

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL  
INSTITUTO DE INFORMÁTICA  
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ANDY RUIZ GARRAMONES

**IntroSpectView: uma ferramenta de  
análise visual para sistemas de  
recomendação do Globoplay**

Monografia apresentada como requisito  
parcial para a obtenção do grau de Bacharel  
em Ciência da Computação

Orientadora: Profa. Dra. Renata Galante  
Co-orientador: Prof. Dr. Weverton Cordeiro

Porto Alegre  
2021

**UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL**

Reitor: Prof. Carlos André Bulhões

Vice-Reitora: Prof<sup>a</sup>. Patricia Pranke

Pró-Reitora de Graduação: Prof<sup>a</sup>. Cíntia Inês Boll

Diretora do Instituto de Informática: Prof<sup>a</sup>. Carla Maria Dal Sasso Freitas

Coordenador do Curso de Ciência de Computação: Prof. Rodrigo Machado

Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

## **AGRADECIMENTOS**

Gostaria de salientar que, para mim, esta conquista não é apenas individual, mas de mérito coletivo também. Eu jamais seria capaz de realizar tal façanha sem todo o apoio, amor e ajuda do meu pai, da minha mãe e da minha vó, que tiveram um papel fundamental e especial durante toda minha vida, não só (mas também) na etapa estudantil. Também queria agradecer à minha namorada Marina por todo o apoio durante estes tempos difíceis, tanto de pandemia quanto de TCC, por todo o amor e carinho que me ajudaram a aguentar até o fim.

A universidade, principalmente a UFRGS, por muitas vezes pode se tornar um ambiente hostil e complicado. É por este motivo que agradeço de coração a todos os amigos que fiz durante este percurso e que foram responsáveis, em maior ou menor grau, por eu conseguir chegar onde cheguei e estar concluindo agora o curso. Em especial, queria agradecer ao Guilherme Haetinger, ao Rafael Audibert, ao Leonardo Vianna, ao João Vitor e à minha madrinha de curso Valéria. No entanto, há inúmeras pessoas que me ajudaram ao longo do curso, das mais diversas formas, me apoiando e estando ao meu lado, nos bons e maus momentos, e que também contribuíram para eu conseguir chegar até aqui. É por este motivo que não posso deixar de agradecer à Lize, o Hagen, o Stepien, o homem Alegre, à Aline, à Laura e a tantas outras pessoa por toda a vossa ajuda, sou deveras grato por tudo.

Por último, e não menos importante, queria agradecer a Globo Comunicação e Participações, ao Globoplay e ao Felipe Ferreira pela confiança para realizar esta parceria e conceder o conjunto de dados essencial para a realização deste trabalho.

## RESUMO

O acesso à Internet cresceu substancialmente nas últimas décadas, alcançando a marca de bilhões de usuários conectados com os mais diversos dispositivos. Uma grande parcela do volume de dados consumidos por esses usuários tem origem em plataformas de entretenimento e *streaming*, como *Netflix* e *YouTube*, que foram responsáveis por 26% do tráfego mundial em 2020. Tal proeminência exige uma correta administração dos recursos desses espaços, tanto no que diz respeito à infraestrutura envolvida quanto com relação aos próprios conteúdos criados e hospedados pelas plataformas. Para auxiliar nisso, sistemas de recomendação são empregados para identificar padrões de comportamento dos usuários, como suas preferências e interesses com relação aos conteúdos, de modo a aplicar tais informações no aprimoramento das plataformas. De maneira complementar, informações semelhantes também podem ser identificadas por meio de ferramentas de visualização de dados, cujo principal objetivo é impulsionar a capacidade humana de tomar decisões com base na identificação de padrões visuais. Neste trabalho, apresentamos a ferramenta IntroSpectView (ISV), uma ferramenta *web* que oferece ao usuário a possibilidade de realizar diversas análises visuais sobre os comportamentos dos usuários da plataforma Globoplay, um serviço de *streaming* de vídeos sob demanda desenvolvida pelo Grupo Globo. A ferramenta oferece diversos tipos de visualização diferentes, bem como possibilidades de customização dos gráficos. Demonstramos seu funcionamento utilizando um conjunto de casos de aplicabilidade real e discutindo tecnicamente as consultas SQL implementadas. O resultados das visualizações são analisados criticamente de modo a extrair informações úteis que podem ser usadas por empresas reais para aperfeiçoar ainda mais seus serviços oferecidos.

**Palavras-chave:** Visualização de dados. Banco de dados. SQL. Sistemas de recomendação. Big Data.

# **IntroSpectView: a visual analysis tool for Globoplay's recommendation system**

## **ABSTRACT**

Internet access has grown substantially in recent decades, reaching the mark of billions of users connected with the most diverse devices. A large portion of the volume of data consumed by these users comes from entertainment and streaming platforms, such as Netflix and YouTube, which accounted for 26% of worldwide traffic in 2020. Such prominence requires proper management of the resources of these spaces, both in regards to the infrastructure involved and to the contents created and hosted by the platforms. To help with this, recommendation systems are used to identify user behavior patterns, such as their preferences and interests in relation to the content, in order to apply such information to improve the platforms. Complementarily, similar information can also be identified through data visualization tools, whose main objective is to boost the human capacity to make decisions based on the identification of visual patterns. In this work, we present the IntroSpectView (ISV) tool, a web tool that offers the user the possibility of performing various visual analyzes on the behavior of users of the Globoplay platform, an on-demand video streaming service developed by Grupo Globo. The tool offers several different types of visualization, as well as possibilities for customizing the graphics. We demonstrate how it works using a set of real cases and technically discussing the implemented SQL queries. The results of the views are critically analyzed in order to extract useful information that can be used by real companies to further improve their services.

**Keywords:** Data visualization. Databases. SQL. Recommendation systems. Big Data..

## LISTA DE FIGURAS

Figura 2.1 Arquitetura do sistema <i>Recommendation Dashboard</i> , representando a extensão de navegador desenvolvida, um sistema de recomendação, o acesso a diversas fontes de dados e a camada de painel de controle ( <i>Dashboard</i> ) .....	16
Figura 2.2 <i>Word cloud</i> baseada em títulos de conteúdos da plataforma Netflix, onde palavras que aparecem em tamanho maior indicam maior frequência.....	18
Figura 2.3 Visualização disponível no portal <i>Flourish</i> exibindo os resultados das eleições do Parlamento Europeu de 2014 por meio de um gráfico de agrupamento em círculos. ....	19
Figura 2.4 Visualização do portal <i>Flourish</i> que apresenta a distribuição dos deputados eleitos nas eleições estadunidenses de 2018 de acordo com sua religião, fazendo uso de agrupamento personalizável.	20
Figura 2.5 <i>Dashboard</i> dedicado ao Covid-19 do portal <i>Our World In Data</i> ..	22
Figura 2.6 Página dedicada à visualização do andamento mundial da vacinação contra Covid-19 do portal <i>Our World In Data</i> .....	23
Figura 2.7 Página inicial do portal <i>Painel Coronavírus RS</i> dedicado à visualização da situação da pandemia do Covid-19 no estado do Rio Grande do Sul. É apresentada uma visão geral dos casos confirmados, recuperados, óbitos e taxa de ocupação hospitalar. ....	24
Figura 2.8 Destaque da página <i>Monitoramento da Imunização Covid-19</i> dedicado à visualização da situação da vacinação do Covid-19 no estado do Rio Grande do Sul. ....	25
Figura 3.1 Representação gráfica dos conceitos gerais da modelagem conceitual utilizados em um diagrama entidade-relacionamento. Figura obtida em Heuser (2008). .....	32
Figura 4.1 Diagrama da modelagem entidade relacionamento representando as entidades e relações observadas no universo de discurso da plataforma Globoplay. Fonte: o Autor.....	43
Figura 4.2 Diagrama da Modelagem Lógica representado por tabelas do SGBD, com suas propriedades e conexões a outras tabelas por meio de chaves estrangeiras. Fonte: o Autor. ....	45
Figura 4.3 Visão geral da arquitetura da ferramenta <i>IntroSpectView</i> . Fonte: o Autor. ....	46
Figura 4.4 Arquitetura <i>frontend</i> do sistema com distintos usuários acessando a página web e suas aplicações fazendo requisições para o servidor central simultaneamente. Fonte: o Autor.....	48
Figura 4.5 Diagrama de sequência do fluxo de requisições tratado pelo <i>backend</i> para fornecer os dados necessários a uma visualização da aplicação <i>frontend</i> . Fonte: o Autor. ....	49
Figura 5.1 Imagem da página inicial da ferramenta <i>IntroSpectView</i> com as opções de consulta possíveis de serem selecionadas.....	53
Figura 5.2 Consulta SQL dos reprodutores mais comuns por dispositivo, fazendo uso de uma função SQL para facilitar o processamento de reprodutores por dispositivo.....	55

Figura 5.3 Visualização da consulta dos reprodutores mais comuns por dispositivo por meio de uma visualização de agrupamento em círculos.	57
Figura 5.4 Consulta SQL da distribuição da quantidade de títulos diferentes assistidos pelos usuários, fazendo uso de agrupamentos ( <i>GROUP BY</i> ) na consulta principal assim como na subconsulta .....	59
Figura 5.5 Visualização distribuição da quantidade de títulos diferentes assistidos pelos usuários a través de um histograma em barras assim como a caixa de texto mostrada passar o <i>mouse</i> por cima de algum dos grupos. ....	60
Figura 5.6 Consulta SQL da correlação entre diferentes gêneros de conteúdo. ....	62
Figura 5.7 Gráfico em setores com dois níveis de profundidade para analisar a correlação entre dois gêneros e conteúdos distinos.....	63
Figura 5.8 Consulta SQL da influência dos reprodutores nos gêneros de conteúdo. ....	65
Figura 5.9 Consulta SQL da influência dos reprodutores nos gêneros de conteúdo por meio de um gráfico de cordas. ....	67
Figura 5.10 Consulta SQL da distribuição dos dispositivos dentre os 10 títulos mais vistos. ....	69
Figura 5.11 Visão padrão do gráfico de barras empilhadas da distribuição dos dispositivos dentre os 10 títulos mais vistos. ....	70
Figura 5.12 Visão do gráfico de barras empilhadas filtrando por dispositivos. ....	71
Figura 5.13 Visão do gráfico de barras agrupadas da distribuição dos dispositivos dentre os 10 títulos mais vistos. ....	71

## **LISTA DE TABELAS**

Tabela 4.1 Descrição das colunas do dataset.....	41
Tabela 4.2 Tradução dos nomes do modelo conceitual para os novos nomes empregados no modelo lógico.....	44

## LISTA DE ABREVIATURAS E SIGLAS

ACID	<i>Atomicidade, Consistência, Isolamento, Durabilidade</i>
API	<i>Application Programming Interface</i>
BD	Banco de Dados
CDN	<i>Content Delivery Networks</i>
CNPJ	Cadastro Nacional de Pessoas Jurídicas
COVID19	<i>Coronavirus Disease 2019 2019-nCoV</i>
CSS	<i>Cascading Style Sheets</i>
CSV	<i>Comma-separated values</i> (Arquivo com valores separados por vírgulas)
CRUD	<i>Create, Retrieve, Update, Delete</i> (Criar, Recuperar, Atualizar, Remover)
DOM	<i>Document Object Model</i>
ER	Entidade-Relacionamento
HTML	<i>Hypertext Markdown Language</i>
HTTP	<i>Hypertext Transfer Protocol</i>
REST	<i>Representational State Transfer</i>
RPC	<i>Remote Procedure Call</i>
RS	Rio Grande do Sul
SGBD	Sistema Gerenciador de Banco de Dados
SQL	<i>Structured Query Language</i>
SVG	<i>Scalable Vector Graphics</i>
UI	<i>User Interface</i> (Interface do Usuário)
WAL	<i>Write-ahead Logging</i> (Escrita antes do registro)

## SUMÁRIO

<b>1 INTRODUÇÃO .....</b>	<b>12</b>
<b>2 TRABALHOS RELACIONADOS .....</b>	<b>16</b>
<b>2.1 EEXCESS Recommendation Dashboard .....</b>	<b>16</b>
<b>2.2 Visualização de Dados da Plataforma Netflix Utilizando Python .</b>	<b>17</b>
<b>2.3 Flourish .....</b>	<b>18</b>
<b>2.4 Our World In Data .....</b>	<b>20</b>
<b>2.5 Painel Coronavírus RS .....</b>	<b>23</b>
<b>3 CONCEITOS E TECNOLOGIAS .....</b>	<b>27</b>
<b>3.1 Modelo Cliente-Servidor.....</b>	<b>27</b>
<b>3.2 API HTTP .....</b>	<b>28</b>
<b>3.3 Cache .....</b>	<b>29</b>
<b>3.4 Modelagem dos Dados .....</b>	<b>29</b>
3.4.1 Modelagem Conceitual .....	30
3.4.2 Modelagem Lógica.....	31
<b>3.5 Tecnologias .....</b>	<b>32</b>
<b>3.6 PostgreSQL .....</b>	<b>32</b>
3.6.1 Python.....	34
3.6.1.1 Pandas.....	35
3.6.2 Javascript .....	36
3.6.2.1 NodeJS .....	37
3.6.2.2 Express .....	37
3.6.2.3 React.....	38
3.6.2.4 D3 .....	39
<b>4 INTROSPECTVIEW: UMA FERRAMENTA DE ANÁLISE VISUAL PARA SISTEMAS DE RECOMENDAÇÃO DO GLOBOPLAY .....</b>	<b>40</b>
<b>4.1 Conjunto de Dados.....</b>	<b>40</b>
<b>4.2 Modelagem dos Dados.....</b>	<b>41</b>
4.2.1 Modelagem Conceitual .....	42
4.2.2 Modelagem Lógica.....	44
<b>4.3 Arquitetura do Sistema .....</b>	<b>45</b>
4.3.1 Frontend .....	46
4.3.2 Backend .....	48
<b>4.4 Carregamento dos dados .....</b>	<b>50</b>
4.4.1 Pré-Processamento dos Dados.....	50
4.4.2 Inserção dos dados no Banco .....	51
<b>5 DEMONSTRAÇÃO, ANÁLISE E APLICABILIDADE DA FERRAMENTA.....</b>	<b>52</b>
<b>5.1 Metodologia de Análise .....</b>	<b>52</b>
<b>5.2 Reprodutores mais comuns por dispositivos.....</b>	<b>54</b>
5.2.1 Consulta SQL.....	55
5.2.2 Visualização .....	56
<b>5.3 Distribuição da quantidade de títulos diferentes assistidos pelos usuários.....</b>	<b>58</b>
5.3.1 Consulta SQL .....	58
5.3.2 Visualização .....	59
<b>5.4 Correlação entre gêneros de conteúdos.....</b>	<b>61</b>
5.4.1 Consulta SQL .....	61
5.4.2 Visualização .....	63

<b>5.5 Influência dos reprodutores nos gêneros dos conteúdos.....</b>	<b>64</b>
5.5.1 Consulta SQL .....	65
5.5.2 Visualização .....	67
<b>5.6 Distribuição dos dispositivos dentre os 10 títulos mais vistos ....</b>	<b>68</b>
5.6.1 Consulta SQL .....	69
5.6.2 Visualização .....	70
<b>6 CONCLUSÃO.....</b>	<b>73</b>
<b>REFERÊNCIAS.....</b>	<b>75</b>

## 1 INTRODUÇÃO

A grande difusão do acesso à tecnologia transformou a Internet como conhecemos hoje no maior sistema já criado pela humanidade, com bilhões de usuários conectados por meio dos mais diversos dispositivos (computadores, celulares, sistemas de segurança, relógios, carros...) (KUROSE; ROSS, 2012). Uma grande parte desses acessos é realizado a plataformas de *streaming* e aplicações voltadas para o entretenimento, como *Netflix*, *Amazon Prime*, *Hulu*, *GloboPlay*, *Spotify*, *Disney+*, dentre incontáveis outros. O relatório emitido pela Sandvine em 2020 indica que plataformas de vídeo digitais foram responsáveis por aproximadamente 57% do volume de *downstream* da Internet mundial (SANDVINE, 2020), sendo que apenas o *YouTube* e a *Netflix* já são responsáveis por 15% e 11% de todo o tráfego da rede mundial, respectivamente (SANDVINE, 2020).

Essa crescente quantidade de acessos tem tornado a administração desses espaços uma tarefa complexa, fazendo com que muitas plataformas de *streaming* façam uso de infraestrutura de terceiros para hospedar e distribuir seus conteúdos (KUROSE; ROSS, 2012). Um exemplo disso é observado no caso da *Netflix*, que executa suas páginas *web* e seus servidores inteiramente em servidores em nuvem da *Amazon* (KUROSE; ROSS, 2012). Funções como o processamento de conteúdo (geração de diferentes formatos para cada conteúdo) e carregamento dessas versões para Redes de Distribuição de Conteúdo (*Content Delivery Networks (CDN)*) passam a ser executadas em grandes *clusters* da *Amazon*.

Além de aspectos relacionados à infraestrutura, plataformas de *streaming* também precisam realizar um bom planejamento sobre a criação de novos conteúdos para garantir uma experiência cada vez mais cativante aos usuários (DYE et al., 2020). Dados do histórico de acessos dos usuários sempre foram utilizados para auxiliar na caracterização do comportamento dos mesmos e no direcionamento de novas produções, identificando títulos similares, a forma na qual eles são similares, o possível tamanho da audiência de um novo título, dentre outros fatores. No entanto, a crescente variedade de conteúdos sendo assistidos e o alto número de acessos têm tornado a

identificação de tais características uma tarefa cada vez mais desafiadora (DYE et al., 2020).

Uma das consequências desse cenário é o desenvolvimento de sistemas de recomendações para essas aplicações e plataformas de *streaming* (HALDER; SEDDIQUI, 2014; DYE et al., 2020). Sistemas de recomendação são sistemas baseados em uma série de algoritmos de Aprendizado de Máquina, como *collaborative filtering* (IAQUINTA et al., 2007; CHEN et al., 2008; Merve Acilar; ARSLAN, 2009) usados para filtrar as informações vitais das grandes quantidades de dados geradas dinamicamente pelos acessos dos usuários (ISINKAYE et al., 2015). Tais sistemas identificam os padrões dos usuários, como suas preferências, interesses e comportamentos com relação a um dado item (como produtos e conteúdos) de maneira a traçar um perfil dos mesmos, o que pode ser utilizado na predição de seus comportamentos futuros (ISINKAYE et al., 2015). As informações extraídas podem ser úteis para aprimorar as plataformas e torná-las mais interessantes ao público, explorando tendências e gostos pessoais que levem a um maior engajamento e identificação com o conteúdo.

De maneira complementar e mais amigável aos usuários, sistemas de visualização de dados têm sido desenvolvidos para sumarizar todo tipo de informação gráfica e interativamente. Tais sistemas aproveitam a capacidade de detecção de padrões presente no sistema visual humano para auxiliar na resolução de tarefas de *decision-making* (MUNZNER, 2015), como o exemplo de criação de novos conteúdos para plataformas de *streaming* citado anteriormente. Dessa forma, o usuário continua e é essencial no processo de tomada de decisões sobre os dados disponíveis ao invés de simplesmente ser substituído por mecanismos de decisão automatizados (MUNZNER, 2015), como no caso de sistemas de recomendação.

Decidir quando aplicar sistemas de visualização combinados com sistemas automatizados ou quando aplicar apenas sistemas automatizados é uma questão que depende do tipo de pergunta que se deseja responder (MUNZNER, 2015). Quando tem-se pleno conhecimento de quais são as perguntas a serem respondidas, então sistemas puramente computacionais e estatísticos são perfeitamente aplicáveis. Um exemplo de tal situação são aplicações sensíveis ao tempo como o mercado de ações,

onde quer-se tomar decisões a respeito de compra e venda de ações rapidamente quando uma certa condição for satisfeita. Depender da análise visual humana de tal condição pode ser ineficiente, e transferir a responsabilidade a um sistema automatizado faz mais sentido. No entanto, quando não se conhece suficientemente o problema ou não se sabe exatamente como solucioná-lo, permitir que usuários trabalhem com a informação disponível de forma estruturada e visual pode auxiliar na detecção de padrões e informações úteis que não são percebidas por sistemas inteligentes.

Dois exemplos de ferramentas de visualização e que tiveram extrema importância no contexto da pandemia de Covid-19 são o painel *Coronavirus Pandemic (COVID-19)* (RITCHIE et al., 2020), desenvolvido pelo portal *Our World in Data* (LAB, 2019), e o *Painel Coronavírus RS* (SUL, 2021), desenvolvido pela Secretaria da Saúde do Estado do Rio Grande do Sul. Para que a população tivesse acesso e pudesse compreender a grande quantidade de informações relacionadas à pandemia, foi necessário modelar esses dados e apresentá-los por meio de plataformas gráficas e interativas. Da mesma forma, estratégias de modelagem e visualização também podem ser empregadas pra auxiliar na análise do grande volume de dados gerados a partir da interação dos usuários com as diversas plataformas de *streaming*. Diversos padrões e tendências desses sistemas podem ser identificados por meio da visualização dos dados, e podem auxiliar no gerenciamento e compreensão das atividades observadas nas plataformas. Mais do que isso, elas ainda podem ser aplicadas juntamente com sistemas computacionais de tomada de decisão para avaliar o desempenho de tais sistemas a partir do julgamento humano (MUNZNER, 2015).

Este trabalho apresenta a ferramenta IntroSpectView, uma ferramenta *web* que permite realizar uma análise visual dos comportamentos e padrões dos usuários de um serviço de vídeos sob demanda. Mais especificamente, o conjunto de dados utilizado foi fornecido pelo Globoplay<sup>1</sup>, a plataforma digital de *streaming* de vídeos e áudios sob demanda desenvolvida e operada pelo Grupo Globo. O serviço disponibiliza um catálogo de conteúdos para os seus assinantes pagos poderem assistir, além de oferecer diversas sugestões

---

<sup>1</sup><<https://globoplay.globo.com/>>

e recomendações para seu conteúdo. Com base nesse conjunto de dados, foram desenvolvidas consultas de aplicabilidade real a fim de demonstrar como a ferramenta pode ser utilizada por empresas sobre casos de usos reais. Detalhamos o código SQL das consultas realizadas, exemplos de visualizações disponibilizadas pela ferramenta, bem como possíveis customizações e as possíveis conclusões a ser tiradas a partir dos gráficos gerados.

