UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE CIÊNCIAS EXATAS E DA TERRA
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO
MESTRADO ACADÊMICO EM SISTEMAS E COMPUTAÇÃO

Dashboards para Desenvolvimento de Aplicações e Visualização de Dados para Plataformas de Cidades Inteligentes

Douglas Arthur de Abreu Rolim

Natal-RN, Brasil
2020

Douglas Arthur de Abreu Rolim

Dashboards para Desenvolvimento de Aplicações e Visualização de Dados para Plataformas de Cidades Inteligentes

Dissertação de Mestrado apresentado ao Programa de Pós-Graduação em Sistemas e Computação do Centro de Ciências Exatas e da Terra da Universidade Federal do Rio Grande do Norte como requisito parcial para a obtenção do título de Mestre em Sistemas e Computação.

Linha de pesquisa:
Engenharia de *Software*

Orientadora: Prof.ª Dra. Thais Vasconcelos Batista

Natal-RN, Brasil
2020

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Setorial Prof. Ronaldo Xavier de Arruda - CCET

Rolim, Douglas Arthur de Abreu.
Dashboards para desenvolvimento de aplicações e visualização de dados para plataformas de cidades inteligentes / Douglas Arthur de Abreu Rolim. - 2020.
87f.: il.

Dissertação (Mestrado) - Universidade Federal do Rio Grande do Norte, Centro de Ciências Exatas e da Terra, Programa de Pós-Graduação em Sistemas e Computação. Natal, 2020.
Orientadora: Thais Vasconcelos Batista.

1. Computação - Dissertação. 2. Desenvolvimento de aplicações para cidades inteligentes - Dissertação. 3. SGeoL - Dissertação. 4. Dashboard de desenvolvimento e visualização - Dissertação. I. Batista, Thais Vasconcelos. II. Título.

RN/UF/CCET CDU 004

Elaborado por Joseneide Ferreira Dantas - CRB-15/324

DOUGLAS ARTHUR DE ABREU ROLIM

"Dashboards para Desenvolvimento de Aplicações e Visualização de Dados para Plataformas de Cidades Inteligentes"

Esta Dissertação foi julgada adequada para a obtenção do título de mestre em Sistemas e Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte.

Presidente: **Dra. THAIS VASCONCELOS BATISTA**
(Orientadora - UFRN)

Profª. Drª. Anne Magaly de Paula Canuto
(Coordenadora do Programa)

Banca Examinadora

Examinador Interno: **Dr. EVERTON RANIELLY DE SOUSA CAVALCANTE**
(UFRN)

Examinador Interno: **Dr. NÉLIO ALESSANDRO AZEVEDO CACHO** (UFRN)

Examinadora Externa: **Dra. ROSSANA MARIA DE CASTRO ANDRADE** (UFC)

Março, 2020

Resumo

As aplicações de cidades inteligentes são inerentemente caracterizadas pela integração de dados de fontes heterogêneas e pela necessidade de considerar informações geográficas que representam o espaço urbano do mundo real. Visando atender tais necessidades, algumas plataformas foram propostas nos últimos anos, oferecendo serviços comuns para facilitar o desenvolvimento de aplicações de cidades inteligentes. No entanto, a maioria das plataformas existentes não oferecem interfaces de alto nível para fornecerem aos desenvolvedores ferramentas adequadas que podem reduzir a complexidade do desenvolvimento de aplicações, nem uma interface para organizar a visualização de dados para os usuários finais. Com o objetivo de lidar com essas limitações, este trabalho propõe *dashboards* baseados na *Web* para dar suporte ao desenvolvimento e visualização de dados em aplicações de cidades inteligentes: o primeiro é adaptado para desenvolvedores de aplicações, enquanto o segundo é adequado para usuários visualizarem dados. Este trabalho: (i) propõe uma arquitetura para as interfaces de alto nível da plataforma de cidade inteligente que considera dados georreferenciados; (ii) implementa essa arquitetura no contexto do *middleware Smart Geo Layers* (SGeoL), incluindo interfaces de *dashboard* específicas para desenvolvedores de aplicações e usuários interessados em aplicações criados usando a plataforma; (iii) discute como os dashboards definidos nesse trabalho são usados em aplicações reais, no contexto do SGeoL, que é uma plataforma que combina dados georreferenciados, resolve problemas de interoperabilidade e heterogeneidade e atualmente é aplicada no contexto da cidade de Natal. Esse trabalho também apresenta os resultados de um experimento controlado realizado com os usuários visando avaliar os benefícios trazidos pelos *dashboards* propostos em termos de esforço de desenvolvimento de aplicações e usabilidade.

