Mapa digital para gestão do conhecimento

A construção de um sistema com o software Visão

Organizador
Milton Shintaku

Autores
Rebeca dos Santos de Moura
Janinne Barcelos
Lucas Angelo da Silveira
Tiago Emmanuel Nunes Braga

PRESIDÊNCIA DA REPÚBLICA

Jair Messias Bolsonaro
Presidente da República

Hamilton Mourão
Vice-Presidente da República

MINISTÉRIO DA MULHER, DA FAMÍLIA E DOS DIREITOS HUMANOS

Damares Alves
Ministra da Mulher, da Família e dos Direitos Humanos

Sérgio Luiz Cury Carazza
Secretário Executivo

SECRETARIA NACIONAL DA JUVENTUDE

Jayana da Silva
Secretária Nacional da Juventude

Luis Vannucci Cantanhede Cardoso
Secretário Adjunto

Flaviane Agustini Stedille
Chefe de Gabinete

Lívia Lopes de Souza
Gerente de Projetos

Douglas Pinheiro Azevedo de Souza Andrade
Coordenador-Geral de Cidadania

Lucas Cardozo Dalló
Coordenador-Geral de Relações Institucionais

Eduardo Zimmermann e Silva
Coordenador-Geral de Políticas Finalísticas

Rafael Davi Campos
Secretário-Executivo do Conselho Nacional da Juventude

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES

Cecília Leite Oliveira
Diretora

Reginaldo de Araújo Silva
Coordenação de Administração - COADM

José Luis dos Santos Nascimento
Coordenação de Planejamento, Acompanhamento e Avaliação - COPAV

Bianca Amaro de Melo
Coordenadora-Geral de Pesquisa e Manutenção de Produtos Consolidados - CGPC

Arthur Fernando Costa
Coordenador-Geral de Pesquisa e Desenvolvimento de Novos Produtos - CGNP

Gustavo Silva Saldanha
Coordenador de Ensino e Pesquisa, Ciência e Tecnologia da Informação - COEPE

Tiago Emmanuel Nunes Braga
Coordenador-Geral de Tecnologias de Informação e Informática - CGTI

Milton Shintaku
Coordenador de Articulação, Geração e Aplicação de Tecnologia - COTEC

MINISTÉRIO DA MULHER,
DA FAMÍLIA E DOS DIREITOS
HUMANOS

Secretaria Nacional
da Juventude

MINISTÉRIO DA CIÊNCIA,
TECNOLOGIA, INOVAÇÕES
E COMUNICAÇÕES

Instituto Brasileiro de Informação
em Ciência e Tecnologia

Mapa digital para gestão do conhecimento

A construção de um sistema com o software Visão

Organizador
Milton Shintaku

Rebeca dos Santos de Moura
Janinne Barcelos
Lucas Angelo da Silveira
Tiago Emanuel Braga

Brasília
2019

Projeto de pesquisa:
"Estudo para sistematização, desenvolvimento e manutenção do Sistema Nacional de Juventude"

Coordenador do projeto:
Milton Shintaku - Coordenador de Articulação, Geração e Aplicação de Tecnologia (Cotec/Ibict)

Revisão de Conteúdo
Rafael Teixeira de Souza

Design Gráfico, Diagramação e Ilustrações
Nuielle Medeiros
Mariela Muruga

Normalização
Ingrid Schiessl

Autores
Rebeca dos Santos de Moura
Janinne Barcelos
Lucas Angelo da Silveira
Tiago Emmanuel Braga

Dados Internacionais de Catalogação-na-Publicação (CIP)
Bibliotecária: Ingrid Schiessl CRB1/ 3084

M297

Mapa digital para gestão do conhecimento: a construção de um sistema com o software Visão/ Milton Shintaku (org.); Rebeca dos Santos de Moura et al. Brasília: Ibict, 2019.

92 p.
ISBN 978-85-7013-163-8
DOI:10.18225/9788570131638

1. Sistemas de informação. 2. Gestão do conhecimento. 3. Gestão da informação. 4. Mapa digital. 5. Tecnologia da Informação. I. Shintaku, Milton. II. Moura, Rebeca dos Santos. III. Barcelos, Janinne. IV. Silveira, Lucas Angelo da. V. Braga, Tiago Emanuel. VI. Título.

CDD 658.4038
CDU 002.6:658

COMO CITAR
SHINTAKU, Milton (Org.). **Mapa digital para gestão do conhecimento**: a construção de um sistema com o software Visão. Brasília: Ibict, 2019. 92 p. DOI: 10.18225/9788570131638

Setor de Autarquias Sul (SAUS) Quadra 05 Lote 06, Bloco H – 5º andar
Cep:70.070-912 – Brasília, DF Telefones: 55 (61) 3217-6360/55 /(61)3217-6350

www.ibict.br

Sumário

Agradecimentos

Ao finalizar uma obra, visualizamos o longo caminho que nos levou até os resultados. Nesse percurso, várias pessoas nos apoiaram, algumas vezes nos confortando, outras nos ajudando a prosseguir, de modo que esta obra não é um trabalho somente dos autores, mas representa um pouco de cada colaborador que direta ou indiretamente contribuiu para a sua realização. Ao agradecermos, esperamos que as pessoas que nos ajudaram sintam um pouco da nossa retribuição e reconhecimento.

A nossas famílias, tanto as biológicas quanto as de coração, que nos confortam nos momentos mais difíceis e nos acompanham nos momentos de alegria.

Aos amigos, companheiros de vida, pelos momentos inspiradores.

Aos colegas da Coordenação de Articulação, Geração e Aplicação de Tecnologia, pelo convívio diário que nos ajuda a continuar.

Ao Instituto Brasileiro de Informação em Ciência e Tecnologia, por meio de sua diretora, a Dra. Cecília Leite de Oliveira, pelo seu apoio constante. À equipe da Coordenação de Tecnologia Aplicada, por meio de seu coordenador Tiago Emanuel Braga. À equipe de apoio aos projetos, por meio de sua coordenadora Valéria Paiva. À Leda, Adreana e tantos outros que participam da vida diária no instituto.

À Secretaria Nacional da Juventude, por meio de sua secretária Jayana Nicaretta da Silva, pelo fomento mediante o qual essa obra se tornou possível.

A todos o nosso muito obrigado.

O organizador.

Prefácio

O Sistema Aberto de Observatório para Visualização de Informações (Visão) surge como uma proposta do Instituto Brasileiro de Informação em Ciência e Tecnologia (Ibict) para um sistema de gestão dos dados governamentais a partir da disponibilização de mecanismos complexos de visualização. Sua criação está apoiada nas filosofias abertas, defendidas pelo Instituto ao longo das últimas décadas, e busca possibilitar que a sociedade tenha garantido seu direito de acesso pleno às informações governamentais.

O Brasil avançou muito nos últimos anos em relação à abertura de dados. Os compromissos firmados por meio da Open Government Partnership (OGP) serviram para situar o País em uma posição privilegiada no que diz respeito à disponibilização de dados abertos. Iniciativas como a Infraestrutura Nacional de Dados Abertos (INDA) e a Infraestrutura Nacional de Dados Espaciais (INDE) permitiram novas possibilidades de entendimento da utilização da máquina pública. No entanto, a abertura de dados não necessariamente indica que os dados são utilizados. O Visão busca fortalecer o processo de utilização dos dados governamentais ao mesmo tempo que possibilita a criação de narrativas resultantes de diferentes perspectivas. Entende-se que a diversidade de visões permitirá que os temas caros à população sejam abordados com a profundidade necessária de uma nação tão plural como a nossa.

Os desafios para a concretização dos objetivos do Visão são vários. Desde a definição de padrões e preparação dos metadados, passando pela identificação e aplicação de tecnologias, entende-se que o sistema ainda está em sua fase inicial.

Coube à Coordenação de Articulação, Geração e Aplicação de Tecnologia o árduo trabalho de desvendar o Visão e suas aplicações. Seguindo a linha de trabalho já adotada pela coordenação em outras ocasiões, os aprendizados obtidos durante a jornada foram sintetizados a fim de serem compartilhados com a comunidade. Este manual registra a fase inicial do Visão em sua versão beta. O foco dele é partilhar os procedimentos práticos necessários para a utilização do software, bem como orientar o usuário na utilização do sistema em construção.

Tiago Emanuel Braga
Coordenador-Geral de Tecnologias da Informação e Informática.

Apresentação

O ***Mapa digital para gestão do conhecimento: a construção de um sistema com o software Visão*** é um dos resultados do projeto de pesquisa intitulado "Estudos para sistematização e desenvolvimento do Sistema Nacional de Juventude (Sinajuve)" do Instituto Brasileiro de Informação em Ciência e Tecnologia (Ibict) em parceria com a Secretaria Nacional da Juventude (SNJ). O projeto visa desenvolver estratégias para gestão de políticas públicas voltadas à juventude brasileira e ao mapeamento de todos os estabelecimentos promotores de políticas públicas de juventude nas esferas federal, estadual e municipal.

A obra busca definir e apresentar os conceitos e requisitos contemplados pelo Sistema Aberto de Observatórios para Visualização de Informação (Visão), considerado a ferramenta ideal para atender ao objetivo do ***Mapa das Unidades de Juventude***. O mapeamento das unidades de juventude está previsto na estrutura do Sinajuve como parte integrante do Subsistema de Informação, Monitoramento e Avaliação (SIMA), portanto, a realização deste livro atende aos objetivos do projeto no que tange ao desenvolvimento de estudos voltados para a criação de uma estratégia de implantação do Sinajuve.

Criado pela Coordenação de Tecnologias Aplicadas a Novos Produtos (COTEA) do Ibict, o Visão é conceituado como um Sistema de Informação Geográfica (SIG) que possibilita o gerenciamento de dados pautados no componente geográfico de território, por meio de armazenamento, manipulação, análise, demonstração e relatos de dados referenciados geograficamente. Foi desenvolvido em código aberto e tem sido mantido por uma comunidade de desenvolvedores com o objetivo de dar suporte, por meio de um mapa interativo, à tomada de decisão na gestão pública e à pesquisa técnico-científica ao integrar inteligência de localização com dados geoespaciais.

Com este livro, os autores também esperam contribuir para o contínuo aprimoramento do software, fornecendo informações necessárias aos profissionais envolvidos no desenvolvimento, teste, implementação, personalização e operação do Visão. Além disso, espera-se que o livro

possa oferecer informações relevantes tanto para os produtores quanto para os usuários de informações geoespaciais.

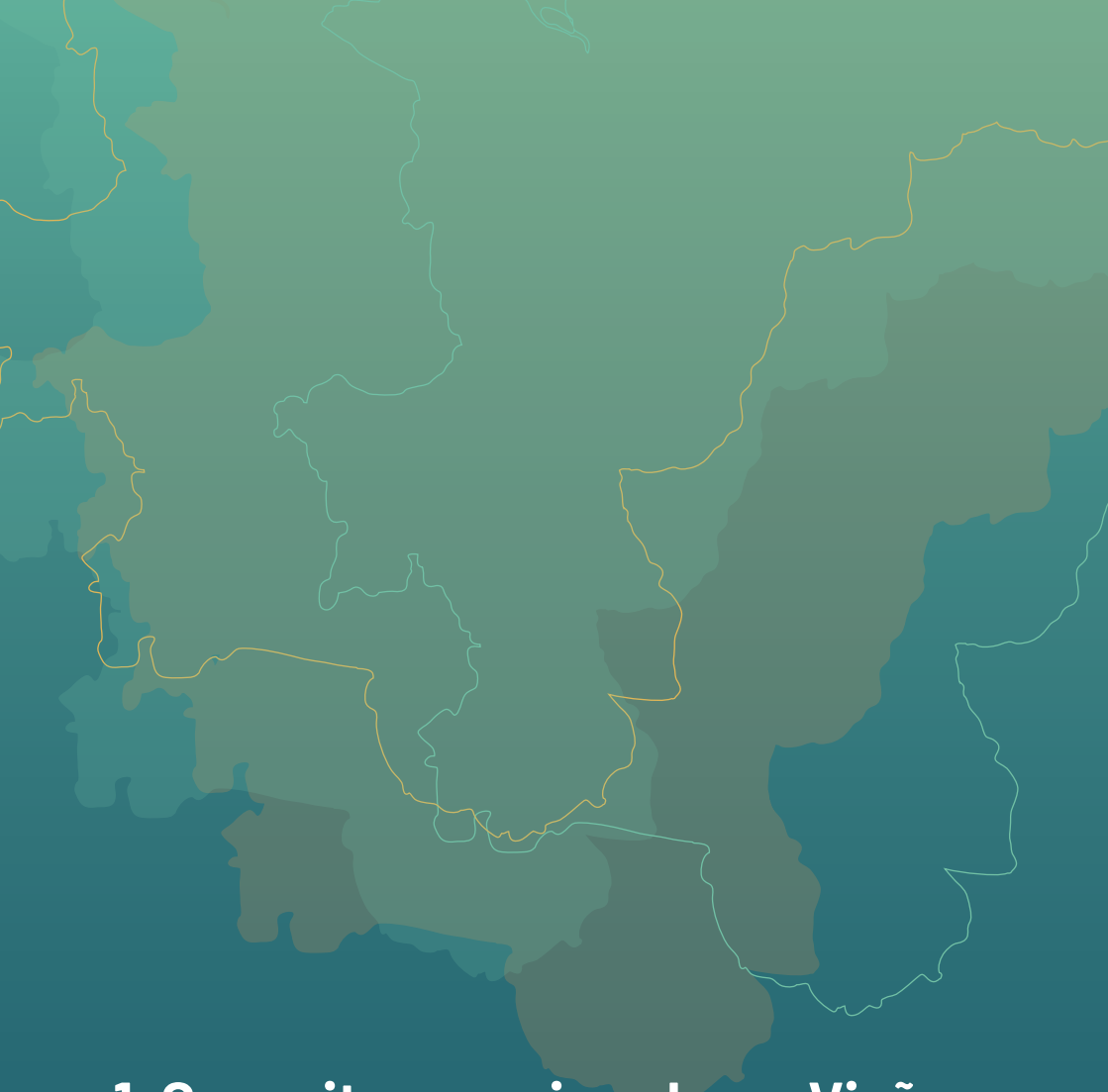
O livro fornece elementos conceituais da criação do Visão, na versão 2.1, do ramo teste-visão do GitHub[1], que foi usado como base para o desenvolvimento do ***Mapa das Unidades de Juventude***, assim como uma descrição compreensível de suas funções implementadas, concentrando-se essencialmente nas características técnicas, na modelagem dos dados e na usabilidade.

A primeira parte do ***Mapa digital para gestão do conhecimento*** apresenta breve relato da evolução dos mapas, discutindo a importância de seu uso, desde a antiguidade até os tempos atuais. No capítulo 2, são descritas características fundamentais do Visão, incluindo os tipos de dados compatíveis com a plataforma, o mapa interativo e o menu de ferramentas, contendo indicadores, filtros geográficos e camadas de marcadores.

No capítulo ***Instalação do Visão*** são descritas as principais tecnologias utilizadas no projeto de desenvolvimento do software e há um tutorial completo que detalha desde a preparação do ambiente até a disponibilização para uso em produção. Seguem-se os capítulos 4 e 5, que tratam, respectivamente, do banco de dados relacional e do guia de uso da aplicação.

O capítulo Visão no ***Sistema Nacional de Juventude*** (Sinajuve) discorre sobre a utilização do software na construção do ***Mapa das Unidades de Juventude***. Por fim, em ***Versões futuras***, o autor apresenta a perspectiva conceitual do Visão e suas projeções de futuro como uma proposta ambiciosa do Instituto Brasileiro de Informação em Ciência e Tecnologia de criar um sistema de gestão focado nas diversas possibilidades de visualização de dados em filosofia aberta.

1 Disponível em: <https://github.com/IBICT/visao/tree/teste_visao>, Acesso em: 10 set. 2019.



1. Conceitos gerais sobre o Visão

Janinne Barcelos

Lucas Angelo da Silveira

Rebeca dos Santos de Moura

Mais do que simplesmente desenhos e imagens, os mapas são representações da realidade que ilustram de forma reduzida um determinado ponto geográfico. Uma leitura alternativa da cultura e da história humana relata que, antes mesmo da invenção da escrita, a humanidade desenhou mapas nas paredes das cavernas, por meio de pinturas rupestres, com a intenção de representar o caminho dos locais onde havia caça. Segundo diversos historiadores, o mapa é uma das principais tecnologias desenvolvidas para que a humanidade suprisse a necessidade elementar de se localizar, de saber onde se está, de entender o lugar das pessoas no universo.

"Em seu nível mais simples, os mapas são uma coleção de linhas (as ruas) e espaços (os quarteirões). É evidente, contudo, que a realidade possui muito mais dados, e fazer um mapa melhor implica em colocar mais desses dados nele" (SUMARES, 2016). Essa é a razão pela qual os mapas oferecem uma gama de informações muito maior do que num primeiro momento. Para além das questões de localização, os mapas também são considerados importantes instrumentos de comunicação e conhecimento de aspectos culturais, estratégicos, bélicos, econômicos e religiosos, apoiando fundamentalmente o desenvolvimento de várias ciências e profissões (BURDA, 2014).

Os mapas tiveram grandes participações, por exemplo, no início das navegações europeias, quando muitos continentes foram descobertos, explorados e habitados, e muitos dados e informações sobre baías, enseadas, montanhas, rios e clima, entre outros, foram coletados. Tais fatos influenciaram no desenvolvimento dos atlas e marcaram o começo da cartografia moderna, período em que o conhecimento geocartográfico passou a se preocupar mais não só com a elaboração do mapa em si, como também com a comunicação estabelecida com seu usuário (MATIAS, 1996).

Os principais instrumentos usados para coletas de dados e informações geográficas no período das grandes navegações eram as bússolas, os astrolábios (antiga ferramenta astronômica usada para resolver problemas relacionados ao tempo, à posição do sol e das estrelas no céu), os quadrantes e sextantes. Séculos mais tarde, os instrumentos evoluíram para o moderno sensoriamento remoto (obtenção de informações por meio de sensores acoplados em aeronaves, satélites e balões), fotos aéreas (extraídas a partir de câmeras fixadas em aviões) e imagens coletadas por monitoramento de satélites (MARTINELLI, 2011).

Com a expansão das tecnologias de posicionamento global ou GPS (Global Positioning System) na última década, os mapas de papel cederam espaço para os mapas digitais, que hoje ocupam boa parte do dia a dia da população mundial na solução de diversas situações rotineiras, como localizar o mercado mais próximo, se situar dentro de um shopping center, encontrar um caminho pela cidade, conferir o estado do trânsito ou ver onde está o transporte que você espera. Em muitos casos, os usuários podem escolher entre mapas virtuais por satélite (com vista aérea) e vistas híbridas (uma combinação de mapa virtual e vistas aéreas), ampliando a visão sobre a superfície do planeta e atraindo novos usuários a cada dia (RAMOS, 2005; MARTINELLI, 2011).

Nesse contexto, a cartografia tem inovado as formas de representação da realidade e de apresentação de dados buscando se adequar à novas mídias digitais, cujas ferramentas e funcionalidades permitem ao usuário realizar consultas ao banco de dados, selecionar as variáveis a serem visualizadas e ter seus resultados exibidos no mapa. As mídias possibilitam que todos os usuários, mesmo aqueles sem co-

nhecimento específico, se comuniquem e visualizem dados geoespaciais na rede mundial de computadores (World Wide Web ou apenas web), tornando o uso de mapas mais atrativo e mais evidente do que nunca. Até mesmo Sistemas de Informação Geográfica (SIGs) tradicionais têm buscado adequar suas funcionalidades ao novo ambiente (RAMOS, 2005; BURDA, 2014).