O restante desse trabalho está organizado da seguinte maneira: o Capítulo 2 apresenta alguns trabalhos e plataformas de análise visual de dados voltados para os mais variados contextos; o Capítulo 3 descreve os conceitos e tecnologias que embasaram o desenvolvimento da ferramenta; o Capítulo 4 discorre sobre a metodologia envolvida no desenvolvimento propriamente dito, detalhando desde a modelagem dos dados até a descrição da arquitetura da ferramenta; o Capítulo 5 faz uso de diversos exemplos de aplicabilidade real para demonstrar o funcionamento da ferramenta, exibindo a consulta escrita em linguagem SQL e diversas visualizações diferentes; o Capítulo 6 apresenta a conclusão desse trabalho, apresentando as principais contribuições e os trabalhos futuros.

## 2 TRABALHOS RELACIONADOS

Neste Capítulo apresentamos alguns trabalhos e ferramentas de análise visual de dados voltados para os mais variados contextos, como política, a pandemia de Covid-19, plataformas de entretenimento, dentre outros. Cada uma das plataformas é descrita e analisada levando em consideração as diferentes abordagens propostas, funcionalidades e opções de visualização disponíveis, seus objetivos e seus aspectos diferenciais.

### 2.1 EEXCESS *Recommendation Dashboard*

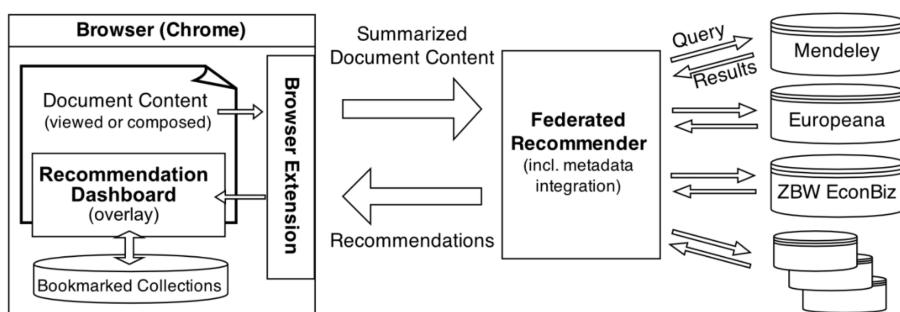
Tendo em mente a dificuldade de se visualizar a crescente quantidade de dados observada em sistemas de recomendação, Tschinkel et al. (2015) propuseram a ferramenta *EEXCESS Recommendation Dashboard*. O sistema é conectado com servidores que possuem dados de contextos culturais, educacionais e geográficos, como o portal Europeana<sup>1</sup>, que contém informações sobre heranças culturais, e o site Mendeley<sup>2</sup>, que contém artigos científicos. A Figura 2.1 descreve a arquitetura do sistema, composta por uma extensão de navegador, um sistema recomendador, diversas conexões a fontes de dados e a ferramenta de *Dashboard* em si.

De maneira resumida, a extensão de navegador captura informações sobre o conteúdo sendo acessado e criado pelo usuário na página web. Com

<sup>1</sup><<https://www.europeana.eu/en>>

<sup>2</sup><<https://www.mendeley.com/>>

Figura 2.1: Arquitetura do sistema *Recommendation Dashboard*, representando a extensão de navegador desenvolvida, um sistema de recomendação, o acesso a diversas fontes de dados e a camada de painel de controle (*Dashboard*).



tais informações, a extensão acessa cada página acessada pelo usuário e requisita os conteúdos para então extrair as palavras chave da página e enviar ao sistema recomendador. A partir das palavras chave, o sistema acessa resultados de várias fontes de dados e integra os metadados obtidos a fim de possibilitar a geração das visualizações pela ferramenta de *Dashboard*.

O *Recommendation Dashboard* (RD) provê diversas formas de o usuário explorar, organizar e filtrar as recomendações realizadas pelo sistema de recomendação. O painel de visualização é composto por cinco diferentes áreas: há um painel de pesquisa que permite ao usuário realizar uma pesquisa sobre os dados; um painel de conjunto de dados por meio do qual o usuário pode escolher qual conjunto de dados será exibido pelo RD; uma lista exibindo as fontes de dados que o sistema de recomendação encontrou com relação ao dado tópico, com informações como título, fonte e linguagem; e um painel de controle com configurações de eixos e cores disponíveis, que permite ao usuário escolher a visualização a ser exibida no painel central. É possível ver uma demonstração da ferramenta em vídeos disponibilizados no YouTube<sup>3</sup>.

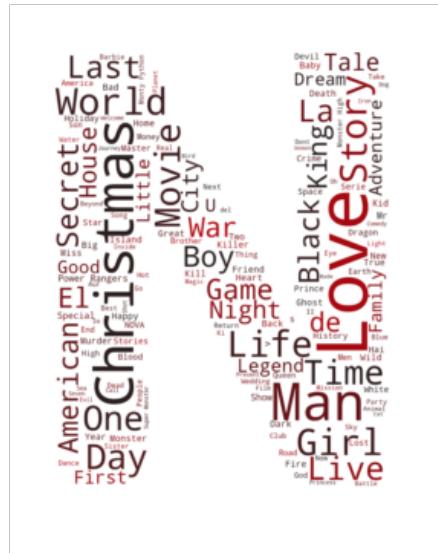
## **2.2 Visualização de Dados da Plataforma Netflix Utilizando Python**

Em uma abordagem diferente de ferramentas *web*, Pradhan (2021) apresentou uma metodologia de visualização de dados em Python (ROSSUM, 1991) em formato de tutorial. O tutorial debate a crescente necessidade da visualização de dados na era do *Big Data*, onde trilhões de linhas de dados são gerados todos os dias (PRADHAN, 2021) e é importante traduzir os dados para formatos mais facilmente comprehensíveis. Diferentes tipos de visualização são apresentados, como gráficos de barra, linhas do tempo, e *word clouds*, como a exibida na Figura 2.2, que representa os títulos da plataforma que aparecem com maior frequência.

---

<sup>3</sup><[https://www.youtube.com/watch?v=IKwEexY\\_Ay4](https://www.youtube.com/watch?v=IKwEexY_Ay4)>

Figura 2.2: *Word cloud* baseada em títulos de conteúdos da plataforma Netflix, onde palavras que aparecem em tamanho maior indicam maior frequência.



O tutorial é baseado na linguagem de programação Python (ROSSUM, 1991) e faz uso das bibliotecas de visualização matplotlib, seaborn, numpy e pandas, mostrando o código utilizado para cada uma das visualizações. Uma introdução a pré processamento dos dados é feita, onde os dados faltantes são analisados e substituídos de acordo com os valores mais frequentes do conjunto de dados e dados inválidos são removidos. Além de demonstrar formas de visualizar os dados, o tutorial ainda apresenta opções de personalização das visualizações, como adição de anotações sobre os gráficos e troca da paleta de cores.

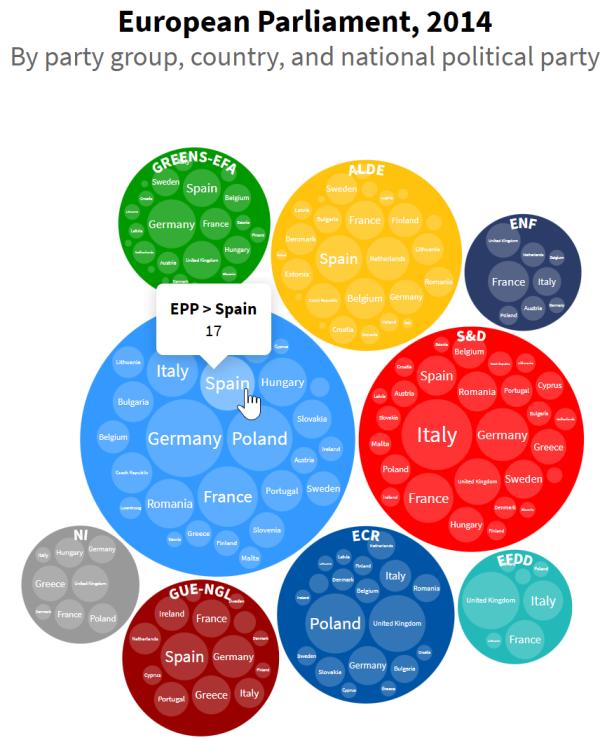
### 2.3 *Flourish*

O portal *Flourish* (LTD, 2016) é uma plataforma de apresentação de matérias e histórias a partir de representações visuais. O seu foco é facilitar a criação de visualizações interativas a partir de arquivos de texto separado por vírgulas (*comma-separated values (CSV)*), conhecido popularmente como csv ou planilha, de modo a melhorar a apresentação e explicação dos dados. Em 15 de Abril de 2019 foi publicada a matéria *Ten ways to visualize elections data*<sup>4</sup>, uma página dedicada à exploração de dados sobre eleições por meio de ilustrações. Usando como base os dados de algumas votações

<sup>4</sup><<https://flourish.studio/2019/04/15/report-on-elections-with-flourish/>>

Figura 2.3: Visualização disponível no portal *Flourish* exibindo os resultados das eleições do Parlamento Europeu de 2014 por meio de um gráfico de agrupamento em círculos.

#### 4. Treemap

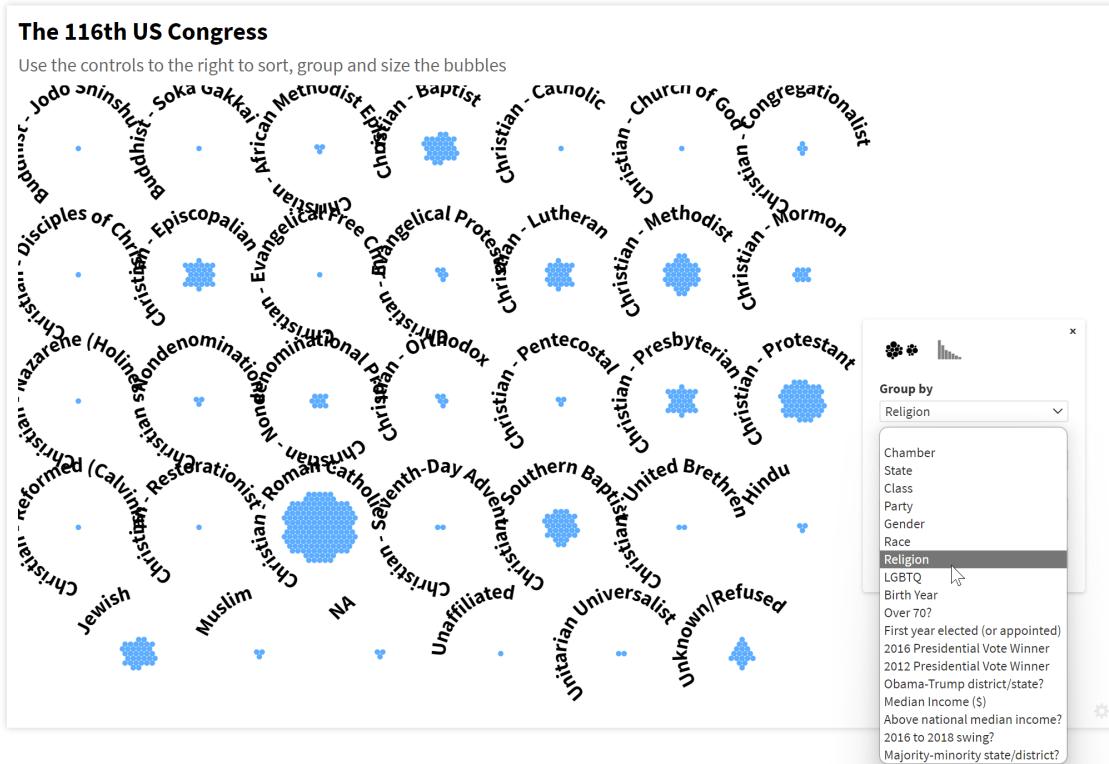


internacionais de significativa relevância geopolítica recentes, como as eleições do Parlamento Europeu ou as eleições dos Estados Unidos, distintas formas de realizar uma análise sobre os resultados dessas eleições são apresentadas utilizando-se dos mais diversos gráficos.

A Figura 2.3 apresenta a utilização de um gráfico de agrupamento em círculos para mostrar os resultados das eleições do Parlamento Europeu de 2014. Cada círculo externo possui uma cor própria que referencia um grupo político distinto, e contém círculos menores que representam quantos deputados daquele partido foram eleitos em cada país. O usuário pode interagir com os círculos menores para saber o número de deputados que o respectivo país elegeu, bem como o nome do partido. Por exemplo, vemos que a Espanha elegeu 17 deputados do partido EPP.

Já a Figura 2.4 mostra um gráfico de agrupamento personalizável a respeito dos congressistas eleitos nas eleições estadunidenses de 2018. O

Figura 2.4: Visualização do portal *Flourish* que apresenta a distribuição dos deputados eleitos nas eleições estadunidenses de 2018 de acordo com sua religião, fazendo uso de agrupamento personalizável.



usuário tem diversas opções de agrupamento a serem escolhidas. No exemplo da Figura, é possível observar a distribuição dos deputados eleitos de acordo com sua religião, permitindo identificar os grupos religiosos com maior (ou menor) representatividade. Com a ajuda desses recursos visuais e interativos torna-se muito mais acessível e intuitiva a análise de dados mais complexos e densos. Por meio de simples e poderosas visualizações, é possível observar e extrair diversas informações interessantes, tais como a tendências dos votantes de um dado grupo étnico, a popularidade de um partido político em uma determinada região ou até a distribuição de idade dos políticos eleitos, a fim de identificar se está ocorrendo uma mudança geracional, por exemplo.

## 2.4 Our World In Data

O portal *Our World In Data* (LAB, 2019) é uma plataforma online de publicação científica especializada em pesquisas e análise de dados sobre as mais diversas questões, como pobreza, doenças, guerras, mudanças

climáticas e desigualdades socioeconômicas. O projeto foi desenvolvido pela *Global Change Data Lab*<sup>5</sup>, uma organização não governamental cujo time é composto por pesquisadores da Universidade de Oxford. Além do cunho informativo sobre questões de interesse global, a plataforma busca gerar gráficos e visualizações interativas sobre os dados. Desta forma, o usuário não só recebe as principais informações de forma visual, como também lhe é permitido interagir e explorar estas representações visuais interativas da forma que melhor lhe convir. A missão do portal é "publicar pesquisas e dados para progredir com relação aos maiores problemas do mundo" (ROSER, 2021).

Com o surgimento e avanço da pandemia do Covid-19, soluções capazes de sintetizar as informações a respeito da situação da doença se tornaram comuns, principalmente as que o faziam através de gráficos e representações visuais. O *Our World In Data* se destacou por lançar a página *Coronavirus Pandemic (COVID-19)* (RITCHIE et al., 2020), dedicada exclusivamente para acompanhar todas as informações relacionadas ao Covid-19 no mundo, como observado na Figura 2.5. Uma das visualizações disponíveis é a *Coronavirus (COVID-19) Vaccinations*<sup>6</sup>, que mostra a situação da vacinação do Covid-19 em 225 regiões ou entidades soberanas. Como pode ser observado na Figura 2.6, a página exibe a porcentagem de população vacinada com uma ou duas doses por meio de um gráfico de barras disposto horizontalmente. A seção da esquerda permite ao usuário escolher quais regiões ele quer visualizar no gráfico, fazendo com que a visualização fique mais ou menos compacta.

Por causa da dinamicidade dos eventos durante a pandemia, surgiu a necessidade de um portal que reunisse a maior quantidade de informações em um único lugar. O fato de ser um portal científico voltado à livre contribuição de informação auxiliou na criação de parcerias com as mais diversas instituições. Órgãos como a Organização Mundial da Saúde, a Organização Pan-Americana da Saúde e o Ministério da Saúde do Brasil apoiaram o projeto disponibilizando seus dados oficiais sobre o estado da pandemia. Com isso, o portal foi fundamental para acompanhar a situação da

---

<sup>5</sup>Disponível em <<https://global-change-data-lab.org/>>. Último acesso em 20/10/2021

<sup>6</sup>Disponível em <<https://ourworldindata.org/covid-vaccinations>>. Último acesso em 17/10/2021.

Figura 2.5: Dashboard dedicado ao Covid-19 do portal *Our World In Data*.

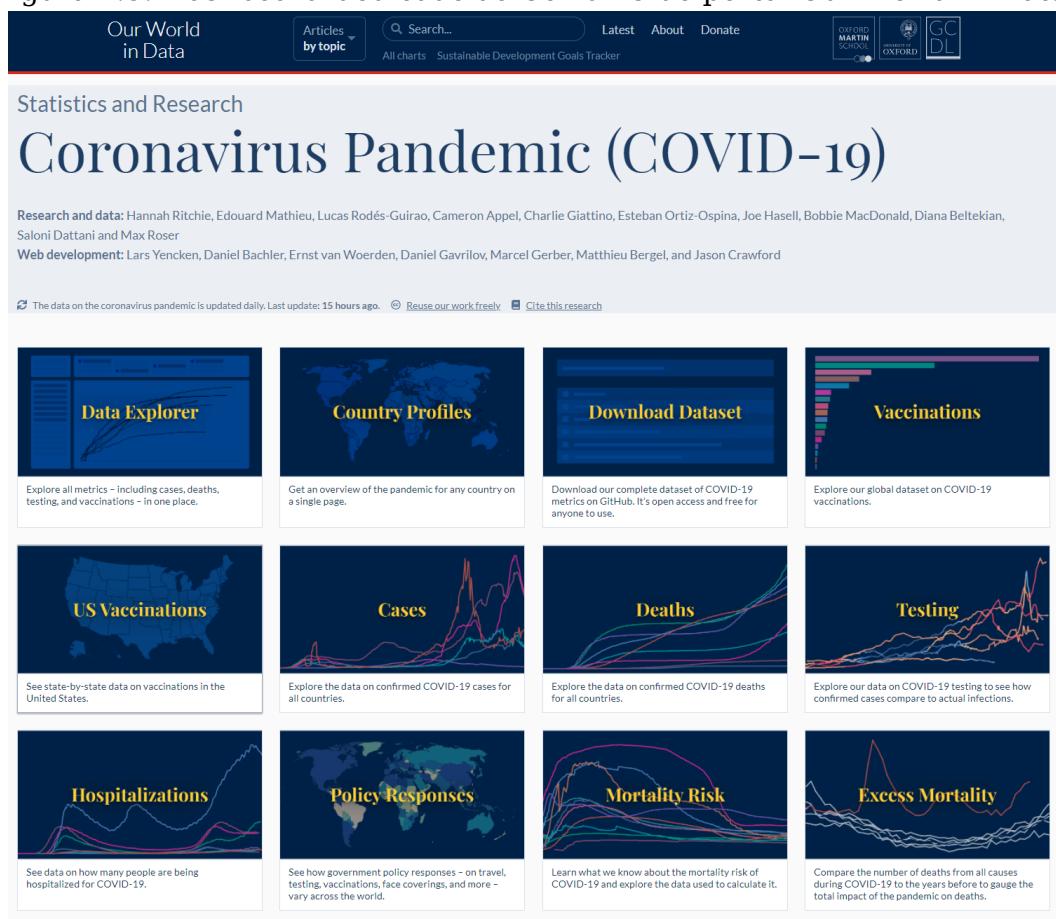
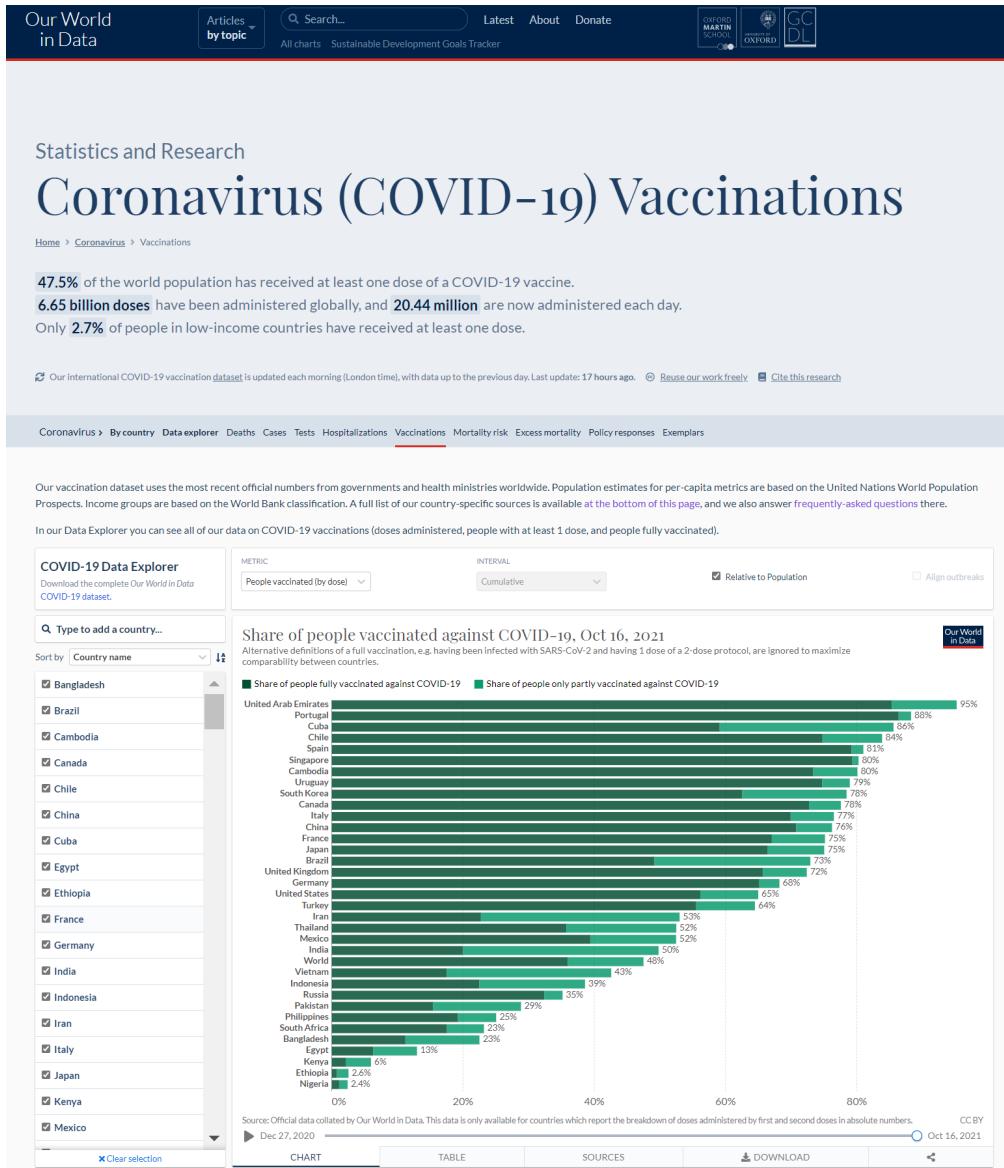


Figura 2.6: Página dedicada à visualização do andamento mundial da vacinação contra Covid-19 do portal *Our World In Data*.