Palavras-chave: Desenvolvimento de Aplicações para Cidades Inteligentes. SGeoL. *Dashboard de Desenvolvimento. Dashboard de Visualização.*

Abstract

Smart city applications are inherently characterized by the integration of data from heterogeneous sources and the need of considering geographical information that represents the real-world urban space. To address these concerns, some platforms have been proposed in recent years offering common services and facilities to ease the development of smart city applications. Nonetheless, the existing platforms do not offer high-level interfaces to provide developers with proper tools that could reduce the complexity of developing applications, neither an interface to organize data visualization to end-users. Aiming at tackling such limitations, this work presents Web-based dashboards to support development and data visualization in smart city applications: the former is tailored to application developers, whereas the latter is suited to visualize data. This research: (i) proposes an architecture for the smart city platform interface that considers georeferenced data; (ii) implements such architecture in the context of Smart Geo Layers (SGeoL) middleware, including specific dashboard interfaces for application developers and users interested in applications built using the platform; (iii) discuss how the dashboards defined in this work are used in real applications, in the SGeoL context, which is a platform that combines georeferenced data, adresses interoperability and heterogeneity problems, and it is currently used in the context of the city of Natal. A controlled experiment with users was performed aimed to assess the benefits brought by the proposed dashboards in terms of application development effort and usability.

Keywords: Internet of Things. Application Development. Smart cities. SGeoL. Dashboards.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface (Interface de Programação de Aplicações)
CAD	Computer Aided Design (Desenho assistido por computador)
CoAP	Constrained Application Protocol (Protocolo de Aplicação Restrita)
CE	Comunidade Europeia
DXF	Drawing Exchange Format (Formato de troca de desenho)
FI	Future Internet (Internet do Futuro)
GE	Generic Enablers (Habilitadores Genéricos)
GQM	Goal-Question-Metric (Objetivo-Pergunta-Métrica)
HTTP	Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto)
IBGE	Instituto Brasileiro de Geografia e Estatística
IM	Identity Manager
IPTU	Imposto Predial e Territorial Urbano
IoT	Internet of Things (Internet das Coisas)
KPI	Key Performance Indicator (Indicadores Chave de Desempenho)
LD	Linked Data (Dados Vinculados)
LGIMM	Layered-based Geographic Information Management Module (Módulo de gerenciamento de informações geográficas baseado em camadas)
MPRN	Ministério Público do Rio Grande do Norte
MQTT	Message Queuing Telemetry Transport (Transporte de Telemetria de Enfileiramento de Mensagens)
NGSI	Next Generation Service Interfaces (Arquitetura de Interfaces de Serviço de Próxima Geração)
NGSI-LD	Next Generation Services Interfaces Linked Data (Arquitetura de Interfaces de Serviço de Próxima Geração de Dados Vinculados)

PDP	Policy Decision Point (Ponto de decisão política)
QP	Questão de Pesquisa
RDF	Resource Description Framework (Estrutura de descrição de recursos)
SAEB	Sistema Nacional de Avaliação da Educação Básica
SEMURB	Secretaria Municipal de Meio Ambiente e Urbanismo
SEMUT	Secretaria Municipal de Tributação
SESED	Secretaria Estadual da Segurança Pública e da Defesa Social
SGBD	Data Base Management System (Sistema de Gerenciamento de Banco de Dados)
SUS	System Usability Scale (Escala de Usabilidade do Sistema)
TIC	Tecnologias da Informação e Comunicação
UE	União Européia
UFRN	Universidade Federal do Rio Grande do Norte