Em virtude desse boom da geolocalização, ou seja, da proliferação de softwares e aplicativos que possibilitam uma cartografia digital interconectada, vários órgãos governamentais têm recorrido aos serviços de mapa para basear suas tomadas de decisão a partir de uma perspectiva que envolve medidas latitudinais e longitudinais. De acordo com Koinski e Hoppen (2017), as variáveis podem revelar padrões e aspectos fundamentais de comportamento de forma rápida e intuitiva, proporcionando o desenvolvimento de projetos com alto valor agregado e o aperfeiçoamento de estratégias nas futuras decisões e políticas públicas.

Algumas instituições possuem até mesmo geoportais e produtos cartográficos como informações ambientais, dados geodésicos e atlas digitais, buscando disponibilizar dados públicos para a consulta e download. Os atlas digitais, que fazem parte dos SIGs, têm por objetivo a geração e a visualização de materiais para análises de fenômenos com expressão territorial, como é o caso de dados censitários de população e do Índice de Desenvolvimento Humano (IDH). A comunicação de informações geoespaciais por meio dos atlas tende a ser mais rica, mais eficiente e personalizável. Além disso, é também mais amigável se comparada à visualização de dados brutos em tabelas do Instituto Brasileiro de Geografia e Estatística (IBGE), por exemplo.

Com o aumento da demanda pela utilização de inteligências de localização, de atlas digitais e consequentemente de SIGs, o Ibict tem se aliado aos esforços governamentais para desenvolver uma infraestrutura de informação capaz de catalogar, integrar e harmonizar dados geoespaciais: o Visão. Integrando funcionalidades de visualização geográfica e de refinamento da busca por meio de indicadores personalizáveis, o Visão permite o armazenamento e a disponibilização de dados abertos de governo e de pesquisa.

Diariamente, informações georreferenciadas são produzidas e utilizadas por todos os setores da sociedade. A distribuição da população e suas variáveis sócioeconômicas são apenas alguns dos milhares de dados que podem servir como aporte a pesquisas científicas e à construção de políticas públicas. Não obstante, no Brasil, a disponibilização de dados abertos é normatizada pela Lei de Acesso à Informação (Lei n° 12.527, de 18 de novembro de 2011), que regulamenta o direito constitucional de acesso às informações públicas, e pela Política de Dados Abertos do Poder Executivo Federal (Decreto n° 877 de 11 de maio de 2016).

Contudo, atualmente os dados estão distribuídos em bases de instituições distintas, em formatos e padrões divergentes e, por vezes, em sistemas sem interoperabilidade. Devido a essa dispersão informacional, a tarefa de cruzar dados de diferentes fontes para criação de novos indicadores tem se mostrado inviável. Com efeito, nos últimos anos, o Governo Federal tem protagonizado ações que estimulam a interação entre bases de organizações de todas as esferas públicas e áreas temáticas de políticas públicas nos três poderes.

Como grande incentivador do acesso aberto no Brasil, cabe ao Ibict justamente a missão de "promover a competência, o desenvolvimento de recursos e a infraestrutura de informação em ciência e tecnologia para a produção, socialização e integração do conhecimento científico e tecnológico" (IBICT, 2018). Assim, o Visão foi criado pelo Ibict para atender às necessidades de gestão e visualização de informação integrada de diversas organizações governamentais responsáveis pelo desenvolvimento social.

Embora já existisse no Brasil alguns sistemas de visualização de dados geográficos originários de diferentes bases, como por exemplo o visualizador da Infraestrutura Nacional de Dados Espaciais (INDE), os sistemas não suportam a criação de novos indicadores tampouco a manipulação destes dados de maneira mais flexível. Em contrapartida, o Visão traz como inovação uma interface que possibilita a manipulação dos dados de diferentes bases e a criação de novos indicadores, filtros e camadas de marcadores.

Referências

BURDA, N. A. **Cartografia e patrimônio arquitetônico**: a elaboração do atlas eletrônico do sítio histórico urbano na Lapa. 2014. 330 f. Tese (Doutorado) - Curso de Pós-graduação em Geografia Humana, Faculdade de Filosofia, Letras e Ciências Humanas da Universidade de São Paulo, Universidade de São Paulo, São Paulo, 2014. Disponível em: <http://www.teses.usp.br/teses/disponiveis/8/8136/tde-20032014-105146/pt-br.php>. Acesso em: 07 ago. 2019.

KOINSKI, V.; HOPPEN, J. Inteligência artificial em análise georreferenciada. **Aquarela**, 13 out. 2017. Disponível em: <https://www.aquare.la/inteligencia-artificial-em-analises-georreferenciadas/>. Acesso em: 20 set. 2019.

MARTINELLI, M. Cartografia do Turismo e Imaginário. In: RODRIGUES, A. B. (org). **Turismo rural**: práticas e perspectivas. São Paulo: Contexto, 2011, p. 151-170.

MATIAS, L. F. Geotecnologias e patrimônio arquitetônico: potencialidades no mapeamento e análise para fins turísticos. In: PAES, M. T. D.; OLIVEIRA, M. R. da S. (org). **Geografia, Turismo e Patrimônio Cultural**. São Paulo: Annablume, 2010, p. 81-111.

RAMOS, C. da S. **Visualização cartográfica e cartografia multimídia:** conceitos e tecnologias. São Paulo: UNESP, 2005.

SUMARES, G. Tudo sobre os mapas digitais do Facebook, Microsoft e outras empresas. **Olhar digital** [online], 21 set. 2016. Disponível em: <https://olhardigital.com.br/carros-e-tecnologia/noticia/saiba-como-sao-feitos-os-mapas-digitais/62365>. Acesso em: 20 set. 2019.

COMO CITAR
BARCELOS, Janinne; SILVEIRA, Lucas Angelo da; MOURA, Rebeca dos Santos de.Conceitos gerais sobre o Visão. In: SHINTAKU, Milton (Org.). **Mapa digital para gestão do conhecimento**: a construção de um sistema com o software Visão. Brasília: Ibict, 2019. p. 10-15. DOI: 10.18225/9788570131638.cap1

2. Características gerais do produto

Janinne Barcelos

Lucas Angelo da Silveira

Rebeca dos Santos de Moura

Sumariamente, o Visão é uma ferramenta digital para disponibilização, visualização e manipulação de dados baseados em localização geográfica com o objetivo de dar suporte à tomada de decisão e facilitar a realização de pesquisas técnico-científicas.

2.1 ESPECIFICAÇÃO DO TIPO DE DADO A SER ARMAZENADO

Segundo o Decreto n° 6.666, de 27 de novembro de 2008, que institui o INDE, é considerado um dado ou uma informação geoespacial aquele que "se distingue essencialmente pela componente espacial, que associa a cada entidade ou fenômeno uma localização na Terra, traduzida por sistema geodésico de referência, em dado instante ou período de tempo". O decreto diz, ainda, que dados estatísticos podem ser considerados dados geoespaciais, a critério do órgão produtor, se estiverem de acordo com a definição da norma. Isto é, se o dado estiver associado a uma posição sob a superfície terrestre, como, por exemplo, um dado vinculado a um município.

Segundo Palazzi e Tygel (2012), os dados estatísticos podem ser definidos como sequências de observações ao longo de uma ou mais dimensões (tempo, espaço etc.). Para Lacerda (2014), a estatística espacial é

um ramo da estatística que permite, por meio de técnicas descritivas e inferenciais, detectar e caracterizar a distribuição espacial dos dados, incorporando o espaço na análise. Assim, a estatística espacial promove um estudo quantitativo dos eventos com referência espacial, como acontece na análise de acidentes de trânsito que ocorrem em determinado lugar, possibilitando assim uma análise georreferenciada.

Logo, os dados a serem inseridos no Visão devem ser dados estatísticos relevantes para um estudo de localização, como é o caso de dados censitários de população, IDH, saúde, segurança, trabalho e renda — ou seja, dados que podem ser caracterizados como estatísticos e georreferenciados. Mas também podem ser integrados ao banco de dados, informações puramente geográficas (com apenas longitudes e latitudes), isto é, que apontam para um ponto fixo no espaço terrestre, como é o caso de desastres naturais, localização de escolas ou hospitais.

O georreferenciamento de um dado com endereço é definido como sendo o processo de associação desse dado a um mapa e pode ser efetuado de três formas básicas: associação a um ponto, a uma linha ou a uma área. Segundo Barcellos (2008), o resultado desse processo é a criação de elementos gráficos que podem ser usados para a análise espacial.

O Visão utiliza ***GeoJSON***[2], um formato de intercâmbio de dados geoespaciais baseado em ***JSON***[3], para georreferenciar dados. Esse formato define vários tipos de objetos ***JSON*** e a maneira como eles são combinados para representar dados sobre recursos geográficos, suas propriedades e suas extensões espaciais. Assim, estados, mesorregiões, municípios e localizações podem ser georreferenciados no software.

2.2 O MAPA

O Visão aproveita vários recursos de mapas interativos da biblioteca ***JavaScript Leaflet***, em especial, os mapas coropléticos e as camadas de marcadores com associação de *pop-ups*, além das funções básicas disponibilizadas pela biblioteca. Assim, o software oferece, entre suas

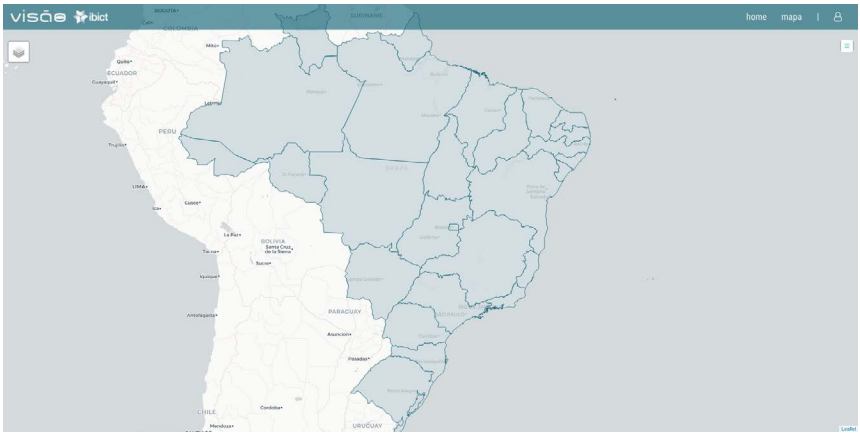
2 Disponível em: <https://tools.ietf.org/html/rfc7946>. Acesso em: 20 set. 2019.

3 JSON, acrônimo de JavaScript Object Notation, é um formato compacto, de padrão aberto para troca de dados simples e rápida entre sistemas.

diversas formas de interação, a escolha da base de mapas e ferramentas no menu lateral que permitem a inclusão de indicadores, a aplicação de filtros geográficos e a apresentação de marcadores no mapa.

O Mapa em branco, sem apresentação de nenhum dado, está ilustrado na Figura 2.1. A camada de cor ciano claro, que representa o Brasil dividido em estados, é área de trabalho do Visão, em que serão exibidos os indicadores, filtros e camadas.

Figura 2.1 - Mapa em branco



Fonte: Captura de tela (2019).

As informações de estados, mesorregiões e municípios foram retiradas da ***API (Application Programming Interface)*** de localidade do IBGE[4], enquanto os dados de exemplo presentes na versão beta do software foram retirados do catálogo de geoserviços do IBGE[5].

2.2.1 MAPAS COROPLÉTICOS

Segundo Archela (2008), mapa coroplético é um tipo de mapa temático que permite a visualização de qualquer tipo de dado de um determinado território a partir de diferentes cores ou tonalidades. Ele é

4 Disponível em: <https://servicodados.ibge.gov.br/api/docs/localidades?versao=1>. Acesso em: 20 set. 2019.

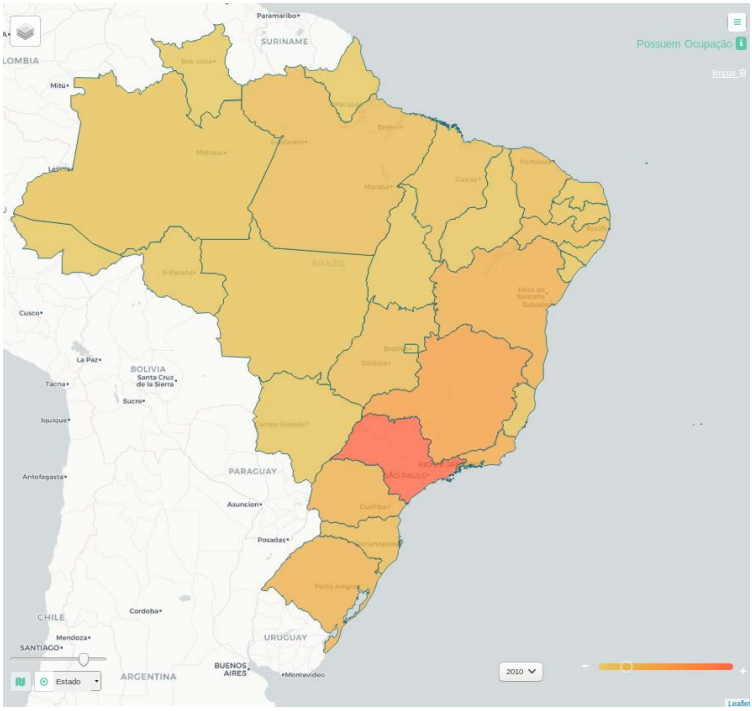
5 Disponível em: <https://geoservicos.ibge.gov.br/>. Acesso em: 20 set. 2019.

indicado para expor distribuição de densidades (habitantes por quilômetro quadrado), de rendimentos (toneladas por hectare) e de índices expressos em percentagens, que refletem a variação da densidade de um fenômeno (taxa de natalidade) ou, ainda, outros valores que são relacionados a mais de um elemento.

Os símbolos criados nos mapas coincidem com as regiões onde foram coletados os dados, o que dá a impressão de que há uniformidade de dados dentro de cada uma das regiões e que as quebras ocorrem sempre nos limites das áreas.

Ao escolher um indicador no menu lateral no Visão, o Mapa reflete a magnitude dos valores numa escala de cores e se torna um mapa coroplético (Figura 2.2).

Figura 2.2 - Mapa coroplético



Fonte: Captura de tela (2019)

2.2.2 Bases de mapas

No Visão, as bases de mapas são constituídas por duas camadas principais: uma de mapa e outra de rótulo. Independentemente da base, a camada do Brasil (área de trabalho) é exibida em primeiro plano.

São usados quatro serviços de mapa que possuem *API* gratuita para fornecimento das camadas:

- Base ***Gaode Maps*** fornecida pelo ***AutoNavi***[6];
- Base ***Mapbox Satellite*** fornecida pela ***Mapbox***[7];
- Base ***ChinaOnlineStreetPurplishBlue*** fornecido pela ***ArcGis***[8];
- Bases ***light_nolabels***, ***light_only_labels***, ***dark_nolabels*** e ***dark_only_labels***, fornecidas pela ***CARTO***[9].

As camadas de rótulos possuem nomes de países, estados e cidades, que são mostrados conforme o *zoom* no mapa.

Assim, o *software* permite a escolha entre cinco diferentes bases de mapas (Figura 2.3), contudo, por padrão, o mapa claro é exibido (visualizações detalhadas nos Quadros 2.1-2.5).

Figura 2.3 - Bases de mapas disponíveis no Visão.

Fonte: Captura de tela (2019)

6 Disponível em: <https://lbs.amap.com/api/webservice/summary/>. Acesso em 20 set. 2019.

7 Disponível em: <https://docs.mapbox.com/help/how-mapbox-works/satellite-imagery/>. Acesso em 20 set. 2019.

8 Disponível em: <https://map.geoq.cn/ArcGIS/rest/services/ChinaOnlineStreetPurplishBlue/MapServer>. Acesso em 20 set. 2019.

9 Disponível em: <https://github.com/CartoDB/cartodb/wiki/BaseMaps-available>. Acesso em 20 set. 2019.

Quadro 2.1 - Base de mapa Gao De Image

	Nome	Gao De Image
	Descrição	Mapa com imagens de satélite e apresentação dos nomes de países e cidades
	Camada de mapa	Gaode Maps
	Camada de rótulo	light_only_labels

Fonte: Elaboração dos autores (2019)

Quadro 2.2 - Base de mapa Mapa tridimensional

	Nome	Mapa tridimensional
	Descrição	Mapa com imagens de satélite sem nuvens e sem apresentação de nomes
	Camada de mapa	Mapbox Satellite
	Camada de rótulo	Não apresenta camada de rótulo

Fonte: Elaboração dos autores (2019)

Quadro 2.3 - Base de mapa GeoQ

	Nome	GeoQ
	Descrição	Mapa com fundo na cor azul escuro que mostra as fronteiras políticas com países vizinhos e apresenta os nomes de países e capitais em chinês
	Camada de mapa	ChinaOnlineStreet PurplishBlue
	Camada de rótulo	ChinaOnlineStreet PurplishBlue

Fonte: Elaboração dos autores (2019)

Quadro 2.4 - Base de mapa Mapa escuro

	Nome	Mapa escuro
	Descrição	Mapa com fundo na cor preta que mostra as fronteiras políticas com países vizinhos e apresenta os nomes de países e capitais
	Camada de mapa	dark_nolabels
	Camada de rótulo	dark_only_labels

Fonte: Elaboração dos autores (2019)

Quadro 2.5 - Base de mapa Mapa claro

	Nome	Mapa claro
	Descrição	Mapa com fundo na cor branca que mostra as fronteiras políticas com países vizinhos e apresenta os nomes de países e capitais
	Camada de mapa	light_nolabels
	Camada de rótulo	light_only_labels

Fonte: Elaboração dos autores (2019)

2.3 Menu lateral

No menu lateral é possível escolher Indicadores, Filtros e Camadas (Figura 2.4). O Indicador escolhido é ressaltado no mapa após clicar em "Aplicar". Os filtros selecionados realizam a filtragem do indicador escolhido por região (não é possível aplicar o filtro sem um indicador), sendo que eles não estão associados ao indicador em si. As camadas são marcadas com um ícone no mapa, de modo que elas não dependem de indicadores ou filtros para aparecer no mapa.

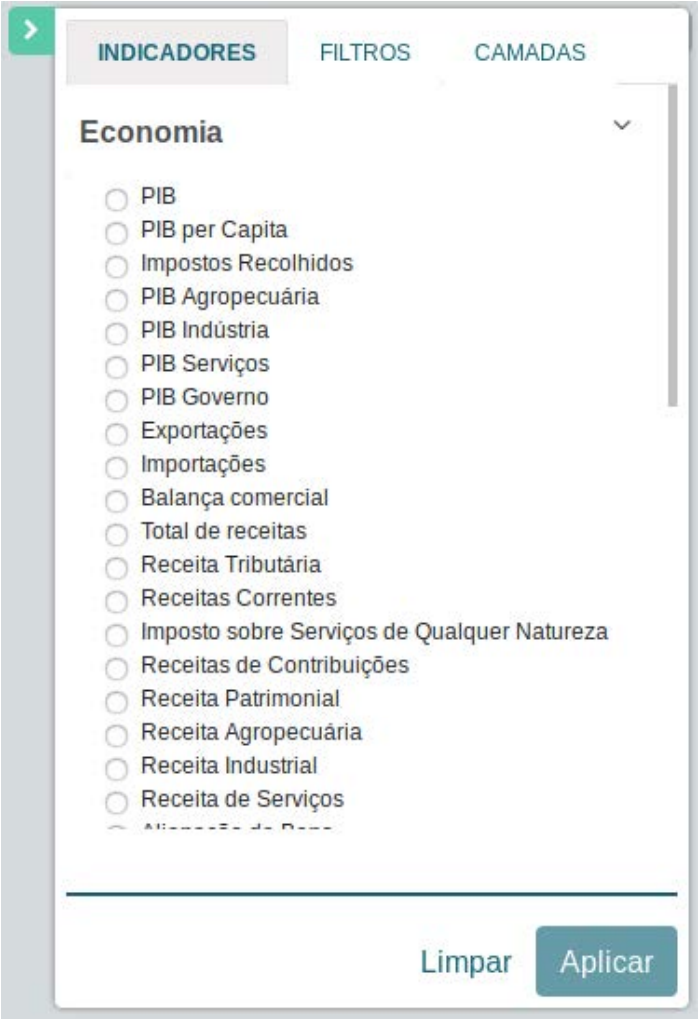
Figura 2.4 - Menu lateral

Fonte: Captura de tela (2019)

2.3.1 Indicadores

A primeira aba do menu lateral (Figura 2.5) apresenta uma lista com os Indicadores disponíveis na aplicação.