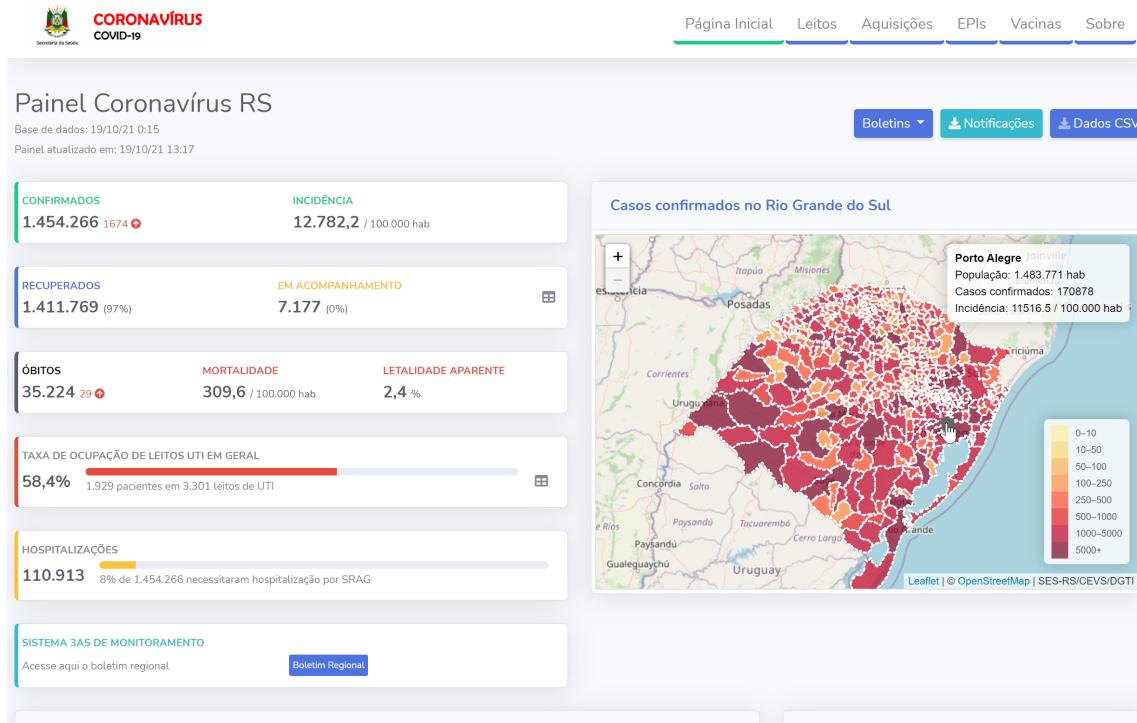


pandemia quase em tempo real dada a frequência e qualidade dos dados disponíveis.

## 2.5 Painel Coronavírus RS

A Secretaria da Saúde do Estado do Rio Grande do Sul lançou o Painel Coronavírus RS (SUL, 2021), um portal dedicado a apresentar os dados epidemiológicos da Covid-19 no estado do Rio Grande do Sul. As informações são obtidas a partir das notificações oficiais do Ministério da Saúde do Brasil. Através dessa ferramenta é possível acompanhar a situação atual da

Figura 2.7: Página inicial do portal *Painel Coronavírus RS* dedicado à visualização da situação da pandemia do Covid-19 no estado do Rio Grande do Sul. É apresentada uma visão geral dos casos confirmados, recuperados, óbitos e taxa de ocupação hospitalar.



pandemia no estado de forma bastante precisa, uma vez que os dados são atualizados diariamente e contém informação sobre a taxa ocupação hospitalar dos leitos, contágios e fatalidades decorrentes do vírus, ritmo da inoculação da vacina do Covid-19 entre outros detalhes.

Como pode ser observado na Figura 2.7, na página principal do painel estão reunidas as principais informações sobre a situação do Rio Grande do Sul de forma resumida: quantidade de casos confirmados e recuperados, incidência acumulada por 100.000 habitantes, óbitos e situação hospitalar. Também é exibido o mapa do estado do Rio Grande do Sul delimitando cada um dos 496 municípios por cor segundo a quantidade de casos na determinada região. Um recurso interessante dessa ferramenta é um mapa estadual com os municípios delimitados, como pode ser observado à direita da Figura. Através dele é possível acompanhar a situação de cada um dos 496 municípios do estado, e conforme o cursor é movido por cima dos municípios, algumas informações sobre a situação de cada local são exibidas. Isto permite um maior detalhamento sobre o andamento da pandemia a nível tanto regional quanto municipal.

Figura 2.8: Destaque da página *Monitoramento da Imunização Covid-19* dedicado à visualização da situação da vacinação do Covid-19 no estado do Rio Grande do Sul.



Na página Monitoramento da Imunização Covid-19, representado pela Figura 2.8, é possível acompanhar o estado da vacinação no estado. São utilizados diversos modelos de gráficos para representar as características dos vacinados. Podemos reparar no gráfico de barras sobre a faixa etária dos vacinados idade, assim como no gráfico de barras empilhadas sobre a proporção de vacinados com apenas uma dose e com o esquema vacinal completo. Também pode ser observado gráficos de pizza, ou por setores, informando características como a raça e sexo da população vacinada. Há também uma visualização sobre a proporção das marcas de vacinas aplicadas, informação que pode ser utilizada juntamente com o tempo de espera necessário para a aplicação da segunda dose. Isso pode auxiliar a entender, por exemplo, porque há faixas etárias com uma porcentagem muito elevada de apenas uma dose havendo sido inoculada (a fabricante utilizada para esse grupo pode requerer um intervalo maior de espera).

Ferramentas de visualização de dados foram amplamente utilizadas no contexto da Covid-19 por permitirem à população uma melhor compreensão das informações envolvendo a pandemia. Dados muitas vezes complexos e contendo relações escondidas para a maior parte das pessoas passaram a ficar ao alcance das mesmas. O desenvolvimento da ferramenta IntroSpectView permite que a mesma estratégia seja aplicada para o contexto de entretenimento tal qual apresentado pelo tutorial mencionado acima, mas expandindo amplamente as capacidades de visualização. Embora sistemas de recomendação contribuam na interação entre usuários e plataformas, a representação visual das informações permite que diferentes comportamentos e necessidades sejam observados, como a necessidade de suporte ou descontinuação para um dado tipo de dispositivo e tendências no comportamento do público afim de direcionar novas produções. A ferramenta desenvolvida nesse trabalho auxiliará na resolução de tais atividades da mesma forma que o portal *Our World in Data* e o *Painel Coronavírus RS* auxiliaram na compreensão do estado da pandemia do Covid-19.

### 3 CONCEITOS E TECNOLOGIAS

Neste Capítulo, introduzimos os conceitos que serviram como base para o desenvolvimento da ferramenta IntroSpectView, como o modelo de sistema Cliente-Servidor, o que é a API HTTP e como o conceito de *cache* é utilizado for ferramentas *web*. Além disso, também detalhamos as tecnologias utilizadas para dar sustentação à implementação dos conceitos mencionados, como linguagens de programação e suas bibliotecas.

#### 3.1 Modelo Cliente-Servidor

A arquitetura de aplicações *web* é baseada no modelo Cliente-Servidor, por meio do qual separam-se as funções de requisição de serviços (o lado do cliente) e de oferecimento de serviços (o lado do servidor) (COULOURIS et al., 2011; KUROSE; ROSS, 2012). O processamento dos dados é comumente de responsabilidade do servidor, que encaminha os resultados de tal processamento para os clientes que requisitaram as informações. Tal modelo de organização exige a criação de protocolos de comunicação por meio dos quais as informações são compartilhadas entre o servidor e o cliente. O Protocolo de Transferência de Hipertexto (*Hypertext Transfer Protocol – HTTP*) (FIELDING et al., 1999) é um protocolo de camada de aplicação que implementa o modelo cliente-servidor e define um padrão de comunicação entre as partes.

Organizar as aplicações *web* em um formato cliente-servidor permite dividir o processamento em diferentes máquinas, bem como reduzir a replicação de dados por centralizar o armazenamento das informações no lado do servidor. Dessa forma, o lado do cliente, que executa no navegador do usuário, contém principalmente a lógica de apresentação da aplicação, o que requer menos recursos e esforços de programação. Ao mesmo tempo, o servidor tem a capacidade de responder a diversos clientes e concentrar a maior parte dos recursos da ferramenta em questão.

No contexto de engenharia de software e desenvolvimento *web*, além de empregar o conceito de cliente-servidor, utilizam-se também os termos *frontend* e *backend*. Costuma-se designar o termo *frontend* (ou *front-end*)

para a parte cliente de uma aplicação, ou seja, aquela voltada ao usuário, responsável tanto pela apresentação visual quanto pela interação com este. O *frontend* é o responsável por interagir com o *backend* (ou *back-end*), o responsável pelas regras de negócio da aplicação, garantindo aspectos de segurança como autenticação, autorização, sanitização, consulta a banco de dados, processamento de dados, serialização, dentre outras.

### 3.2 API HTTP

Uma *Application Programming Interface* (API, em português Interface de Programação de Aplicativos) é um conjunto de métodos e padrões estabelecidos por um *software* de forma que consiga entregar recursos e serviços a outras aplicações sem que estas precisem conhecer a implementação interna das funcionalidades, sendo necessário apenas seguir os requisitos definidos no contrato da interface. No contexto de rede, uma API HTTP é um serviço exposto para a rede por meio de um servidor *web* através de diversas rotas HTTP. Dessa forma, cada rota é definida pelo seu endereço único e por um conjunto de parâmetros que uma dada aplicação precisa passar quando requisitar pela determinada rota a fim de acessar as funcionalidades disponíveis corretamente.

Hoje em dia, APIs HTTP são comumente estruturadas no formato REST (NOTTINGHAM, 2010; SHELBY, 2012) ou GraphQL (FACEBOOK, 2018; BRITO; VALENTE, 2020), embora no passado tenha-se usado com frequência APIs HTTP SOAP (MANNIE; PAPADIMITRIOU, 2006) ou RPC (BUELTHOFF; MALESHKOVA, 2014; NEUMANN et al., 2018). A abstração chave da informação em uma API REST é um "recurso útil". Quaisquer informações que possam ser nomeadas podem ser um recurso útil: um relatório ou imagem, um serviço temporal, um conjunto de diferentes posses, um item não-virtual (uma pessoa), e assim por diante. REST utiliza um identificador único para os recursos, que por sua vez também possuem relacionamentos com diferentes outros recursos. Ao utilizar uma API REST, temos um número fixo de ações (ou verbos) para realizar nesses recursos (comumente chamado de CRUD - *Create, Retrieve, Update, Delete* - Criar, Recuperar, Atualizar, Remover) (SUJAN et al., 2020).

### 3.3 Cache

Caches são estruturas que armazenam objetos e dados recentemente acessados em locais mais próximos do usuário do que do local de origem dos dados (COULOURIS et al., 2011; KUROSE; ROSS, 2012). O conceito de *caching* pode ser aplicado em diversos contextos da computação, como arquitetura e serviços web. Quando um objeto é recebido de um servidor web, o mesmo é adicionado à *cache* local do navegador web e substitui-se os valores antigos quando necessário. No momento em que o usuário ou a aplicação cliente requisitar um dado, verifica-se primeiramente se o dado encontra-se em *cache*. Em caso positivo, o dado é disponibilizado ao cliente rapidamente, caso contrário, a requisição é encaminhada ao servidor que contém as informações necessárias (COULOURIS et al., 2011; KUROSE; ROSS, 2012).

A aplicação de *web caching* tem potencial de reduzir significativamente o tempo de resposta para uma requisição do cliente por estar fisicamente mais próxima do mesmo (KUROSE; ROSS, 2012). Tal capacidade é ainda mais importante ao considerarmos o possível gargalo de transferência de dados que pode existir entre o cliente e o servidor, que tende a ser superior ao possível gargalo existente entre cliente e *cache* (KUROSE; ROSS, 2012). Além disso, a aplicação de *cache* também colabora com a diminuição do tráfego do sistema com a Internet como um todo.

### 3.4 Modelagem dos Dados

Ao desenvolver-se aplicações que trabalham com dados, o projetista deve estar ciente de problemas como a redundância de dados não controlada, quando as informações podem aparecer representadas diversas vezes no sistema. Tal problema acontece quando a aplicação não tem controle sobre a manutenção e representação das informações, deixando a tarefa nas mãos do usuário (HEUSER, 2008). Para evitar tais situações, o compartilhamento de dados pode ser empregado, onde os dados ficam centralizados em um único mecanismo de armazenamento chamado de banco de dados (BD). Ao compartilharmos os dados, a estrutura interna dos mesmos torna-se mais

complexa, uma vez que passa a ser necessário atender aos diferentes sistemas que acessam esses dados. A tarefa de gerenciar e manter os banco de dados é função dos sistemas de gerência de banco de dados (SGBD) (HEUSER, 2008), que implementa funções de definição, recuperação e alteração de dados em um BD.

Uma vez que tem-se um sistema específico para o armazenamento e gerenciamento dos dados de uma aplicação, é necessário gerar um modelo de dados, que envolve a descrição formal dos tipos de informações armazenadas no BD. A ação de modelar os dados é o processo por meio do qual cria-se um modelo de dados que auxilia na representação visual das informações e reforça as regras existentes entre os dados do domínio, bem como as políticas de administração dos mesmos (HEUSER, 2008). Modelos de dados são representações abstratas que garantem a consistência de convenções de nomenclatura, valores padrões, aspectos semânticos e questões de segurança (HEUSER, 2008). É possível mapear os dados para diferentes níveis de abstração, como a modelagem conceitual, que fornece um modelo de dados útil para exibição ao usuário, sem detalhes técnicos e relacionados ao SGBD, e como a modelagem lógica, na qual são detalhados aspectos mais técnicos e sobre como as informações estão organizadas internamente no SGBD.

### **3.4.1 Modelagem Conceitual**

A criação de um esquema conceitual envolve uma modelagem em alto nível das informações que compõem o universo de discurso, independente de SGBD, de modo a descrever quais são os conceitos e como eles se relacionam entre si (HEUSER, 2008). O objetivo é apresentar uma visão geral do sistema a ser modelado reconhecendo as entidades individualmente identificáveis e como elas se relacionam. O modelo conceitual tanto pode ser utilizado para definir as entidades da organização que possuem informações armazenadas no BD, quanto como um modelo abstrato do banco de dados, que define quais são os arquivos que farão parte do BD (HEUSER, 2008). O processo de construção de um BD comumente se inicia pela definição de um modelo conceitual que envolve a participação do usuário, uma vez que é ele quem

detém o maior conhecimento a respeito de seus dados e pode auxiliar na descrição e organização dos mesmos.

Para esse nível de abstração, as informações são representadas por meio da abordagem entidade-relacionamento (ER) (HEUSER, 2008). Um modelo ER é uma representação gráfica que, por meio de um diagrama ER, apresenta os conceitos gerais da modelagem conceitual: entidade, relacionamento, atributo, generalização/especialização e entidade associativa (HEUSER, 2008):

- Entidade: representa um conjunto de objetos da realidade que está sendo modelada e sobre os quais se deseja manter informações;
- Relacionamento: informações a respeito da associação entre diferentes objetos/entidades;
- Atributo: informações relacionadas às ocorrências de entidades ou de relacionamentos;
- Generalização/Especialização: mecanismos de atribuição de propriedades a entidades. Propriedades particulares a um subconjunto das ocorrências (especializadas) de uma entidade genérica;
- Entidade associativa: usada para permitir que haja uma relação entre um relacionamento N:N entre duas entidades e uma terceira entidade. Mais especificamente, um relacionamento N:N é um relacionamento no qual uma ocorrência de uma das entidades envolvidas pode se relacionar com N ocorrências da outra entidade, e vice versa.

A Figura 3.1 mostra a representação gráfica dos conceitos mencionados acima.

### **3.4.2 Modelagem Lógica**

Uma vez tendo definida e realizada a modelagem conceitual, passamos para a elaboração do projeto lógico. Nesta etapa, o objetivo principal é fazer o mapeamento do modelo de dados do esquema conceitual para um esquema lógico relacionado ao SGBD (HEUSER, 2008). Ou seja, escolhemos um SGBD e trabalhamos no mapeamento do modelo de alto nível para um modelo lógico: um projeto que introduza entidades operacionais e relacionais de

Figura 3.1: Representação gráfica dos conceitos gerais da modelagem conceitual utilizados em um diagrama entidade-relacionamento. Figura obtida em Heuser (2008).

Conceito	Símbolo
Entidade	
Relacionamento	
Atributo	
Atributo identificador	
Relacionamento identificador	
Generalização/ especialização	
Entidade associativa	

modo a obedecer as especificidades do SGBD, utilizando-se de seus recursos e levando em consideração suas limitações.

### 3.5 Tecnologias

Nesta Seção, detalhamos as linguagens de programação, bibliotecas e tecnologias utilizadas para o desenvolvimento da ferramenta IntroSpectView.

### 3.6 PostgreSQL

O banco de dados escolhido para sustentar os dados da IntroSpectView foi o PostgreSQL (STONEBRAKER; ROWE, 1986), ao qual se realizam consultas por meio de uma conexão aberta através da biblioteca Psycopg2 (DIGREGORIO; VARRAZZO, 2010). PostgreSQL é um poderoso banco de dados objeto-relacional de código aberto que utiliza e expande a

linguagem SQL com várias funcionalidades para armazenar e escalar as cargas de trabalho mais intensas de forma segura. As suas origens remontam a 1986 como parte do projeto *POSTGRES* na *University of California* em Berkeley, e possui mais de 30 anos de desenvolvimento no núcleo de sua plataforma (POSTGRES, 2021). Pelo fato de ser código aberto, muitos outros bancos de dados relevantes são desenvolvidos baseados na implementação do PostgreSQL, como EdgeDB<sup>1</sup>, TimeScale<sup>2</sup>, Apache HadoopDB<sup>3</sup> e Amazon Redshift<sup>4</sup>. No PostgreSQL podem ser definidas funções SQL utilizando duas linguagens: a linguagem padrão *SQL*<sup>5</sup> e a linguagem específica do PostgreSQL, chamada *PG/SQL*<sup>6</sup>. A linguagem padrão SQL facilita a portabilidade das consultas SQL em caso de reutilização ou migração para outros SGDBs que sejam compatíveis com a especificação padrão do SQL.

Embora o mercado tenha sido dominado por bancos de dados como MySQL (WIDENIUS et al., 2002) e Microsoft SQL Server (MICROSOFT, 2021) no passado, PostgreSQL tem atraído muitos usuários nos últimos anos, sendo o segundo banco de dados mais utilizado pelos desenvolvedores em 2021 (OVERFLOW, 2020). É altamente extensível, permitindo criar seus próprios tipos de dados, funções customizadas, *plugins*, além de estar de acordo com a especificação SQL (SQL, 2011). Oferece proteção ACID (Atomicidade, Consistência, Isolação e Durabilidade), utilizando WAL (*Write-ahead Logging* - Traduzido livremente como "Escrita com antecedência") como seu sistema de *journaling* a fim de evitar falhas catastróficas, além de possuir sistemas de replicação assíncrona, síncrona ou lógica.

Seu modelo objeto-relacional é padrão como o esperado de bancos SQL. Possui diferentes tabelas que representam diferentes entidades do universo de discurso, sendo tais tabelas estruturadas com colunas e tipos definidos à priori. Cada objeto do universo de discurso é representado como

---

<sup>1</sup><<https://www.edgedb.com/>>

<sup>2</sup><<https://www.timescale.com/>>

<sup>3</sup><<http://dslam.cs.umd.edu/hadoopdb/hadoopdb.html>>

<sup>4</sup><<https://aws.amazon.com/pt/redshift/>>

<sup>5</sup>Disponível em <<https://www.postgresql.org/docs/14/xfunc-sql.html>>. Último acesso em 18/10/2021.

<sup>6</sup>Disponível em <<https://www.postgresql.org/docs/14/plpgsql.html>>. Último acesso em 18/10/2021.

uma linha dessa tabela, contendo exatamente sempre as mesmas informações. No entanto, esse universo de discurso é mutável, o que permite adicionar novas colunas com o passar do tempo e atribuir valor nulo a algumas colunas (*NULL*). Para facilitar a busca de itens tem-se o conceito de índices, que são árvores de pesquisa capazes de rapidamente encontrar nodos que respeitem uma *query* sem precisar percorrer toda a tabela à procura dos dados.

