOFFKA, K. Principles of gestalt psychology. v. 2, 1935. Citado na página 19.

MANLIHEROVA, M. The influence of visual information. *Faculty of Journalism and Mass Communication, Sofia University*, 2020. Disponível em: <https://www.ceeol.com/search/chapter-detail?id=882141>. Citado na página 17.

Ministério da Saúde. *Câncer de pele*. 2023. Disponível em: <https://www.gov.br/saude/pt-br/assuntos/saude-de-a-a-z/c/cancer-de-pele>. Citado na página 13.

MOTA, J. P.; BARJA, P. Classificação de fototipos de pele. *São José dos Campo: Universidade do Vale do Paraíba Instituto de Pesquisa e Desenvolvimento*, 2006. Citado na página 40.

PACHECO, A. et al. Sade: Software de análise dermatológica - um sistema de coleta, gerenciamento e triagem de lesões de pele. In: *Anais Estendidos do XXIX Simpósio Brasileiro de Sistemas Multimídia e Web*. Porto Alegre, RS, Brasil: SBC, 2023. p. 111–114. ISSN 2596-1683. Disponível em: <https://sol.sbc.org.br/index.php/webmedia_estendido/article/view/25686>. Citado 2 vezes nas páginas 13 e 14.

RICHARD, K. *Data Visualization: Processing Images at 1250MB/s*. 2015. Disponível em: https://wiredcraft.com/blog/data-visualizations-images/. Citado 2 vezes nas páginas 7 e 17.

RODRIGUES, A. A.; DIAS, G. A. Estudos sobre visualização de dados científicos no contexto da data science e do big data. *BRAPCI*, v. 12, 2017. Citado na página 13.

STERNBERG, R. J. *Psicologia Cognitiva*. [S.l.]: Cengage Learning, 2000. Citado na página 18.

TREISMAN, A. Preattentive processing in vision. *Computer Vision, Graphics, and Image Processing*, v. 31, n. 2, p. 156–177, 1985. ISSN 0734-189X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0734189X85800049>. Citado na página 18.

TUFTE, E. R. The visual display of quantitative information. *Graphics Press*, v. 1, 2001. Citado na página 24.

WARD, M.; KEIM, D.; GRINSTEIN, G. Interactive data visualization: foundations, techniques, and applications. *CRC Press*, AK Peters, 2015. Citado na página 14.

WARE, C. Information visualization: Perception for design: Second edition. In: _____. [S.l.: s.n.], 2004. Citado 3 vezes nas páginas 7, 18 e 19.

WERTHEIMER, M. Untersuchungen zur lehre von der gestalt. ii. *Psychologische Forschung*, v. 4, n. 1, p. 301–350, 1923. Citado na página 19.

Apêndices

.1 Exposição da lógica no *front-end*

Na Listagem 1 é mostrado o conteúdo do arquivo **data-analysis.component.ts**. A lógica desse arquivo principal do componente é aparentemente trivial devido à separação dos elementos visuais do módulo de visualização em diversos sub-componentes. Dessa forma, este arquivo é, basicamente, responsável por receber quais gráficos foram selecionados para visualização e qual paleta de cor foi escolhida. Os gráficos selecionados são armazenados na variável **graphs** e a paleta de cor escolhida é guardada na variável **colorPalette**. Essas variáveis são utilizadas no arquivo **data-analysis.component.html**, que pode ser visto na Listagem 2.

Listagem 1 – Código do arquivo que contém a lódica principal do componente data-analysis.

```
1  import { Component } from "@angular/core";
2  import { firstPalette } from "./utils/color-palettes";
3
4  @Component({
5    selector: "app-data-analysis",
6    templateUrl: "./data-analysis.component.html"
7  })
8  export class DataAnalysisComponent {
9    colorPalette: string[] = firstPalette;
10   graphs: any;
11
12   onColorPalleteSelected(colorPalete: string[]) {
13     this.colorPalette = colorPalete
14   }
15
16   onGraphsSelected(graphs: any) {
17     this.graphs = graphs;
18   }
19 }
```

As paletas de cores pré-definidas, são armazenadas no arquivo **color-palettes.ts**, dentro diretório "utils". Isso foi feito para centralizar a definição dessas paletas em um único local, facilitando não somente o acesso a elas pelos componentes do módulo de visualização, mas também a criação de novas paletas de cores no sistema. A Listagem 3 mostra como essas paletas de cores são armazenadas no componente principal. Todas essas paletas foram criadas com o auxílio da ferramenta online *Coolors* [1], que consiste basicamente em um website que possibilita a criação de paletas de cores, incluindo uma seção exclusiva para paletas acessíveis às pessoas daltônicas. Assim, todas as três paletas de cores disponíveis no sistema foram criadas considerando esses aspectos, cuja importância foi discutida na seção 2.3.2.