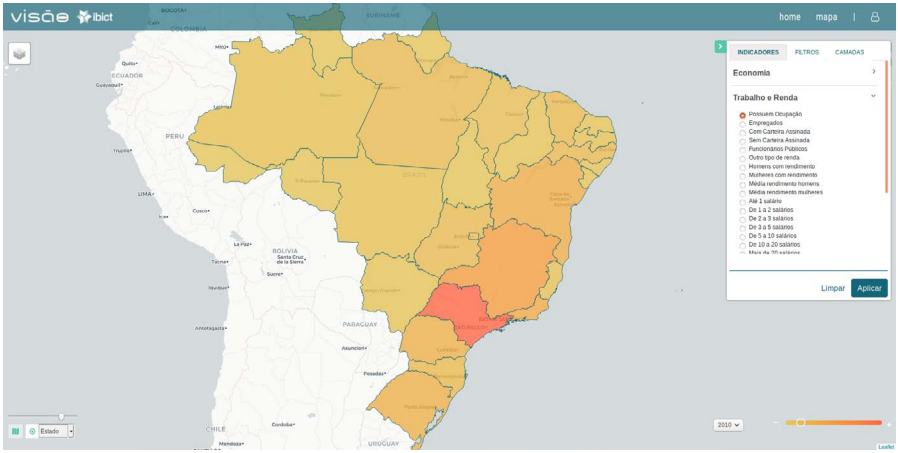
Figura 2.5 - Aba de Indicadores no menu lateral



Fonte: Captura de tela (2019)

Ao aplicar um Indicador, o mapa reflete a magnitude dos valores numa escala de cores e se torna um mapa coroplético (Figura 2.6).

Figura 2.6 - Mapa com indicador aplicado

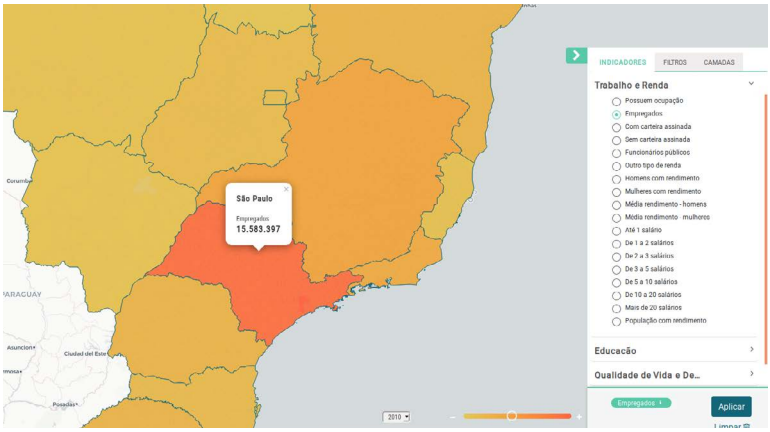


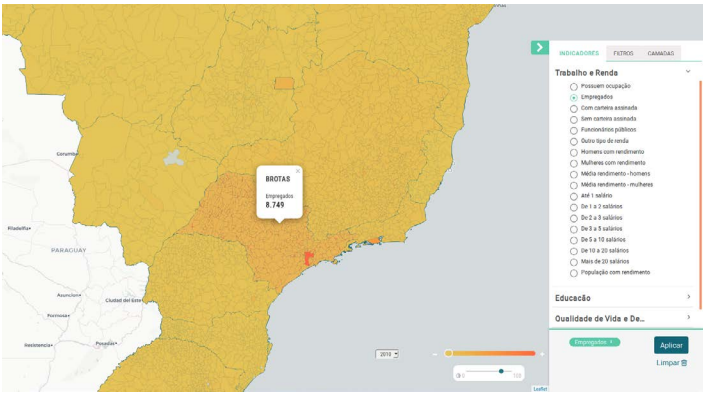
Fonte: Captura de tela (2019)

A partir do indicador escolhido, é possível escolher o nível da informação mostrada no mapa: estadual ou municipal. Por exemplo, o Indicador "Trabalho e Renda > Com emprego" permite a exibição do valor para o estado de São Paulo (Figura 2.7a) ou para municípios (Figura 2.7b)

Figura 2.7 - Exemplo do Indicador "Trabalho e Renda > Com emprego"

a.

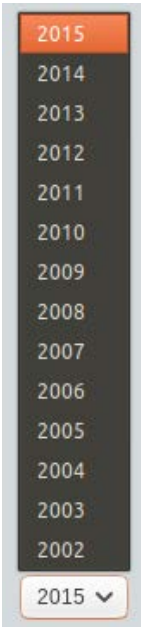


b. 

Fonte: Captura de tela (2019)

Também é possível escolher o ano de interesse da informação (se o indicador permitir) e a opacidade das cores no mapa. As ferramentas para a escolha de anos, opacidade e nível estadual/municipal estão mostradas na Figura 2.8.

Figura 2.8 - Seleção de anos (a) e opacidade dos dados (b).

a. 

b. 

Fonte: Captura de tela (2019)

2.3.2 Filtros geográficos

Na segunda aba do menu lateral "Filtros", é possível realizar um recorte geográfico que pode ser personalizado em estado, município e mesorregião. Essa aba está vinculada à de indicadores e só pode ser ativada se houver um Indicador aplicado. Importa ressaltar que não é possível aplicar um filtro sobre o Indicador em si, mas apenas o recorte sobre a região associada a ele.

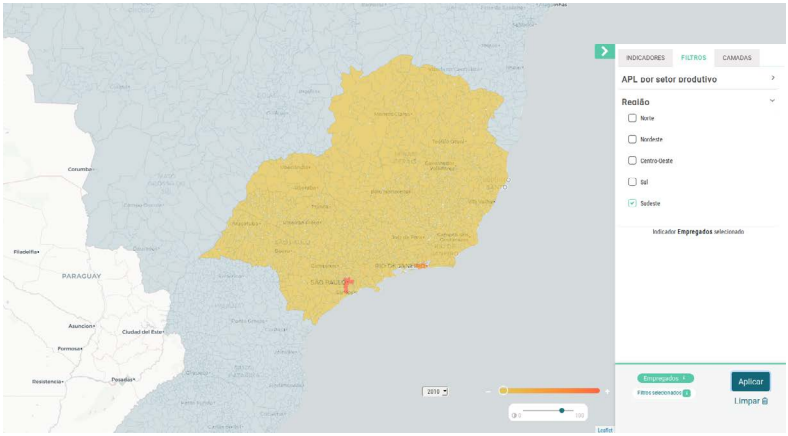
Atualmente, o Visão permite o recorte por grandes regiões brasileiras: Norte, Nordeste, Centro-Oeste, Sul e Sudeste (Figura 2.9).

Figura 2.9 - Aba de Filtros no menu lateral

Fonte: Captura de tela (2019)

Um exemplo de aplicação do filtro "Região > Sudeste" no Indicador "Trabalho e Renda > Empregados" é exibido na Figura 2.10. Pode-se ver que apenas a região escolhida é ressaltada no mapa, gerando um novo mapa coroplético com uma escala específica.

Figura 2.10 - Filtro Região > Sudeste aplicado ao Indicador "Trabalho e Renda > Com emprego"

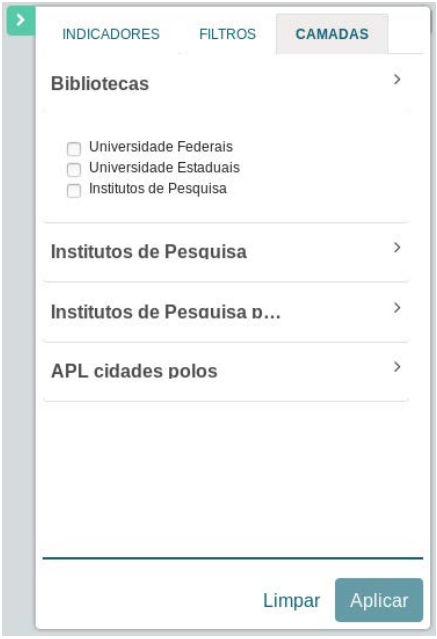


Fonte: Captura de tela (2019)

2.3.3 Camadas de marcadores

As camadas de marcadores não dependem de Indicadores ou Filtros e podem ser aplicadas diretamente ao mapa, por meio da terceira aba do menu lateral, de título Camadas (Figura 2.11). Os marcadores apontam para um ponto específico de coordenadas (latitude e longitude).

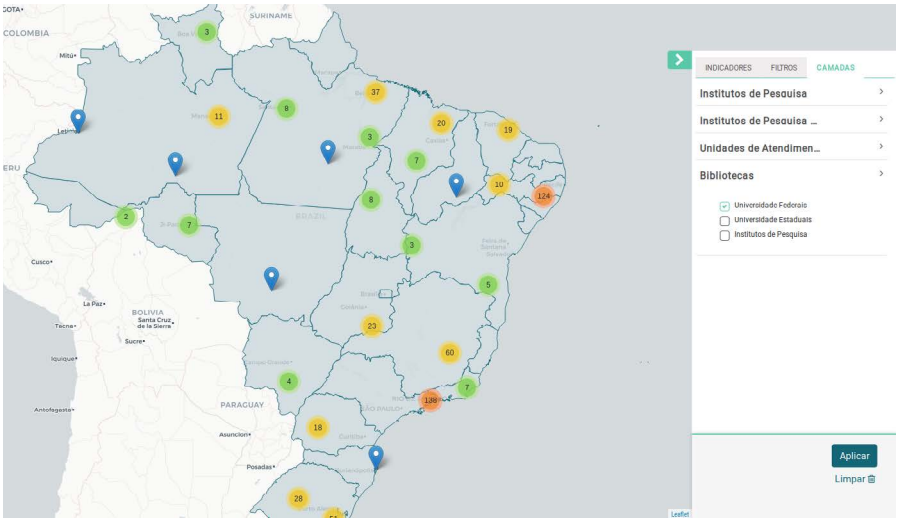
Figura 2.11 - Aba de Camadas no menu lateral



Fonte: Captura de tela (2019)

Os marcadores podem ser agrupados em *clusters* por um *plugin*[10] da biblioteca Leaflet. Cada *cluster* indica a quantidade de marcadores existentes naquela região do mapa. A área que o *cluster* abrange é realçada ao passar o *mouse* sobre ela. Ao clicar dentro de uma dessas áreas, o *zoom* é aumentado e o mapa passa a mostrar o próximo nível de *clusters*. Quando um *cluster* tem apenas um item, ele é mostrado como um marcador. Os *clusters* gerados podem ter diferentes cores. Por padrão, o mínimo de marcadores necessários para clusterização são dois e sua cor é verde. A partir de 10 marcadores, a cor do *cluster* passar a ser amarela. Por fim, a partir de 100, a cor é laranja. Um exemplo de marcadores e *clusters* aparece na Figura 2.12.

Figura 2.12 - Marcadores e clusters no Visão

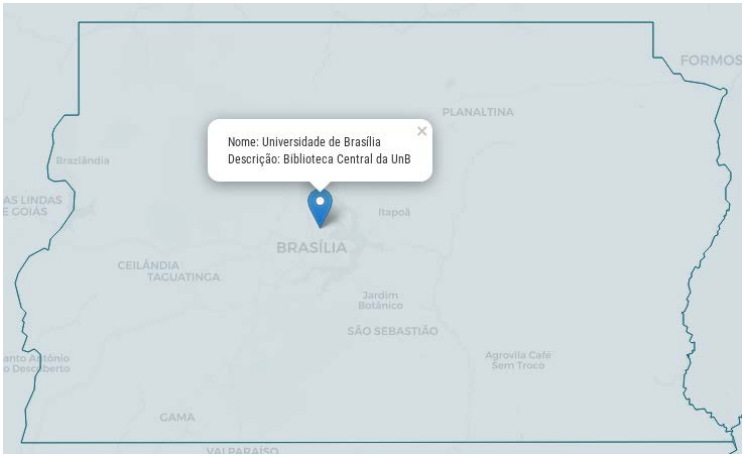


Fonte: Captura de tela (2019)

Cada marcador está associado a uma pop-up, que é uma pequena janela auxiliar que aparece sob o mapa, contendo mais informações sobre o marcador. Por padrão, são mostrados os dados de Nome e Descrição (inseridos durante a criação) do marcador (Figura 2.13).

10 Disponível em: < https://github.com/Leaflet/Leaflet.markercluster>. Acesso em: 20 set. 2019.

Figura 2.13 - Marcador e pop-up no Visão



Fonte: Captura de tela (2019)

Ao se utilizar dessas ferramentas, o sistema se comunica com o usuário, de forma a garantir maior interação do que outros sistemas de mapas digitais, ressaltando a parte estética do software, que torna a interface agradável para quem a utiliza.

Referências

ARCHELA, R. S.; THÉRY, H. Orientação metodológica para construção e leitura de mapas temáticos. **Confins** [Online], n. 3, jun. 2008. Disponível em: <http://journals.openedition.org/confins/3483>. Acesso em : 20 set. 2019. DOI : 10.4000/confins.3483.

BARCELLOS, C. de C. et al. Georreferenciamento de dados de saúde na escala submunicipal: algumas experiências no Brasil. **Epidemiol. Serv. Saúde**, Brasília, v. 17, n.1, p. 59-70, jan./mar. 2008. Disponível em: <https://www.arca.fiocruz.br/handle/icict/1290>. Acesso em: 20 set. 2019.

BRASIL. Decreto nº 6.666, de 27 de novembro de 2008. Institui, no âmbito do Poder Executivo federal, a Infra-Estrutura Nacional de Dados Espaciais - INDE, e dá outras providências. **Diário Oficial da União**, Brasília, 28 nov. 2008. Disponível em: <http://www.planalto.gov.br/ccivil_03/_Ato2007-2010/2008/Decreto/D6666.html>. Acesso em: 20 set. 2019

LACERDA, C. J. **Análise de dados georreferenciados para obter a distribuição estatística espacial das vítimas fatais em acidentes de trânsito em Goiânia**. 2014. 58 f. Dissertação (Mestrado em Engenharia) - Pontifícia Universidade Católica de Goiás, GOI NIA, 2014. Disponível em: <http://localhost:8080/tede/handle/tede/2456>. Acesso em 28 de março de 2018.

PALAZZI, D.; TYGEL, A. Visualização de dados estatísticos representados como dados abertos ligados. **Relatório final da disciplina de Visualização de Informações**, trimestre 2012/3. Disponível em: < http://cirandas.net/alantygel/academico/relatorio-visualizacao-de-dados--estatisticos-representados-como-dados-abertos-ligados.pdf>. Acesso em 28 de março de 2018.

COMO CITAR
BARCELOS, Janinne; SILVEIRA, Lucas Angelo da; MOURA, Rebeca dos Santos de. Características gerais do produto. In: SHINTAKU, Milton (Org.). **Mapa digital para gestão do conhecimento**: a construção de um sistema com o software Visão. Brasília: Ibict, 2019. p. 16-32. DOI: 10.18225/9788570131638.cap2

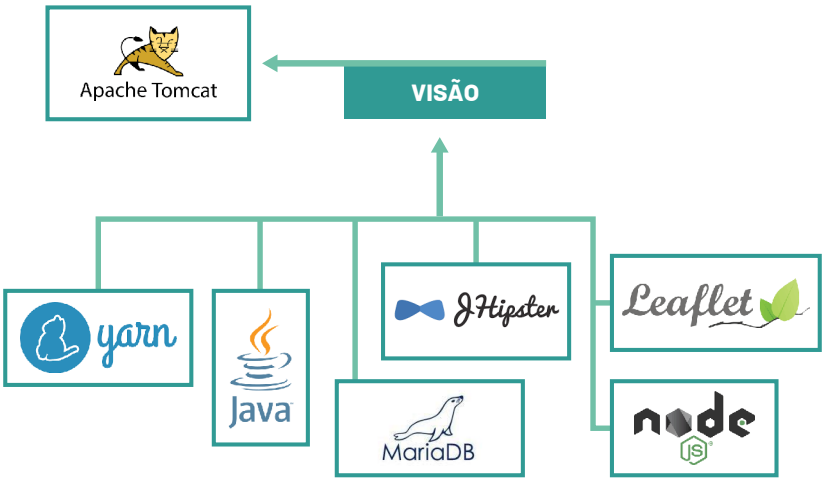
3. Instalação do Visão

Lucas Angelo da Silveira

Rebeca dos Santos de Moura

O Visão é uma aplicação web desenvolvida em Java por meio de *frameworks* livres e de código aberto (*open source*). A utilização de *frameworks* em projetos de desenvolvimento tem se tornado cada vez mais rotineira, uma vez que tais ferramentas agilizam o processo de desenvolvimento da aplicação e facilitam a contribuição por parte dos desenvolvedores. A Figura 3.1 apresenta as principais tecnologias envolvidas no projeto. A seguir, cada uma das tecnologias é discutida.

Figura 3.1- Principais tecnologias utilizadas no projeto do Visão.



Fonte: Elaboração dos autores (2019)

3.1 Apache Tomcat

O software ***Tomcat***, desenvolvido pela Fundação Apache, é um container de código aberto lançado no fim da década de 1990 para aplicações desenvolvidas em Java, que utiliza tecnologias *Servlets* e JSPs (*Java Server Pages*). Ele foi criado inicialmente como uma ramificação do Apache-Jakarta, porém, devido a sua rápida popularidade, ganhou espaço e se tornou um projeto independente na empresa, apoiado por um grupo de colaboradores pertencentes à comunidade de código aberto do Java.

Diferentes versões do ***Tomcat*** estão disponíveis para muitas versões das especificações de Servlets e JSP. O mapeamento entre as especificações e as respectivas versões do ***Apache Tomcat*** é especificado na Quadro 3.1, onde cada versão do ***Tomcat*** é suportada por qualquer versão estável do Java que atenda aos requisitos da coluna final da tabela.

Quadro 3.1 - Versões do Tomcat

Servlet	JSP	Versão do Tomcat	Última versão lançada	Java suportado
4.0	2.3	9.0.x	9.0.24	8 e anteriores
3.1	2.3	8.5.x	8.5.43	7 e anteriores
3.1	2.3	8.0.x	8.0.53	7 e anteriores
3.0	2.2	7.0.x	7.0.96	6 anteriores
2.5	2.1	6.0.x	6.0.53	5 e anteriores
2.4	2.0	5.5.x	5.5.36	1.4 e anteriores
2.3	1.2	4.1.x	4.1.40	1.3 e anteriores
2.2	1.1	3.3.x	3.3.2	1.1 e anteriores

Fonte: Traduzido de APACHE TOMCAT[11] (2019, website)

Em ambiente de produção, é útil ter a capacidade de gerenciar os aplicativos web sem precisar desligar e reiniciar o ***Tomcat***. Para isso, o software

11 Disponível em: <http://tomcat.apache.org/whichversion.html>. Acesso em: 10 set. 2019.

oferece uma interface web para gerenciamento dividida em seis seções: *message* (mensagem), *manager* (gerenciador), *applications* (aplicações), *deploy* (implantar), *diagnostics* (diagnósticos) e *server information* (informação do servidor), que serão discutidas a seguir.