Sumário

1 Introdução

Este capítulo apresenta o contexto do tema deste trabalho, bem como a motivação e os objetivos. A seção 1.1 apresenta o contexto do estudo. A seção 1.2 discorre sobre a motivação do trabalho. A seção 1.3 discute os objetivos do trabalho; a seção 1.4 apresenta uma visão geral do trabalho; a seção 1.5 introduz as questões de pesquisa e, por fim, a seção 1.6 apresenta a organização do trabalho.

1.1 Contexto

As cidades e comunidades ao redor do mundo têm enfrentado novos tipos de problemas trazidos por uma urbanização em rápido crescimento, como poluição, congestionamentos, restrições de energia, falta de infra-estrutura, etc. O surgimento e a incorporação dos princípios do desenvolvimento sustentável no planejamento e gestão urbana elevaram a atenção dos efeitos negativos desta urbanização.

O conceito emergente de cidade inteligente foi reconhecido como um meio de alcançar tal sustentabilidade, alavancando os avanços na Tecnologia da Informação e Comunicação (TIC) (JING et al., 2019). Uma gestão urbana focada em TIC é baseada na integração de informações provenientes de diversos sistemas complexos e em interação que evoluem em várias escalas espaço-temporais e que atendem a domínios específicos. Nesse âmbito, vários desafios cercam o ecossistema de cidades inteligentes, tanto na fase de desenvolvimento de aplicações, quanto no uso das tecnologias relacionadas com o gerenciamento de informações geográficas e no consumo da enorme quantidade de dados heterogêneos que geralmente estão disponíveis nesse cenário.

Nesse contexto, de acordo com Teixeira et al. (2011), plataformas de *middleware* têm surgido como soluções promissoras para facilitar o desenvolvimento de aplicações, provendo interoperabilidade para possibilitar a integração de dispositivos, pessoas, sistemas e dados, e uma série de serviços adicionais necessários para aplicações.

Entretanto, o desenvolvimento de aplicações é uma tarefa custosa e propensa a erros, no qual os desenvolvedores precisam lidar com as complexidades relacionadas com a organização geoespacial da cidade, a plataforma de *middleware* subjacente e o ambiente de programação. Do ponto de vista dos usuários finais, os gestores urbanos e os cidadãos precisarão fazer a avaliação correta das situações urbanas, sendo necessário lidar com o grande número de dados geoespaciais gerados diariamente, entendê-los e saber como eles estão relacionados.

De acordo com Santana et al. (2016), embora as plataformas de *middleware* de cidades inteligentes potencialmente apoiem o desenvolvimento e a integração de aplicações para cidades inteligentes, vários desafios significativos permanecem abertos, principalmente

relacionados à necessidade de ter um amplo conhecimento das especificidades da plataforma para criar aplicações e organizar a visualização de dados para usuários finais. Algumas plataformas existentes herdam a ideia bem conhecida de *middleware* baseado na Internet das Coisas (IoT), que fornece *mashups* de fluxo de dados baseados na Web, permitindo que os desenvolvedores combinem fluxos de dados e os visualizem através de aplicações Web (KLEINFELD et al., 2014).

No entanto, as necessidades do desenvolvimento de aplicações de cidades inteligentes vão além do uso de *mashup*, exigindo suporte fácil para o gerenciamento de dados geoespaciais, integrando informações heterogêneas da cidade à sua localização geográfica e consultando, compondo, e exibindo dados de cidades inteligentes.