### 3.6.1 Python

Python (ROSSUM, 1991) é uma linguagem de programação interpretada, com tipagem dinâmica, orientada a objetos, de alto nível, com uma sintaxe simples e intuitiva para o programador iniciante. Sua sintaxe simples enfatiza a legibilidade, tornando o processo de aprendizado da linguagem bastante acessível. Possui estruturas de dados de alto nível embutidas em sua biblioteca padrão (como pode ser visto pelo seu slogan de "*batteries included*", indicando que pretende possuir tudo que o desenvolvedor necessita diretamente na sua biblioteca padrão) que, combinadas com a tipagem dinâmica, tornam Python atraente para o rápido desenvolvimento de aplicações, dado à facilidade que um programador tem de construir aplicações complexas. Python também é bastante utilizada como linguagem de *script* e como um mecanismo de conexão ou "cola" de outros componentes. O grande ambiente proporcionado pela linguagem e sua sintaxe simples proporcionam um baixo custo de manutenção dos programas desenvolvidos em Python, além de grande facilidade para expandir seu código. É também muito utilizado no ambiente acadêmico (IRVING et al., 2021), principalmente àquele voltado à análise de dados, possuindo muitas bibliotecas famosas para esse tipo de aplicação (como NumPy<sup>7</sup> e Pandas<sup>8</sup>).

Python apresenta um gerenciador de pacotes incorporado à linguagem, o *pip* (FOUNDATION, 2011), que proporciona uma capacidade poderosa de expandir seu código utilizando um amplo ambiente de bibliotecas distribuídas na Internet. As bibliotecas são hospedadas pelo

---

<sup>7</sup><<https://pypi.org/project/numpy/>>

<sup>8</sup><<https://pypi.org/project/pandas/>>

repositório oficial de bibliotecas *PyPi* (FOUNDATION, 2003) ou por repositórios extra-oficiais como o *Anaconda* (ANACONDA, 2012).

### 3.6.1.1 Pandas

Pandas (MCKINNEY, 2020) é uma biblioteca de código aberto e uso livre para a linguagem de programação Python dedicada à análise e manipulação de dados. Oferecendo um amplo leque de estruturas de dados e operações, tem como objetivo tornar fácil e flexível a manipulação de conjuntos de dados grandes e complexos. Dentre as tarefas simplificadas pela biblioteca, pode-se citar:

- Carregamento de dados: possui primitivas para ler arquivos de texto ou arquivos CSV e carregar já em memória para estruturas próprias chamadas de DataFrame.
- Limpeza: permite a identificação da nulidade de vários tipos de dados e validação dos valores, e é compatível tanto com valores numéricos, quanto com *strings*, datas e até estruturas complexas utilizando-se de regras personalizáveis.
- Normalização: existem diversos métodos para normalização dos dados de uma determinada coluna ou um conjunto de colunas quando se trabalha com dados em formato de planilha.
- Integração: são oferecidos métodos para realizar operações mais complexas e avançados sobre os conjuntos de dados, como é o caso de processamento de cálculo integral ou infinitesimal.
- Visualização: a biblioteca matplotlib (D., 2007), uma das mais famosas bibliotecas de visualização em Python, possui integração com o Pandas e recomenda o uso de DataFrames para estruturar os dados antes de serem renderizados.
- Análise estatística: diversas primitivas e recursos são oferecidos para executar análises estatísticas sobre os DataFrames, tais como cálculo de media e mediana, agregação, agrupamento e discretização.

### 3.6.2 Javascript

Javascript (EICH, 1995), formalmente ECMAScript 2021 (TC39, 2021), é uma linguagem de programação utilizada principalmente em páginas web, embora também possa ser utilizado em servidores (NodeJS) ou para construir aplicações *desktop* (Electron<sup>9</sup>). É uma linguagem de alto nível, com tipagem dinâmica, possuindo uma sintaxe muito parecida com a sintaxe de linguagens mais antigas como C ou Java. A linguagem é comumente referida como orientada a protótipos, uma abstração mais complexa do que a orientação à objetos. Embora as versões mais recentes da linguagem possuam recursos populares como classes e objetos, estes são dados apenas por *syntactic sugar* (LANDIN, 1964) (açúcar sintático em tradução livre), pois são implementados a partir de recursos e funcionalidades que já existiam anteriormente na linguagem. A versão ES6 da linguagem foi a responsável pela modernização da linguagem, trazendo-a aos níveis esperados de uma linguagem moderna base para a estrutura da Internet mundial, adicionando funcionalidades como iteradores funcionais, assincronicidade, escopo estático, etc.

O Javascript tornou-se a principal linguagem da *web*, tendo evoluído juntamente com a mesma e contribuído para tal. Estima-se que hoje em dia Javascript está presente em 95.2% (AGIRA, 2021) do conteúdo disponível na Internet. A linguagem possui uma comunidade de pessoas desenvolvedoras muito ativa e engajada, em constante evolução e desenvolvendo bibliotecas e recursos de livre uso a todo momento. Dessa forma, os principais produtos e serviços passaram a oferecer suporte à linguagem. Esta comunidade é conhecida por construir muitas bibliotecas de uso aberto, hospedadas e distribuídas através do NPM<sup>10</sup>. Essas bibliotecas não se restringem somente a fins de desenvolvimento web, mas expandem seu alcance para as mais diversas áreas tais como: computação gráfica (como a biblioteca P5.js (PROCESSING, 2014)), inteligência artificial e aprendizado de máquina (Tensorflow.js (SMILKOV et al., 2019)), financeiro, dentre outras.

---

<sup>9</sup><<https://www.electronjs.org/>>

<sup>10</sup><<https://www.npmjs.com/>>

### 3.6.2.1 NodeJS

NodeJS (DAHL, 2009), muitas vezes considerado apenas como um *framework*, é na verdade um ambiente de execução para a linguagem Javascript que atua no lado do servidor, sem necessidade de um navegador *web* para sua execução. Para que Javascript possa rodar fora do *browser*, o NodeJS utiliza de uma plataforma chamada V8 (GOOGLE, 2008c) para executar o código JS. Essa plataforma, desenvolvida pela empresa Google, também é utilizada nos navegadores Google Chrome (GOOGLE, 2008b) e Chromium (GOOGLE, 2008a), a versão de código aberto do Google Chrome.

Uma característica importante do NodeJS é seu ambiente "pseudo *multithread*". O programador é capaz de criar código que aparenta ser assíncrono e/ou *multithread* através das primitivas conhecidas como *promises* e *callbacks*. No entanto, o NodeJS cria somente uma *thread* no sistema operacional, e faz uso de um mecanismo chamado de *event loop* para realizar essas operações "aparentemente" simultâneas de forma sequencial.

Por se tratar de uma extensão da linguagem de programação Javascript, é possível aproveitar tudo o que esta oferece: uma linguagem alto nível, com sintaxe acessível e de fácil legibilidade, com suporte para os paradigmas de programação tanto de orientação a objeto quanto de programação funcional. Também, pelo fato de estarmos executando fora de um navegador, temos acesso a alguns recursos adicionais como sistemas de arquivos e ao *shell* da máquina. Estas funcionalidades extras nos permitem realizar tarefas mais poderosas do que aquelas que temos na *sandbox* do navegador.

### 3.6.2.2 Express

Express (HOLOWAYCHUK; STRONGLOOP, 2010) é um *framework* minimalístico e flexível para a criação de servidores *web* que provê um conjunto robusto de funcionalidades. É facilmente extensível e configurável, possuindo uma grande variedade de *middlewares* que facilitam a customização do servidor e como ele tratará as requisições recebidas. Sua principal característica está justamente na facilidade de criar uma API (interface da aplicação) baseada em rotas HTTP, disponibilizando diferentes

partes da nossa aplicação e/ou lógica de negócio em diferentes rotas consumíveis pelo cliente. Podemos, por exemplo, requisitar uma rota <*seudominio.com.br/rota1*> e o servidor será capaz de identificar a requisição a essa rota específica e diferenciá-la de outro cliente executando uma requisição à rota <*seudominio.com.br/rota2*>.

É importante notar, porém, que o Express é uma aplicação *singlethread* (pelos motivos já explicados na Seção 3.6.2.1) e, portanto, não é capaz de processar muitas requisições simultâneas. Para contornar esse problema, costuma-se utilizar uma aplicação como a PM2<sup>11</sup> para executar várias aplicações Express ao mesmo tempo, uma em cada núcleo da máquina. O PM2 é responsável pelo balanceamento de carga das requisições entre as diferentes instâncias Express, porém, essas instâncias não compartilham nenhum estado e nem sequer sabem que as outras instâncias existem.

### 3.6.2.3 React

React (FACEBOOK, 2013) é uma biblioteca para *frontend* escrita na linguagem de programação JavaScript e voltada para a criação de páginas *web* interativas. Seu funcionamento é baseado no conceito de "componentes", que representam cada uma das discretas partes que formam a página *web*. Cada componente pode possuir um estado lógico, o que influencia na sua aparência e funcionamento, além de possuir componentes filhos (em uma direta relação com a estrutura em árvore do formato HTML utilizado para renderizar páginas *web*).

O React trabalha com um ciclo de estados, que é responsável por checar se é necessário atualizar os dados atualmente renderizados no navegador *web* a partir de eventos causados por interação do usuário ou por tempo. Tal controle é feito por meio de um DOM virtual mantido em memória que representa o que deveria ser renderizado no DOM do navegador. Assim que é detectado alguma alteração, são atualizados somente os componentes que realmente tiveram sua aparência atualizada. Pelo fato de cada componente gerenciar seu próprio estado, é possível criar interfaces de usuário complexas com cada parte da interface sendo responsável por garantir a integridade do seu próprio estado e de sua própria aparência.

---

<sup>11</sup> <<https://pm2.keymetrics.io>>

Dessa forma, torna-se mais fácil manter as informações dos dados sincronizadas com as informações disponibilizadas para o usuário do navegador.

#### 3.6.2.4 D3

D3<sup>12</sup>, ou D3.js, é uma biblioteca JavaScript com foco em visualização de dados. A sua premissa é conseguir gerar gráficos interativos de forma simples e fácil a partir de dados estruturados. Utilizando-se de SVG, HTML e CSS, responsáveis respectivamente por imagens, texto e estilização, a ferramenta oferece inúmeros gráficos personalizáveis, permitindo estilizar e adicionar comportamentos de interação de forma simples.

O grande diferencial desta ferramenta é que ela visa não apenas tornar fácil e intuitiva a criação de representações visuais, como também coloca bastante ênfase na interação do usuário com estas representações. Com isto, o usuário consegue explorar as visualizações e interagir com elas, gerando um maior engajamento, e um consequente maior interesse e melhor entendimento na análise que está sendo realizada sobre os dados.

Na sua página oficial, o D3 disponibiliza uma série de exemplos e guias sobre como utilizar a ferramenta, além de tutoriais sobre os métodos e recursos oferecidos. Dentre eles podemos destacar gráficos de barras, tanto agrupados como empilhados, gráficos de linhas, de dispersão, pizza/torta, entre muitos outros.

---

<sup>12</sup><<https://d3js.org/>>

## 4 INTROSPECTVIEW: UMA FERRAMENTA DE ANÁLISE VISUAL PARA SISTEMAS DE RECOMENDAÇÃO DO GLOBOPLAY

Neste capítulo discorremos sobre a metodologia empregada para o desenvolvimento da ferramenta IntroSpectView. Iniciamos pela descrição e contextualização do *dataset* fornecido pelo Globoplay. Detalhamos como os dados foram modelados, descrevendo o universo de discurso e as respectivas modelagens conceitual e lógica. Seguimos com a descrição da arquitetura de software que compõe o projeto, como este se organiza e o seu funcionamento detalhado, tanto com relação à página web, quanto ao servidor HTTP, e como este se comunica com o banco de dados. Por fim, explicamos como foi realizada a etapa de processamento e tratamento dos dados, junto com a posterior inserção destes no banco de dados.

### 4.1 Conjunto de Dados

O conjunto de dados (*dataset*) utilizado como base para este trabalho foi oferecido pelo serviço Globoplay a partir da parceria realizada entre a empresa *Globo Comunicação e Participações S.A.* e o grupo de pesquisa da Prof<sup>a</sup>. Dr<sup>a</sup>. Renata Galante e do Prof. Dr. Weverton Cordeiro, do Instituto de Informática da Universidade Federal do Rio Grande do Sul.

O Globoplay<sup>1</sup> é uma plataforma digital de *streaming* de vídeos e áudios sob demanda, desenvolvida e operada pelo Grupo Globo. O *dataset* cedido que foi utilizado nesta ferramenta é composto por informações de acesso à plataforma organizadas em um conjunto de arquivos de texto separado por vírgulas (*comma-separated values (CSV)*), conhecido popularmente como *csv* ou planilha. As entradas do conjunto de dados representam o registro das sessões de acesso que os usuários da plataforma realizaram a conteúdos, totalizando 2.195.785 (dois milhões e cento e noventa e cinco mil e setecentos e oitenta e cinco) de entradas. As informações são referentes ao período das 00:00:00 horas do dia 11 de maio de 2021 até às 23:59:59 horas do dia 13 de julho de 2021. Além disso, cada acesso registrado é detalhado

---

<sup>1</sup><<https://globoplay.globo.com/>>

em diversas colunas que contêm informações como o título assistido, o usuário que o assistiu e as propriedades do dispositivo utilizado, dentre outras informações omitidas por questões de sigilo de dados. Os campos usados neste trabalho são descritos na Tabela 4.1.

Tabela 4.1: Descrição das colunas do dataset.

	<b>Descrição</b>	<b>Tipo</b>	<b>Exemplo</b>
<b>user_ID</b>	Identificador único de um usuário	Numérico	54712
<b>title_ID</b>	Identificador único do título digital assistido	Numérico	2289
<b>genres</b>	Gêneros do respectivo título	Lista de String	Comédia, Drama
<b>deviceGroup</b>	Nome do dispositivo utilizado	String	TV
<b>playertype</b>	Nome do reproduutor utilizado	String	Sony

## 4.2 Modelagem dos Dados

Para realizar a modelagem dos dados, primeiro precisamos compreender o contexto e o universo do discurso que está sendo abordado. Ao entender a realidade da organização ou do serviço em questão, prosseguimos para o levantamento e análise dos requisitos necessários para a realização do trabalho proposto. A ferramenta IntroSpectView trabalha sobre uma plataforma para assistir mídias digitais, o Globoplay, um serviço de vídeo sob demanda onde um usuário pode navegar pela plataforma explorando os diversos títulos disponíveis para assistir. Ao selecionar um título, seja um filme, uma série ou um show musical, o serviço exibe uma pequena descrição resumindo do que trata esse conteúdo. Conforme o usuário vai utilizando a plataforma, esta lhe sugere alguns conteúdos que estão em alta, sendo bastante assistidos entre usuários com características similares, como usuários que tenham assistido filmes relacionados, usuários que usem os mesmos tipos de dispositivos para acessar a plataforma, que marquem como favorito os mesmo gêneros de conteúdos etc. Uma vez tendo escolhido o que assistir, o usuário pode então decidir reproduzir o conteúdo em algum dos seus dispositivos: seja na televisão, em algum dispositivo móvel ou até mesmo no computador pessoal. Internamente, o Globoplay coleta métricas sobre este acesso do usuário, tais como informações de que

região ele está assistindo, o dispositivo pelo qual está acessando o serviço, em qual reproduutor selecionou assistir o conteúdo, qual o título escolhido e suas características.

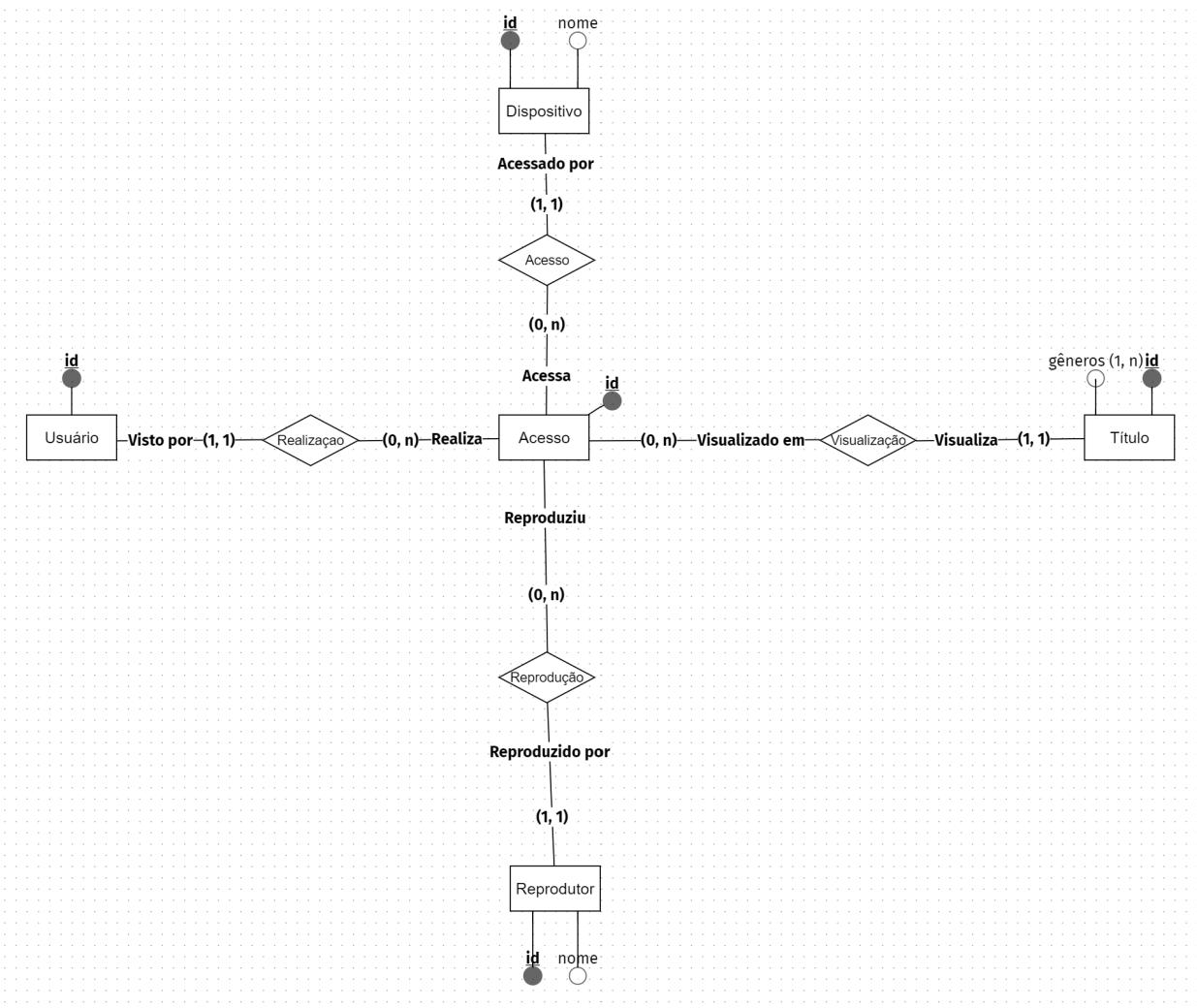
Após entender melhor o funcionamento do Globoplay e ele é organizado, é possível identificar as principais entidades que interagem entre si. Tais entidades e relações devem ser mapeadas em diferentes níveis de abstração de modo a permitir o correto funcionamento do serviço seguindo os diversos fluxos de funcionamento possíveis:

- Usuário: representa uma pessoa do mundo real cadastrada no sistema.
- Título: representa um título de série, filme ou áudio.
- Dispositivo: representa um tipo de dispositivo que pode acessar a plataforma.
- Reprodutor: representa um tipo de reproduutor no qual é possível assistir algum título.
- Acesso: representa um acesso de visualização de um título, por um usuário, através de um dispositivo e em um reproduutor.

#### **4.2.1 Modelagem Conceitual**

Como foi introduzido anteriormente, a ferramenta IntroSpectView é composta de seis entidades: Usuário, Título, Dispositivo, Reprodutor, Acesso. Após definir estas 5 entidades, prosseguimos para a modelagem entidade-relacionamento. Nesta etapa realizamos o mapeamento de todas as entidades juntamente com seus atributos que as identificam e descrevem. Estas entidades funcionam de forma conjunta, havendo portanto relações entre elas para que haja coesão na sua existência e funcionamento. O diagrama entidade-relacionamento exibido na Figura 4.1 mostra a lógica da modelagem conceitual envolvendo as seis entidades pré-definidas. É possível observar que o diagrama ER gira em torno da entidade *Acesso*, a entidade central à qual se relacionam todas as outras entidades.

Figura 4.1: Diagrama da modelagem entidade relacionamento representando as entidades e relações observadas no universo de discurso da plataforma Globoplay. Fonte: o Autor.



#### 4.2.2 Modelagem Lógica

Para a etapa da modelagem lógica precisamos mapear os elementos do modelo ER para tabelas de banco de dados. A fim de seguir os padrões mais comuns da indústria e manter compatibilidade com as bibliotecas utilizadas, utilizamos os nomes das tabelas a serem criadas na língua inglesa. Para uma melhor compreensão, a tradução dos nomes utilizados na seção 4.2.1 para os nomes utilizados nesta seção podem ser encontrados na Tabela 4.2.

Tabela 4.2: Tradução dos nomes do modelo conceitual para os novos nomes empregados no modelo lógico.