3.1.1 Message

Esta seção exibe informações sobre o sucesso ou a falha do último comando executado no software. Se for bem sucedido, um "OK" é exibido e pode ser seguido por uma mensagem de sucesso. Se falhar, será exibido "FAIL", seguido por uma mensagem de erro. A lista completa de mensagens de falha para cada comando pode ser encontrada na documentação oficial[12].

3.1.2 Manager

Esta seção apresenta operações do gerenciador geral, como listagens e ajuda. A seção *Manager* tem três links:

» *List Applications* (listar aplicações): mostra uma lista de aplicações disponíveis.
» *HTML Manager Help* (ajuda do gerenciador de HTML): link para um documento de ajuda.
» *Manager Help* (ajuda do gerenciador): link para o documento de ajuda sobre o *Manager App*.

3.1.3 Applications

Esta seção lista informações sobre todos os aplicativos web instalados e fornece links para gerenciá-los. Para cada aplicativo web, as seguintes informações são exibidas:

a. Path: o caminho da aplicação.
b. Display Name: o nome para exibição da aplicação, caso exista um nome configurado em seu arquivo "web.xml".

12 Disponível em: <http://tomcat.apache.org>. Acesso em: 10 set. 2019.

c. Running: se a aplicação está em execução e disponível (*true* ou verdadeiro) do contrário (*false* ou falso).
d. Sessions: o número de sessões ativas para usuários remotos da aplicação. O número de sessões é um link que exibe mais detalhes sobre o uso da sessão pela aplicação na caixa de mensagem.
e. Commands: lista todos os comandos que podem ser executados na aplicação, como: Start: para iniciar uma aplicação; Stop: para parar uma aplicação que esteja executando; e Reload: recarrega a aplicação, para que um arquivo ".jar" ou novas classes sejam incorporadas.

3.1.4 Deploy

Esta seção faz referência à implantação de aplicativos web. Os aplicativos podem ser implantados usando arquivos ou diretórios localizados no servidor do ***Tomcat*** ou o usuário pode enviar um arquivo *Web Application Archive* (WAR ou Arquivo de Aplicação Web) para o servidor. Para instalar uma aplicação, é necessário preencher os campos apropriados para o tipo de instalação que se deseja fazer e, em seguida, enviá-la usando o botão Instalar.

3.1.5 Diagnostics

Esta seção identifica os potenciais problemas que podem acometer as aplicações, em especial os vazamentos de memória das aplicações hospedadas no ***Tomcat***. O diagnóstico de localização de vazamento aciona uma coleta de lixo completa e, consequentemente, deve ser usado com extrema cautela em sistemas de produção. O serviço identifica as aplicações que apresentaram vazamentos de memória quando foram interrompidas ou recarregadas. Os resultados devem sempre ser confirmados com um perfilador. O diagnóstico usa funcionalidade adicional fornecida pela implementação do StandardHost[13].

O diagnóstico lista os caminhos para as aplicações que foram paradas ou recarregadas e quais classes das execuções anteriores ainda estão presentes na memória, que são, portanto, um vazamento de memória.

13 Implementação padrão da interface do hospedeiro (host).

Deve-se atentar que, caso uma aplicação tenha sido recarregada várias vezes, ela poderá ser listada várias vezes no diagnóstico.

3.1.6 Server Information

Esta seção exibe informações sobre o ***Tomcat***, sobre o sistema operacional do servidor no qual o software está hospedado e sobre o Tomcat sendo executado, além do nome do host principal do servidor (pode não ser o nome do host usado para acessar o ***Tomcat***) e do endereço IP principal do servidor (pode não ser o endereço IP usado para acessar o ***Tomcat***).

3.2 MariaDB

O ***MariaDB Server***[14] é um dos sistemas de gerenciamento de banco de dados (SGBD) mais populares do mundo. Atualmente na versão 10.3.13, lançada em fevereiro de 2019, o projeto foi iniciado pelos desenvolvedores originais do ***MySQL***[15] em 2009, com o propósito de ser exclusivamente *open source*. Dentre o vasto público que aderiu ao gerenciador, destacam-se grandes corporações, tais como Wikipédia, WordPress e Google.

O ***MariaDB***, em suma, é um substituto aprimorado para o ***MySQL*** que obteve uma ascensão vertiginosa no mercado de tecnologia por ser rápido, escalável e robusto. Além disso, apresenta um rico ecossistema de mecanismos de armazenamento, *plugins* e outras ferramentas que o tornaram muito mais versátil.

Com relação à arquitetura, o gerenciador foi desenvolvido e disponibilizado de forma *open source*, isto é, seu código-fonte é aberto e de livre acesso. Seu modelo de banco de dados é relacional e fornece uma interface SQL (*Structured Query Language*)[16] para criar, acessar e modificar dados. As versões mais recentes do ***MariaDB*** incluem suporte a recursos SIG e ***JSON***.

---

14 Disponível em: <https://mariadb.org/>. Acesso em: 10 set. 2019.

15 O MySQL é um SGBD que apresenta a opção de duas licenças de uso: a Licença Pública Geral GNU ou a licença comercial padrão da Oracle.

16 SQL (Structured Query Language), ou Linguagem de Consulta Estruturada é a linguagem de pesquisa declarativa padrão para banco de dados relacional.

As semelhanças entre o ***MariaDB*** e o ***MySQL*** são muito patentes, dado que o ***MariaDB*** é baseado no ***MySQL*** de código aberto. Um nada mais é que uma ramificação do código-fonte do outro, de forma que os desenvolvedores usaram o código do ***MySQL*** livremente disponível e agregaram funcionalidades a ele. Conforme mudanças são feitas ou novos recursos são adicionados ao ***MySQL***, os mantenedores do ***MariaDB*** revisam e incorporam as novidades.

Os usuários do ***MariaDB*** provavelmente notam que, embora os nomes dos pacotes disponíveis sejam diferentes, os nomes dos arquivos e binários de instalação e a localização padrão de cada um deles são os mesmos que o ***MySQL***. Além disso, é perceptível que a linguagem *SQL* e os comandos são os mesmos entre ambos os gerenciadores. Até os arquivos de configuração são parecidos, com poucas peculiaridades específicas do ***MariaDB***.

Há tantas semelhanças entre ***MySQL*** e ***MariaDB*** que o usuário pode se perguntar "por que o ***MariaDB*** existe?" ou "o que o ***MariaDB*** fornece que o ***MySQL*** não oferece?". Como resposta às perguntas, o ***MariaDB*** tem novos recursos, melhorias de desempenho, melhores testes e correções de *bugs* que não são encontrados no ***MySQL***. Alguns deles foram desenvolvidos internamente pelos desenvolvedores do ***MariaDB***, enquanto outros vêm de colaboradores externos e empresas grandes como Facebook, Twitter e Google.

Um dos principais objetivos dos desenvolvedores do ***MariaDB*** é melhorar continuamente o seu desempenho, assim, vários aprimoramentos do software são relacionados ao tópico. Uma área de concentração é o otimizador localizado no núcleo do ***MariaDB*** e, portanto, do ***MySQL***, que tem como tarefa é transformar os comandos SQL recebidos em instruções para o banco de dados. Em cargas de trabalho complexos, o otimizador aprimorado do ***MariaDB*** desempenha essa função de maneira significativamente mais rápida que o ***MySQL***.

A tabela *Elimination* é outra área de concentração. A otimização dessa tabela é especialmente útil ao usar *views* (visualizações no banco de dados) para acessar dados altamente normalizados. A ideia básica é se aproveitar da possibilidade de resolver consultas sem acessar algumas

das tabelas às quais a consulta se refere, no intuito de diminuir o número de acesso a tabelas e, consequentemente, concluir mais rápido a consulta.

Um componente-chave para melhorar o código é o escopo dos testes de validação do software. Para tanto, os desenvolvedores do ***MariaDB*** realizaram grandes melhorias na infraestrutura de testes do gerenciador, que incluem:

» Maior conjunto de testes: dezenas de novos testes foram adicionados, tanto para avaliar *bugs* recém-descobertos quanto para fornecer uma maior cobertura do código-fonte;
» Correções de erros no conjunto de testes: os desenvolvedores testam tanto o conjunto de testes quanto outras partes do ***MariaDB***;
» Testes construídos com diferentes opções de configuração e para várias combinações de sistema operacional e processador, para obter melhores avaliações de recursos;
» Remoção de testes inválidos.

Todo esforço é feito para resolver o maior número possível de erros em todas as versões lançadas do ***MariaDB***, além de não introduzir novos erros. Tanto a infraestrutura avançada de testes quanto o profundo conhecimento e experiência dos desenvolvedores contribui nesse aspecto.

Um exemplo são os avisos de compilador que indicam alguma desconformidade no software, mas não impedem seu funcionamento. Alguns profissionais da área não concordam, mas os desenvolvedores do ***MariaDB*** se preocupam com esses avisos, visto que são quase tão ruins quanto os *bugs* e tentam consertar o maior número possível deles. Como resultado, houve uma grande redução no número total de erros e avisos de compiladores no software. Para saber mais o leitor é convidado a consultar a documentação oficial do ***MariaDB***[17].

17 Disponível em: <https://mariadb.com/kb/en/library/documentation>. Acesso em: 10 set. 2019.

3.3 Node.js

Node.js é uma plataforma criada em *JavaScript* por Ryan Dahl em maio de 2009, com objetivo de oferecer recursos para a criação de aplicações rápidas e dimensionáveis capazes de operar independentemente de hardware ou software adicionais. O sucesso do ***Node.js*** produziu uma infinidade de desenvolvedores e empresas de consultoria com experiência na implementação de aplicativos utilizando sua tecnologia.

Para isso, o ***Node.js*** emprega um modelo de Entrada/Saída sem encadeamento, orientado a eventos, assíncrono e sem bloqueio. Ao contrário de abordagens mais comuns, baseadas na geração de fios de execução (*threads*) do sistema operacional por solicitação e em um laço de eventos que bloqueia no início da operação, o ***Node.js*** nunca executa operações de Entrada/Saída inerentemente, o que não ocasiona o bloqueio do processo.

Node.js é uma tecnologia de código aberto que pode ser usada livremente sob a licença do MIT (*Massachusetts Institute of Technology*), ambiguamente conhecida como a licença do Expat, que essencialmente permite o uso livre do software por conta e risco do usuário. O ***Node.js*** é protegido por direitos autorais e patrocinado pela *Joyent Inc.*[18], que oferece suporte comercial corporativo.

Como já mencionado, o ***Node.js*** usa um processo de Entrada/Saída sem bloqueio, de encadeamento único. Assim, essa tecnologia não é uma boa escolha para operações de processamento intensivo na unidade central de processamento ou CPU (*Central Processing Unit*), uma vez que os processos de computação pesada realizados em um "laço de eventos" podem impedir que o núcleo faça seu trabalho de manipular eventos. As operações intensivas de CPU desse tipo devem ser delegadas para uma plataforma mais adequada e, quando concluídas, elas simplesmente respondem ao processo do ***Node.js*** como um retorno de chamada (ou evento) de volta ao "laço de eventos".

O princípio geral é permitir que o processo do ***Node.js*** aceite solicitações e forneça respostas, o que é adequado para muitos aplicativos baseados

18 Joyent Inc. é uma empresa de software e serviços com sede em San Francisco, Califórnia, especializada em virtualização de aplicativos e computação em nuvem.

na web que simplesmente trocam dados entre cliente e servidor. O ***Node.js*** se destaca ao considerar *APIs* que usam *REST (Representational State Transfer)*[19] e aplicativos de página única, uma vez que esses tipos de aplicação exigem por respostas de baixa latência e alto rendimento.

Atualmente, há muitos casos de sucesso do uso dessa tecnologia, e como exemplo pode-se citar a *Walmart*. Em 2012, a empresa implementou o ***Node.js*** para tratar do tráfego móvel de suas aplicações[20]. Isso não só diminuiu drasticamente os tempos de resposta, como reduziu a infraestrutura necessária.

O uso do ***Node.js*** altera a maneira como os aplicativos web tradicionais são pensados. O modelo típico de aplicativo web é baseado na abordagem de requisição-resposta, com o cliente sempre iniciando a comunicação. Porém, um dos objetivos do criador do ***Node.js*** era criar sites em tempo real com capacidade de envio de requisições, o que quebra a visão tradicional de como os sites são estruturados e permite que a troca de informações seja iniciada de ambos os lados.

Isso proporciona a possibilidade de usar a mesma linguagem no cliente e no servidor, o que efetivamente elimina a parede divisória que existe entre os desenvolvedores *front-end* e *back-end*. Esse fato, por si só, pode resultar em maior eficiência ao longo da vida de um projeto ou aplicativo e, além disso, permite que um desenvolvedor experiente trabalhe desimpedido em vários ambientes e configurações. Por padrão, esse tipo de desenvolvimento pode reduzir os requisitos de pessoal de uma organização, permitindo que usem os mesmos recursos de maneira mais holística.

A segurança de aplicações é uma área de precaução legítima e que não pode ser tomada de ânimo leve. Independentemente da tecnologia utilizada, sempre haverá uma preocupação a ser gerenciada nessa área. Como outras tecnologias, o ***Node.js*** não está livre de invasores.

---

19 REST (Representational State Transfer), ou Transferência de Estado Representacional, é uma abstração da arquitetura da web, que consiste em um conjunto coordenado de restrições arquiteturais aplicadas a componentes, conectores e elementos de dados dentro de um sistema de hipermídia distribuído.

20 Disponível em: <https://venturebeat.com/2012/01/24/why-walmart-is-using-node-js/>. Acesso em: 10 set. 2019.

Os esforços criados e estabelecidos por iniciativas de segurança e pelo *OWASP* (*Open Web Application Security Project*)[21] ajudam a educar desenvolvedores para criar aplicativos confiáveis. Conhecimento do desenvolvedor, padrões e técnicas de integração contínua são partes essenciais do sucesso na área.

No entanto, existem desenvolvedores que escrevem códigos que se aproveitam de brechas no sistema e *bugs*, muitos dos quais relacionados à segurança. A boa notícia é que, com o considerável aumento nas implantações de produção de aplicativos usando novas tecnologias, esse não é um assunto ignorado. O ***Node.js*** está na vanguarda de muitas implantações em larga escala de algumas das maiores empresas da atualidade (*Walmart, LinkedIn, PayPal, etc.*), e isso beneficia a todos da comunidade.

Atualmente o ***Node.js*** encontra-se na versão 10.14.1, lançada em novembro de 2018. A legião de usuários está em expansão devido às perspectivas de mercado a longo prazo. A documentação de referência da *API*[22] fornece informações detalhadas sobre funções e objetos da plataforma.

3.4 Leaflet

Leaflet é uma biblioteca JavaScripit de código aberto usada para criar aplicativos de mapas interativos na web. Lançada em maio de 2011, atualmente ela se encontra na versão 1.5.1 e oferece suporte para a maioria das plataformas móveis e de desktops. Juntamente com a *OpenLayers* e a *API* do *Google Maps*, é uma das bibliotecas de mapas mais populares, sendo utilizada por sites famosos, como o *FourSquare, o Pinterest e o Flickr.*

Usando a ***Leaflet***, desenvolvedores e entusiastas de SIG podem criar facilmente mapas na web com uma vasta gama de funcionalidades, entre as quais estão as camadas prontas para uso, as ferramentas de interação, de performance e visuais e controles de mapas. Além disso, a

21 OWASP (Open Web Application Security Project), ou Projeto Aberto de Segurança em Aplicações Web, é uma comunidade on-line que cria e disponibiliza, de forma gratuita, artigos, metodologias, documentação, ferramentas e tecnologias no campo da segurança de aplicações web.

22 Disponível: <https://nodejs.org/api>. Acesso em: 10 set. 2019.

biblioteca proporciona a opção de carregar dados no formato *GeoJSON*, estilizá-los e criar, por exemplo, camadas interativas, com marcadores que mostram *pop-ups* quando clicados.

Um uso típico da ***Leaflet*** envolve vincular um elemento *map* da biblioteca a um elemento *div* do *HTML*[23]. Camadas e marcadores são então adicionados ao elemento de mapa. Um exemplo simples de criação de mapa é exibido no código a seguir:

```
// cria um mapa na div "map", define a visualização para um local e aplica o zoom
var map = L.map('map').setView([51.505, -0.09], 13);

 // adicionar uma camada de mapa do OpenStreetMap
 L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
    attribution: '&amp;copy; <a href="http://osm.org/copyright">OpenStreetMap</a>
contributors'
 }).addTo(map);
```

Nota: A instância da Leaflet em si é acessível por meio da variável L.

Com relação às camadas de mapas a serem utilizadas, a ***Leaflet*** suporta *Web Map Service* (WMS)[24], *GeoJSON* e gráficos vetoriais. Outros tipos de camadas são suportadas por meio de *plugins*. Como outras bibliotecas de mapas, o modelo básico de exibição implementado pela ***Leaflet*** é um mapa base, com zero ou mais sobreposições translúcidas e zero ou mais objetos vetoriais exibidos na parte superior. Além disso, a ***Leaflet*** trabalha com vários tipos de camadas, entre os quais, os principais são:

- Camadas de varredura (*TileLayer* e *ImageOverlay*)
- Camadas vetoriais (linhas, polígono e tipos específicos, como círculo)
- Camadas de agrupamento (*LayerGroup* e *FeatureGroup*)

23 HTML (Hypertext Markup Language), ou Hipertexto de Marcação de Linguagem é uma linguagem de marcação para a web.

24 Disponível em: <https://www.opengeospatial.org/standards/wms>. Acesso em: 10 set. 2019.

A biblioteca é frequentemente comparada com a *OpenLayers*, já que ambas são bibliotecas *JavaScript* de código aberto do lado do cliente. Porém, a ***Leaflet*** como um todo é muito menor, com cerca de 7 mil linhas de código em comparação com as 230 mil da *OpenLayers*, além de apresentar uma pegada de código menor. Por sua vez, a base de código é mais recente e aproveita os recursos mais novos do JavaScript, além de *HTML5*[25] e *CSS3*[26]. No entanto, a *OpenLayers* suporta mais recursos que a *Leaflet*, como o *Web Feature Service* (WFS)[27] e projeções diferentes do *Google Web Mercator*[28].

A Leaflet também é comparada à *API* de código fechado do *Google Maps*[29] e do *Bing Maps*[30], já que ambas *APIs* incorporam recursos significativos do servidor para fornecer serviços como geocodificação, roteamento, pesquisa e integração com outras ferramentas como o *Google Earth*. A *API* do *Google Maps* fornece velocidade e simplicidade, mas não é flexível e só pode ser usada para acessar os serviços do Google.

Vale ressaltar que a documentação da ***Leaflet***[31], além de intuitiva, tem potencial para ajudar os desenvolvedores a iniciar uma aplicação do zero até realizar eventuais customizações.