1.2 Motivação

Como visto na seção 1.1, as aplicações para cidades inteligentes são inerentemente caracterizadas pela integração de dados de fontes heterogêneas e pela necessidade de considerar informações geográficas que representam o espaço urbano do mundo real. Para tratar esses aspectos, algumas plataformas foram propostas nos últimos anos, oferecendo serviços e instalações comuns para facilitar o desenvolvimento de aplicações de cidades inteligentes.

De acordo com Santana et al. (2018), um dos propósitos das plataformas para Cidades Inteligentes é também facilitar o desenvolvimento de aplicações dispondo de ferramentas que auxiliem no desenvolvimento de aplicações. No entanto, a maioria das plataformas existentes não oferecem interfaces gráficas de alto nível que forneçam aos desenvolvedores ferramentas adequadas que podem reduzir a complexidade do desenvolvimento de aplicações, nem uma interface para organizar a visualização de dados para os usuários finais.

Além disso, desenvolver aplicações para cidades inteligentes fazendo uso desta plataformas pode ser uma tarefa complexa para usuários sem conhecimento aprofundado, visto que é preciso lidar com algumas situações como: visualização dos dados geográficos, gerenciamento da segurança, estruturação dos dados, gerenciamento de camadas e entidades, edições de pontos ou vetores geográficos, conversões de tipo de dados geográficos, cruzamento de dados, importações de dados, entre outros. Ainda é comum que os desenvolvedores precisem criar novas aplicações para lidar com alguns destes problemas ou mesmo para explorar os recursos que, comumente, um outro desenvolvedor já vivenciou ao usar a plataforma. Isto acaba, por vezes, resultando no desenvolvimento de soluções adaptadas às suas necessidades e públicos específicos.

Tendo em vista estas dificuldades e o fato de que nem todo usuário é um desenvolvedor com conhecimento técnico profundo, é indispensável que quaisquer desenvolvedores consigam criar novas aplicações ou que, ao menos, usufruam das contribuições das apli-

cações existentes. Nesse sentido, o desenvolvimento de uma interface de alto nível pode superar os problemas supracitados e, de forma geral, fornecer facilidades e otimizar o trabalho de desenvolvedores não especialistas, até mesmo provendo a comunicação dos usuários comuns com os dados de seu interesse, o que é imprescindível para a difusão da plataforma.

Entretanto, como plataformas para cidades inteligentes contam com o armazenamento e o processamento de grande volume de variados tipos de informações, o desenvolvimento de uma interface torna-se complexa devido à quantidade e diversidade de dados. Tendo em vista esta grande quantidade e a variedade de dados, este cenário pode ser ideal para o desenvolvimento de uma interface de *dashboard*. *Dashboards* são ferramentas que ajudam as pessoas a identificarem, visualmente, tendências, padrões e anomalias, possibilitando o raciocínio sobre o que veem e ajudando a orientá-las em direção a decisões efetivas (BRATH; PETERS, 2004)

1.3 Objetivos

Nas subseções seguintes serão apresentados o objetivo geral e os objetivos específicos.

1.3.1 Objetivo geral

O objetivo deste trabalho é propor uma arquitetura conceitual para dois *dashboards* baseado na web, *dashboard* de desenvolvimento e *dashboard* de visualização, e implementá-la no contexto de uma plataforma de desenvolvimento de aplicações para cidades inteligentes.

1.3.2 Objetivos específicos

Como objetivos específicos podem ser citados:

- Propor uma arquitetura conceitual para os *dashboards* de visualização e desenvolvimento;
- Concretizar a arquitetura proposta através de uma implementação da mesma no contexto de uma plataforma de desenvolvimento de aplicações para cidades inteligentes;
- Realizar uma avaliação dos *dashboards* propostos com relação a esforço de desenvolvimento de aplicações e usabilidade.

