Construindo um *Dashboard* para a Base de Trabalhos Acadêmicos iFBox

Carlos Torres [1] **, Luigi Kita** [1] **, Victor Fiamoncini** [1]

[1]IFC

Abstract. *Data is a very valuable asset, even more so on the current times, making the comprehension of this data even more important. Data visualization is a great way of illustrating patterns and characteristics of samples, specially when done so by means of graphical representatinos. Dashboards are systems focused on gathering a variety of graphics, organizing them in a understandable fashion. The usage of frameworks and libraries greatly assists the development of a web system, not only making the process smoother but also giving it more features.*

Resumo. *Dados são um ativo extremamente valioso, ainda mais nos dias atuais, o que torna a compreensão desses dados ainda mais importante. A visualização de dados é uma ótima maneira de ilustrar padrões e características de amostras, especialmente quando isso é feito por meio de representações gráficas. Os dashboards são sistemas focados em reunir uma variedade de gráficos, organizando-os de maneira compreensível. O uso de frameworks e bibliotecas ajuda amplamente o desenvolvimento de um sistema Web, não apenas tornando o processo mais fluído, mas também oferecendo mais recursos a este.*

1. Introdução

A habilidade de se visualizar informações é de enorme importância, ainda mais nos dias atuais, uma vez que a quantidade de dados gerados em nosso cotidiano só tende a aumentar. Conforme aponta Runkler[1] (2012), quantidades crescentes de dados são armazenadas e processadas em empresas e na indústria, com o objetivo de otimizar processos produtivos e também para aumentar as vendas, mas este recurso pode ser utilizado para qualquer corpo institucional. Os utilitários gráficos são uma das melhores formas de realizar essa visualização, dado seu apelo inato á visão.

O método padrão para visualização de dados é a plotagem (Runkler[1], 2012), produzindo figuras ilustrativas que podem representar como os dados são distribuídos e os relacionamentos presentes neles. Como vários gráficos podem apresentar mais detalhadamente a amostra de estudo do que uma única representação, um sistema capaz de reunir de maneira organizada e hierárquica essas múltiplas abordagens proporcionaria grandes vantagens, sendo que esse tipo de sistema é denominado *dashboard*. Por esse motivo, este artigo procura introduzir a implementação de um *dashboard* para o repositório *iFBox*. A primeira seção exploram a base teórica desse tipo de sistema. A segunda explica quais métodos foram usados na implementação. A terceiro explora os resultados obtidos. A última encerra este trabalho.

2. Fundamentação Teórica

A informação é um bem com extrema importância, não apenas para indivíduos mas também para grupos como governos, corporações e organizações em geral. Essa realidade se evidência cada vez mais, especialmente nos dias atuais onde o volume de dados cresce exponencialmente a cada momento, dito o constante barateamento de componentes de armazenamento como *Hard-Drives* e SSDs, juntamente ao avanço de técnicas para a captura de dados, como afirmam Kotu e Deshpande[2] (2019). Porém, nem todos os dados coletados se mostram por serem uteis, não sendo de forma alguma equivalentes a informação em si. Bourgeois[3] define dados como sendo a parte bruta da informação, podendo ser quantitativa como números ou qualitativa como letras e palavras (2014).

Dado essa grotesca quantia de dados, a tarefa de criação e manutenção de sistemas que tratam e formatam a informação, é de altíssima importância, para que dessa forma se possa aumentar sua visualização e por consequência, sua compreensão. Esse tipo de sistema é descrito por Laudon e Laudon[4] da seguinte maneira:

> Um Sistema de Informação (SI) pode ser definido tecnicamente como um conjunto de componentes inter-relacionados que coletam (ou recuperam), processam, armazenam e distribuem informações destinadas a apoiar a tomada de decisões, a coordenação e o controle de uma organização. (2011, p. 12)

Dessa forma, pode-se mencionar um tipo específico de sistema de informação que são focados na apresentação das informações, sendo estes os "Sistemas de Apoio a Decisão" (SAD). Tal tipo de sistema tem se estabelecido nas plataformas web em vastas quantidades, dito a simplicidade de acesso aos mesmos uma vez que uma conexão de internet está disponível, dessa forma, permitindo aos usuários usufruir da interatividade dos recursos gráficos (Laudon e Laudon[4], 2011).

3. Métodos

Ao projetar a interface do *Dashboard*, a intuitividade e a interatividade foram os dois fatores que receberam maior ênfase. Por esse motivo, um conjunto de *frameworks* e bibliotécas foram aplicadod, imensamente auxiliando a criação de cada componente. O primeiro deles foi o *Bootstrap*, amplamente aplicado em cada página pois fornece muitas classes preparadas de CSS juntamente a outras extensões *JQuery*[5] (Bootstrap, Online), o que instantaneamente aumenta o apelo visual da *User Interface* (*UI*, ou Interface do Usuário) e também a torna mais acessível para usuários que não estão o acessando de um desktop, dito que seu sistema de grades fornece responsividade por padrão.