3.5 JHipster

O ***JHipster*** é um projeto de código aberto que combina três *frameworks*

---

25 HTML5 é a mais recente evolução do padrão que define HTML

26 CSS3 é a terceira mais nova versão do CSS (Cascading Style Sheets), que permite a definição de estilos para um projeto na web

27 Disponível em: <https://www.opengeospatial.org/standards/wfs>. Acesso em: 10 set. 2019.

28 Disponível em: <https://epsg.io/3857>. Acesso em: 10 set. 2019.

29 Disponível em: <https://developers.google.com/maps/documentation/>. Acesso em: 10 set. 2019.

30 Disponível em: <https://docs.microsoft.com/en-us/bingmaps/>. Acesso em: 10 set. 2019.

31 Disponível em: <https://leafletjs.com>. Acesso em: 10 set. 2019.

muito bem sucedidos no desenvolvimento web: *Bootstrap*[32], *Angular*[33] e *Spring Boot*[34]. Ele foi iniciado por Julien Dubois em outubro de 2013 e sua primeira versão pública (versão 0.3.1) foi lançada em 7 de dezembro de 2013. Desde então, já teve mais de 172 lançamentos. O projeto usa a licença Apache 2.0 e conta com uma equipe principal de 19 desenvolvedores e mais de 430 colaboradores.

Em suma, o ***JHipster*** fornece ferramentas para gerar, desenvolver e implementar uma aplicação web com uma pilha Java no lado do servidor (usando o *Spring Boot*) e um *front-end* responsivo no lado do cliente (com *Angular* e *Bootstrap*), podendo criar uma pilha de microsserviços com suporte para *Netflix OSS*, *Docker* e *Kubernetes*.

O termo "***JHipster***" vem de "*Java Hipster*", já que seu objetivo inicial era usar todas as ferramentas modernas e "da moda" (*hype*) disponíveis na época. Atualmente, o projeto alcançou um objetivo mais empresarial, com um forte foco na produtividade e qualidade do desenvolvedor, além de suas ferramentas.

As principais funcionalidades do ***JHipster*** são: geração de microsserviços e aplicações *full stack* (com software para ambos lados de cliente e servidor); geração de entidades *CRUD*[35]; migrações de banco de dados com o *Liquibase*[36]; suporte a bancos de dados *NoSQL*[37],

32 Bootstrap é um framework web de código aberto para desenvolvimento de componentes de interface e front-end para sites e aplicações web usando HTML, CSS e JavaScript.

33 Angular é uma plataforma de aplicações web de código aberto com front-end baseado em TypeScript liderado pela Equipe Angular do Google.

34 Spring Boot é a evolução do Spring, um framework open source para a plataforma Java baseado nos padrões de projeto de inversão de controle e injeção de dependência, facilitando a resolução das múltiplas configurações básicas necessárias.

35 CRUD são as quatro operações básicas utilizadas em bases de dados relacionais: CREATE, READ, UPDATE e DELETE.

36 Liquibase é uma biblioteca independente de banco de dados open source para rastreamento, gerenciamento e alterações no esquema do banco de dados.

37 NoSQL é uma classe de banco de dados que fornece um mecanismo para armazenamento e recuperação de dados modelados de formas diferentes das relações tabulares usadas nos bancos de dados relacionais.

como *Cassandra*[38] e *MongoDB*[39]; suporte a *Elasticsearch*[40]; suporte a *Websockets*[41]; e implantação automática para *Cloud Foundry*[42], *Heroku*[43], *OpenShift*[44] e *Amazon Web Services (AWS)*[45].

No lado do cliente, além do Bootstrap, o sistema usa as tecnologias: *HTML5*; *Angular JS*[46]; *Angular 2+*[47]; *React*[48]; completo suporte de internacionalização com o *Angular Translate*; e suporte opcional ao *Sass* para design de CSS.

Já no lado do servidor, além de recursos *Spring* como *Security*, *Spring MVC*[49] *REST + Jackson e Spring Data JPA*[50] *+ Bean Validation*[51], as seguintes

38 Cassandra é um projeto de sistema de banco de dados distribuído altamente escalável de segunda geração, que reúne a arquitetura do DynamoDB, da Amazon Web Services e modelo de dados baseado no BigTable, do Google.

39 MongoDB é um software de banco de dados orientado a documentos livre, de código aberto e multiplataforma que usa documentos semelhantes a JSON com esquemas.

40 Elasticsearch é um servidor de buscas distribuído open source baseado no Apache Lucene.

41 WebSocket é uma tecnologia que permite a comunicação bidirecional por canais full duplex sobre um único soquete TCP (Transmission Control Protocol, ou protocolo de controle de transmissão).

42 O Cloud Foundry é uma plataforma de aplicativos de nuvem open-source.

43 O Heroku é uma plataforma em nuvem como um serviço que suporta várias linguagens de programação.

44 OpenShift é um produto de software da Red Hat para implantação e gerenciamento de softwares baseados em container.

45 Amazon Web Services (AWS), ou Serviços Web da Amazon, é uma plataforma de serviços de computação em nuvem.

46 AngularJS é um framework JavaScript open source, mantido pelo Google, que auxilia na execução de aplicações de página única.

47 Angular2+ é a versão atualizada do Angular construída do zero.

48 React é uma biblioteca JavaScript open source para criar interfaces de usuário.

49 Spring MVC é um framework que ajuda no desenvolvimento de aplicações Web utilizando os conceitos sobre Model, View e Controller.

50 JPA é uma API padrão da linguagem Java que descreve uma interface comum para frameworks de persistência de dados definindo um meio de mapeamento objeto-relacional para objetos Java simples e comuns, denominados beans de entidade.

51 Bean Validation é uma especificação Java que permite expressar restrições em modelos de objetos por meio de anotações, permitindo escrever restrições personalizadas

tecnologias são usadas: monitoramento de métricas; suporte *WebSocket* opcional com *Spring Websocket*; atualizações de banco de dados com o *Liquibase*; e suporte a *Elasticsearch*, *MongoDB* e *Cassandra*.

O ***JHipster*** também usa algumas ferramentas automáticas configuradas e prontas para uso: *Yeoman*[52]; *Webpack*[53]; *BrowserSync*[54]; *Maven*[55] ou *Gradle*[56]; e editor visual e textual de *Datamodeling* (modelagem de dados).

A documentação do ***JHipster*** está no site oficial[57] e seu código-fonte está escrito em *JavaScript* (33%), *Java* (27%), *TypeScript* (25%) e *HTML* (12%).

3.6 Yarn

O ***Yarn*** foi anunciado pelo Facebook com a colaboração das empresas *Exponent*, *Google* e *Tilde* em outubro de 2016, com a proposta de ser um novo gerenciador de código aberto para pacotes *Javascript* que fosse mais rápido, seguro e confiável que o npm[58], o consagrado gerenciador de pacotes do ***Node.js***, perante a comunidade desenvolvedora de aplicações.

de maneira extensível com APIs para validar os objetos.

52 Yeoman é uma ferramenta open source executada como uma interface de linha de comando escrita para o Node.js e combina várias funções em um único local, como gerar um modelo inicial, gerenciar dependências, executar testes de unidade, fornecer um servidor de desenvolvimento local e otimizar o código de produção para implantação.

53 Webpack é um bundler de módulo JavaScript open source. É um módulo empacotador, principalmente para JavaScript, mas pode transformar recursos de front-end, como HTML, CSS e imagens, se os plug-ins correspondentes forem incluídos.

54 Browser-sync é um módulo do Node.js disponível para uso sob a licença Apache 2.0. Ele proporciona uma série de funcionalidades como instalação e execução de qualquer lugar, histórico de URL, sincronização de arquivos, acelerador de rede e etc.

55 Maven é uma ferramenta de automação de compilação utilizada primariamente em projetos Java.

56 Gradle é um sistema de automação de compilação open source que se baseia nos conceitos do Ant e do Maven.

57 Disponível em: <https://www.jhipster.tech/documentation-archive>. Acesso em: 10 set. 2019.

58 O npm ou Node Package Manager (gerenciador de pacotes do Node) consiste de dois componentes principais: um grande repositório on-line para publicação de projetos Node.js de código aberto e uma ferramenta de linha de comando para interação com o repositório e seus pacotes.

O ***Yarn*** permite a utilização e compartilhamento de códigos de seus usuários com outros desenvolvedores de todo o mundo, possibilitando o uso de soluções de terceiros para diferentes problemas, o que facilita o desenvolvimento de software e proporciona que os usuários se concentrem no que importa: a criação de novos produtos e recursos.

O uso de um gerenciador de pacotes no projeto introduz um novo fluxo de trabalho em torno das dependências. O ***Yarn*** faz o possível para não atrapalhar o desenvolvedor e tornar cada etapa do fluxo de trabalho simples de entender. Um fluxo de trabalho padrão basicamente é composto pelas seguintes atividades:

a. criar um novo projeto;
b. adicionar, atualizar ou remover dependências;
c. instalar ou reinstalar dependências;
d. trabalhar com controle de versão;
e. integração contínua.

O ***Yarn*** fornece um rico conjunto de comandos de *CLI* (*command-line interface* ou interface de linha de comando) para ajudar o usuário com vários aspectos do gerenciador, incluindo instalação, administração e publicação. Alguns dos comandos mais populares são:

» *yarn_add*: adiciona um pacote para usar em seu pacote atual.
» *yarn_init*: inicializa o desenvolvimento de um pacote.
» *yarn_install*: instala todas as dependências definidas em um arquivo *package.json*.
» *yarn_publish*: publica um pacote para um gerenciador de pacotes.
» *yarn_remove*: remove um pacote não utilizado do seu pacote atual.

Um pacote é um diretório com algum código e um arquivo *package.json*, que fornece informações ao ***Yarn*** sobre o seu pacote. A maioria dos pacotes usa algum tipo de sistema de controle de versão, sendo que o mais comum é o *git*[59], todavia, o gerenciador não se importa com qual ferramenta usuário escolher.

Para configurar o ambiente, o ***Yarn*** procura por arquivos *package.json*

59 Git é um sistema de controle de versões distribuído.

no intuito de identificar cada pacote e configurar seu comportamento durante a execução dentro desse ambiente. Como exemplo, apresentamos a configuração de um pacote fictício denotado *Ibict*, que contém nome, versão, arquivo principal e dependências, e se encontra em *Ibict/package.json*:

```
{
 "name": "Ibict",
 "version": "0.1.0",
 "main": "ibict.js",
 "dependencies": {
 "hand": "1.0.0"
 }
}
```

O gerenciador também usa um arquivo de bloqueio, chamado *yarn.lock*, na raiz do seu projeto para tornar a resolução de dependências rápida e confiável. Não é necessário que o usuário interaja com o arquivo, uma vez que é papel do ***Yarn*** mudá-lo ao gerenciar as dependências. Para garantir que a aplicação funcione de forma consistente, é aconselhável salvar o arquivo *yarn.lock* no repositório do código.

Vale ressaltar que o ***Yarn*** é um projeto comunitário com contribuições patrocinadas de várias empresas e aberto ao público, com uma documentação localizada no seu site oficial[60], no qual qualquer pessoa pode se envolver e contribuir com projeto.

3.7 Configurar o ambiente de produção

Neste exemplo de instalação do Visão, será utilizado o sistema operacional ubuntu server 18.04 com 64 bits, que tem distribuição livre. Para o servidor a ser utilizado, é indicado uma configuração mínima de 8GB de memória RAM e 80GB de HD.

O Visão é um aplicação desenvolvida em Java, assim, para fins de compilação e execução, faz-se necessário ter o *Java Development Kit* (JDK)

60 Disponível em: <https://yarnpkg.com/lang/pt-br/docs/>. Acesso em: 10 set. 2019.

instalado no servidor. Também são necessários os pacotes *git*, *nodejs*, *npm e yarn*.

3.7.1 Instalando o JDK 9

Para armazenar os dados fornecidos pelos usuários e produzidos pelo sistema, optou-se por utilizar o sistema de gerenciamento de banco de dados de código aberto *MariaDB*.

a. Adicionar o repositório do Java 9:

```
$ sudo add-apt-repository ppa:webupd9team/java
```

b. Atualizar o gerenciador de pacotes:

```
$ sudo apt-get update
```

c. Instalar o Java 9:

```
$ sudo apt-get install oracle-java9-installer
```

d. Testar o compilador java:

```
$ javac -version
```

e. Testar o ambiente de execução do Java:

```
$ java -version
```

Observação: para os itens ***d)*** e ***e)***, deve aparecer terminal mensagens indicando a versão do Java instalado, que nesse exemplo é o Java 9.

f. Instalar o sistema de gerenciamento de banco de dados MariaDB:

```
$ sudo apt-get install mariadb-server
```

É importante anotar a senha do root, pois será necessária posteriormente.

3.7.2 Git, nodejs, npm e yarn

Seguem os passos para instalar os pacotes adicionais:

a. Instalar o git para buscar o código-fonte da aplicação:

```
$ sudo apt install git
```

b. Instalar o Node.js para para construir aplicações web usando JavaScript:

```
$ sudo apt-get install nodejs
$ sudo apt-get install npm
```

c. Instalar o Yarn para gerenciar as dependência do Node.js:

```
$ curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo
apt-key add -
$ echo "deb https://dl.yarnpkg.com/debian/ stable main" |
sudo tee /etc/apt/sources.list.d/yarn.list
$ sudo apt update
$ sudo apt-get install yarn
```

3.8 Implantação

Inicialmente é necessário baixar o código-fonte do Visão no local de preferência a partir do GitHub[61] do mantenedor. Depois são criados o banco de dados e os usuários e são realizadas algumas configurações para a aplicação.

a. Baixar o código-fonte:

```
$ cd /home/visao
$ git clone https://github.com/IBICT/visao.git VISAO
```

b. Criar o usuário e o banco de dados fornecendo os devidos privilégios:

```
$ sudo su
$ mysql -u root
        CREATE USER 'user'@'localhost' IDENTIFIED BY
'your_password';
        CREATE DATABASE banco DEFAULT CHARACTER
SET utf8;
        GRANT ALL ON banco.* TO user@localhost IDEN-
TIFIED BY 'your_password';
```

c. Navegar até o diretório **/home/visao/VISAO**, que contém os códigos-fonte da aplicação. Abra o arquivo:

```
VISAO/src/main/resources/config/application-prod.yml
```

Edite as linhas 30, 31 e 32 para configurar o banco:

```
url: jdbc:mariadb://localhost:3306/banco
 username: user
 password: your_password
```

61 Disponível em: <https://github.com/IBICT/visao.git>. Acesso em: 11 set. 2019.

d. Abra o arquivo:

```
VISAO/src/main/resources/config/application-prod.yml
```

Configure o serviço de e-mail nas linhas 52, 53, 54 e 55:

```
mail:
    host: servidor smtp
    port: número da porta
    username: usuário do servidor
    password: senha do usuário
```

e. Navegar pelo terminal até o diretório **/home/visao/VISAO** e executar os seguintes comandos:

```
$ yarn install
$ yarn build
$ gradlew -Pprod bootWar
```

Agora é necessário usar um servidor web para disponibilizar a aplicação. Optou-se por utilizar o servidor web Java livre Apache Tomcat.

f. Baixar o binário do tomcat 9 no site oficial[62]:

```
$sudo su
 cd /opt

wget
http://mirror.nbtelecom.com.br/apache/tomcat/tomcat-9/
v9.0.22/bin/apache-tomcat-9.0.22.tar.gz

tar -vzxf apache-tomcat-9.0.22.tar.gz
mv apache-tomcat-9.0.22 tomcat

chown -R visao:visao tomcat
exit
```

62 Disponível em: <https://tomcat.apache.org/download-90.cgi>. Acesso em 11 set. 2019.

g. Carregar a aplicação no servidor tomcat:

```
$ cp /home/VISAO/build/libs/*.war  /opt/tomcat/webapp/ROOT
```

h. Configurar a codificação de apresentação dos dados para utf-8 no arquivo */opt/tomcat/conf/server.xml* na linha 71:

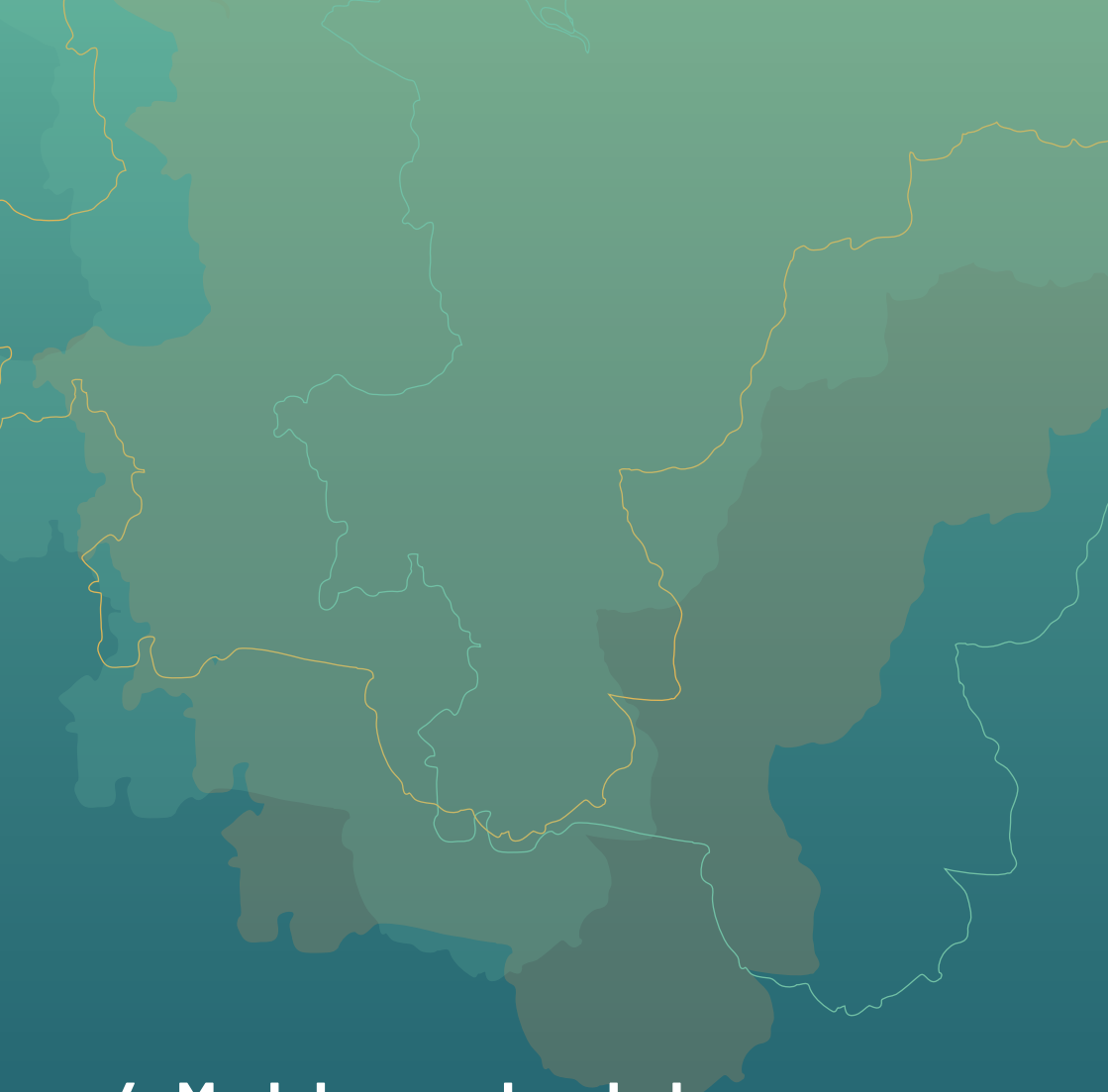
```
<Connector port="8080" protocol="HTTP/1.1"
connectionTimeout="20000"
redirectPort="8443" URIEncoding="UTF-8" />
```

i. Iniciar a aplicação:

```
$ cd /opt/tomcat/bin
$ ./startup.sh
```

Alcançando todos os passos anteriores, tem-se o Visão pronto para ser utilizado. Basta utilizar o navegador de preferência e acessar a URL http://localhost:8080.