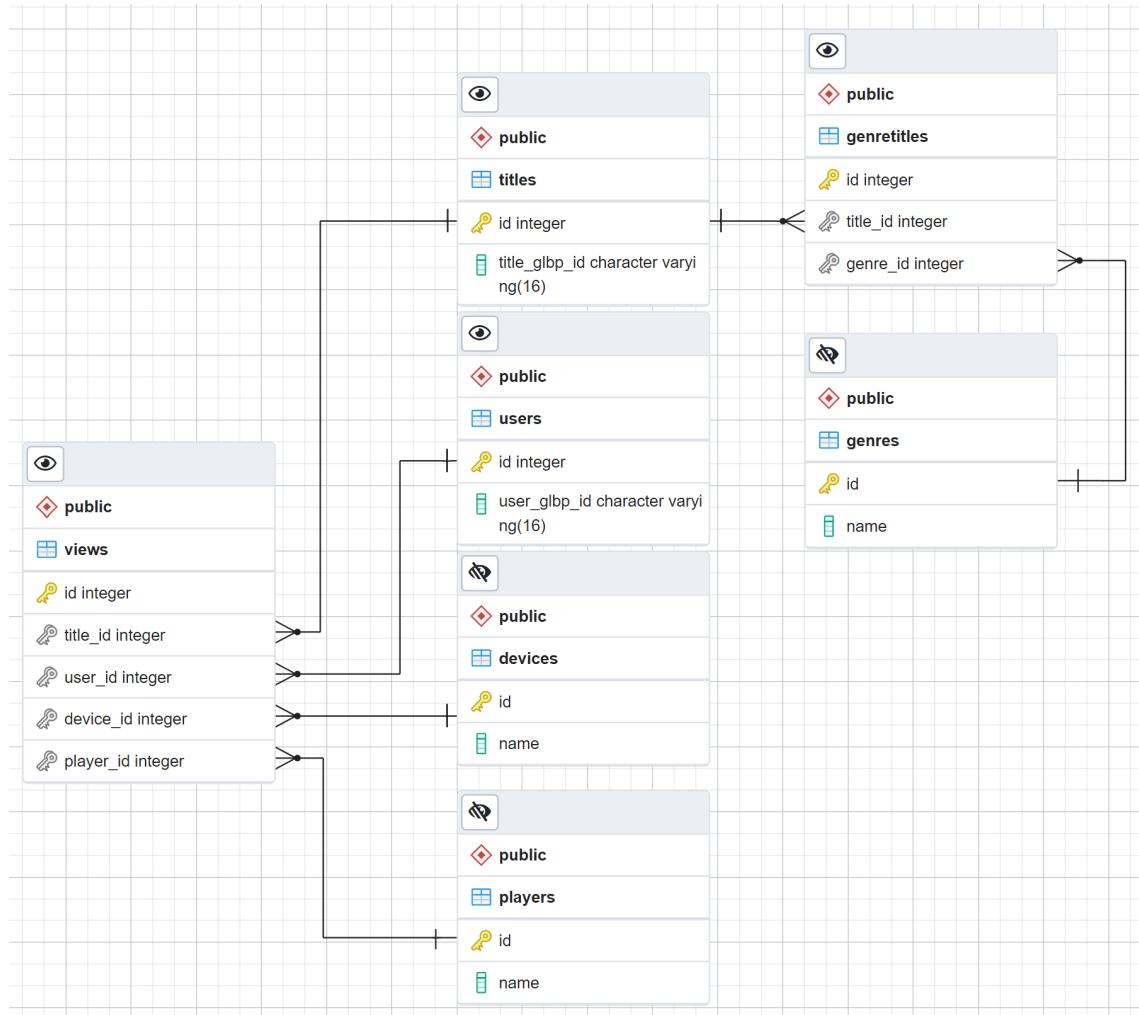
Nome conceitual	Nome lógico
Usuário	Users
Título	Titles
Dispositivo	Devices
Reprodutor	Players
Acesso	Views

As relações com cardinalidades  $(1 - 1) \leftrightarrow (0 - n)$  são facilmente representadas por meio de 2 tabelas, 1 para cada entidade. A entidade que possuir a cardinalidade  $(0 - n)$  no modelo ER conterá a chave estrangeira da outra tabela. Isto é aplicado em todas as entidades que possuíam um relacionamento desta cardinalidade. Por exemplo, a entidade Acesso foi mapeada para uma tabela chamada Views que possui chaves estrangeiras para as tabelas Titles, Users, Devices e Players.

Como mostra a Figura 4.1, a entidade Título possui uma quantidade variável de elementos para o atributo gêneros. Pelo fato destes valores fazerem parte de um conjunto finito e previamente conhecido de gêneros, optou-se pela criação de uma nova tabela Genres. Esta tabela possui um relacionamento muitos-para-muitos para a tabela Titles, o que implica na necessidade de uma tabela de junção (também conhecida como *link table* ou *join table*), criada com o nome de GenreTitles. Foi tomada esta decisão de implementação a fim de evitar duplicação de dados em distintas entradas na tabela e evitar assim um superdimensionamento da mesma. Se fosse optado por um campo gêneros diretamente na tabela, precisaríamos lidar com uma lista de *strings*, o que tornaria o trabalho do SGBD mais complexo, pois

algumas ferramentas não aceitam este tipo de dado. Na nossa implementação, a tabela auxiliar criada mantém as 2 chaves estrangeiras de tipo inteiro, para as tabelas *Genres* e *Titles*, o que torna-se muito menos custoso do que armazenar uma lista variável de strings. O resultado do mapeamento do modelo conceitual para o modelo lógico pode ser observado na Figura 4.2

Figura 4.2: Diagrama da Modelagem Lógica representado por tabelas do SGBD, com suas propriedades e conexões a outras tabelas por meio de chaves estrangeiras. Fonte: o Autor.



### 4.3 Arquitetura do Sistema

A ferramenta IntroSpectView é composta por uma aplicação *frontend*, uma interface *web* que se comunica com uma aplicação *backend*; o servidor central HTTP que contém todas as lógicas e regras de negócio; e o banco de

Figura 4.3: Visão geral da arquitetura da ferramenta *IntroSpectView*. Fonte: o Autor.



dados PostgreSQL. O servidor *backend* é responsável por fazer o papel intermediário entre a interface *web* e o banco de dados, estabelecendo uma conexão segura com o mesmo, realizando as consultas SQL necessárias para satisfazer a requisição do cliente. Dessa forma, quando o usuário seleciona uma das opções de visualizações que a página *web* oferece, uma requisição é enviada ao servidor central a partir do navegador. O servidor, por sua vez, faz as consultas necessárias ao banco de dados, reestrutura os resultados destas consultas e os retorna à aplicação *web* para que esta renderize o gráfico correspondente com os dados recebidos.

A Figura 4.3 descreve a relação acima, exibindo uma comunicação bidirecional entre a aplicação *frontend* que é executada no navegador do usuário e o servidor central. Uma comunicação bidirecional também ocorre entre o servidor e o banco de dados, tornando explícito o papel intermediário interpretado pelo servidor. Nas próximas seções, descrevemos com mais detalhes a implementação de cada uma das partes que compõem a arquitetura da ferramenta *IntroSpectView*.

#### 4.3.1 *Frontend*

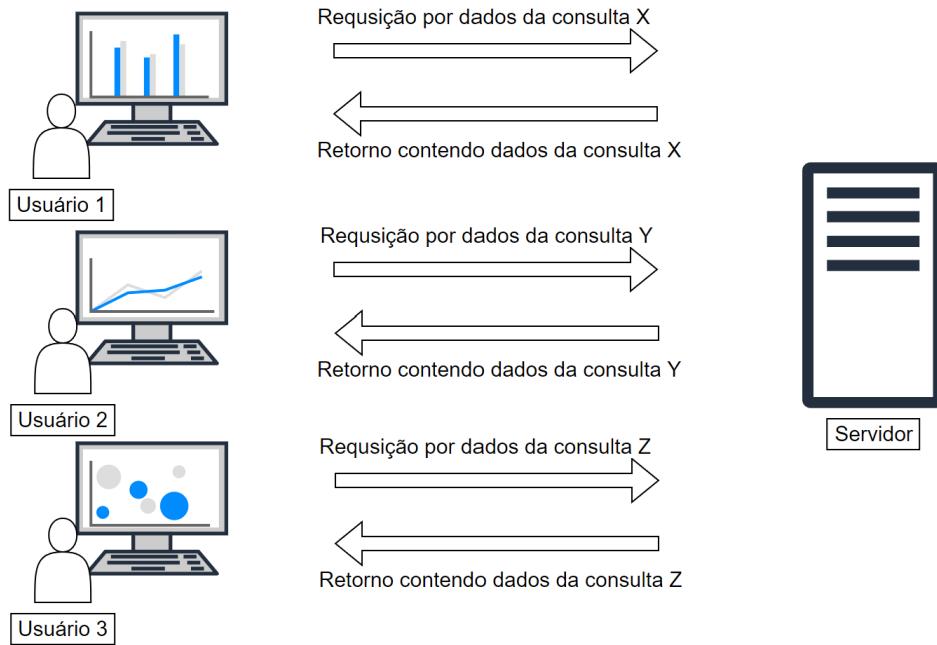
A aplicação *frontend* foi desenvolvida utilizando a biblioteca de código aberto React, que é baseada na linguagem de programação JavaScript e busca facilitar a criação de interfaces de usuário (*user interface – UI*) e seus componentes de forma encapsulada. A página *web* foi desenvolvida para oferecer uma interface amigável e interativa ao usuário, com a opção de escolher entre diferentes tipos de consultas a serem realizadas e analisadas a partir de representações visuais que são geradas. Para exibição do resultado das consultas dos usuários de forma iterativa e dinâmica foi empregada a

biblioteca D3: Data-Driven Documents. A biblioteca é baseada em JavaScript e é voltada para a criação de gráficos e visualizações interativas orientada a dados, com ênfase nos padrões de desenvolvimento web.

Quando o usuário escolhe a consulta a ser realizada, uma nova página no navegador é aberta. A aplicação *frontend* envia uma requisição *HTTP* ao servidor central requisitando os dados necessários para gerar a visualização escolhida (tal processo será descrito em maiores detalhes na Seção 4.3.2). Ao receber o retorno da requisição, estes dados são estruturados pela biblioteca React de forma que a biblioteca D3 consiga gerar uma visualização interativa com os dados da consulta.

Por ser uma ferramenta *frontend*, a aplicação executa inteiramente no navegador do usuário e apenas depende do servidor *backend* para requisitar os dados que estão armazenados no banco de dados. Isto permite que diversos usuários possam acessar a página ao mesmo tempo em seus navegadores, navegar por ela e consultar as representações de dados disponíveis, como exibido na Figura 4.4. Só há interação com o servidor quando uma requisição por dados é realizada, e uma vez que os dados são retornados pelo *backend*, o usuário analisa e interage com a visualização montada sem mais necessidade de comunicação externa. Dessa forma, independente de falhas no servidor ou de conexão com a Internet, a página *web* continua funcionando ainda sendo possível visualizar, interagir e analisar os dados.

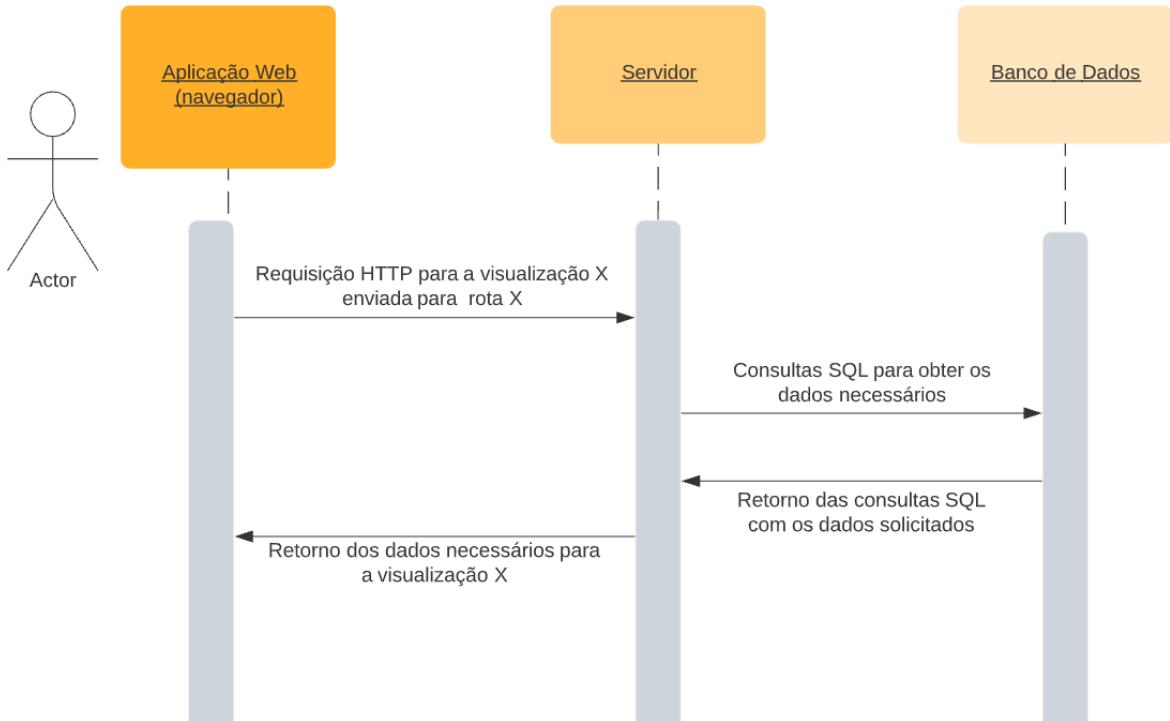
Figura 4.4: Arquitetura *frontend* do sistema com distintos usuários acessando a página web e suas aplicações fazendo requisições para o servidor central simultaneamente. Fonte: o Autor.



#### 4.3.2 Backend

O *backend* da ferramenta IntroSpectView é constituído por uma aplicação NodeJS, um *framework* de código aberto da linguagem de programação JavaScript para execuções no lado do servidor. Utilizando a biblioteca Express, que permite criar servidores *web* de forma simples e rápida, foi criado um servidor *web* HTTP para atender às requisições da aplicação *frontend* meio de uma API HTTP. O servidor disponibiliza diferentes rotas que podem ser acessadas por meio de requisições HTTP, cada uma referente a uma das consultas disponíveis. Como é sumarizado pela Figura 4.5, ao receber uma requisição em uma das rotas HTTP expostas, a aplicação que executa no servidor realiza as consultas necessárias ao banco de dados a fim de obter os dados necessários para a requisição recebida. Depois de realizar as consultas necessárias e receber as devidas respostas, o servidor reorganiza e estrutura todos os dados recebidos e os retorna como resposta da requisição original enviada pelo *frontend*.

Figura 4.5: Diagrama de sequência do fluxo de requisições tratado pelo *backend* para fornecer os dados necessários a uma visualização da aplicação *frontend*. Fonte: o Autor.



O servidor web aceita múltiplas requisições em paralelo e, numa implementação mais ingênuia, realizaria uma consulta SQL para cada uma das requisições recebidas, mesmo se elas fossem exatamente a mesma consulta. Isto é, apesar de as consultas ao banco de dados serem processadas em paralelo, cada requisição do *frontend* sempre esperará por, no mínimo, o tempo necessário para o processamento de sua respectiva consulta SQL. Em casos de consultas SQL mais complexas, com subconsultas ou *joins* complexos, o tempo de processamento pode chegar a 3 minutos. Isso significa que se vários usuários estiverem acessando o sistema ao mesmo tempo e tentando realizar essa mesma consulta SQL, cada uma poderá levar 3 minutos para receber os dados de sua visualização.

Uma vez que não se espera nenhuma atualização nos dados armazenados no banco de dados com muita frequência devido à natureza da ferramenta, foi proposto uma melhoria em cima desta visão: criar uma cache com tempo de expiração para cada uma das rotas existentes. Ao receber uma requisição em uma dada rota  $R$  (e após as devidas consultas e estruturação

dos dados), o servidor salva em memória os dados estruturados visando melhorar o desempenho da ferramenta. Durante os próximos 5 minutos, todas as requisições para esta rota serão retornadas imediatamente com os dados salvos previamente em memória. Após esse período, o servidor executa novamente a consulta SQL e atualiza a *cache* de forma proativa, fazendo com que as próximas requisições recebidas do *frontend* sejam retornadas com os dados atualizados. Com isto, as requisições do *frontend* são respondidas de forma imediata, pois acessar a memória é muito mais eficiente do que realizar um acesso ao banco de dados, e o usuário apenas precisa esperar para carregar a visualização na sua tela.

## 4.4 Carregamento dos dados

Nesta seção explicamos como foi realizada a etapa de carregamento do nosso conjunto de dados no Banco de dados PostgreSQL. Iniciamos descrevendo a etapa de pré-processamento e tratamento dos dados e, a seguir, aprofundamos a explicação da ferramenta desenvolvida para realizar as inserções de dados no banco de dados.

### 4.4.1 Pré-Processamento dos Dados

Os arquivos do *dataset* fornecido foram extraídos do banco de dados da Globo, que é populado a todo instante com informações em tempo-real por terceiros. Por depender de fatores externos como o envio de dados através da rede, e pelo fato de os serviços serem acessados pelos mais diversos tipos de dispositivos que muitas vezes não são completamente compatíveis, é possível que haja perda ou corrupção dos dados salvos. Dessa forma, não há como garantir a integridade destes dados, o que faz com que seja necessário realizar um pré processamento dos mesmos para remover entradas inválidas e que comprometessem a qualidade das análises.

Para esta fase inicial de tratamento e processamento de dados, foi empregada a linguagem de programação Python, juntamente com a ferramenta de análise e manipulação de dados Pandas (MCKINNEY, 2020).

Com a Pandas, cada um das planilhas foi lida e carregada em memória, removendo-se todas as linhas com algum campo cujo valor não condizia com seu tipo especificado, como indicado na Tabela 4.1, bem como linhas com algum campo nulo. Por exemplo, registros em que o gênero do título assistido era uma lista vazia, assim como casos em que os campos *deviceGroup* e *playerType* eram nulos resultaram na remoção das entradas, uma vez que tais informações não poderiam ser recuperadas.

#### **4.4.2 Inserção dos dados no Banco**

Após tratar todos os dados do nosso conjunto, prosseguimos para a etapa de inserção dos mesmos no banco de dados. Para isto, desenvolvemos uma aplicação Python que, utilizando a biblioteca Psycopg2 (DIGREGORIO; VARAZZO, 2010), estabelece uma conexão com o PostgreSQL a fim de realizar as devidas consultas SQL.

Cada uma das tabelas que podem ser observadas na Figura 4.2 foi mapeada para uma Classe, aproveitando-se do paradigma de Orientação a Objeto oferecido pela linguagem Python. Os arquivos CSV eram lidos sequencialmente e para cada linha lida eram criadas 7 instâncias, uma de cada classe. Como estas instâncias precisam ter referências entre si uma vez que possuem chaves estrangeiras compartilhadas, é necessário realizar a inserção no banco de dados de maneira ordenada. Dessa forma, é possível obter-se a chave primária gerada pela inserção e usá-la para preencher as demais classes que apontam para esta recém inserida. Após preencher cada instância com os atributos necessários, é realizada a transação que persiste no banco de dados todas as instâncias.

Se for necessário inserir mais dados posteriormente, caso receba-se mais um conjunto de dados seguindo o mesmo esquema, a aplicação apenas requer que esses novos arquivos sejam salvos no mesmo diretório onde já está presente o conjunto de dados atual. Ao rodar a aplicação, esta mantém um histórico de todos os arquivos que já foram processados e inseridos no banco de dados, o que provocaria que apenas os novos arquivos CSV tenham suas entradas mapeadas para instâncias das classes e posteriormente persistidas no banco de dados.

## 5 DEMONSTRAÇÃO, ANÁLISE E APLICABILIDADE DA FERRAMENTA

Para fins de demonstração da ferramenta desenvolvida, neste Capítulo apresentamos uma coleção de consultas realizadas sobre o conjunto de dados disponível. Para cada uma destas, apresentamos a sua visualização, explicando o gráfico utilizado, o motivo da sua escolha e algumas observações que podem ser extraídas a partir da sua análise. Também é apresentado o código SQL que sustenta tal consulta, analisando o seu funcionamento, os pontos fortes e fracos da implementação, além de comentar eventuais otimizações que podem ser aplicadas.

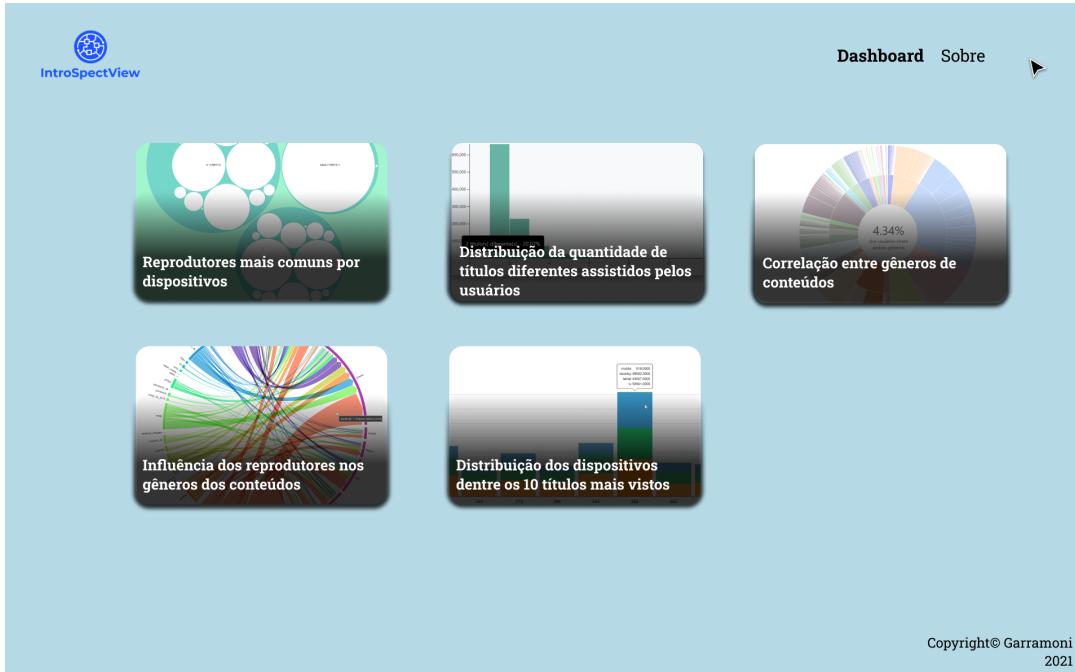
### 5.1 Metodologia de Análise

Durante 5 semanas, foram realizadas reuniões semanais com integrantes da equipe de *Big Data* e Recomendação da GLOBO COMUNICAÇÃO E PARTICIPAÇÕES S/A com o intuito de realizar um levantamento de requisitos e itens de análise, definindo o escopo que este trabalho teria e a sua contribuição para a empresa. As consultas que são apresentadas nas próximas Seções têm como característica principal serem de aplicabilidade real, mostrando como a ferramenta pode, de fato, ser utilizada no dia a dia de empresas sobre casos de uso reais.