A Figura 34 mostra a estrutura do componente "config". Ele é responsável por gerenciar as configurações possíveis das visualizações, sendo elas a paleta de cores e a escolha de quais gráficos serão expostos ao usuário. A Listagem 4 mostra a lógica do componente de configurações. Nas linhas 7, 8 e 9 são definidas variáveis necessárias

[1] <https://coolors.co/palettes/blindness>

Listagem 2 – Código do arquivo que contém o HTML principal do componente data-analysis.

```
<app-config (colorPalleteEmitter)="onColorPalleteSelected($event)" (graphsEmitter
    )="onGraphsSelected($event)">
</app-config>

<app-fitzpatrick-distribution-graph
  *ngIf="graphs?.fitzpatrickGraphSelected" [colorPalette]="colorPalette">
</app-fitzpatrick-distribution-graph>

<app-age-bar-distribution-graph
  *ngIf="graphs?.ageBarGraphSelected" [colorPalette]="colorPalette">
</app-age-bar-distribution-graph>

<app-age-boxplot-distribution-graph
  *ngIf="graphs?.ageBoxplotGraphSelected" [colorPalette]="colorPalette">
</app-age-boxplot-distribution-graph>

<app-age-per-diagnostic-graph
  *ngIf="graphs?.agePerDiagnosticGraphSelected" [colorPalette]="colorPalette">
</app-age-per-diagnostic-graph>

<app-anatomic-region-distribution-graph
  *ngIf="graphs?.anatomicRegionGraphSelected" [colorPalette]="colorPalette">
</app-anatomic-region-distribution-graph>

<app-gender-distribution-graph
  *ngIf="graphs?.genderGraphSelected" [colorPalette]="colorPalette">
</app-gender-distribution-graph>

<app-top-lesions-graph
  *ngIf="graphs?.topLesionsGraphSelected" [colorPalette]="colorPalette">
</app-top-lesions-graph>
```

para o gerenciamento da paleta de cores escolhida. O método **changeColorPalette**, é responsável por mudar a paleta de cores a ser exportada com base no número fornecido como argumento. Enquanto isso, a propriedade **dataAnalysisFilterForm** representa um formulário de filtro para os gráficos selecionados. Por fim, podemos destacar também a função **saveConfig**, que consiste em um método que emite eventos usando o decorador **@Output** para informar outras partes do módulo sobre a paleta de cores atual e as configurações dos gráficos selecionados.

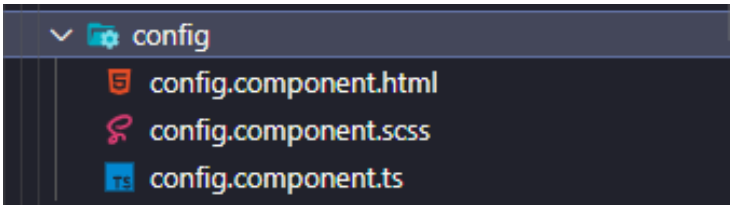

Figura 34 – Estrutura do componente `config`, que contém todo o código correspondente as configurações possíveis dos gráficos.

Conforme mencionado na seção 4.1, cada gráfico possui um componente específico. Essa configuração concede aos gráficos maior poder de customização individual, sendo possível a adição de filtros específicos para cada um e a alteração da forma com que a

Listagem 3 – Código que define as paletas de cores disponíveis no sistema

```
export const firstPalette = [
  "#264653ff",
  "#287271ff",
  "#2a9d8fff",
  "#8ab17dff",
  "#e9c46aff",
  "#f4a261ff",
  "#e76f51ff",
  "#e97c61ff",
];

export const secondPalette = [
  "#e63946ff",
  "#ec9a9aff",
  "#efcac4ff",
  "#f1faeeff",
  "#cdeae5ff",
  "#a8dadcff",
  "#457b9dff",
  "#1d3557ff",
];

export const thirdPalette = [
  "#606c38ff",
  "#283618ff",
  "#93987cff",
  "#fefae0ff",
  "#f6e4c0ff",
  "#eece9fff",
  "#dda15eff",
  "#bc6c25ff",
];
```

visualização é construída. Os arquivos de lógica de todo componente de gráfico criado neste módulo possui uma função que recebe os dados do *back-end*. Um exemplo de função assim está exposto na Listagem 5.