COMO CITAR
SILVEIRA, Lucas Angelo da; MOURA, Rebeca dos Santos de. Instalação do Visão. In: SHINTAKU, Milton (Org.). **Mapa digital para gestão do conhecimento**: a construção de um sistema com o software Visão. Brasília: Ibict, 2019. p. 33-55. DOI: 10.18225/9788570131638.cap3



4. Modelagem dos dados

Lucas Angelo da Silveira

Rebeca dos Santos de Moura

O Visão possui uma estrutura de banco de dados relacional composta por dezenove tabelas. Dentre elas, sete estão vinculadas exclusivamente ao sistema do software e as outras doze tem a função de mapear os dados que possam vir a ser inseridos.

As tabelas de software serão abordadas brevemente, pois foram criadas automaticamente pelo framework JHipster e são alimentadas por processos internos do software. Em seguida, as quinze tabelas de dados serão discutidas com mais abrangência.

4.1 Tabelas de sistema

As sete tabelas de sistema podem ser divididas em três grupos. O primeiro grupo é relacionado aos usuários do sistema e compreende as tabelas mostradas na Figura 4.1 e Quadro 4.1.

Figura 4.1 - Esquema relacional do primeiro grupo de tabelas de sistema

Fonte: Captura de tela (2019).

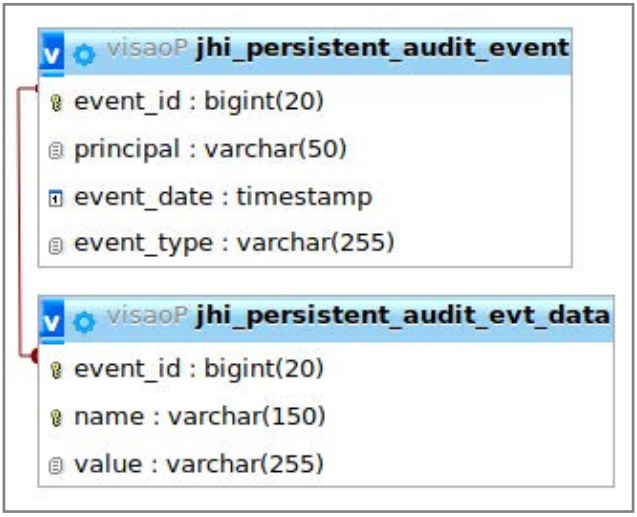
Quadro 4.1 - Tabelas de sistema relacionado aos usuários.

Nome das tabelas	Descrição
jhi_authority	Armazena os possíveis perfis de autenticação de usuários: admin e user.
jhi_user_authority	Armazena os perfis de autenticação de cada usuário cadastrado no sistema.
jhi_persistent_token	Armazena o token de acesso do usuário. Atualmente a tabela não está implementada.

Fonte: Elaboração dos autores (2019).

O segundo grupo de tabelas (Figura 4.2) faz alusão aos eventos de autenticação de usuários no sistema.

Figura 4.2 - Esquema relacional do segundo grupo de tabelas de sistema



Fonte: Captura de tela (2019)

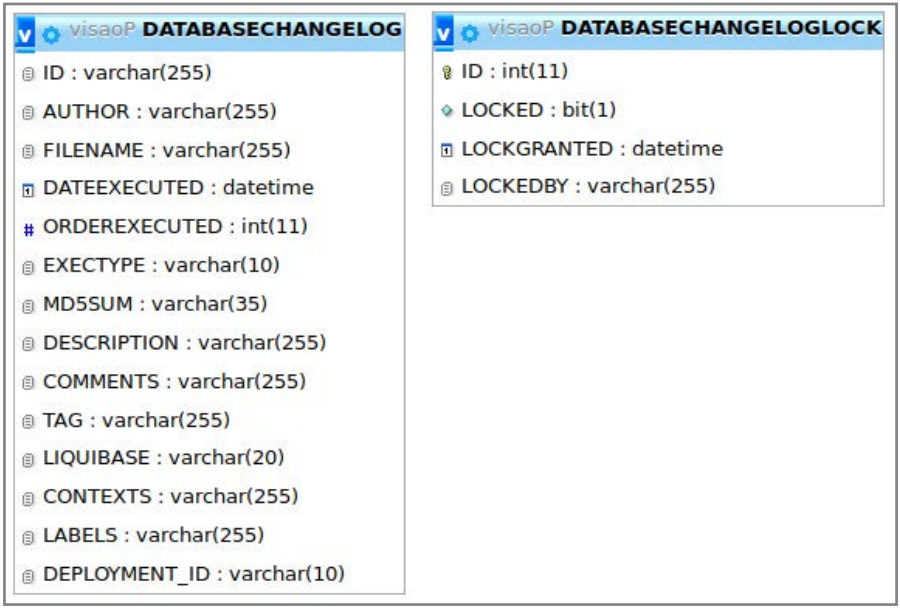
Quadro 4.2 - Tabelas de sistema para autenticação de usuários.

Nome das tabelas	Descrição
jhi_persistent_audit_event	Armazena os eventos de autenticação no sistema que podem ser do tipo AUTHENTICATION_FAILURE (falha) ou AUTHENTICATION_SUCCESS (sucesso).
jhi_persistent_audit_evt_data	Armazena os dados dos eventos de autenticação no sistema.

Fonte: Elaboração dos autores (2019).

O terceiro e último grupo de tabelas está ilustrado na Figura 4.3 e se refere às operações de manutenção e histórico (*changelog*) do banco de dados.

Figura 4.3 - Esquema relacional do terceiro grupo de tabelas de sistema.



Fonte: Captura de tela (2019).

Quadro 4.3 - Tabelas de sistema referente às operações de manutenção e histórico.

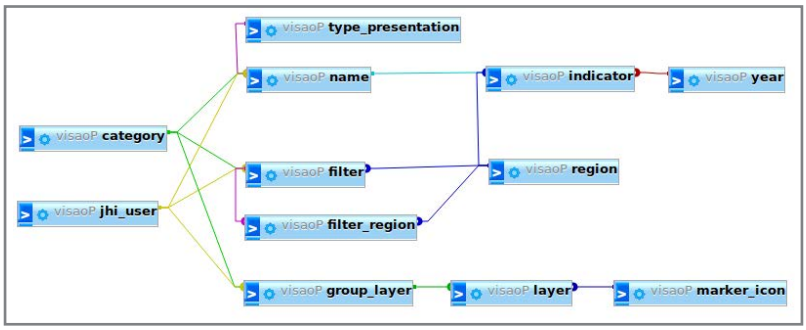
Nome das tabelas	Descrição
DATABASECHANGELOG	Armazena um histórico das operações realizadas no banco de dados.
DATABASECHANGELOGLOCK	Armazena informações sobre o bloqueio de changelog do banco de dados para garantir que apenas uma instância da tabela DATABASECHANGELOG esteja em operação.

Fonte: Elaboração dos autores (2019).

4.2 Tabelas de dados

As doze tabelas de dados estão diagramadas na Figura 4.4. A seguir, será apresentado uma descrição de cada uma delas em ordem alfabética.

Figura 4.4 - Esquema relacional do banco



Fonte: Captura de tela (2019)

category
Armazena as possíveis categorias de indicadores e possui os seguinte atributos:

» *bigint id*: chave primária incremental.
» *varchar name:* o nome da categoria.
» *varchar jhi_type*: há opção de criar uma categoria escolhendo entre os três tipos fixados no sistema: *LAYER*, *FILTER*, *INDICATOR*. Não há a possibilidade de alteração dos três tipos por parte do usuário.

filter
Armazena as regiões do sistema e possui os seguinte atributos:

» *bigint id*: chave primária incremental.
» *varchar name*: insere o nome da região exemplo: Norte, Nordeste etc.
» *bit active*: representa um bit: *1* para ativar a região a ser inserida, ou *0* para não usar a região.
» *longtext description*: texto livre descrevendo a região.
» *varchar key_word*: palavras-chave para a região.
» *datetime jhi_date*: inserido automaticamente pela aplicação para salvar a data e o horário de inserção.
» *varchar source*: campo para indicar a fonte da informação.
» *datetime data_change*: caso tenha alteração no registro, o campo terá uma data diferente do jhi_date.
» *longtext note*: nota sobre a região a ser inserida.
» *bigint cidade_polo_id*: identificador da cidade polo, representando uma chave estrangeira da tabela ***region***.
» *bigint category_id*: chave estrangeira da tabela ***category***.
» *bigint user_id*: designa o usuário que inseriu, chave estrangeira da tabela ***jhi_user***.

filter_region
Armazena o identificador do filtro inserido na tabela ***filter*** e o identificador da tabela ***region***, de forma a correlacionar ambos. Possui os seguinte atributos:

» *bigint regions_id*: chave estrangeira da tabela ***region***.
» *bigint filters_id*: chave estrangeira da tabela ***filter***.

group_layer
Armazena as camadas de marcadores que serão plotadas no mapa. Possui os seguintes atributos:

» *bigint id*: chave primária incremental.
» *varchar name*: nome da camada a ser apresentada no mapa.
» *bit active*: bit para representar se a camada está ativa.
» *varchar key_word*: palavras-chave para descrever a camada.
» *bigint category_id*: chave estrangeira da tabela ***category***.

» *bigint user_id*: chave estrangeira da tabela ***jhi_user***.

jhi_user

Tabela para cadastrar os usuários do Visão. Por padrão, o sistema possui quatro usuários cadastrados, listados a seguir:

a. Admin: usuário com maior privilégio no sistema, responsável por gerenciar a aplicação.

b. User: usuário com segundo maior privilégio no sistema. Comparado ao *Admin*, esse usuário só não tem acesso ao menu de administração, mas todas as demais funcionalidades são acessíveis.

c. System: usuário responsável pelas operações automáticas no sistema, tais como criar usuários e tabelas na implantação do software.

d. Anonymoususer: usuário não logado pode apenas acessar as funcionalidades que o sistema oferece.

A tabela possui os seguinte atributos:

» *bigint id*: chave primária incremental.
» *varchar login*: nome para ser usado como usuário no sistema.
» *varchar password_hash*: a senha codificada utilizando hash.
» *varchar first_name*: primeiro nome do usuário.
» *varchar last_name*: último nome do usuário.
» *varchar email*: email para contatar o usuário.
» *varchar image_url*: imagem para o perfil do usuário.
» *bit activate*: representa um bit: *1* indica o usuário ativo, ou *0* o usuário será cadastrado mas não está visível no sistema.
» *varchar lang_key*: idioma utilizado pelo usuário.
» *varchar create_by*: é inserido automaticamente pela aplicação e faz referência ao usuário System.
» *timestamp created_date*: data de criação do usuário.
» *timestamp reset_date*: data da alteração de senha.
» *varchar last_modified*: usuário que fez a modificação nos dados.
» *timestamp last_modified_date*: data de alteração dos dados do usuário.

layer

Armazena os marcadores nas camadas e possui os seguintes atributos:

» *bigint id*: chave primária incremental.
» *varchar name*: nome da camada.
» *longtext geo_json*: localização geográfica do mundo real da ca-

mada na perspectiva latitude-longitude.
» *varchar jhi_type*: formato a ser plotado no mapa entre polygon, marker, cycle.
» *varchar description*: descrição da camada.
» *datetime jhi_date*: data da inserção da camada.
» *varchar source*: campo para indicar a fonte da informação.
» *datetime date_change*: data da última atualização.
» *varchar note*: alguma nota sobre a camada.
» *bigint icon_id*: chave estrangeira da tabela ***marker_icon***.
» *bigint group_id*: chave estrangeira da tabela ***group_layer***.

name
Mapeia os indicadores e possui os seguintes atributos:
» *bigint id*: chave primária incremental.
» *varchar jhi_value*: nome do indicador.
» *bit active*: representa um bit: *1* para ativar o indicador.
» *longtext description*: descrição do indicador.

marker_icon
Armazena um novo ícone para representar um marcador no mapa. Possui os seguintes atributos:
» *bigint id*: chave primária incremental.
» *blob icon*: imagem a ser utilizada como ícone.
» *varchar incont_content_type*: formato da imagem.
» *blob shadow*: imagem para ser usada como sombra.
» *varchar shadow_content_type*: o tipo da imagem a ser usada como sombra.
» *varchar icon_size*: tamanho da imagem usada como ícone.
» *varchar shadow_size*: tamanho da imagem usada como sombra.
» *varchar icon_anchor*: customiza o campo icon.
» *varchar shadow_anchor*: customiza o campo shadow.
» *varchar popup_anchor*: campo de customização.

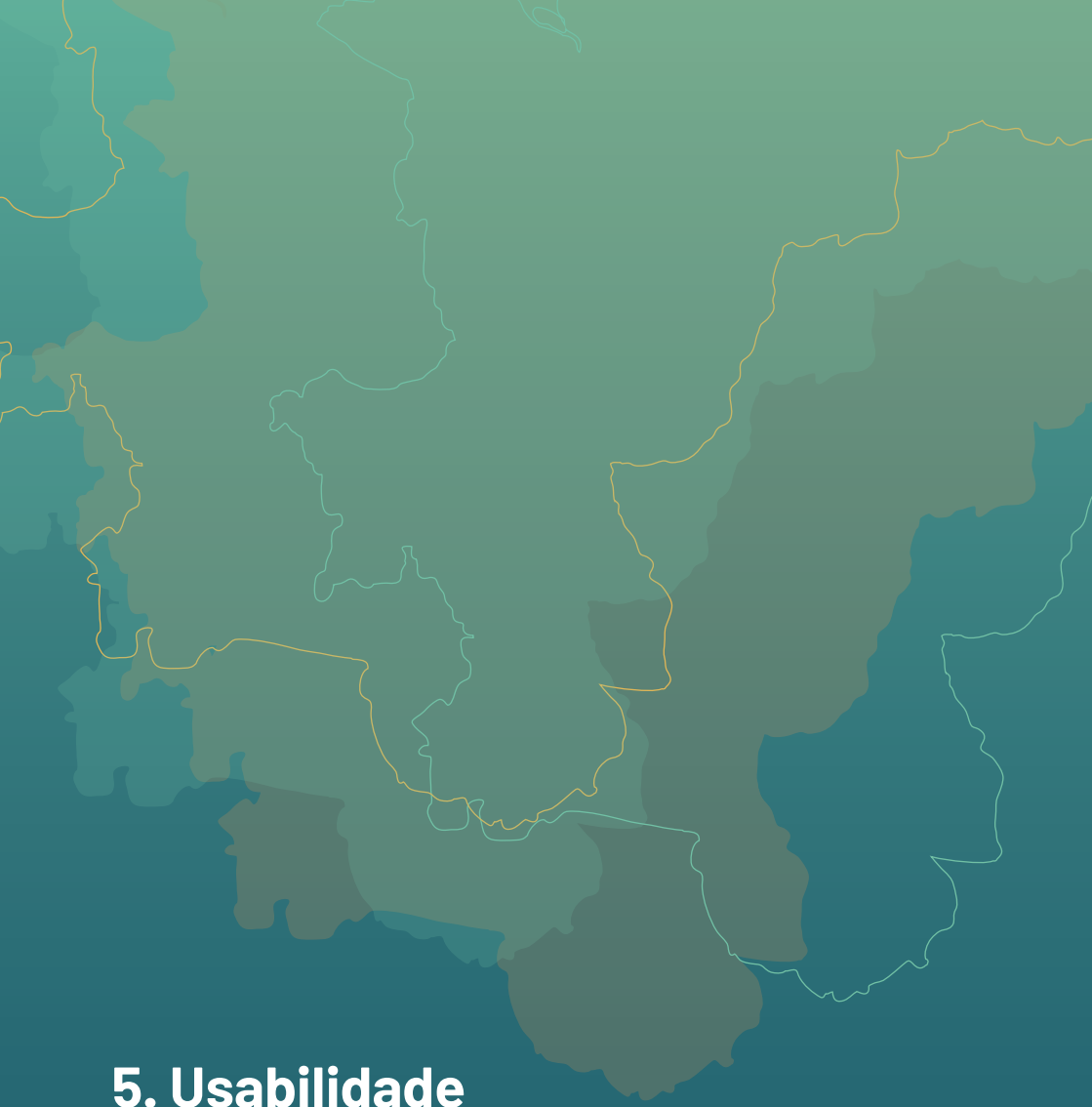
region
Mapeia as regiões no sistema, como estados, mesorregiões e municípios. Possui os seguintes atributos:
» *id*: chave primária incremental.
» *name*: nome do estado, mesorregião ou município.
» *uf*: sigla para indicar o estado. Para estados brasileiros, é utili-

zado sua respectiva sigla.
» *geo_code*: código de objeto geoespacial para coordenadas geográficas.
» *type_region*: define se é estado, mesorregião ou município. Não há a possibilidade de alteração dos três tipos por parte do usuário.

type_presentation
Armazena os tipos de dados a serem apresentados nos indicadores. Possui os seguintes atributos:
» *bigint id*: chave primária incremental.
» *varchar name*: o nome dado ao tipo.
» *varchar display*: o nome que será plotado no mapa.
» *int geo_code*: código de geolocalização.
» *varchar description*: descrição do tipo de dado.

year
Armazena os anos a serem usados na inserção dos dados de indicadores. Possui os seguintes atributos:
» *bigint id*: chave primária incremental.
» *varchar jhi_date*: insere um ano.

COMO CITAR
SILVEIRA, Lucas Angelo da; MOURA, Rebeca dos Santos de. Modelagem dos dados. In: SHINTAKU, Milton (Org.). **Mapa digital para gestão do conhecimento**: a construção de um sistema com o software Visão. Brasília: Ibict, 2019. p. 56-64. DOI: 10.18225/9788570131638.cap4



5. Usabilidade

Lucas Angelo da Silveira

Rebeca dos Santos de Moura

No Visão, existem três tipos de usuário: o anônimo que não tem login, aquele que possui cadastro e o administrador. Todos os usuários podem interagir com o mapa e o menu lateral, mas o usuário cadastrado e o administrador possuem mais opções de interação e gerenciamento do sistema.

5.1 Barra de menu principal

Na Figura 5.1 são mostradas as três possíveis barras de menu. O usuário anônimo pode entrar no sistema ou realizar o cadastro. O usuário cadastrado pode ver o menu de Entidades e suas configurações de usuário. Por fim, o usuário administrador tem acesso ao menu de Entidades, menu de Administração e também às suas configurações.

Figura 5.1 - Barra de menus para os diferentes usuários: anônimo (a), cadastrado (b) e administrador (c)

a. 

b.

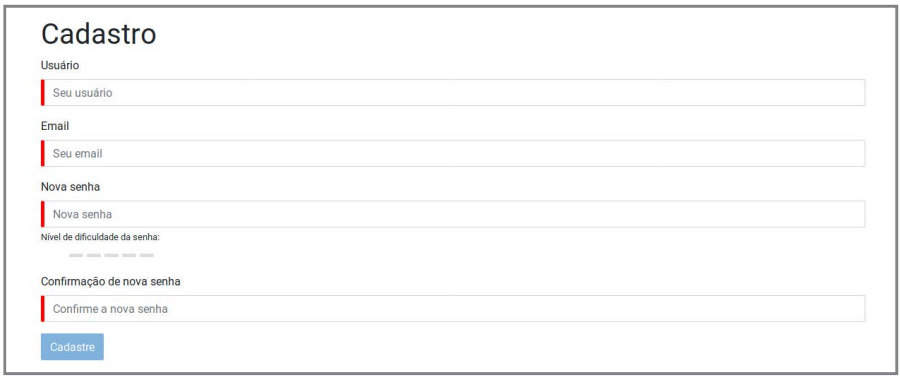
c.