1.4 Visão geral do trabalho

Os *dashboards* baseados na Web para o desenvolvimento de aplicações para cidades inteligentes são compostos por duas interfaces de *dashboards*, sendo: (i) o *dashboard de*

desenvolvimento fornece suporte para o desenvolvimento de aplicações via uma interface amigável que reduz a necessidade de escrever códigos complexos e específicos da plataforma subjacente de desenvolvimento para cidades inteligentes; (ii) o *dashboard de visualização*, fornece uma interface com o *dashboard* geoespacial, incluindo o mapa da cidade e os dados plotados sobre ele, além de serviços para vincular, consultar e analisar informações da cidade através de localização geográfica. Enquanto o *dashboard* de desenvolvimento reduz as barreiras técnicas dos desenvolvedores, o *dashboard* de visualização permite suporte à decisão e monitoramento em tempo real de vários dados da cidade.

Os *dashboards* propostos podem ser instanciados sob qualquer plataforma de desenvolvimento de aplicações para cidades inteligentes, no entanto, para sua validação, foi realizada uma implementação sob a plataforma de desenvolvimento para cidades inteligentes, *Smart Geo Layers* (SGeoL) (SOUZA et al., 2018).

O SGeoL é uma plataforma de *middleware* que integra vários tipos de fontes de dados governamentais e não governamentais, fornecidos por indivíduos e empresas, reunindo uma série de *camadas georreferenciadas*. O conceito de camadas é originado dos Sistemas de Informação Geográficos (SIG), onde cada *camada* corresponde a um domínio e, juntas, fornecem um vasto conjunto de informações. As camadas possuem uma origem e algumas peculiaridades. Por exemplo, a camada de escolas agrupa informações sobre as diferentes escolas de uma cidade, como: sua localização geográfica, número de alunos, número de professores, notas de alunos etc. No SGeoL, cada camada oferece uma visão diferente da cidade e elas pode se sobrepor, com base em suas propriedades geográficas. A implementação dos *dashboards* é realizada usando tecnologias bem consolidadas de ambientes da Web.

O SGeoL usa o FIWARE[1], uma plataforma desenvolvida na Comunidade Europeia (CE) que é utilizada em diversas aplicações para cidades inteligentes nos mais variados cenários. O FIWARE atende a uma ampla gama de requisitos considerados relevantes para cidades inteligentes e provê diversos componentes funcionais genéricos, extensíveis e reutilizáveis, que podem facilitar o desenvolvimento de aplicações, fazendo com que essa plataforma seja uma opção interessante a ser adotada nesse contexto (ROLIM et al., 2018). A Figura 1 apresenta, de forma didática, as interconexões do FIWARE, SGeoL e dos *dashboards* de desenvolvimento e visualização.

Figura 1 – FIWARE, SGeoL e os Dashboards de desenvolvimento e visualização

[1] <https://www.fiware.org/>

A escolha da plataforma SGeol para ser usada nesse trabalho justifica-se pelo fato do trabalho estar inserido no contexto de um projeto amplo, o SmartMetropolis. Tal plataforma tem sido usada no desenvolvimento de várias aplicações, por exemplo: para a Prefeitura Municipal de Natal, para a Secretaria Estadual da Segurança Pública e da Defesa Social (SESED) do Rio Grande do Norte, para o Ministério Publico do Rio Grande do Norte (MPRN) e para a Universidade Federal do Rio Grande do Norte (UFRN).

1.5 Questões de Pesquisa

Com base nos objetivos definidos, este trabalho é guiado por três Questões de Pesquisa (QP):

- **Questão 1** - Quais elementos são necessários para a construção *dashboards* de desenvolvimento e visualização para plataformas de Cidades Inteligentes centradas em dados geográficos?

O objetivo dessa questão de pesquisa é estabelecer quais elementos necessários para construir *dashboards* com foco em desenvolvimento e em visualização e como esses elementos devem ser definidos.

- **Questão 2** - O uso do *dashboard* de desenvolvimento facilita o desenvolvimento de aplicações?