Uma descrição do caso de uso, contexto e aplicação das consultas é apresentada, bem como possíveis resultados e conclusões que podem ser observados a partir das visualizações geradas. Também descreve-se a estrutura e a lógica relacionadas à consulta escrita em linguagem SQL, analisando o seu funcionamento durante a execução, os pontos positivos e negativos. Para cada caso, o resultado é apresentado por meio de uma visualização diferente, e discorre-se sobre o tipo de visualização escolhida, como é o seu funcionamento e quais informações podem ser obtidas a partir dela. Por fim, analisamos alguns dos dados extraídos da correspondente visualização, as conclusões que foram observadas, sua relevância e os possíveis casos de uso.

Assim sendo, e com o objetivo de caracterizar o comportamento e o perfil dos usuários de plataformas e serviços de entretenimento, o seguinte

Figura 5.1: Imagem da página inicial da ferramenta IntroSpectView com as opções de consulta possíveis de serem selecionadas.



conjunto de consultas foi realizado afim serem analisados por meio de representações visuais:

- Distribuição da quantidade de títulos diferentes assistidos por cada usuário;
- Reprodutores mais comuns por dispositivos;
- Correlação entre gêneros de conteúdos;
- Influência dos reprodutores nos gêneros de conteúdos;
- Distribuição por dispositivos nos títulos mais vistos;
- Distribuição da quantidade de títulos diferentes assistidos por usuário único.

A Figura 5.1 exibe a tela principal da ferramenta IntroSpectView. Pode-se observar as diferentes consultas que podem ser acessadas diretamente dessa página, com seu respectivo título e uma pequena pré-visualização de seu resultado. O usuário pode clicar no item *Dashboard* no canto superior direito para voltar para esta tela inicial a partir de qualquer uma das visualizações. Cada uma das consultas oferecidas ao usuário serão explicadas mais detalhadamente nas próximas Seções.

## 5.2 Reprodutores mais comuns por dispositivos

O objetivo desta Seção é apresentar uma consulta sobre os reprodutores mais comuns por tipo de dispositivo. Explorar esta característica é motivado pelo fato de existirem diversas formas de consumir entretenimento hoje em dia, seja por meio de dispositivos portáteis ou por meio dos tradicionais reprodutores de mídia e televisões. Cada tipo de dispositivo possui diferentes características atreladas ao seu contexto específico. Entender como os usuários estão acessando o conteúdo ajuda a entender melhor o público alvo de cada plataforma e as possíveis demandas e melhorias que precisam ser adotadas pela empresa fornecedora do conteúdo.

Para dispositivos do tipo *televisão*, reprodutores são geralmente usados por algumas fabricantes específicas. Dessa forma, é possível identificar quais são as marcas mais comuns entre os usuários e optar pela continuação ou descontinuação do suporte técnico oferecido às mesmas. No caso dos dispositivos *mobile*, reprodutores representam o tipo de sistema operacional, como *Android* ou *iOS*. Esta informação pode ser aplicada, por exemplo, para optar pelo desenvolvimento de um aplicativo específico para cada plataforma, ou por um híbrido que seja compatível para ambas.

### 5.2.1 Consulta SQL

Figura 5.2: Consulta SQL dos reprodutores mais comuns por dispositivo, fazendo uso de uma função SQL para facilitar o processamento de reprodutores por dispositivo.

```

1 CREATE OR REPLACE FUNCTION count_players_per_device (int)
2   RETURNS TABLE (
3     total_views bigint,
4     name varchar
5   )
6   AS $$ 
7   SELECT
8     count(distinct user_id) AS total_views,
9     Players.name
10  FROM
11    Views
12  JOIN Players ON player_id = Players.id
13  WHERE
14    device_id = $1
15  GROUP BY
16    Players.name
17  ORDER BY
18    total_views DESC
19 $$;
20 LANGUAGE SQL;
21
22
23 SELECT
24   counter.total_views AS total_views,
25   counter.name AS player,
26   d.name as on_device
27 FROM Devices d, count_players_per_device(d.id) AS counter;
```

A Figura 5.2 mostra o código SQL implementado para a consulta. Foi utilizado de uma função SQL a fim de facilitar o processamento e cálculo de reprodutores por dispositivo. Nessa sintaxe, declaramos o nome da função seguido por uma lista dos tipos dos parâmetros recebidos (neste caso, somente um parâmetro de tipo *int*). Para nos referirmos aos parâmetros, nesta e nas subsequentes consultas em que for descrito o funcionamento de funções SQL, utilizaremos a sintaxe *\$1*, *\$2*, ..., *\$n* *\$1*, *\$2*, *\$3*, ..., *\$n*, que indica o primeiro parâmetro, o segundo parâmetro, e assim por diante, até o n-ésimo parâmetro. Com relação ao retorno das funções, o mesmo é declarado utilizando uma *TABLE*. Por padrão, retornaria-se os campos de forma não estruturada, o que impediria de acessá-los pelo nome. Quando retorna-se uma tabela, tem-se a possibilidade de acessá-los por nome às custas da necessidade de definir os

tipos de antemão (o que não é, necessariamente, um problema).

A execução da função SQL é realizada por meio de um *join* da tabela *Views V* a partir do campo *V.player\_id* seguido do agrupamento de todas as visualizações do correspondente *Player P*, retornando a quantidade total de visualizações junto com o nome do reproduutor. A *query* propriamente dita, definida na linha 23, é responsável apenas por chamar a função previamente definida no banco de dados para cada um dos dispositivos existentes. Por fim, a *query* retorna tuplas  $\{visualizações\ totais, nome\ do\ reproduutor, nome\ do\ dispositivo\}$ .

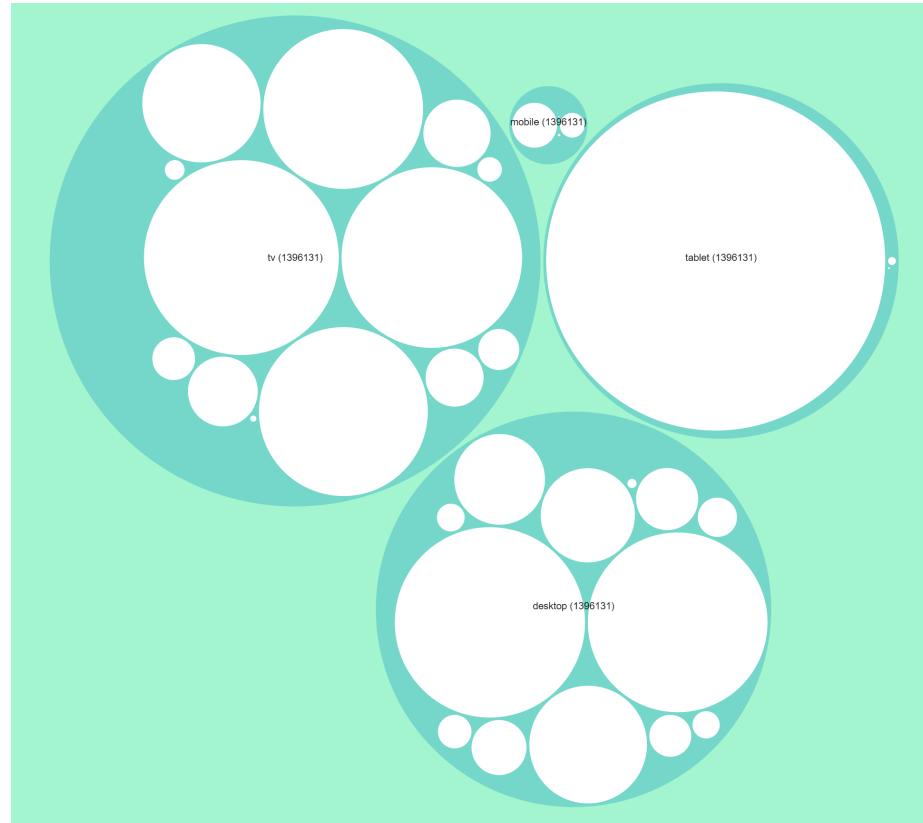
### 5.2.2 Visualização

O resultado da consulta é representado visualmente por meio de um gráfico de agrupamento em círculos, como exibido na Figura 5.5. Este tipo de representação é muito útil para casos de semelhanças categóricas, como é o caso dos tipos de reprodutores e tipos de dispositivos, pois ajuda a facilmente identificar os grupos de usuários que compartilham das mesmas características. Desta forma, torna-se simples para o usuário final identificar a quantidade de grupos diferentes, bem como a magnitude de cada grupo, pois um maior tamanho de grupo decorre de uma maior quantidade de indivíduos com aquelas características.

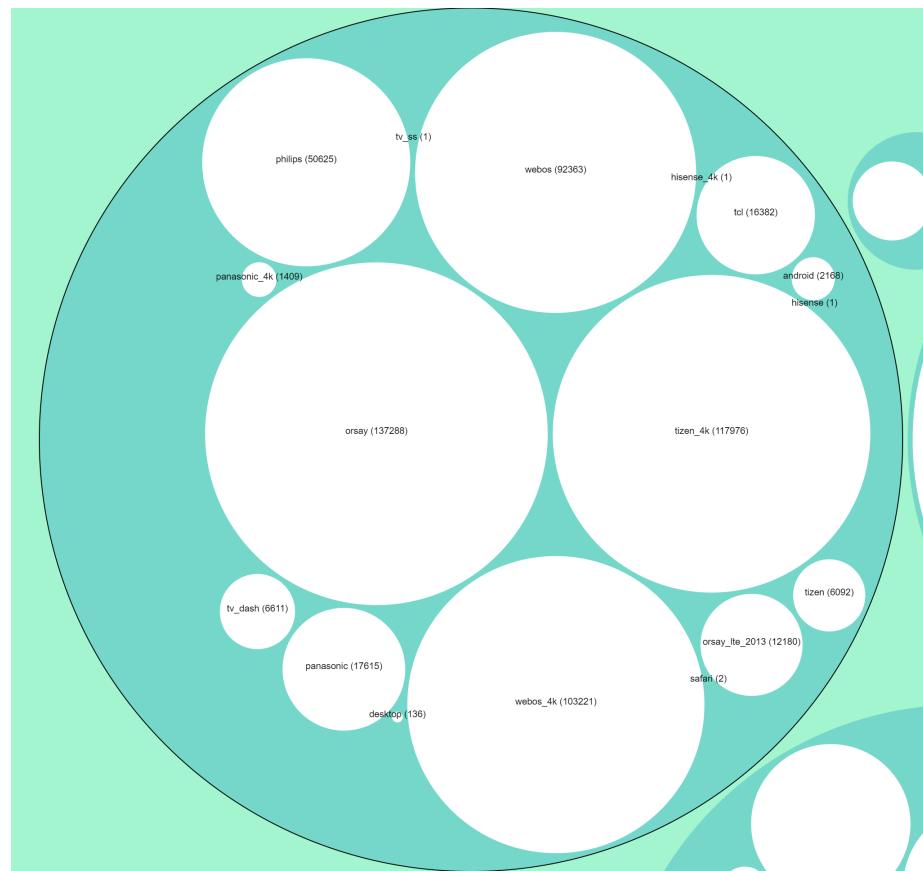
A Figura 5.3a mostra a visualização padrão do gráfico sobre a consulta em questão. Esta visualização permite interagir com os grupos de dispositivos, representados pelos agrupamentos de círculos mais externos. Ao clicar sobre um destes, a ferramenta realiza um *zoom* sobre o mesmo e mostra os reprodutores e a quantidade de usuários únicos do respectivo dispositivo, conforme pode ser observado na Figura 5.3b. Ao clicar fora do grupo onde foi realizada a aproximação, o gráfico desfará o *zoom* e voltará à visão original com todos os grupos.

Pode-se perceber uma diferente distribuição de reprodutores dependendo do dispositivo analisado na Figura 5.3b. Enquanto nos *tablets* há um total monopólio de um reproduutor só e no caso de *mobile* aprecia-se um duopólio, nos grupos de *tv* e *desktop* os acessos são muito mais distribuídos. No caso das televisões, há uma grande concentração em 5 grupos: *Philips*,

Figura 5.3: Visualização da consulta dos reprodutores mais comuns por dispositivo por meio de uma visualização de agrupamento em círculos.



(a) Visão geral do gráfico.



(b) Zoom no conjunto de um dos dispositivos.

*Webos*, *Tizen\_4k*, *Orsay* e *Webos\_4k*. Esta informação ajuda a definir quais os principais reprodutores e marcas que precisam de suporte no contexto de desenvolvimento para TV's. No entanto, apesar de ser um grupo menor, TCL ainda possui mais de 16 usuários, o que precisa ser analisado na hora de tomar decisões sobre abandonar totalmente o suporte ou manter um mínimo de dedicação. Afinal, é um valor absoluto relativamente elevado, podendo gerar críticas ao serviço e prejudicando sua imagem caso parassem de poder assistir o conteúdo em caso de abandono de suporte prestado a estes sistemas.

### **5.3 Distribuição da quantidade de títulos diferentes assistidos pelos usuários**

Nesta Seção, analisamos a distribuição da quantidade de títulos diferentes assistidos por pelos usuários fazendo uso de histogramas em barras. Pelo fato de existirem diferentes tipos de conteúdo disponíveis, como filmes, programas de músicas ou séries, a forma como os usuários distribuem seu consumo pode variar de acordo com o tipo do título e suas características. Por exemplo, um usuário que tem preferência por séries tem uma tendência maior a assistir menos títulos diferentes, uma vez que se concentrará nos vários episódios de uma mesma série. Tal tendência de consumo dos usuários é mais uma variável que pode ser analisada para auxiliar na tomada de decisões para recomendações, seja sugerindo novos títulos semelhantes aos que o usuário já assistiu, recomendando episódios das séries já visualizadas ou até mostrando algo completamente diferente para incentivar uma maior exploração.

#### **5.3.1 Consulta SQL**

A consulta SQL para gerar essa visualização com base nos recursos oferecidos nativamente pela linguagem SQL é mais simples do que as demais consultas deste Capítulo. Como exibido na Figura 5.4, fazemos uso de uma subconsulta (nas linhas 3 a 5) para contar quantos títulos diferentes cada um

Figura 5.4: Consulta SQL da distribuição da quantidade de títulos diferentes assistidos pelos usuários, fazendo uso de agrupamentos (*GROUP BY*) na consulta principal assim como na subconsulta

```

1 SELECT viewed_titles, count(*)
2 FROM (
3     SELECT user_id, COUNT( distinct title_id) AS viewed_titles
4     FROM VIEWS
5     GROUP BY user_id
6 ) as foo
7 GROUP BY viewed_titles

```

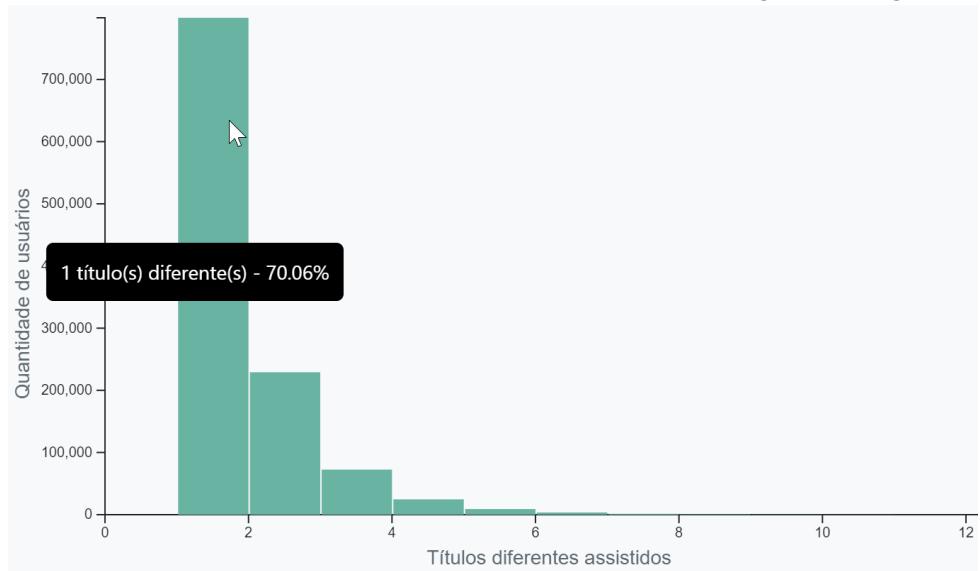
dos usuários assistiu, aplicando uma cláusula *GROUP BY* no *id* dos usuários. Posteriormente, apenas agrupamos o resultado da subconsulta pela quantidade de títulos visualizadas, gerando um histograma da frequência de visualização de diferentes títulos dos usuários.

### 5.3.2 Visualização

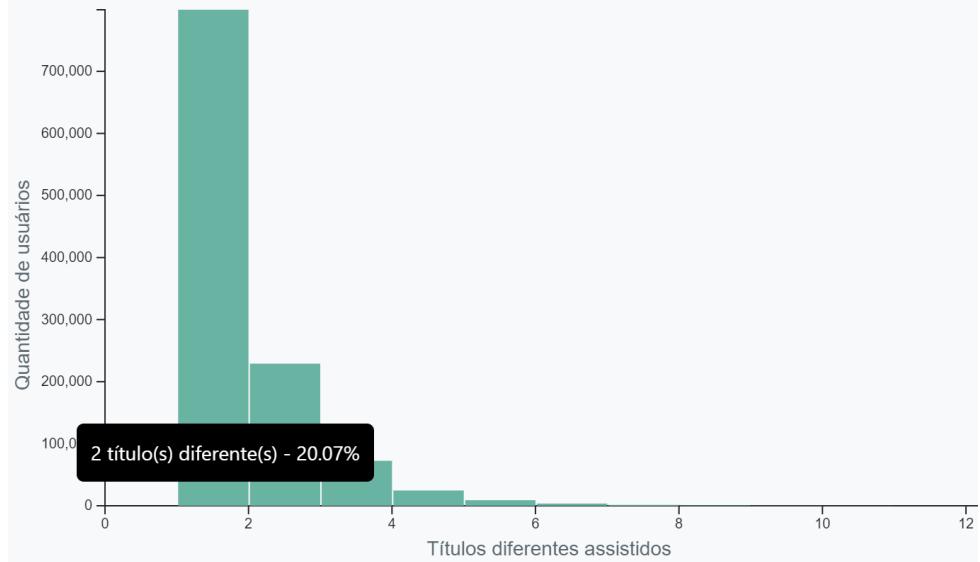
A visualização escolhida para esta consulta foi um histograma em barras, um tipo de visualização semelhante ao gráfico de barras verticais que tem por objetivo analisar a distribuição de frequências. Para o caso do dataset aplicado neste trabalho, cada barra vertical representa no eixo Y a quantidade  $N$  de usuários que visualizaram  $x$  títulos diferentes, representados no eixo X. Ao se pensar nesse gráfico, a expectativa era a de que teríamos uma distribuição normal bimodal com foco direito maior, definido por  $f(x) = pg_1(x) + (1 - p)g_2(x)$ , com  $\mu_1 > \mu_2$  (onde  $\mu_1$  é a média de  $g_1(x)$ , e  $\mu_2$  é a média de  $g_2(x)$ ) e  $p > 0.5$  (onde  $p$  é o coeficiente de mistura). Isto é, teríamos uma grande quantidade de usuários no intervalo de poucos títulos diferentes, por exemplo [1, 4], que seriam as pessoas que utilizam a plataforma para visualizar séries (alta frequência, pouca quantidade de títulos diferentes), e depois de um intervalo teríamos uma nova elevação no gráfico (porém menor que a primeira) das pessoas que assistem filmes (baixa frequência, muitos títulos diferentes). Essa hipótese, porém, não se confirmou.

Analisando a visualização gerada pela nossa ferramenta, podemos observar na Figura 5.5a que a maioria dos usuários (mais de 70%) visualizaram apenas um título. Se somarmos aqueles usuários que assistiram

Figura 5.5: Visualização distribuição da quantidade de títulos diferentes assistidos pelos usuários a través de um histograma em barras assim como a caixa de texto mostrada passar o *mouse* por cima de algum dos grupos.



(a) Visão do histograma com o *mouse* em cima da barra de maior ocorrência.



(b) Visão do histograma com o *mouse* em cima da barra com a segunda maior ocorrência.

dois títulos diferentes, como mostra a Figura 5.5b, o total de usuários chega a 90%. Tal observação pode indicar que a maior parte do público que utilizou o Globoplay no período analisado o fez de forma experimental. Este pode ter sido influenciado por campanhas de vendas oferecendo acesso gratuito ao serviço por um número determinado de dias ou mesmo pela disponibilização gratuita de algum conteúdo muito popular. Esse tipo de abordagem tem como objetivo atrair novos *leads* (futuros possíveis clientes), esperando que uma parte desses usuários experimentais assinem o serviço Globoplay após o fim da campanha de captação.