Fonte: Captura de tela (2019)

5.2 Usuário não logado

Um usuário não logado terá acesso apenas à visualização padrão do sistema, isto é, o mapa e o menu lateral, com suas aplicações. O usuário pode realizar seu cadastro no sistema. Basta criar um nome de usuário, e-mail e senha (Figura 5.2).

Figura 5.2 - Cadastro para novos usuário



Fonte: Captura de tela (2019)

Após realizar o cadastro, o usuário pode se autenticar no sistema usando login e senha (Figura 5.3).

Figura 5.3 - Login no sistema

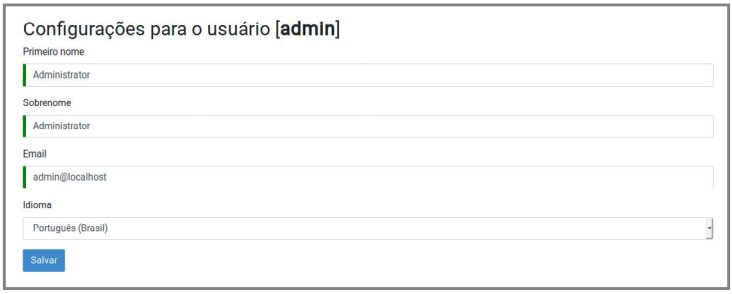
Fonte: Captura de tela (2019)

5.3 Usuário logado

O usuário cadastrado e o administrador têm acesso a um menu que contém as opções de Configuração, Senha, Sessões e Sair.

Na opção de Configuração (Figura 5.4), o usuário pode editar seu primeiro nome, sobrenome, e-mail e idioma da aplicação.

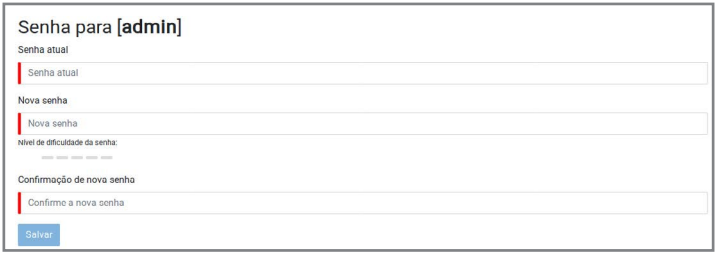
Figura 5.4 - Opção de configuração do usuário



Fonte: Captura de tela (2019).

Na opção de Senha (Figura 5.5), o usuário pode redefinir sua senha.

Figura 5.5 - Opção de senha do usuário



Fonte: Captura de tela (2019)

Na opção de Sessões (Figura 5.6), o usuário pode ver suas sessões ativas. Por fim, ele pode sair do sistema ao clicar na opção de Sair.

Figura 5.6 - Opção de sessão do usuário

Fonte: Captura de tela (2019).

5.4 Menu de Entidades

Figura 5.7 - Entidades do Visão

Fonte: Captura de tela (2019)

As entidades definidas pelo Visão (Figura 5.7) são: Grupos de Indicadores, Indicadores, Ano, Meta Dado, Filtro, Grupos de Camadas, Camadas, Ícones, Categorias, Regiões e Apresentação dos Dados.

Cada entidade está relacionada a uma tabela de dados. São elas:

» *Grupo de Indicadores*: lista dos indicadores que ficarão disponíveis na aba Indicadores do menu lateral.
» *Indicadores*: dados dos Indicadores listados em "Grupo de indicadores".

» *Ano*: lista com anos disponíveis para os indicadores.
» *Meta Dado*: tabela curinga para associar dados (imagens, textos etc.) que ainda não foi implementada.
» *Filtro*: lista dos filtros que ficarão disponíveis na aba Filtros do menu lateral.
» *Grupo de Camadas*: lista das camadas que ficarão disponíveis na aba Camadas do menu lateral.
» *Ícones*: ícones disponíveis para uso nos marcadores das camadas.
» *Categorias*: lista das categorias disponíveis no sistema, elas podem ser *indicator* (indicador), *filter* (filtro) ou *layer* (camada).
» *Regiões*: dados das regiões do mapa. Elas podem ser mesorregião, estado ou município.
» *Apresentação dos dados*: nome, display e descrição dos tipos de dados para apresentação de indicadores.

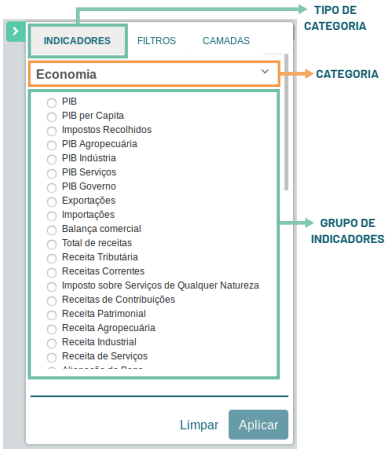
Nas próximas seções serão apresentadas descrições mais detalhadas das entidades e seus relacionamentos entre si.

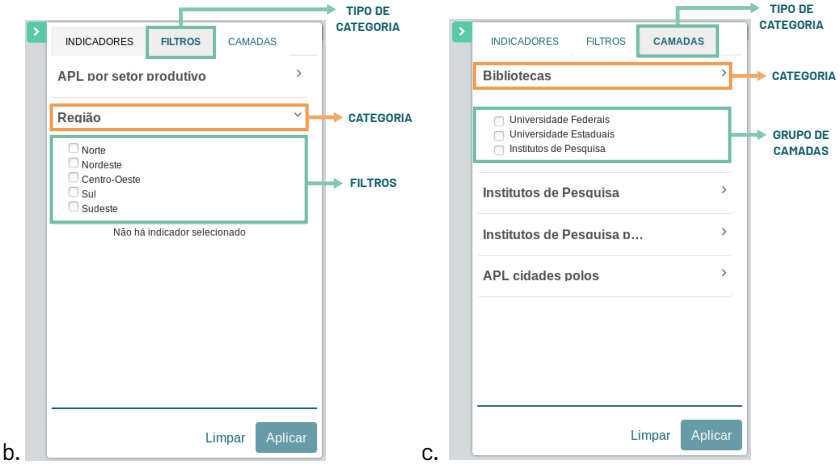
5.4.1 Categorias

A entidade Categoria é a base para a apresentação dos dados no mapa do Visão. Ela delimita os tipos possíveis de dados no sistema: indicadores, filtros e camadas. Uma categoria deve ter um nome e um tipo, que pode ser Indicador (*INDICATOR*), Filtro (*FILTER*) ou Camada (*LAYER*). A categoria aparecerá no primeiro nível do menu lateral, embaixo de cada um dos tipos, como exemplificado na Figura 5.8 para os três tipos existentes.

Figura 5.8 - Ilustração da entidade Categoria para os tipos Indicador (a), Filtros (b) e Camadas (c)

a.





Fonte: Elaboração dos autores (2019).

5.4.2 Grupo de indicadores

As instâncias da entidade Grupo de indicadores são listadas na aba de Indicadores, no menu lateral. Uma instância de Grupo de indicadores precisa ter Valor obrigatoriamente e ainda pode conter os campos:

» Ativo: checkbox para alternar entre ativo e inativo
» Descrição: campo de texto para descrição do grupo
» Palavra-Chave: lista de palavras-chave separadas por ";"
» Data: data de criação do grupo
» Origem: campo de texto para inserção da origem do dado
» Data de Modificação: data de modificação do grupo
» Nota: campo de texto para notas sobre o grupo
» Categoria: uma lista com as categorias do tipo Indicador
» Apresentação dos dados: lista dos possíveis tipos de apresentação de dados
» Usuário: usuário criador do grupo, pode ser *admin*, *user* e *system*.

Apesar de opcionais para o registro no banco de dados, os campos de Categoria e Apresentação dos dados precisam ser preenchidos para o correto funcionamento do Mapa.

5.4.3 Indicadores

Cada instância da entidade Indicadores deve estar ligada a um Grupo de indicadores. Uma instância de Indicadores precisa ter Valor obrigatoriamente, mas para o correto funcionamento do Mapa ela deve conter os campos:

- Região: estado, município ou mesorregião associada ao Valor
- Nome: lista com os Grupos de indicadores existentes
- Ano: lista com os anos existentes no sistema

5.4.4 Filtro

As instâncias da entidade Filtro são listadas na aba de Filtros do menu lateral. Cada instância da entidade precisa ter Nome obrigatoriamente e ainda pode ser conter os campos:

- Ativo: checkbox para alternar entre ativo e inativo
- Descrição: campo de texto para descrição do filtros
- Palavra-Chave: lista de palavras-chave separadas por ";"
- Data: data de criação do filtro
- Origem: campo de texto para inserção da origem do dado
- Data de Modificação: data de modificação do filtro
- Nota: campo de texto para notas sobre o filtro
- Cidade Pólo: campo não implementado.
- Categoria: uma lista com as categorias do tipo Filtro
- Usuário: usuário criador do filtro, pode ser *admin*, *user* e *system*.
- Região: lista das regiões às quais o filtro será associado no mapa.

Apesar de opcionais para o registro no banco de dados, os campos de Categoria e Região precisam ser preenchidos para o correto funcionamento do Mapa.

5.4.5 Grupo de camadas

As instâncias da entidade Grupo de camadas são listadas na aba de Camadas no menu lateral. Uma instância desta entidade precisa ter Nome obrigatoriamente e ainda pode conter os campos:

» Ativo: checkbox para alternar entre ativo e inativo
» Palavra-Chave: lista de palavras-chave separadas por ";"
» Usuário: usuário criador da camada, pode ser *admin*, *user* e *system*.

5.4.6 Camada (de marcadores)

Cada instância da entidade Camada deve estar ligada a um Grupo de camadas. Uma instância de camada, que na verdade é um marcador no mapa, deve ter obrigatoriamente:

» Nome: Nome da unidade
» Geo Json: geolocalização do marcador no formato "[Latitude, Longitude]"
» Tipo: lista dos tipos disponíveis de marcador (*Marker, Circle, Polygon, Icon*)

Ainda podem ser inseridos os campos de:

» Descrição: campo de texto para descrição da camada
» Data: data de criação da camada
» Origem: campo de texto para inserção da origem do dado
» Data de Modificação: data de modificação da camada
» Nota: campo de texto para notas sobre a camada
» Categoria: uma lista com as categorias do tipo Camada
» Ícone: lista com os Ícones existentes
» Grupo: lista com os Grupos de camadas existentes

Apesar da existência do campo Categoria, ele não é levado em consideração para visualização no mapa.

5.4.7 Regiões

A entidade de Regiões é extremamente importante para o correto funcionamento do mapa. Nela são listadas as regiões geográficas (município, estado ou mesorregião) às quais os dados podem ser associados. Uma instância de Regiões precisa ter obrigatoriamente:

» Nome: Nome da região
» Geo Code: geolocalização da região para representação no mapa (recurso implementado pela biblioteca *Leaflet*)
» Tipo de Região: lista dos tipos disponíveis de regiões (município, estado, mesorregião)

Ainda podem ser inseridos os campos de:

» UF: campo de texto para inserção da Unidade Federativa
» Relação: campo não implementado

O campo relação é usado para relacionar uma região com a outra. Por exemplo, para definir quais municípios fazem parte de um estado.

5.4.8 Ano

As instâncias da entidade Ano são utilizadas pela entidade Indicadores para agregar informação de ano aos dados inseridos. Uma entidade Ano contém obrigatoriamente o ano específico.

5.4.9 Apresentação dos dados

As instâncias da entidade Apresentação dos dados são utilizadas pela entidade Indicadores para agregar informação de representação de tipo de dado aos dados inseridos. Uma entidade Apresentação dos dados contém:

» *Name*: nome do tipo
» *Display*: nome que será exibido no mapa
» *Description*: descrição do tipo

5.4.10 Ícones

As instâncias da entidade Ícone são usadas para definir os ícones disponíveis para as entidades Camadas de marcadores. Uma instância de Ícone pode conter:

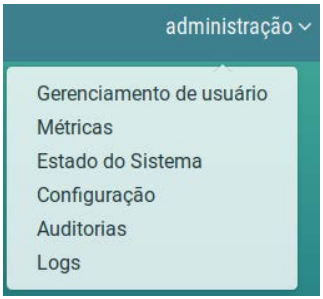
» Icon: arquivo de ícone
» Shadow: arquivo de sombra do ícone

» Icon Size: tamanho da imagem inserida no campo Icon
» Shadow Size: tamanho da imagem inserida no campo Shadow
» Icon Anchor: campo para customização do ícone
» Shadow Anchor: campo para customização da sombra do ícone
» Popup Anchor: campo para customização da pop-up

5.5 Menu de Administrador

O menu de Administrador permite acesso às seguintes opções (Figura 5.9): Gerenciamento de usuário, Métricas, Estado do Sistema, Configuração, Auditorias, Logs.

Figura 5.9 - Menu de Administrador

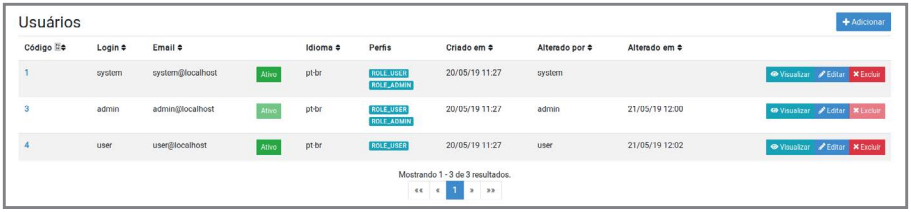


Fonte: Captura de tela (2019)

5.5.1 Gerenciamento de usuário

A página de gerenciamento de usuário (Figura 5.10) permite visualizar, editar ou excluir os usuários atualmente cadastrados no sistema e adicionar novos.

Figura 5.10 - Gerenciamento de usuário

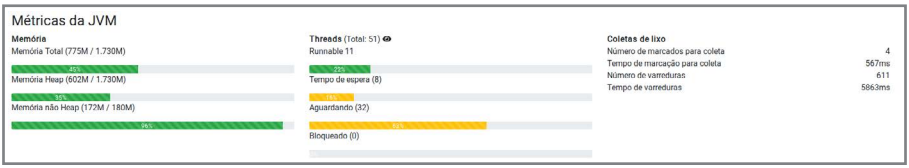


Fonte: Captura de tela (2019)

5.5.2 Métricas

A página de métricas disponibiliza várias métricas para gerenciamento da aplicação: Métricas da JVM (Figura 5.11), Requisições HTTP, Estatísticas dos serviços, Estatísticas da cache e Estatísticas do *datasource*.

Figura 5.11 - Métricas da JVM



Fonte: Captura de tela (2019)

5.5.3 Estado do sistema

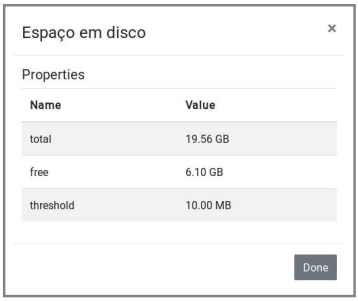
A página de estado do sistema (Figura 5.12) permite a visualização em detalhes do espaço em disco (Figura 5.13a) e do banco de dados (Figura 5.13b) da aplicação.

Figura 5.12 - Estado do sistema

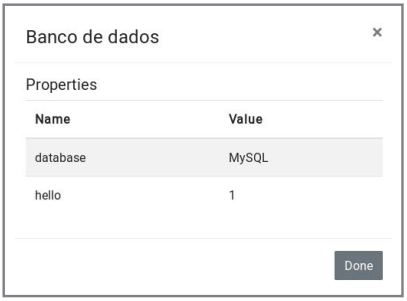
Fonte: Captura de tela (2019)

Figura 5.13 - Detalhes do espaço em disco (a) e do banco de dados (b) da aplicação.

a.



b.



Fonte: Captura de tela (2019)

5.5.4 Configuração

A página de Configuração mostra detalhes de vários parâmetros de configuração da aplicação, separados nas seguintes categorias: *Spring configuration*, *servletContextInitParams*, *systemProperties*, *systemEnvironment*, *applicationConfig*, *class path resource* e *defaultProperties*. A página ainda oferece um simples filtro nas informações apresentadas.

5.5.5 Auditorias

A página de Auditorias fornece ao usuário um histórico das tentativas de login realizadas no sistema classificadas como sucesso ou falha e com registro do IP responsável. A página ainda fornece um filtro por data (Figura 5.14).

Figura 5.14 - Auditorias

Fonte: Captura de tela (2019).

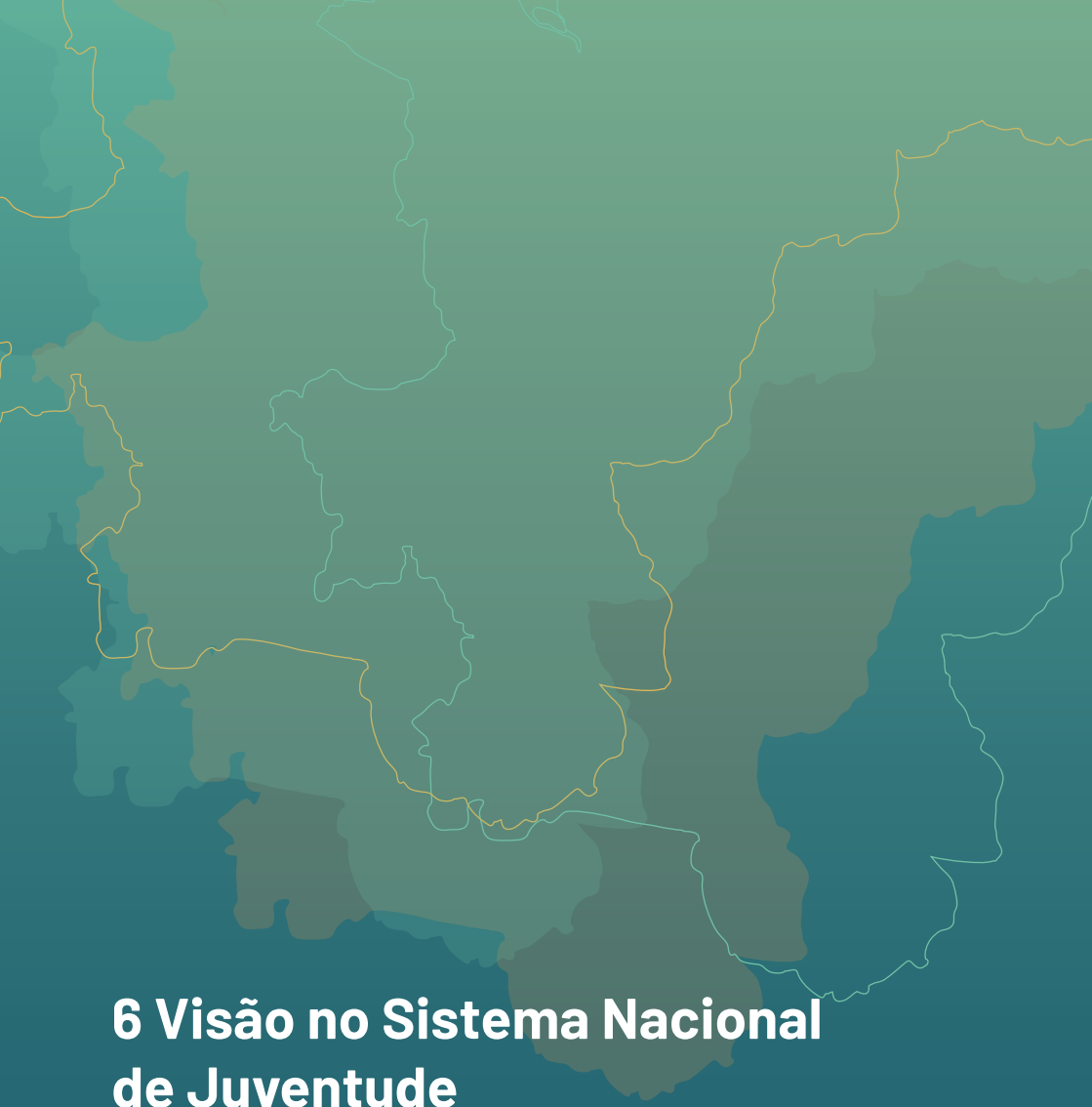
5.5.6 Logs

A página de logs (Figura 5.15) apresenta uma lista de logs, classificados por nível: *TRACE*, *DEBUG*, *INFO*, *WARN*, *ERROR* e *OFF*. A página ainda oferece um simples filtro nas informações apresentadas.