O objetivo dessa questão de pesquisa é verificar se o *dashboard* de desenvolvimento proposto é capaz de facilitar o desenvolvimento de aplicações para cidades inteligentes, como também administrar usuários e seus papéis, gerenciar camadas e entidades, facilitar a interação entre importações e atualizações dos dados em seus variados tipos e formatos, proporcionar interação confortável aos seus desenvolvedores.

- **Questão 3** - O uso do *dashboards* de visualização facilita o acesso e a compreensão dos dados?

O objetivo dessa questão de pesquisa é investigar se o *dashboard* de visualização permite que interessados possam usufruir das aplicações criadas com o uso da plataforma, e ao mesmo tempo, possibilitar a fácil compreensão dos dados por meio de *dashboards*, por meio de mapas através da exibição em múltiplas camadas dos dados georreferenciados e através consultas amigáveis a dados.

1.6 Organização do trabalho

O presente trabalho está estruturado da seguinte forma: o capítulo 2 descreve os conceitos básicos necessários para a compreensão desse trabalho; o capítulo 3 apresenta os

dashboards de desenvolvimento e visualização, como também detalha suas funcionalidades, a arquitetura e seu desenvolvimento e implementação; o capítulo 4 discute os trabalhos relacionados encontrados na literatura; o capítulo 5 apresenta o processo de avaliação e os resultados e, por fim, o capítulo 6 apresenta o cronograma para a conclusão do trabalho e propõe um modelo de avaliação para da plataforma proposta.

2 Fundamentação Teórica

Este capítulo descreve os conceitos necessários para a compreensão deste trabalho. A seção 2.1 discute sobre *dashboards*, abordando o conceito e sua sistemática de desenvolvimento, como também os tipos de *dashboard*; a seção 2.2 explica a plataforma FIWARE; e, por fim, a seção 2.3 discorre sobre o SGeoL, apresentando também sua arquitetura.

2.1 Dashboards

De acordo com Rasmussen, Bansal e Chen (2009), *dashboard* (ou painel de informação) é uma ferramenta de análise de dados utilizada com a finalidade de rastrear e exibir indicadores-chave de desempenho (KPI), métricas e pontos-chave de dados provendo uma forma organizada de visualização de informações.

Um *Dashboard* sintetiza informações, evitando excessos e operando de modo a reunir e exibir dados de maneira uniforme. Dado que a capacidade de processamento humano é limitada e, muitas vezes, tendenciosa, *dashboards* tentam evitar quaisquer possíveis problemas na leitura de dados, ajudando o usuário a entender as informações. É neste ponto que os *dashboards* diferenciam-se de outros métodos de leitura e análise de dados, que operam compilando outros tantos serviços. Na computação, utiliza-se diversos tipos de informações: estado atual de serviços e dispositivos IoT, dados geográficos, estatísticas geradas a partir da análise de grande volume de dados (Big Data), relatórios de dados em forma de gráficos, etc., o que acaba por segregar, em muitas plataformas, os dados obtidos, deixando que, por muitas vezes, estes se percam.

A sistemática da elaboração de um *dashboard* funciona seguindo uma ordem que combina coleta de dados e orientações sobre como traduzi-los de modo a aproveitá-los maximamente para, a partir daí, trabalhar na elaboração e aplicabilidade destes, garantindo efetividade das métricas utilizadas.

A integração da tríade de dados, processos e diferentes pontos de vista é um fator indispensável para criação de *dashboards*, tendo em vista ser necessário entender como funcionam os dados e qual a conjuntura que estes são utilizados. Em seguida, então, os dados podem ser monitorados e correlacionados com outros tipos de informação para que, desse modo, seja possível avaliar o resultado obtido de acordo com as expectativas dos interessados.

A Figura 2 ilustra os passos necessários para o desenvolvimento de um *dashboard*. Para isso é necessário que, primeiramente, sejam selecionadas as métricas a serem utilizadas. Em seguida, deve-se preencher o *dashboard* com dados e, então, estabelecer relações causais entre os itens dispostos neste para que somente informações indispensáveis sejam reunidas e de modo que se façam compreensíveis a todos os envolvidos (PAUWELS et al., 2009).