O parâmetro opcional *filters*, é passado para a função para determinar quais conjuntos de dados irão retornar da API do SADE. É possível observar que o valor padrão do parâmetro define um amplo intervalo de tempo e um gênero não especificado para a busca dos dados, isso faz com que inicialmente, os gráficos sejam carregados com todos os dados contidos no banco. Na linha 5, vemos que o *service* de *data-analys* chama um a função **getAgeDistribution** que retorna um **Observable**, que são amplamente utilizados - no contexto do Angular - para lidar com dados assíncronos, como resultados de requisições HTTP. O método **subscribe** permite executar ações assim que as requisições responderem. Em caso de sucesso, os dados vindos do *back-end* são tratados e é chamada a função auxiliar **ageBoxplotGraphInit** para construir as principais informações referentes a construção do gráfico. Esta função vem do arquivo auxiliar **age-distribution.ts**, que se encontra dentro da pasta "utils".

A Listagem 6, mostra como as configurações obtidas na linha 15 da Listagem 5 são utilizadas no componente disponibilizado pela biblioteca ApexCharts. É possível

Listagem 4 – Código que define a lógica do componente de configuração

```
export class ConfigComponent {
  faIcons = {
    faGears,
    faSave,
  };

  numberPalette: number = 1;
  colorPalette: string[] = firstPalette;
  palettes = [firstPalette, secondPalette, thirdPalette];


  dataAnalysisFilterForm: FormGroup = this.fb.group({
    genderGraphSelected: false,
    ageBarGraphSelected: false,
    ageBoxplotGraphSelected: false,
    fitzpatrickGraphSelected: false,
    topLesionsGraphSelected: false,
    anatomicRegionGraphSelected: false,
    agePerDiagnosticGraphSelected: false,
  });

  @Output()  colorPalleteEmitter = new EventEmitter<string[]>();
  @Output()  graphsEmitter = new EventEmitter<any>();

  constructor(private fb: FormBuilder) {}

  changeColorPalette(numberPalette: number) {
    switch (numberPalette) {
      case 2:
        this.colorPalette = secondPalette;
        break;
      case 3:
        this.colorPalette = thirdPalette;
        break;
      default:
        this.colorPalette = firstPalette;
    }
  }

  saveConfig() {
    this.colorPalleteEmitter.emit(this.colorPalette)
    this.graphsEmitter.emit({
      genderGraphSelected: this.dataAnalysisFilterForm.value.genderGraphSelected,
      ageBarGraphSelected: this.dataAnalysisFilterForm.value.ageBarGraphSelected,
      ageBoxplotGraphSelected: this.dataAnalysisFilterForm.value.
          ageBoxplotGraphSelected,
      fitzpatrickGraphSelected: this.dataAnalysisFilterForm.value.
          fitzpatrickGraphSelected,
      topLesionsGraphSelected: this.dataAnalysisFilterForm.value.
          topLesionsGraphSelected,
      anatomicRegionGraphSelected: this.dataAnalysisFilterForm.value.
          anatomicRegionGraphSelected,
      agePerDiagnosticGraphSelected: this.dataAnalysisFilterForm.value.
          agePerDiagnosticGraphSelected,
    })
  }
}
```

observar, na linha 2, que o gráfico é mostrado apenas quando os dados do *back-end* já foram carregados. A variável **ageBoxplotData** contém todas as legendas e os conteúdos do gráfico exposto. Já a variável **ageBoxplotOptions**, contém informações relativas às configurações mais gerais do gráfico, como escala, responsividade e localização das legendas e dos eixos. O parâmetro de filtragem, **filters**, passado na função de busca, é construído com base nos dados fornecidos pelo usuário nos campos de filtro que cada gráfico possui. Todos os componentes de gráficos criados possuem basicamente a mesma lógica central: recuperação e tratamento de dados do *back-end*, construção dos dados de configuração dos gráficos e aplicação dos dados nas visualizações.