## 5.4 Correlação entre gêneros de conteúdos

Nesta Seção, explora-se as relações que podem existir entre os distintos tipos de gêneros de filmes, séries e programas musicais. Em um contexto de produção massiva de conteúdo audiovisual e alta competitividade entre as diversas plataformas e serviços de entretenimento, a capacidade de manter o usuário ativo por longos períodos de tempo é fundamental. Para se constituir um sistema de recomendação de conteúdos eficiente, uma análise sobre as preferências e gostos pessoais dos usuários é um dos principais pontos a serem investigados. Uma prática muito comum é agrupar títulos que pertencem ao mesmo gênero e utilizá-los como sugestões por considerar a probabilidade de possuírem um maior grau de similaridade. No entanto, muitas vezes é possível também correlacionar gêneros distintos e identificar como eles se influenciam entre si, ou seja, se um usuário que assiste a um determinado gênero também assiste a outro. Identificar estas propriedades ou padrões de comportamentos e gostos pode ser o fator chave que manterá os usuários satisfeitos e consumindo mais o conteúdo oferecido.

### 5.4.1 Consulta SQL

A Figura 5.6 mostra a consulta SQL realizada para visualizar as possíveis correlações entre gêneros de conteúdo. Novamente, fazemos uso de uma função SQL, na qual o parâmetro \$1 indicar sobre qual gênero

Figura 5.6: Consulta SQL da correlação entre diferentes gêneros de conteúdo.

```

1 CREATE OR REPLACE FUNCTION count_except_category (int)
2   RETURNS TABLE (
3     unique_viewers_ids bigint,
4     name varchar
5   )
6   AS $$ 
7   SELECT
8     count(DISTINCT user_id) AS unique_viewers_ids,
9     genres.name
10  FROM
11    Views
12    JOIN GenreTitles ON Views.title_id = GenreTitles.title_id
13 WHERE
14   genre_id != $1
15   AND user_id IN ( SELECT DISTINCT
16     user_id
17     FROM
18       Views
19       JOIN GenreTitles ON Views.title_id = GenreTitles.title_id
20     WHERE
21       genre_id = $1
22     GROUP BY
23       genre_id,
24       user_id)
25   GROUP BY
26     genres.id
27   ORDER BY
28     unique_viewers_ids DESC
29 $$;
30 LANGUAGE SQL;
31
32 SELECT
33   counter.unique_viewers_ids AS unique_viewers,
34   g.name AS viewed,
35   counter.name AS also_viewed
36 FROM Genres g, count_except_category(g.id) AS counter;

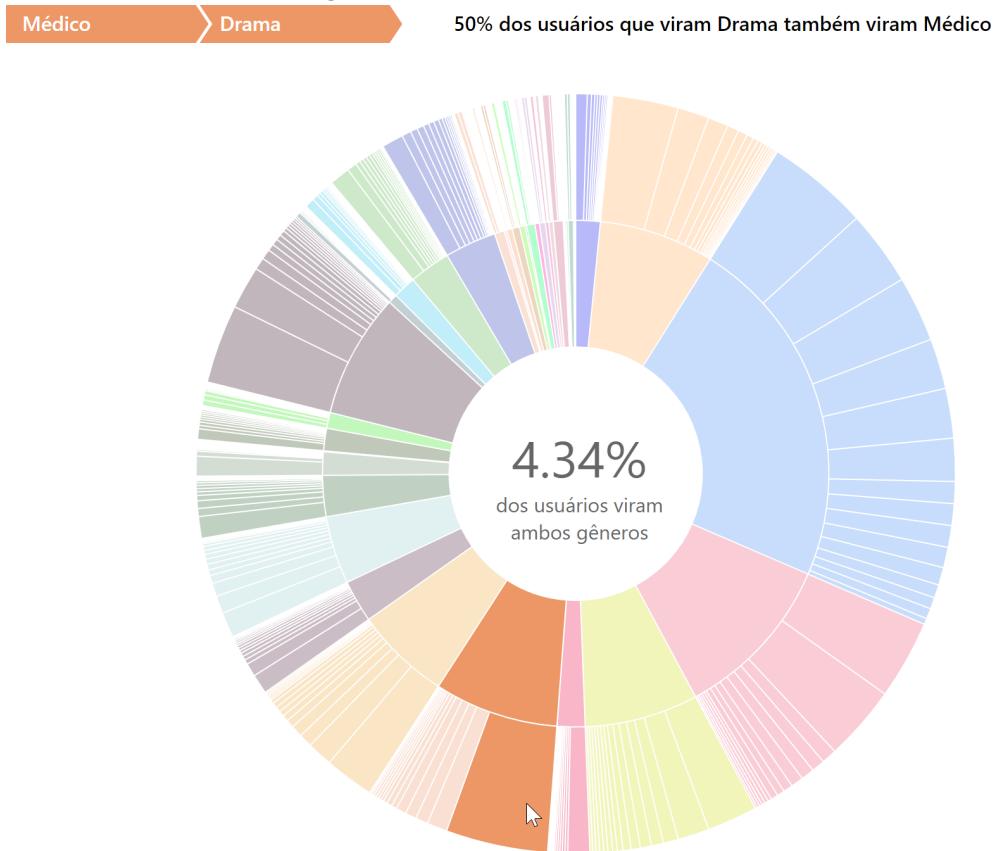
```

estamos querendo visualizar as possíveis correlações. Para esta consulta, um recurso SQL adicional é empregado: uma *subquery* ou subconsulta. Para cada gênero  $G$ , a função SQL primeiro realiza um *join* das tabelas *Views*, *GenreTitles* e *Genres*, eliminando as visualizações do gênero  $\$1$  por meio do *WHERE* da linha 15. Isso resulta em todas as visualizações dos demais gêneros, uma vez que estamos agrupando por  $G.id$  na linha 30.

A subconsulta filtra os usuários que assistiram  $\$1$  e encontra todos aqueles usuários únicos que também assistiram a cada gênero  $G$ . O retorno da *subquery* é dado pelo nome dos demais gêneros assistidos e a quantidade de usuários únicos que assistiram a cada um. Por fim, a consulta SQL chama a função para cada cada um dos gêneros e seu retorno é composto por tuplas  $\{gênero comum, gênero também assistido, total de usuários únicos\}$ .

#### 5.4.2 Visualização

Figura 5.7: Gráfico em setores com dois níveis de profundidade para analisar a correlação entre dois gêneros e conteúdos distintos.



A representação da Figura 5.7 é um gráfico de pizza, ou gráfico de setores, com dois níveis de profundidade. O primeiro nível, ou o nível interior, é separado em diferentes gêneros raiz. No segundo nível, temos subdivisões que representam os gêneros assistidos pelo mesmo conjunto de usuários que assistiram conteúdos do respectivo "gênero raiz". Desta forma, podemos visualizar quantas pessoas assistiram um gênero A e também um gênero B (formalmente,  $A \wedge B$ ). Conforme o cursor é movimentado por cima do gráfico, a sequência atual é destacada. Também é informado, textualmente, quantos usuários viram ambos os gêneros dentre todos os usuários que assistiram algum conteúdo, além da porcentagem dentre os conjuntos destacados.

Pelo gráfico escolhido, pode-se perceber facilmente como alguns setores exteriores representam, proporcionalmente, uma grande parte do respectivo setor interior. Isso significa que, dentre os que assistiram o gênero do círculo interior, uma grande parte assistiu o gênero da parte exterior. Este é o caso do gênero *Médico*, no qual 50% dos usuários que assistiram algum conteúdo deste gênero assistiram também algum título dramático. Este tipo de informação é muito valioso para aprimorar a avaliação da recomendação de conteúdos para os usuários, pois explora relações que à priori não seria possível inferir.

## 5.5 Influência dos reprodutores nos gêneros dos conteúdos

Outro aspecto que pode auxiliar no aprimoramento de um sistema de recomendação de conteúdo é a possível relação existente entre o reproduutor sendo utilizado e o tipo de conteúdo assistido. Nesta Seção, analisamos como os gêneros de conteúdos são influenciados por distintos reprodutores a fim de explorar possíveis padrões de comportamentos atrelados a essa relação. Por exemplo, identificar que um gênero específico é frequentemente acessado em um certo tipo de reproduutor pode auxiliar na recomendação desse gênero aos demais usuários do mesmo ou na experimentação de possíveis afinidades. Esta influência será avaliada a partir dos valores absolutos de dependência entre gêneros e reprodutores, dos quais a relação é diretamente proporcional ao número de visualizações de um determinado gênero por meio de um reproduutor.

Figura 5.8: Consulta SQL da influência dos reprodutores nos gêneros de conteúdo.

```

1 CREATE OR REPLACE FUNCTION count_player_influence_on_genres (int, int[])
2 RETURNS TABLE (
3     total_views numeric,
4     name varchar
5 )
6 AS $$$
7 SELECT
8     SUM(subquery.total_views) AS total_views,
9     CASE WHEN subquery.position_top_10 = 0 THEN 'Other' ELSE STRING_AGG(subquery.name, '') END AS name
10    FROM (
11        SELECT
12            count(*) AS total_views,
13            genres.name,
14            CASE WHEN genres.id = ANY($2) THEN genres.id ELSE 0 END AS position_top_10
15        FROM
16            Views
17            JOIN Players ON Views.player_id = Players.id
18            JOIN GenreTitles ON Views.title_id = GenreTitles.title_id
19            JOIN Genres ON GenreTitles.genre_id = Genres.id
20        WHERE
21            player_id = $1
22        GROUP BY
23            genres.name,
24            genres.id
25    ) AS subquery
26    GROUP BY
27        subquery.position_top_10
28    ORDER BY total_views DESC
29 $$$
30 LANGUAGE SQL;
31
32 WITH top_10_categories AS (
33     SELECT genres.id
34     FROM Views
35         JOIN GenreTitles ON Views.title_id = GenreTitles.title_id
36         JOIN Genres ON GenreTitles.genre_id = Genres.id
37     GROUP BY genres.id, genres.name
38     ORDER BY
39         count(*) DESC
40     LIMIT 10
41 )
42 SELECT
43     counter.total_views AS total_views,
44     counter.name AS genre,
45     p.name as on_player
46    FROM
47        Players p,
48        count_player_influence_on_genres(p.id, ARRAY(select id from top_10_categories)) AS counter
49    ORDER BY on_player;

```

### 5.5.1 Consulta SQL

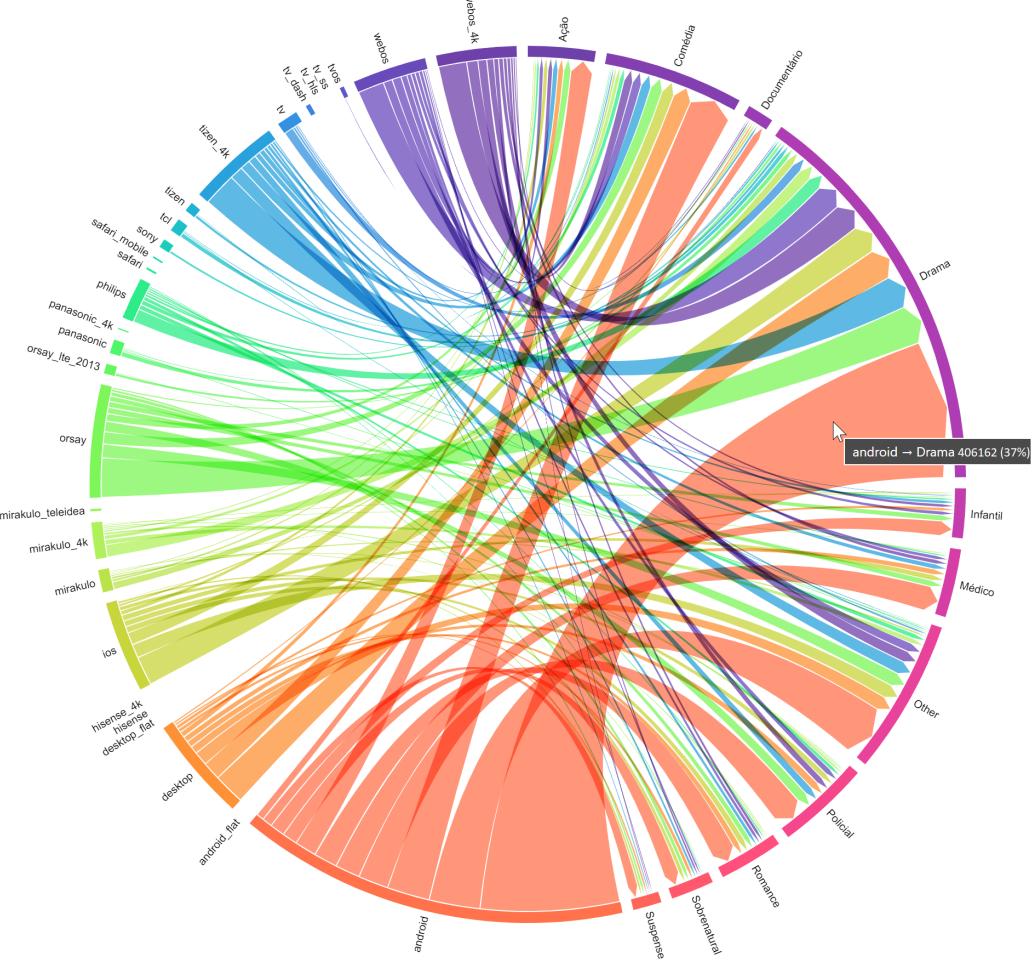
A Figura 5.8 representa o código SQL utilizado para realizar essa consulta. A *query* principal, localizada na linha 42, é responsável por chamar uma *function* SQL previamente definida no banco de dados para cada um dos  $P$  *Players*. A definição de tal *function* SQL é a primeira otimização implementada para melhorar o desempenho dessa consulta. O papel da *function* é realizar o pré cálculo das dez categorias mais assistidas por meio da cláusula *WITH*. Sendo assim, evita-se recalcular os dez gêneros mais assistidos a cada vez que a função é chamada para cada um dos  $N$  reprodutores únicos. A fim de obter-se uma melhor visualização, definimos

que seriam analisadas apenas as dez categorias de conteúdo mais assistidas.

Na chamada da *function* é passado o *id* do *player* que estamos buscando, como parâmetro  $\$1$ , e a lista pré-computada das categorias que queremos analisar, como parâmetro  $\$2$ . Calcula-se então quantas visualizações foram realizadas através do reproduutor  $P$  para cada um dos gêneros em  $\$2$ . O retorno dessa função é dado pelas dez categorias principais e uma categoria adicional na qual são agrupadas todas as demais categorias, denominada "*Other*". Justamente pelo fato de querermos essa categoria extra que nossa *query* se torna um pouco mais complicada. Como queremos poder agrupar as visualizações pelo *id* de cada gênero, não temos como fazer isso sem uma *subquery*. Se tentássemos comparar os *ids* dos *Genres G* de modo a saber se eles pertenceriam a essa categoria "*Other*", seríamos obrigados a ter  $G.id$  na cláusula *GROUP BY*. Isto anularia o fato de estarmos tentando juntá-los, gerando, na verdade, várias linhas com o nome "*Other*" ao invés de agruparmos todas elas em uma só linha. Dessa forma, por isso temos uma *subquery* responsável por gerar um valor "único" para todas as categorias que irão pertencer a "*Other*", o que é descrito na linha 14 da Figura 5.8. Caso esse gênero  $G$  pertença ao Top 10, nós estamos utilizando seu identificador único *id*, e para todos os outros, utilizamos um *id* de valor zero, que não existe no banco de dados e, portanto, será utilizado para agrupar todos eles pelo *GROUP BY* da linha 27.

Note, portanto, que na linha 9 estamos checando esse valor zero para atestar se o nome da categoria deveria ser "*Other*" ou o seu nome original. Também é importante notar que somos obrigados a usar *STRING\_AGG* para agregar todos os nomes das categorias das quais realmente estamos usando o nome, pois poderíamos potencialmente ter várias categorias caindo naquela condição. No entanto, uma vez que sabemos que pela forma como fazemos nossa cláusula interna isso nunca irá acontecer, utilizamos *STRING\_AGG* sem um separador simplesmente pelo PostgreSQL obrigar-nos a usá-lo. Adicionalmente, vemos na linha 12 que a *query* interna é responsável por contar quantas *Views V* temos para cada *Player P*, enquanto que na linha 8 estamos realizando a soma de todos os "*Other*", já na *query* principal da *function*.

Figura 5.9: Consulta SQL da influência dos reprodutores nos gêneros de conteúdo por meio de um gráfico de cordas.



### 5.5.2 Visualização

A abordagem escolhida para representar esta consulta foi o gráfico de cordas, ou *chord diagram*, observado na Figura 5.9. Este tipo de visualização costuma ser utilizado para representar relações ou dependências entre objetos de mesma ordem, indicando a força ou a quantidade de influência nas relações pelo tamanho da largura das cordas. Separamos os "nodos" do diagrama em dois grupos de elementos que representam os reprodutores, na metade da esquerda da Figura 5.9, e os conteúdos, na metade da direita. Para melhor observar esta separação, desenhamos uma seta na ponta das cordas que incidem sobre os gêneros de conteúdo. Com isso, é possível representar quantas visualizações cada gênero de conteúdo teve a partir de cada um dos reprodutores existentes.

De maneira formal, para a demonstração no nosso sistema foi montado

um grafo bipartido  $G = (E, V)$ , onde  $E$  são as arestas em  $G$ , e  $V$  seus respectivos nodos. Como o grafo é bipartido, temos que os vértices podem ser separados em 2 componentes  $V_1$  e  $V_2$ , onde  $V_1 \wedge V_2 = \emptyset$ . Uma aresta  $e$ , portanto, é definida por  $e = (v_{V_1}, v_{V_2})$ , onde  $v_{V_1} \in V_1, v_{V_2} \in V_2$ . Os nodos  $v_{V_1}$  representam os dispositivos, enquanto os elementos  $v_{V_2}$  são os dez gêneros de conteúdo mais assistidos. As arestas  $(v_{V_1}, v_{V_2})$  representam, portanto, a influência, em números absolutos, que o dispositivo teve no gênero. Quanto mais larga a aresta, maior a quantidade de visualizações do determinado gênero a partir do reproduutor em questão.

Como pode ser observado na Figura 5.9, ao passar o cursor por cima de alguma das conexões, ou cordas, aparece uma caixa de texto indicando quantas visualizações daquele gênero ocorreram a partir do reproduutor em questão. Esses dados são demonstrados tanto em valor absoluto quanto em porcentagem em relação ao total de visualizações daquele gênero. No caso do gênero Drama, mais de 1 terço dos acessos (37%) ocorreram a partir de algum reproduutor android. Esta informação pode ajudar definir que, por exemplo, no lançamento de um novo conteúdo dramático seja promovida mais sugestões desse título a usuários que reproduzem conteúdo do Globoplay através de reprodutores android.

## 5.6 Distribuição dos dispositivos dentre os 10 títulos mais vistos

O objetivo desta Seção é explorar a distribuição dos dispositivos dentre os 10 títulos mais vistos. Uma vez que a facilidade de acesso à informação e o compartilhamento da mesma vêm aumentando a um ritmo muito elevado nos últimos tempos, é muito comum um título subitamente tornar-se muito popular e receber um acesso massivo por parte dos usuários. Entender como estão sendo acessados os títulos mais famosos do momento permite identificar possíveis tendências e colabora com a elaboração de novas sugestões. Por exemplo, saber que os títulos mais acessados são majoritariamente visualizados a partir de um dado dispositivo  $X$  ou identificar onde novos títulos se encaixam nessa distribuição pode ajudar a prever se um novo título será, ou não, de agrado ao público do dispositivo  $X$ . Com tal informação, tanto a distribuidora quanto a produtora dos conteúdos

Figura 5.10: Consulta SQL da distribuição dos dispositivos dentre os 10 títulos mais vistos.

```

1 CREATE OR REPLACE FUNCTION count_devices_per_title (int, int [])
2   RETURNS TABLE (
3     total_views bigint,
4     title int
5   ) AS $$ 
6   SELECT count(*) AS total_views,
7     title_id
8   FROM Views
9     JOIN Devices ON device_id = Devices.id
10  WHERE device_id = $1
11    AND title_id = ANY($2)
12  GROUP BY title_id
13  ORDER BY total_views DESC
14 $$ LANGUAGE SQL;
15
16 WITH top_10_titles AS (
17   SELECT titles.id
18   FROM Views
19     JOIN Titles ON Views.title_id = Titles.id
20  GROUP BY titles.id
21  ORDER BY count(*) DESC
22  LIMIT 10
23 )
24 SELECT
25   counter.total_views AS total_views,
26   counter.title AS title,
27   d.name as on_device
28 FROM
29   Devices d,
30   count_devices_per_title(d.id,ARRAY(select id from top_10_titles)) AS counter;

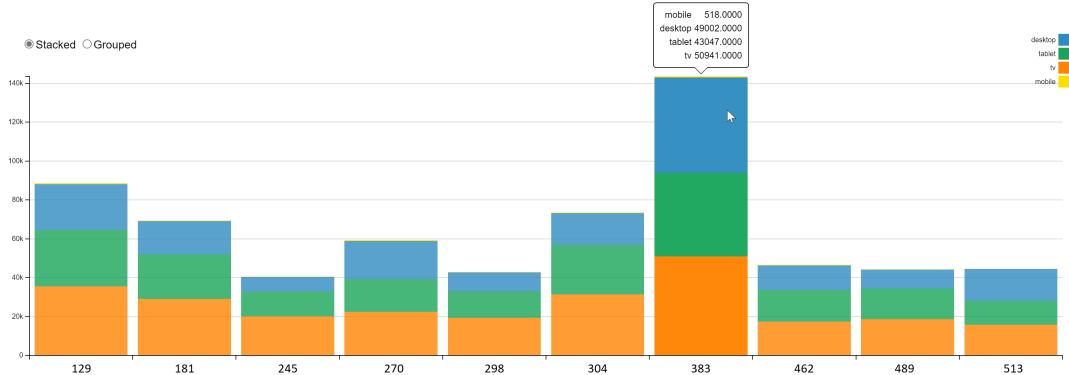
```

têm uma maior segurança no desenvolvimento de novos títulos, pois conhecerão melhor as características do seu público alvo e poderão criar um conteúdo que melhor se adapte a ele.