O segundo deles foi o*React*, que busca tornar o processo de mudança de estado o mais fluido possível, diminuindo a quantidade de recarregamentos necessários para apresentar novas informações. Uma de suas principais características é que, uma vez que um componente altera de estado, somente esse componente específico será recarregado (React, Online)[6]. Sua sintaxe é baseada em *Javascript*, aproveitando o conhecimento já existente. *Redux* foi a terceira biblioteca, fornecendo um mecanismo padronizado para alterações de estado, seguindo os três princípios a seguir:

- **Fonte única de dados:** uma única árvore de estados é mantida, facilitando a tarefa de *debugging* e a inspeção.

- **O estado é *read-only*:** a única maneira pela qual um estado pode ser modificado é através de objetos *action*, que detalham as alterações.

- **Funções puras modificam o estado:** para modificar a árvore de estados, funções puras denominadas *reducers* devem ser implementadas. Estas recebem uma um *action* e o estado como parâmetros (Redux, Online)[7].

Em conjunto destes, a biblioteca *Axios* foi usada para facilitar solicitações *HTTP* em navegadores distintos, executando os ajustes necessários com um único parâmetro *JSON* (Axios, Online)[8], como pode ser visto na listagem 1.

Listagem 1. Requisição POST através da biblioteca *Axios*[8]

```
1 axios({
2   method: 'post',
3   url: '/user/12345',
4   data: {
5     firstName: 'Fred',
6     lastName: 'Flintstone'
7   }
8 });
```

Buscando obter um desempenho razoável na preparação dos dados a serem representados graficamente e exibidos na interface, foi-se utilizada a biblioteca *Lodash*, uma vez que esta fornece funções de rápida manipulação de matrizes (Lodash, Online)[9], diminuindo o custo do pré-processamento.

Por fim, para se plotar os gráficos foi-se utilizada a biblioteca *CanvasJS*. Esta possui um *API* simples e altamente intuitividade, tal como uma performance de renderização consideravelmente alta, sendo capaz de processar uma plotagem com 100 000 pontos em apenas uma fração de segundos (CanvasJS, Online)[10]. Um exemplo de plotagem é ilustrado na listagem 2, sendo renderizado para o gráfico da figura 1.

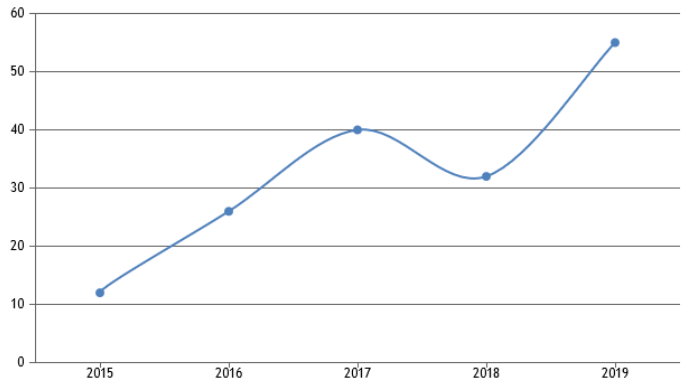

Figura 1. Gráfico gerado apartir do script da listagem 2

Listagem 2. Script básico para gerção de gráficos

```
<html>
<head>
<script type = "text/javascript">
window.onload = function() {
var chart = new CanvasJS.Chart("chartContainer", {
 theme: "light1"
 data: [{
  type: "spline",
  dataPoints: [
   {label: "2015",  y: 12},
   {label: "2016", y: 26},
   {label: "2017", y: 40},
   {label: "2018",  y: 32},
   {label: "2019",  y: 55}]
  }]
 });
chart.render();
}
</script>
</head>
<body>
<div id = "chartContainer"></div>
<script src="https://canvasjs.com/assets/script/canvasjs
    .min.js"> </script>
</body>
</html>
```

A biblioteca permite um ampla variedade de configurações adicionais, como animações e customização de fontes, tal como diversas opções de tipos de gráfico. Dentre as trinta opções de tipo de plotagem, os gráficos de barra, função e circular, sendo reconhecidos pela biblioteca como *bar*, *spline* e *pie* respectivamente, foram selecionados para uso.