Listagem 5 – Código que recupera dados do *back-end* do compoennte que cria o gráfico boxplot de distribuição etária

```
1
2 getAgeDistribution(
3   filters: string = `?city=&initialDate=2000-01-01&finalDate=${this.today}&gender
      =`
4 ) {
5   this.dataAnalysisService.getAgeDistribution(filters).subscribe({
6     next: (result) => {
7       this.ageDistribution = result.map((distribution: any) => {
8         const obj: IAgeDistribution = {
9           age: distribution[0],
10          total: distribution[1],
11        };
12        return obj;
13      });
14
15      const [ageBoxplotData, ageBoxplotOptions] = ageBoxplotGraphInit(
16        this.ageDistribution,
17        this.colorPalette
18      );
19      this.ageBoxplotData = ageBoxplotData;
20      this.ageBoxplotOptions = ageBoxplotOptions;
21    },
22    error: (error) => {
23      console.log(error);
24    },
25    complete: () => {
26      this.loading = false;
27    },
28  });
29 }
```

Listagem 6 – Trecho do código HTML do componente do gráfico boxplot de distribuição etária

```
1 <apx-chart
2   *ngIf="!loading"
3   [series]="ageBoxplotData"
4   [chart]="ageBoxplotOptions.chart"
5   [plotOptions]="ageBoxplotOptions.plotOptions"
6   [title]="ageBoxplotOptions.title"
7   [tooltip]="ageBoxplotOptions.tooltip"
8 ></apx-chart>
```

.2 Exposição da lógica no *back-end*

A Listagem 7 mostra um trecho do código da classe controladora criada, que contém o código de duas rotas da API. É possível reparar que a classe possui a notação **@Crontroller**, isso ajuda o Spring Boot a identificá-la com um *controller*. A classe possui dois parâmetros, passados por injeção de dependências, **patientRepository** e **lessionRepository**, que são instâncias dos repositórios relativos aos *models* de *Lession* e *Patient*, e são necessárias para chamar os métodos de pesquisa no banco de dados.

Listagem 7 – Trecho do código da classe controladora.language

```
1  @Controller
2  public class DataAnalysisController {
3
4      @Autowired
5      private PatientRepository patientRepository;
6
7      @Autowired
8      private LesionRepository lesionRepository;
9
10     @GetMapping(value = "/api/data-analysis/gender-distribution")
11     @ResponseBody
12     public List<Object> getGenderDistribution(
13             @RequestParam(value = "city") String city,
14             @RequestParam(value = "initialDate") String initialDate,
15             @RequestParam(value = "finalDate") String finalDate) {
16         city = "%" + city + "%";
17         List<Object> result = patientRepository.getGenderDistribution(city,
               initialDate, finalDate);
18
19         return result;
20     }
21
22     @GetMapping(value = "/api/data-analysis/anatomic-region-distribution")
23     @ResponseBody
24     public Object getAnatomicRegionDistribution(
25             @RequestParam(value = "city") String city,
26             @RequestParam(value = "initialDate") String initialDate,
27             @RequestParam(value = "finalDate") String finalDate,
28             @RequestParam(value = "gender") String gender) {
29         city = "%" + city + "%";
30         gender = "%" + gender + "%";
31         List<Object> result = lesionRepository.getAnatomicRegionDistribution(city
               , initialDate, finalDate, gender);
32         return result;
33     }
34
35 }
```

A estrutura dos métodos que definem os *endpoints* da API é bastante similar entre eles. Inicialmente, é definido com a notação **@GetMapping** o endereço do *endpoint*, como é possível ver nas linhas 10 e 22. Além disso, são passadas como parâmetros as informações dos filtros realizados na requisição. Por fim, é utilizado a instância do repositório necessário para chamar o método de busca adequado e o resultado dessa pesquisa é retornado ao usuário da API. Na linha 17 é chamado o repositório do paciente, pois o endpoint está relacionado a distribuição dos gêneros. Já na linha 22, o repositório chamado é o de lesão, pois o endpoint é relativo à distribuição anatômica da região da lesão.

Um trecho do código do *repository* de paciente é mostrado na Listagem 8. O código consiste basicamente na definição da interface **PatientRepository** que estende de **PagingAndSortingRepository**, que é uma interface no Spring Framework que oferece métodos para realizar operações básicas de persistência de dados, especialmente em bancos de dos relacionais. **PagingAndSortingRepository** estende a interface **CrudRepository** e fornece funcionalidades adicionais para paginação e ordenação dos resultados.