Figura 5.15 - Logs

Fonte: Captura de tela (2019).

COMO CITAR
SILVEIRA, Lucas Angelo da; MOURA, Rebeca dos Santos de. Usabilidade. In: SHINTAKU, Milton (Org.). **Mapa digital para gestão do conhecimento**: a construção de um sistema com o software Visão. Brasília: Ibict, 2019. p. 65-77. DOI: 10.18225/9788570131638.cap5



6 Visão no Sistema Nacional de Juventude

Janinne Barcelos

Lucas Angelo da Silveira

Rebeca dos Santos de Moura

O *Mapa das Unidades de Juventude* é produto do projeto de pesquisa "Estudos para sistematização e desenvolvimento do Sistema Nacional de Juventude (Sinajuve)" do Ibict em parceria com a SNJ, que utiliza o Visão como plataforma de plano de trabalho. O projeto visa desenvolver estratégias para gestão de políticas públicas voltadas para a juventude brasileira e mapear todos os estabelecimentos promotores de políticas públicas de juventude na esfera federal, estadual, municipal, independentemente de sua natureza administrativa. Tem como princípio norteador as ações e políticas que atendem os cidadãos brasileiros entre 15 e 29 anos. Para tanto, requer estudos voltados para o desenvolvimento de sistema de informação, assim como, métodos e práticas para operacionalizá-lo.

A operacionalização do projeto se efetivou em outubro de 2018. O plano de trabalho delineia que a implantação do Sistema ocorra mediante a estruturação do modelo de sistema de informação. Também se mostrou necessária a aplicação desse modelo, além da avaliação da efetividade do sistema por meio de monitoramentos e, finalmente, a disseminação por meio de publicações técnicas e científicas. Nesse sentido, a elaboração desta obra atende aos objetivos específicos no que tange à difusão do conhecimento científico.

O projeto se justifica pela complexidade da implementação do Sinajuve, devido à diversidade do cenário nacional. Trata-se de um desafio que requer um grande contingente de componente, resultante da quantidade de estados e municípios brasileiros. Por isso demandam-se estudos que apoiem o planejamento e a execução, com embasamento teórico, de modo a assegurar melhores resultados.

Para o desenvolvimento do *Mapa das Unidades de Juventude* foram respeitados os mesmos parâmetros empregados no mapeamento das unidades hospitalares no Sistema Único de Saúde (SUS), com uma representação gráfica, em escala reduzida, da localização das unidades. As informações referentes às unidades de juventude foram coletadas via questionário produzido pela equipe de pesquisadores do projeto, a partir da ferramenta *Google Forms*[63]. O questionário foi produzido pela COTEC e tramitado pela SNJ, responsável por disseminá-lo tanto no Distrito Federal quanto nas outras unidades federativas.

Entre as informações levantadas, estão: nome da unidade; endereço completo; natureza jurídica (ONG, Secretaria, Conselho, Diretoria ou Outros); gestão (municipal, estadual, federal); e área(s) de atuação da unidade, de acordo com as diretrizes do Estatuto de Juventude, Lei nº 12.852, de 2013, nomeadamente: Diversidade e Igualdade, Desporto e Lazer, Comunicação e Liberdade de Expressão, Cultura, Território e Mobilidade, Segurança Pública e Acesso à Justiça, Cidadania, Participação Social e Política e Representação Juvenil, Profissionalização, Trabalho e Renda, Saúde, Educação, Sustentabilidade e Meio Ambiente.

A partir dos dados coletados, o Visão foi alimentado com informações de coordenadas geográficas das unidades de juventude mantidas no país, onde cada unidade é representada como um marcador a partir de sua latitude e longitude (Figura 6.1).

Para melhor apresentação visual do mapa, os marcadores são agrupados em círculos (*clusters*), que mostram a quantidade de marcadores naquela região (círculos verdes e amarelos mostrados na Figura 6.1).

63 Serviço gratuito do Google para criar formulários on-line, que permite produzir pesquisas com questões de múltiplas escolhas ou discursivas, solicitar avaliações em escala numérica, entre outras opções.

Quando o zoom apropriado é aplicado ao mapa, os agrupamentos diminuem até que só os marcadores individuais são mostrados.

Figura 6.1 - Representação das Unidades de Juventude com marcadores do Visão

Fonte: Captura de tela (2019).

Por padrão, os marcadores do Visão são identificados por um ícone de cor azul. A equipe da COTEC implementou customizações com a introdução de novos ícones, nas cores amarela e verde, e fez a separação dos marcadores por esfera governamental da unidade. Dessa forma, uma unidade municipal é representada pela cor amarela, uma estadual pela azul e a federal pela verde.

No Visão, cada marcador está associado a uma *pop-up* que contém mais informações sobre o marcador. Para o *Mapa das Unidades de Juventude*, os descritores foram adequados para exibir informações específicas sobre a unidade de juventude, incluindo nome, telefone, natureza jurídica (levantados a partir do questionário) e um link externo para a central de políticas públicas da SNJ (Figura 6.2).

Figura 6.2 - Representação da *pop-up* para um marcador selecionado no Visão

Fonte: Captura de tela (2019)

Durante o mapeamento das unidades no Visão foi identificada a possibilidade de exploração dos recursos de *Business Intelligence* (BI) providos pelo software. Devido à existência de um grande volume de dados de indicadores sobre a juventude disponibilizados por diversas fontes do Governo, os recursos de BI do Visão foram aproveitados para visualização/demonstração de tais dados, além do mapa das unidades.

Assim, o Ibict buscou dados sobre Demografia, Trabalho, Educação, Saúde e Violência no Instituto Brasileiro de Geografia e Estatística (IBGE), no Sistema de Informação de Agravos de Notificação (SINAN), no Sistema de Informações Hospitalares do SUS (SIH/SUS) e no Sistema de Informações sobre Mortalidade (SIM). O recorte temporal compreendeu o intervalo de 2012 até 2018, considerando o critério de atualidade e a faixa etária estipulada pelo Estatuto da Juventude, que entende como jovens brasileiros de 15 a 29 anos. Os indicadores tabulados estão demonstrados no Quadro 6.1.

Quadro 6.1 - Indicadores selecionados sobre juventude, tabulados no Visão, no intervalo de 2012 até 2018, para jovens brasileiros de 15 a 29 anos.

Indicador	Dados	Discriminação
População Jovem	população residente de jovens	sexo: homem, mulher, total
Trabalho	percentual de jovens que trabalham	pirâmide etária: 15 a 17 anos; 18 a 24 anos; 25 a 29 anos
Educação	percentual de jovens que estudam	pirâmide etária: 15 a 17 anos, 18 a 24 anos; 25 a 29 anos
Saúde	doenças: aids, hanseníase e tuberculose morbidade hospitalar: internações, óbitos e taxa de mortalidade	casos notificados e/ou identificados
Violência	taxa de homicídios, violência física e violência psicológica/moral	casos notificados

Fonte: Elaboração dos autores, 2019.

A estratificação dos dados limitou-se ao grau de sofisticação do recuperador dos bancos de dados das referidas fontes. Logo, os dados de Demografia (população residente) foram retirados da Pesquisa Nacional por Amostra de Domicílios Contínua (PNAD Contínua), os dados de Trabalho e Educação foram retirados da Síntese de Indicadores Sociais (SIS) e os dados de Saúde e Violência foram retirados do portal do SINAN.

Como o Visão permite que cada indicador seja alimentado em associação com uma geolocalização, o usuário pode visualizar a localização na qual o dado está relacionado (Figura 6.3). Com isso, ao explorar a interface do Visão é possível realizar cortes por região brasileira, além de comparar incidências por regiões e estados, para cada um dos indicadores.

Figura 6.3 - Exemplo de visualização de um indicador para um estado específico

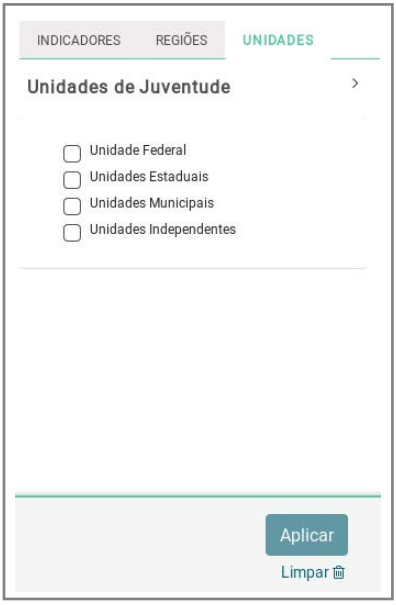
Fonte: Captura de tela (2019)

Assim, as adaptações estruturais realizadas no Visão para o *Mapa das Unidades de Juventude* foram as novas cores dos marcadores e descritores das *pop-ups*. A possibilidade de adequação e seleção de indicadores também foi uma característica explorada, de forma que as abas respectivas do o menu lateral ajustadas podem ser vistas na Figura 6.4.

Figura 6.4 - Abas de Indicadores e Unidades no menu lateral

a.

b.



Fonte: Captura de tela (2019)

Em um país grande como o Brasil, o planejamento e a alocação de recursos são essenciais para a criação e boa execução de políticas públicas. Assim, o *Mapa das Unidades de Juventude* se concretiza como uma ferramenta para visualização de dados de maneira rápida e objetiva. E isso, alinhado à ideia de cadastrar as instituições que trabalham com Juventude em único lugar, viabiliza em parte o monitoramento dos entes.

Assim, o mapa auxilia a correlacionar os dados dos indicadores mapeados para cada estado com as unidades cadastradas, facilitando a gestão de políticas públicas voltadas para a juventude.

COMO CITAR
BARCELOS, Janinne; SILVEIRA, Lucas Angelo da; MOURA, Rebeca dos Santos de. Visão no Sistema Nacional de Juventude. In: SHINTAKU, Milton (Org.). **Mapa digital para gestão do conhecimento**: a construção de um sistema com o software Visão. Brasília: Ibict, 2019. p. 78-85. DOI: 10.18225/9788570131638.cap7

7 Versões futuras

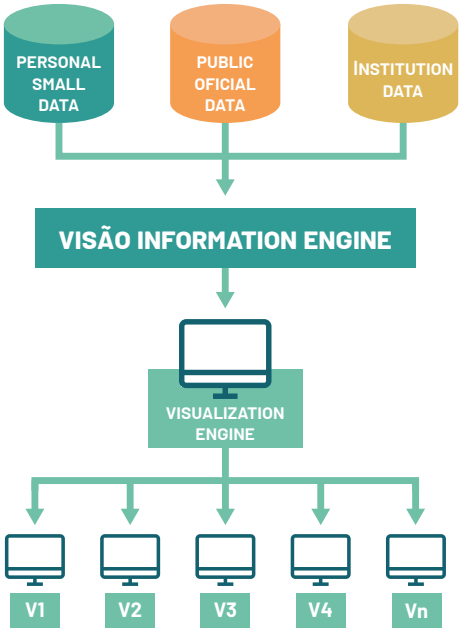
Tiago Emmanuel Braga

O Visão é um sistema vivo e, como tal, está em constante evolução. A versão utilizada atualmente está em sua fase beta, ou seja, os conceitos e ferramentas estão ainda em fase de validação. Enquanto algumas já podem ser consideradas maduras, outras ainda precisam ser aprimoradas. Este capítulo busca fazer um exercício sobre os próximos caminhos a serem trilhados na construção do Visão.

Um dos principais objetivos do Visão é permitir a tomada de decisão. Para isso, faz-se necessário que distintas visões sejam criadas a partir da integração das suas bases de dados. Sendo assim, há necessidade de se pensar na criação dessas visões a partir da organização, construção e disponibilização de indicadores personalizados. A construção de relações entre dados distintos permitirá que novas abordagens de análise de dados sejam experimentadas a partir dos conjuntos de dados já existentes. Entretanto, o acesso às bases já incorporadas no Visão não é suficiente para prover todas as possibilidades possíveis de análise. Então, além de permitir a criação de novos indicadores, é necessário se pensar na incorporação dados fornecidos pelos próprios usuários. A submissão desses pequenos conjuntos de dados, o *small data*, viabiliza que o Visão seja utilizado de forma mais abrangente.

Entre as mudanças conceituais planejadas para o Visão, está é, sem dúvida, a mais marcante. A incorporação de diferentes tipos de dados de forma automatizada a fim de possibilitar a personalização dos conjuntos de indicadores, filtros e camadas e, em última instância, a criação de Visões personalizadas para cada usuário ou instituição a partir de um mesmo sistema integrado.

Figura 7.1 - Perspectiva conceituação do Visão



Fonte: Captura de tela (2019).

Esse tipo de organização permitirá que não só instituições tenham suas próprias instâncias do Visão, com múltiplas visões dentro delas, como também dará a mesma perspectiva para pesquisadores, os quais poderão criar visualizações a partir dos seus resultados de pesquisa.

Se do ponto de vista conceitual o Visão tende a abranger o compartilhamento de visualizações e dados, do ponto de vista tecnológico há, também, muito o que se aprimorar. A versão beta buscou de forma mais incisiva desenvolver recursos e usabilidades voltados para o usuário final.

Todavia, há ainda um caminho a ser percorrido em termos de desempenho. A conexão automática com grandes bases de dados já integradas, como a INDE e a INDA, é um dos principais pontos a serem alcançados. A vinculação a essas bases permitirá o acesso automático a diversos conjuntos de dados já coletados e organizados pelo Governo Federal.

Tecnologias como o Pentaho — que foca na criação de relações entre conjuntos de dados e o GeoServer, voltado para a impressão de mapas — serão adicionadas paulatinamente, bem como bancos de dados voltados para o processamento de dados geográficos. A conexão com bases abertas, como as redes sociais ou os grandes portais de mídia, também é uma perspectiva real no que diz respeito às possibilidades existentes para o Visão.

O Visão é uma proposta ambiciosa do Instituto Brasileiro de Informação em Ciência e Tecnologia de criar um sistema de gestão focado nas diversas possibilidades de visualização de dados. Por trás do sistema há anos de trabalho do Instituto no fortalecimento das filosofias abertas e na busca por fornecer aos cidadãos o direito ao acesso livre e gratuito à informação e, consequentemente, à apropriação da coisa pública. A opção pela utilização de uma arquitetura aberta na construção do Visão é também um convite para que a comunidade colabore com seu desenvolvimento.

Estamos apenas no começo, mas o apoio crescente de diversas instituições e pesquisadores indica perspectivas muito promissoras.

COMO CITAR
BRAGA, Tiago Emanuel. Versões futuras. In: SHINTAKU, Milton (Org.). **Mapa digital para gestão do conhecimento**: a construção de um sistema com o software Visão. Brasília: Ibict, 2019. p. 86-89. DOI: 10.18225/9788570131638.cap7

Sobre os autores

Janinne Barcelos
E-mail: janinnesilva@ibict.br
Lattes: http://lattes.cnpq.br/7729780084365345
Orcid: https://orcid.org/0000-0003-1033-9414

Assistente de pesquisa do projeto Sinajuve. Doutoranda em Ciência da Informação na Universidade de Brasília (UnB). Mestre em Comunicação, Cultura e Cidadania, pela Universidade Federal de Goiás (UFG) e Bacharel em Comunicação Social - Jornalismo, pela Universidade Federal do Tocantins (UFT).

Lucas Ângelo Silveira
E-mail: lucasangelosilveira2018@gmail.com
Lattes: http://lattes.cnpq.br/9490636632029069
Orcid: http://orcid.org/0000-0002-8107-9659

Bacharel em Ciência da Computação pela Universidade Federal de Goiás (UFG), mestre em informática pelo Programa de Pós-Graduação em Informática da Universidade de Brasília (UnB). Atualmente, cursa doutorado em Informática pela UnB com ênfase em bioinformática de alto desempenho em estudos evolutivos entre espécies a partir dos genomas. Atua como pesquisador no Instituto Brasileiro em Ciência e Tecnologia (Ibict).

Rebeca dos Santos de Moura
E-mail: bcasamo@gmail.com
Lattes: http://lattes.cnpq.br/8677193043257356
Orcid: https://orcid.org/0000-0002-7685-8826

Assistente de pesquisa do projeto Sinajuve. Doutoranda em Geografia, Mestra em Engenharia de Sistemas Eletrônicos e de Automação e Bacharel em Engenharia da Computação pela Universidade de Brasília (UnB).

Tiago Emmanuel Nunes Braga
E-mail: tiagobraga@ibict.br
Lattes: http://lattes.cnpq.br/8376134230259399
Orcid: http://orcid.org/0000-0001-6332-7965

Graduado em Sistemas de Informação pela Universidade Católica de Minas Gerais (PUC/MG). Mestre em Educação Tecnológica pelo Centro Federal de Educação Tecnológica de Minas Gerais (CEFET/MG). Doutor em Ciência da Informação com foco em Gestão da Informação pela Universidade de Brasília (UnB). Pesquisador do Instituto Brasileiro de Informação em Ciência e Tecnologia (IBICT) do Ministério da Ciência, Tecnologia, Inovações e Comunicações (MCTIC).

Na moderna cartografia do final do século XX, a web se oferece como plataforma ideal para tornar a comunicação e a apresentação de dados espaciais cada vez mais eficientes e atrativas. A inteligência de localização pode revelar padrões e aspectos fundamentais de comportamento de forma rápida e intuitiva, proporcionando o desenvolvimento de projetos de administração pública com alto valor agregado e o aperfeiçoamento de estratégias para tomadas de decisão nas mais diversas políticas públicas, incluindo as de juventude.

Visando esse cenário, o Instituto Brasileiro de Informação em Ciência e Tecnologia (Ibict), em parceria com a Secretaria Nacional de Juventude (SNJ), tem investido em estudos para a estruturação e o desenvolvimento do Sistema Nacional de Juventude (Sinajuve). O Mapa das Unidades de Juventude, implementado a partir de ferramenta digital desenvolvida pela casa — o software Visão —, é um dos frutos da parceria. Nosso objetivo comum, nesta tarefa desafiadora, é mapear os estabelecimentos promotores de políticas públicas de juventude no Brasil, em todas as esferas governamentais, independentemente de sua natureza administrativa, a fim de unir forças em prol da juventude brasileira.

Cecília Leite de Oliveira
Diretora do Instituto Brasileiro de Informação em Ciência e Tecnologia

ibict 65 anos
Instituto Brasileiro de Informação em Ciência e Tecnologia
SECRETARIA NACIONAL DA JUVENTUDE
MINISTÉRIO DA MULHER, DA FAMÍLIA E DOS DIREITOS HUMANOS
PÁTRIA AMADA BRASIL GOVERNO FEDERAL