Listagem 3. Parte das configurações de um arquivo *JMX* para testes de acesso

```
<stringProp name="ThreadGroup.num_threads">100000</
    stringProp>
<stringProp name="ThreadGroup.ramp_time">1</stringProp>
```

Para avaliar o comportamento do site perante alta demanda, foi-se efetuado testes de requisição *HTTP* através do *framework JMeter*, que é uma aplicação de codigo aberto desenvolvida para automatizar testes de acesso (ApacheJMeter, Online)[11]. Este utiliza arquivos de configuração do tipo *JMX*, que é uma forma estruturada por meio de tags de se persistir dados, semalhante ao *XML*. A listagem 3 contem uma seção das configurações utilizadas.

4. Resultados

Com a expectativa de grande aumento de dados no repositório pensou-se em utilizar métodos para selecionar apenas as informações relevantes para serem representadas nos gráficos, com isso, primeiramente houve uma análise da quantidade máxima a ser atribuída as barras de um gráfico de barra, para isso gráficos de exemplos foram utilizados para comparar a disposição das barras e verificar o número máximo de barras para deixar o gráfico simples e sem poluição visual, onde o valor encontrado foi em torno de dez a dezesseis barras. Decorrente disso foi atribuído o valor de barras dos gráficos como sendo a raiz quadrada do tamanho dos dados, ou seja, com 49 (quarenta e nove) valores no banco de dados, iriam ser criados apenas sete barras no gráfico.

Essa seleção dos itens a serem colocadas seria possível, se o banco de dados possui-se um conjunto pequeno de dados, o que não é muito provável de ocorrer com a atual taxa de submissão de trabalhos e os vários trabalhos antigos que serão colocados no repositório, devido a esses fatores procurou-se outra forma de limitar o número de barras nos gráficos, que após algumas análises decidiu-se por adotar apenas os primeiros dez itens retornados pelo *web-service*, com isso eliminando a variação de barras, simplificando o gráfico e diminuindo o estresse do *web-service* em realizar as consultas.

Para a seleção dos itens a serem colocados na barra lateral, foi analisado a estrutura do repositório para estabelecer as tabelas mais importantes, com isso chegou-se a conclusão de utilizar os itens: *Cursos*, *Artigos*, *Usuários* e *Tags*. O item *Dashboard* redireciona à página inicial do site, tendo tais categorias ilustradas na figura 2. Também foram-se adicionados os itens *Github* e *Site*, onde o primeiro é um link para o repositório do código fonte do projeto, e o segundo redirecionada para o site pelo qual o acesso e postagem dos trabalhos é feito.

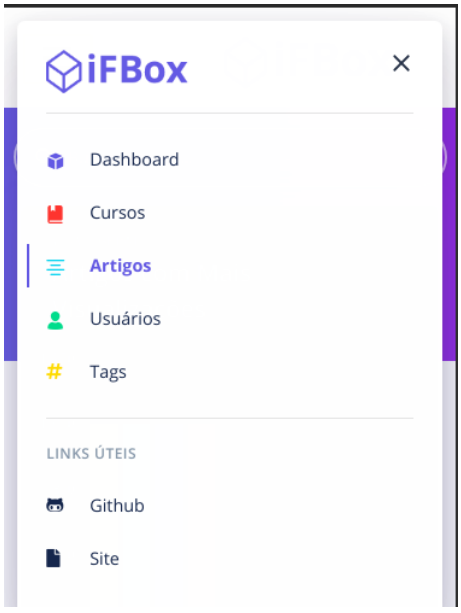

Figura 2. Disposição dos itens de navegação

Outra funcionalidade importante que é comum em agregadores de dados como este é a de pesquisa. Para tal, foi-se posicionado uma barra horizontal no lado superior

direito do site, onde é possível efetuar consultas sobre artigos presentes no repositório do *iFBox*, como pode ser visto na figura 3. Esta captura de tela se refere à página inicial, onde se é exibido a progressão das publicações ao decorrer dos anos. Também se pode notar três pódios representados por cartões flutuantes, localizados logo acima da linha de publicações. Estes exibem o curso com mais visualizações, o usuário com o maior número de publicações, e a tag mais utilizada, respectivamente.

Figura 3. Captura de tela da página inicial do *Dashboard*

Devido a utilização de *frameworks* dinâmicos como o *Bootstrap* e o *React*, a renderização das páginas web do *Dashboard* para plataformas móveis também foi implementada com sucesso. Através do recurso de responsividade, a interface gráfica do mesmo é capaz de se adequar a tela onde esse esta sendo acessado, de forma que os elementos visuais sejam assim dispostos proporcionalmente. A imagem 4 ilustra a interface do *Dashboard* neste contexto.