Listagem 8 – Trecho do código do *repository* de paciente.language

```
@RepositoryRestResource(collectionResourceRel = "patient", path = "patient")
public interface PatientRepository extends PagingAndSortingRepository<Patient,
    Long> {
    @Query(value = """
            SELECT
                gender,
                count(p.id) as total
            FROM
                patient p
            LEFT JOIN
                appointment a
            ON
                a.patient_id  = p.id
            WHERE
                p.city  LIKE :city
                AND a.date > :initialDate
                AND a.date < :finalDate
            GROUP BY
                p.gender
                """, nativeQuery = true)
    List<Object> getGenderDistribution(
            @Param("city") String city,
            @Param("initialDate") String initialDate,
            @Param("finalDate") String finalDate);

    @Query(value = """
            SELECT
                DATE_FORMAT(FROM_DAYS(DATEDIFF(NOW(),p.birth_date)), '%Y') + 0 AS
                    age,
                count(p.id) as total
            FROM
                patient p
            LEFT JOIN
                appointment a
            ON
                a.patient_id  = p.id
            WHERE
                p.city  like :city
                AND a.date > :initialDate
                AND a.date < :finalDate
                AND p.gender like :gender
            GROUP BY
                age
            ORDER BY
                age
             """, nativeQuery = true)
    List<Object> getAgeDistribution(
            @Param("city") String city,
            @Param("initialDate") String initialDate,
            @Param("finalDate") String finalDate,
            @Param("gender") String gender);

}
```

O código mostra a definição de dois métodos que realizam consultas personalizadas ao banco de dados para recuperar distribuições de gênero e idade dos pacientes em uma determinada cidade e período, com base nos dados das consultas e consultas médicas registradas no banco de dados. Segue abaixo uma explicação mais detalhada:

1. **getGenderDistribution**
    - Esta consulta utiliza SQL nativo para recuperar a distribuição de gênero dos pacientes.
    - Parâmetros: a cidade (**city**) dos pacientes e o intervalo de datas (**initialDate** e **finalDate**) para restringir os dados da consulta.
    - A consulta junta as tabelas **patient** e **appointment** usando um **LEFT JOIN**.
    - Filtra os pacientes com base na cidade e nas datas das consultas **a.date** usando cláusulas **WHERE** e agrupa os resultados por gênero **p.gender** e conta o número de ocorrências para cada gênero.
    - Agrupa os resultados por gênero **p.gender** e conta o número de ocorrências para cada gênero.
2. **getAgeDistribution**
    - Esta consulta também utiliza SQL nativo para recuperar a distribuição de idade dos pacientes.
    - Parâmetros: a cidade (**city**) dos pacientes, o intervalo de datas(**initialDate** e **finalDate**) e o gênero (**gender**) dos pacientes para serem considerados na consulta.
    - Assim como na consulta anterior, junta as tabelas **patient** e **appointment** usando um **LEFT JOIN**
    - Além dos filtros por cidade e datas de consulta, também inclui um filtro pelo gênero do paciente.
    - Calcula a idade dos pacientes com base na data de nascimento **p.birth_date** usando funções SQL, agrupa os resultados por idade e conta o número de ocorrências para cada faixa etária

O *repository* de lesão, por sua vez, lida com as pesquisas relacionadas, principalmente, com a entidade **Lesion** no banco de dados. Nas Listagens 9 e 10 é possível ver trechos de código deste *repository*. Na primeira, podemos observar a definição da interface **LesionRepository**, que ocorre de maneira similar ao feito no *repository* de paciente, e a definição do método **getTopLesions**. Na segunda, por sua vez, é visto a estrutura do método **getAnatomicRegionDistribution**.

Na consulta SQL realizada pelo método **getTopLesion**, podemos ver que são recuperadas informações relativas ao diagnóstico clínico das lesões. O *joins* feitos com as tabelas **appointment** e **patient** são responsáveis por possibilitar a filtragem dos resultados por: cidade do atendimento, data do atendimento e gênero do paciente.