### 5.6.1 Consulta SQL

A Figura 5.10 representa o código SQL utilizado para realizar a consulta desejada. A *query* principal possui uma *subquery* na linha 16 responsável por buscar os identificadores dos 10 títulos mais vistos no banco de dados. Ela é extremamente simples, apenas realizando um *join* de todos os títulos com as visualizações existentes, ordenando os títulos pela quantidade de visualizações (em ordem decrescente) e então retornando os 10 primeiros dessa lista. Tal *subquery* é utilizada quando chamamos a função principal na linha 30 (definida na linha 1), de modo a restringir a contagem por dispositivo somente para os 10 títulos principais. Essa função principal

Figura 5.11: Visão padrão do gráfico de barras empilhadas da distribuição dos dispositivos dentre os 10 títulos mais vistos.



retorna a quantidade de visualizações de cada um dos 10 títulos para um determinado dispositivo, sendo executada uma vez para cada um dos dispositivos e, então, agrupada pelo *SELECT* da linha 24 em uma tupla `{total_views, title, device}`.

### 5.6.2 Visualização

Para a visualização desta consulta foi utilizado um gráfico de barras, ou *bar plot*, formado por barras retangulares e frequentemente utilizado para ilustrar comparações entre grandezas numéricas. Algumas variações deste gráfico são utilizadas quando uma mesma barra, que representa um valor sobre uma característica, pode ser dividida em algumas subcategorias. Neste caso, costuma-se representar a barra dividida em segmentos, como no caso do gráfico de barras empilhado ou no gráfico de barras agrupadas. Na nossa aplicação, optamos por dar uma maior liberdade ao usuário na visualização desta consulta, oferecendo 2 versões de gráficos de barra (empilhado e agrupado), além de permitir uma personalização dos dados que serão representados. No contexto da consulta em questão, a altura de cada um dos segmentos indica a quantidade de reproduções em um dado dispositivo.

Na visão padrão, como mostra a Figura 5.11, cada um dos filmes é representado no eixo X por uma barra segmentada. Cada um dos segmentos representa um tipo de dispositivo. O eixo Y indica o tamanho de cada segmento, que é definido pela quantidade de visualizações através daquele elemento. Quando o cursor é posicionado cima da barra, uma caixa de texto é

Figura 5.12: Visão do gráfico de barras empilhadas filtrando por dispositivos.

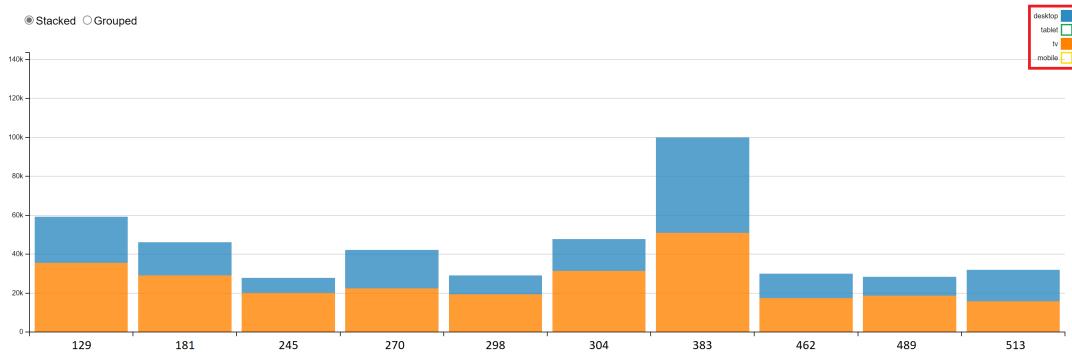
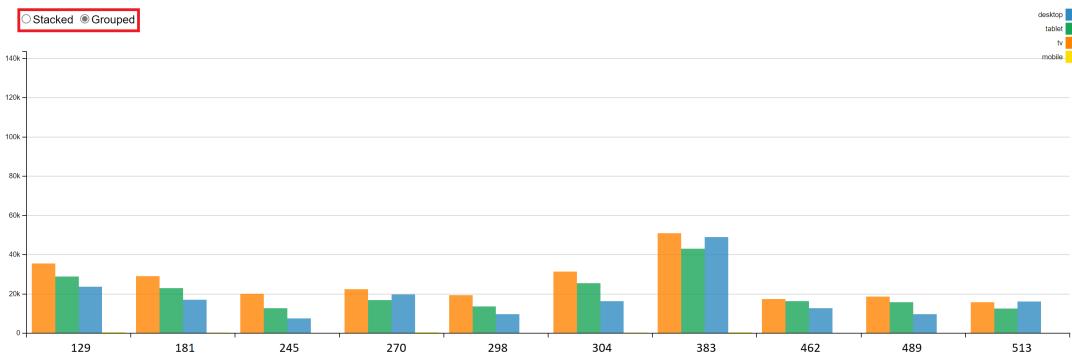


Figura 5.13: Visão do gráfico de barras agrupadas da distribuição dos dispositivos dentre os 10 títulos mais vistos.



exibida indicando o valor absoluto de visualizações daquele título para cada um dos tipos de dispositivos.

Uma das possibilidades de customização que foi oferecida foi a escolha de quais os dispositivos a serem mostrados nas barras. Como pode ser observado na Figura 5.12, no canto superior direito é destacada em vermelho a legenda com caixas de seleção para cada um dos dispositivos. Ao interagir clicando nelas, os segmentos do respectivo dispositivo deixam de ser ilustrados no gráfico. Com isso, o usuário tem a liberdade de escolher entre quais dispositivos fazer as análises e comparações que melhor lhe convir.

A Figura 5.13 ilustra outro recurso oferecido: a possibilidade de escolher entre a visualização em barras empilhadas ou agrupadas. Barras agrupadas separam cada um dos segmentos que formam a barra única em barras menores dispostas lado a lado, o que facilita comparações entre suas alturas uma vez que todas começam do mesmo ponto do eixo Y. Como podemos observar na Figura 5.11, o título 383 possui os segmentos laranja e azul de alturas bem próximas, o que dificulta sua diferenciação. Nestes

casos, a versão de barras agrupadas mostrada na Figura 5.13 auxilia a melhor visualizar que o segmento laranja (*tv*) teve uma quantidade levemente maior de visualizações do que o segmento azul (*desktop*).

Podemos observar que *tv* é o dispositivo mais escolhido para todos os 10 títulos mais visualizados, enquanto *mobile* possui uma quantidade muito baixa de reproduções, representando em média apenas 0.3% das visualizações. Tal resultado vai de contramão ao que seria esperado, dada a enorme pervasividade de dispositivos móveis atualmente (KUROSE; ROSS, 2012). Este resultado pode ser útil para ajudar a traçar um plano de negócios para a empresa, já que com essas estatísticas ela é capaz de saber qual é o seu maior público e através de que tipos de dispositivo está sendo consumido o seu conteúdo oferecido. Estas informações podem ser aplicadas na produção de novos conteúdos que sejam do interesse da maior parcela do seu público ou que melhor desempenhem nos seus dispositivos (como filmar as maiores produções em qualidade de vídeo 4k para aproveitar este recurso dos dispositivos *tv*).

## 6 CONCLUSÃO

Com a crescente quantidade de dados gerados e utilizados pelas incontáveis aplicações, plataformas e serviços que fazem parte do nosso dia a dia, ferramentas de visualização de dados vêm ganhando um espaço tão grande quanto. Este trabalho descreveu o desenvolvimento do IntroSpectView, uma ferramenta de análise visual para sistemas de recomendação do Globoplay. Apesar de sistemas de recomendação serem eficientes em detectar padrões dos usuários, como suas preferências e comportamentos, muitos problemas dos quais não se tem pleno conhecimento podem não ser adequadamente cobertos por tal abordagem. Com isso, permitir que os gerenciadores da plataforma possam visualizar e trabalhar sobre as informações da mesma pode auxiliar na detecção de padrões úteis que passam despercebidos por sistemas inteligentes. Alguns padrões e relações escondidas podem ser fundamentais na identificação de possíveis otimizações e melhorias necessárias ao produto Globoplay. Dessa forma, a aplicação de sistemas de recomendação e ferramentas de visualização lado a lado pode levar o gerenciamento desses sistemas a um novo patamar.

Diversos exemplos da aplicabilidade da ferramenta IntroSpectView foram apresentados ao longo do trabalho. Vimos que, ao analisar visualmente os dados, informações como a necessidade de suporte ou descontinuação para um dado tipo de dispositivo podem ser observadas, bem como relações existentes entre o tipo de conteúdo acessado e o dispositivo pelo qual o acesso ocorreu. Mostramos como podemos identificar relações entre diferentes conteúdos consumidos e como isso pode ser utilizado para fomentar o engajamento do usuário com a plataforma, dentre outros exemplos. É importante ressaltar novamente que o conjunto de consultas usado para demonstração do funcionamento da ferramenta representa exemplos de consultas reais com aplicabilidade real. Com base em tais exemplos, realizamos a análise do código das consultas, dos contextos e possíveis conclusões que poderiam ser observadas a partir das visualizações geradas.

Além disso, diversas formas de visualização das informações foram

abordadas e disponibilizadas para o público, e uma variedade delas foi usada como exemplo das consultadas mencionadas. Mostramos como obter informações por meio de gráficos de barra ou histogramas, gráficos de cordas, gráficos em setores com dois níveis de profundidade e gráficos de agrupamento em círculos. Para cada uma das visualizações discorremos sobre suas características e as respectivas relações que poderiam ser extraídas. Também analisamos as interações disponíveis, como a possibilidade de realizar *zoom* sobre um determinado agrupamento, assim como informações que se tornavam visíveis conforme o cursor se movia por cima da visualização.

A versão atual da ferramenta IntroSpectView possui algumas características engessadas, um ponto que deve ser abordado em versões futuras. Muitos parâmetros das consultas estão codificados de maneira fixa na ferramenta, o que impossibilita que o usuário personalize suas pesquisas e visualizações. Um exemplo disso é a limitação para dez gêneros mais vistos na correlação dos gêneros com os dispositivos. Dessa forma, diversas frentes de otimização da ferramenta devem ser abordadas como trabalhos futuros, de modo a permitir que as visualizações se adaptem às necessidades dos usuários e, ao mesmo tempo, sejam genéricas o suficiente para abranger a maior quantidade de consultas possível. Cada usuário pode ter seus diferentes objetivos, e mesmo que as regras e filtros da pesquisa sejam muito específicas, espera-se que a ferramente forneça a ele os recursos para aplicar tais filtros e testar suas hipóteses sobre o conjunto de dados. As novas versões da ferramenta devem atender tanto às especificidades quanto às variedades das consultas.

## REFERÊNCIAS

- AGIRA, T. **The Timeline Of 25 Successful Years Of JavaScript**. 2021. Disponível na Internet: <<https://www.agiratech.com/the-timeline-of-25-successful-years-of-javascript>>.
- ANACONDA, I. **Anaconda Software Distribution**. Anaconda Inc., 2012. Disponível na Internet: <<https://docs.anaconda.com/>>.
- BRITO, G.; VALENTE, M. T. REST vs graphql: A controlled experiment. **CoRR**, abs/2003.04761, 2020. Disponível na Internet: <<https://arxiv.org/abs/2003.04761>>.
- BUELTHOFF, F.; MALESHKOVA, M. Restful or restless-current state of today's top web apis. Em: . [S.l.: s.n.], 2014. v. 1165, p. 42–51.
- CHEN, L.-S.; HSU, F.-H.; CHEN, M.-C.; HSU, Y.-C. Developing recommender systems with the consideration of product profitability for sellers. **Information Sciences**, v. 178, n. 4, p. 1032–1048, 2008. ISSN 0020-0255. Disponível na Internet: <<https://www.sciencedirect.com/science/article/pii/S0020025507004641>>.
- COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T.; BLAIR, G. **Distributed Systems: Concepts and Design**. 5th. ed. USA: Addison-Wesley Publishing Company, 2011. ISBN 0132143011.
- D., H. J. Matplotlib: A 2d graphics environment. **Computing in Science & Engineering**, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007.
- DAHL, R. **NodeJS**. 2009. (Acessado em 04/11/2021). Disponível na Internet: <<https://nodejs.org/en/>>.
- DIGREGORIO, F.; VARRAZZO, D. **Psycopg2**. 2010. (Acessado em 23/10/2021). Disponível na Internet: <<https://www.psycopg.org/>>.
- DYE, M.; EKANADHAM, C.; SALUJA, A.; RASTOGI, A. **Supporting content decision makers with machine learning**. 2020. (Accessed on 13/11/2021). Disponível na Internet: <<https://netflixtechblog.com/supporting-content-decision-makers-with-machine-learning-995b7b76006f>>.
- EICH, B. **JavaScript**. 1995. (Acessado em 04/11/2021). Disponível na Internet: <<https://www.javascript.com/>>.
- FACEBOOK. **React**. 2013. (Acessado em 04/11/2021). Disponível na Internet: <<https://reactjs.org/>>.
- FACEBOOK. **GraphQL Specification**. 2018. (Accessed on 16/11/2021). Disponível na Internet: <<https://spec.graphql.org/June2018/>>.
- FIELDING, R. T.; GETTYS, J.; MOGUL, J. C.; NIELSEN, H. F.; MASINTER, L.; LEACH, P. J.; BERNERS-LEE, T. **Hypertext Transfer Protocol – HTTP/1.1**. [S.l.], 1999. <<http://www.rfc-editor.org/rfc/rfc2616.txt>>. Disponível na Internet: <<http://www.rfc-editor.org/rfc/rfc2616.txt>>.

- FOUNDATION, P. S. **Python Package Index - PyPI**. 2003. (Accessed on 04/11/2021). Disponível na Internet: <<https://pypi.org/>>.
- FOUNDATION, P. S. **Python Package Index - PyPI**. 2011. (Acessado em 04/11/2021). Disponível na Internet: <<https://pypi.org/>>.
- GOOGLE. **Chromium**. 2008. (Acessado em 07/11/2021). Disponível na Internet: <<https://www.chromium.org/>>.
- GOOGLE. **Google Chrome**. 2008. (Acessado em 07/11/2021). Disponível na Internet: <<https://www.google.com/chrome/>>.
- GOOGLE. **V8 Engine**. 2008. (Acessado em 07/11/2021). Disponível na Internet: <<https://v8.dev/>>.
- HALDER, S.; SEDDIQUI, H. An entertainment recommendation system using the dynamics of user behavior over time. Em: . [S.l.: s.n.], 2014.
- HEUSER, C. A. **Projeto de Banco de Dados**. [S.l.]: Sagra Luzzatto 2001, 2008. (Livros Didáticos Informática UFRGS). ISBN 9788577803828.
- HOLOWAYCHUK, T.; STRONGLOOP. **Express**. 2010. (Acessado em 04/11/2021). Disponível na Internet: <<https://expressjs.com/>>.
- IAQUINTA, L.; GENTILE, A. L.; LOPS, P.; GEMMIS, M. de; SEMERARO, G. A hybrid content-collaborative recommender system integrated into an electronic performance support system. Em: **7th International Conference on Hybrid Intelligent Systems (HIS 2007)**. [S.l.: s.n.], 2007. p. 47–52.
- IRVING, D.; HERTWECK, K.; JOHNSTON, L.; OSTBLOM, J.; WICKHAM, C.; WILSON, G. **Research Software Engineering with Python: Building software that makes research possible**. 1st. ed. USA: Chapman and Hall/CRC, 2021. ISBN 9780367698324.
- ISINKAYE, F.; FOLAJIMI, Y.; OJOKOH, B. Recommendation systems: Principles, methods and evaluation. **Egyptian Informatics Journal**, v. 16, n. 3, p. 261–273, 2015. ISSN 1110-8665. Disponível na Internet: <<https://www.sciencedirect.com/science/article/pii/S1110866515000341>>.
- KUROSE, J. F.; ROSS, K. W. **Computer Networking: A Top-Down Approach (6th Edition)**. 6th. ed. [S.l.]: Pearson, 2012. ISBN 0132856204.
- LAB, G. C. D. **Our World in Data**. 2019. <<https://ourworldindata.org/>>. (Accessado em 23/10/2021).
- LANDIN, P. J. The mechanical evaluation of expressions. 1964.
- LTD, K. E. **Flourish**. 2016. <<https://flourish.studio/>>. (Accessado em 05/11/2021).
- MANNIE, E.; PAPADIMITRIOU, D. **Recovery (Protection and Restoration) Terminology for Generalized Multi-Protocol Label Switching (GMPLS)**. [S.l.], 2006.

MCKINNEY, W. **Pandas**. 2020. (Acessado em 23/10/2021). Disponível na Internet: <<https://doi.org/10.5281/zenodo.3509134>>.

Merve Acilar, A.; ARSLAN, A. A collaborative filtering method based on artificial immune network. **Expert Systems with Applications**, v. 36, n. 4, p. 8324–8332, 2009. ISSN 0957-4174. Disponível na Internet: <<https://www.sciencedirect.com/science/article/pii/S0957417408007100>>.

MICROSOFT. **Microsoft SQL Server Docs**. 2021. (Accessed on 13/11/2021). Disponível na Internet: <<https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15>>.

MUNZNER, T. **Visualization Analysis and Design**. CRC Press, 2015. (AK Peters Visualization Series). ISBN 9781498759717. Disponível na Internet: <<https://books.google.de/books?id=NfkYCwAAQBAJ>>.

NEUMANN, A.; LARANJEIRO, N.; BERNARDINO, J. An analysis of public rest web service apis. **IEEE Transactions on Services Computing**, PP, p. 1–1, 06 2018.

NOTTINGHAM, M. **Web Linking**. [S.l.], 2010. <<http://www.rfc-editor.org/rfc/rfc5988.txt>>. Disponível na Internet: <<http://www.rfc-editor.org/rfc/rfc5988.txt>>.

OVERFLOW, S. **Stack Overflow Developer Survey 2021**. 2020. (Accessed on 13/11/2021). Disponível na Internet: <<https://insights.stackoverflow.com/survey/2021>>.

POSTGRESQL, G. D. G. **About PostgreSQL**. 2021. (Accessed on 13/11/2021). Disponível na Internet: <<https://www.postgresql.org/about/>>.

PRADHAN, K. **Visualizing Netflix Data Using Python!** 2021. (Acessado em 04/11/2021). Disponível na Internet: <<https://www.analyticsvidhya.com/blog/2021/07/visualizing-netflix-data-using-python/>>.

PROCESSING, F. **P5.js**. 2014. (Acessado em 07/11/2021). Disponível na Internet: <<https://p5js.org/>>.

RITCHIE, H.; MATHIEU, E.; RODÉS-GUIRAO, L.; APPEL, C.; GIATTINO, C.; ORTIZ-OSPINA, E.; HASELL, J.; MACDONALD, B.; BELTEKIAN, D.; ROSER, M. Coronavirus pandemic (covid-19). **Our World in Data**, 2020. <Https://ourworldindata.org/coronavirus>.

ROSER, M. Why do we need to know about progress if we are concerned about the world's large problems? June 2021. (Acessado em 23/10/2021).

ROSSUM, G. van. **Python**. 1991. (Acessado em 23/10/2021). Disponível na Internet: <<https://www.python.org/>>.

SANDVINE. The global internet phenomena report: May 2020. October 2020. (Accessado em 23/10/2021).

SHELBY, Z. **Constrained RESTful Environments (CoRE) Link Format.** [S.l.], 2012. <<http://www.rfc-editor.org/rfc/rfc6690.txt>>. Disponível na Internet: <<http://www.rfc-editor.org/rfc/rfc6690.txt>>.

SMILKOV, D.; THORAT, N.; ASSOGBA, Y.; YUAN, A.; KREEGER, N.; YU, P.; ZHANG, K.; CAI, S.; NIELSEN, E.; SOERGEL, D.; BILESCHEI, S.; TERRY, M.; NICHOLSON, C.; GUPTA, S. N.; SIRAJUDDIN, S.; SCULLEY, D.; MONGA, R.; CORRADO, G.; VIÉGAS, F. B.; WATTENBERG, M. Tensorflow.js: Machine learning for the web and beyond. 2019.

SQL, I. C. **SQL Specification.** 2011. (Accessed on 13/11/2021). Disponível na Internet: <<http://www.wiscorp.com/sql20nn.zip>>.

STONEBRAKER, M.; ROWE, L. A. The design of postgres. **SIGMOD Rec.**, Association for Computing Machinery, New York, NY, USA, v. 15, n. 2, p. 340–355, jun. 1986. ISSN 0163-5808. Disponível na Internet: <<https://doi.org/10.1145/16856.16888>>.

SUJAN, Y. M.; SHASHIDHARA, H. D.; NAGAPADMA, D. R. Survey paper: Framework of rest apis. **International Research Journal of Engineering and Technology**, 2020.

SUL, S. da Saúde do Estado do Rio Grande do. **Painel Coronavírus RS.** 2021. <<https://ti.saude.rs.gov.br/covid19/>>. (Acessado em 23/10/2021).

TC39. **ECMAScript 2021 Reference.** 2021. Disponível na Internet: <<https://www.ecma-international.org/publications-and-standards/standards/ecma-262/>>.

TSCHINKEL, G.; SCIASCIO, C. di; MUTLU, B.; SABOL, V. The recommendation dashboard: A system to visualise and organise recommendations. Em: . [S.l.: s.n.], 2015.

WIDENIUS, M.; AXMARK, D.; DUBOIS, P. **Mysql Reference Manual.** 1st. ed. USA: O'Reilly Associates, Inc., 2002. ISBN 0596002653.