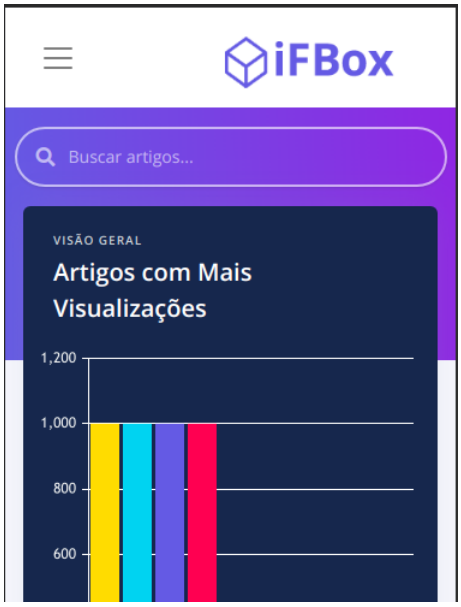

Figura 4. *Dashboard* em um ambiente com dimensões reduzidas

Após se efetuar os testes de requisição *HTTP* com o *framework JMeter*, notou-se que o site permanece estável com 10000 requisições, sendo que estas foram efetuadas

simultaneamente. De maneira similar, foi possível diagnosticar qual é o limite do mesmo, uma vez que um teste com dez vezes a quantia anterior completamente inviabilizou o acesso ao *Dashboard*. Em outro teste, carregando 30000 requisições, o site permaneceu disponível porém uma lentidão significativa também foi detectada, não respondendo à maior parte destas requisições.

5. Conclusão

A implementação do *Dashboard* para o repositório *iFBox*, atestou-se por ser um sucesso, o que também valida a funcionalidade do projeto como um todo. Ao fazer uso do *API* proporcionado pelo repositório, diversas consultas complexas puderam ser simplificadas, diminuindo imensamente a complexidade do site, generalizando o acesso e a distribuição de dados, e também o tornando mais rápido para o cliente por diminuir a quantidade de processamento necessário em seu computador.

Ao se empregar um conjunto de *tags* definidas hierarquicamente, o *frontend* do site se tornou altamente intuitivo para navegação, cumprindo com os propósitos iniciais deste projeto. O uso de bibliotecas permitiu que uma plataforma muito concisa fosse feita, possibilitando expansões futuras mais fluidas, uma vez que sempre que forem publicadas melhorias para tais bibliotecas, as mesmas melhorias inevitavelmente retornam ao *Dashboard*.

Por fim, notou-se que o conjunto de gráficos exibidos visualmente mostrou coincidir bem com o domínio dos dados, o que permitirá a extração de mais informações dos mesmos. As informações que agora podem ser extraídas pela plataforma desenvolvida certamente permitirão obter novos conhecimentos a partir dos dados já disponíveis. O site se mostrou resistent a algumas milhares de requisições, o que o torna adequado ao tamanho de seu público alvo. No geral, o *Dashboard* também possibilitará um método rápido e confiável para propagar o conhecimento gerado no Instituto Federal Catarinense, trazendo uma perspectiva holística para as pesquisas sendo realizadas, além de aproveitar a acessibilidade e extensibilidade trazida pela internet.

Referências

1 RUNKLER, T. A. *Data Analytics*: Models and algorithms for intelligent data analysis. [S.l.]: Spring, 2012.

2 KOTU, V.; DESHPANDE, B. *Data Science*: Concepts and practice. Second edition. [S.l.]: Elsevier, 2019.

3 BOURGEOIS, D. T. *Information Systems for Business and Beyond*: Concepts and practice. [s.e.]. [S.l.]: Saylor Foundation, 2014.

4 LAUDON, K.; LAUDON, J. *Management Information Systems*. 9th edition. ed. [S.l.]: Pearson, 2011.

5 [N.A]. *Bootstrap*. Acesso em: 8-11-2019. Disponível em: ⟨https://getbootstrap.com/⟩.

6 [N.A.]. *React*: A javascript library for building user interfaces. Acesso em: 12-11-2019. Disponível em: ⟨https://reactjs.org/⟩.

7 ABRAMOV, D. *Redux*: Three principles. Acesso em: 12-11-2019. Disponível em: ⟨https://redux.js.org/introduction/three-principles⟩.

8 URALTSEV, N. et al. *axios*: Promise based http client for the browser and node.js. Acesso em: 12-11-2019. Disponível em: ⟨https://github.com/axios/axios⟩.

9 SIROIS, J.-P.; HALL, Z. *Lodash*: A modern javascript utility library delivering modularity, performance extras. Acesso em: 12-11-2019. Disponível em: ⟨https://lodash.com/⟩.

10 [N.A.]. *Beautiful HTML5 JavaScript Charts*: Responsive html5 charting library with a simple api and 10x performance – makes your dashboards fly! Acesso em: 26-11-2019. Disponível em: ⟨https://canvasjs.com/⟩.

11 [N.A.]. *Apache JMeter™*. Acesso em: 5-12-2019. Disponível em: ⟨https://jmeter.apache.org/⟩.