Listagem 9 – Trecho do código do *repository* de lesão.language

```
@RepositoryRestResource(collectionResourceRel = "lesion", path = "lesion")
public interface LesionRepository extends PagingAndSortingRepository<Lesion, Long
    > {
    @Query(value = """
            SELECT
                l.clinical_diagnostic,
                count(l.id) as total
            FROM
                lesion l
            LEFT JOIN appointment a
            ON a.id = l.appointment_id
            LEFT JOIN patient p
            ON p.id = a.patient_id
            WHERE
                (
                    clinical_diagnostic LIKE '%C43%'
                    OR clinical_diagnostic LIKE '%D03%'
                    OR clinical_diagnostic LIKE '%C44%'
                    OR clinical_diagnostic LIKE '%D04%'
                    OR clinical_diagnostic LIKE '%C80%'
                    OR clinical_diagnostic LIKE '%L57%'
                    OR clinical_diagnostic LIKE '%D22%'
                    OR clinical_diagnostic LIKE '%L82%'
                )
                AND a.city like :city
                AND a.date > :initialDate
                AND a.date < :finalDate
                AND p.gender like :gender
            GROUP BY
                clinical_diagnostic
                    """, nativeQuery = true)
    List<Object> getTopLesions(
            @Param("city") String city,
            @Param("initialDate") String initialDate,
            @Param("finalDate") String finalDate,
            @Param("gender") String gender);
}
```

Na Listagem 10 podemos ver a consulta SQL utilizada na construção do método **getAnatomicRegionDestribution**. Essa consulta é um exemplo de tratamento e categorização de dados da coluna **body_region** da tabela **lesion**. Ela é usada para contar o número de lesões agrupadas em diferentes regiões do corpo, categorizando essas regiões de uma maneira mais padronizada e específica. No caso, a estrutura **CASE WHEN** é utilizada para definir categorias específicas para diferentes partes do corpo mencionadas na coluna **body_region**. Essas categorias são associadas a regiões corporais mais genéricas, como “face”, “braço”, “tronco”, “ombros”, “costas” e “perna”.

Após categorizar cada lesão em uma região específica usando o **CASE WHEN**, a consulta conta o número total de lesões em cada categoria usando **COUNT(l.id)**. Isso é feito usando um **LEFT JOIN** com as tabelas **appointment** e **patient**, onde são

aplicados filtros para datas específicas, cidade e gênero do paciente. Por fim, o resultado é agrupado pela categoria de região, fornecendo o total de lesões encontradas em cada parte do corpo especificada na consulta. Essa abordagem permite uma análise mais organizada e agregada das lesões registradas, facilitando a compreensão e o tratamento adequado em diferentes áreas do corpo.

Listagem 10 – Código do da pesquisa .language

```
@Query(value = """
            select
            CASE
                when l.body_region like '%face%' then 'face'
                when l.body_region like '%nuca%' then 'face'
                when l.body_region like '%nariz%' then 'face'
                when l.body_region like '%nasal%' then 'face'
                when l.body_region like '%couro cabeludo%' then 'face'
                when l.body_region like '%orbital%' then 'face'
                when l.body_region like '%auricular%' then 'face'
                when l.body_region like '%zigomatica%' then 'face'
                when l.body_region like '%temporal%' then 'face'
                when l.body_region like '%mandibula%' then 'face'
                when l.body_region like '%labio%' then 'face'
                when l.body_region like '%labial%' then 'face'
                when l.body_region like '%rosto%' then 'face'
                when l.body_region like '%malar%' then 'face'
                when l.body_region like '%hélice%' then 'face'

                when l.body_region like '%braço%' then 'braço'
                when l.body_region like '%mão%' then 'braço'

                when l.body_region like '%FURCULA ESTERNAL%' then 'tronco'
                when l.body_region like '%mama%' then 'tronco'
                when l.body_region like '%colo%' then 'tronco'

                when l.body_region like '%ombro%' then 'ombros'
                when l.body_region like '%trapézio%' then 'ombros'
                when l.body_region like '%DELTÓIDE%' then 'ombros'

                when l.body_region like '%cervical%' then 'costas'

                when l.body_region like '%pé%' then 'perna'
                when l.body_region like '%tornoze%' then 'perna'

            END as region,
            count(l.id) as totalLesions
            from
            lesion l
            join appointment a
            on a.id  = l.appointment_id
            join patient p
            on p.id = a.patient_id
            where
                a.city like :city
                AND a.date > :initialDate
                AND a.date < :finalDate
                AND p.gender like :gender
            group by region
            order by totalLesions
                    """, nativeQuery = true)
    List<Object> getAnatomicRegionDistribution(
            @Param("city") String city,
            @Param("initialDate") String initialDate,
            @Param("finalDate") String finalDate,
            @Param("gender") String gender);
```

Os demais *endpoints* disponibilizados no controlador do módulo criado seguem o mesmo padrão de lógica que os *endpoints* mostrados nesta seção. Essa abordagem padronizada ajuda a manter a consistência no código, facilitando a compreensão e a manutenção do sistema, uma vez que segue um padrão de organização e estruturação das consultas e operações de dados.