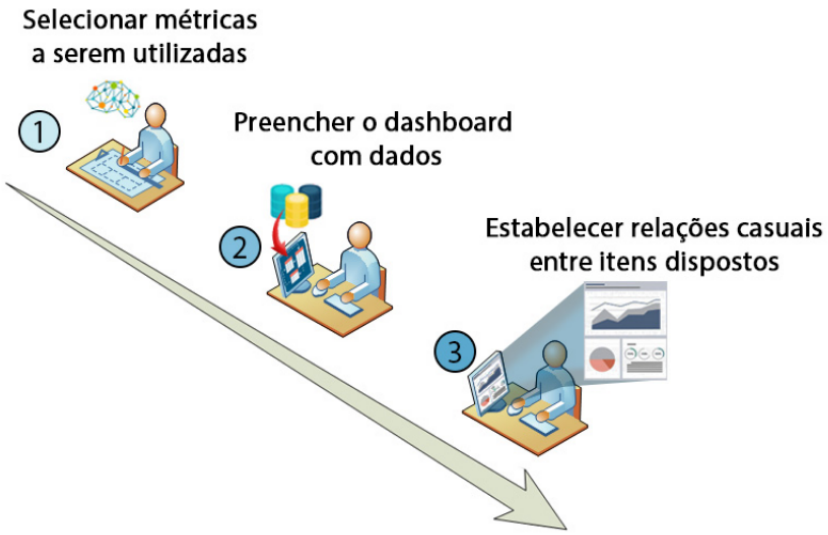

Figura 2 – Passos para o desenvolvimento de um *dashboard*

2.1.1 Dashboards: operacional, estratégico, analítico ou tático

De acordo com Few (2006), Pappas e Whitman (2011), Dobraja, Kraak e Engelhardta (2018) e Rasmussen, Bansal e Chen (2009), os *dashboards* dividem-se em quatro tipos. São eles:

1. operacional;
2. estratégico; e
3. analítico ou tático;

O primeiro, como o nome aponta, funciona em comunhão dos interessados, que analisam os dados disponibilizados e podem detectar um problema – que terá solução pensada por um gestor – e atuarão otimizando o serviço ofertado, garantindo rápida resposta e eficiência no atendimento. Além disso, os *dashboards* operacionais também são usados para monitorar processos, atividades ou eventos complexos. Normalmente, o *dashboard* fornece atualizações mais recentes que ilustram o estado atual da informação. Devido à dinamicidade deste tipo de *dashboard*, eles são usados, geralmente, para descobrir problemas e tomar medidas para a solução. Por exemplo: um analista pode verificar a disponibilidade de serviços, ocorrência de erros ou falhas ou processamento em uso de servidores e, caso necessário, optar por executar determinadas ações.

O segundo tipo de *dashboard* é de interesse da alta gerência de uma empresa. Atualizado com menos frequência, o *dashboard* estratégico inclui medidas globais, externas, de tendência e crescimento, atuando com objetivos de longo prazo. Para tanto, o *dashboard* compila dados para que estes sejam comparados e evidenciem a evolução obtida, exibindo-os, aos seus interessados, que verão de que forma seus desempenhos individuais refletem na

construção e finalização do trabalho. Ainda, *dashboards* estratégicos geralmente concentram vários tipos de dados que compõem uma informação, de forma a garantir que estes estejam alinhados estrategicamente. Exemplo: um gestor da educação monitora o desempenho de inúmeras escolas de um município dentro de determinado período, sendo possível, dessa forma, avaliar a progressão das escolas e realizar ações preditivas.

O terceiro tipo de *dashboard* é de interesse da alta gerência e é o que atuará sobre o do tipo operacional, pois permite cruzar informações importantes, visualizar onde especificamente está ocorrendo uma falha e estabelecer decisões que levam a mudanças operacionais de modo a mobilizar somente recursos estritamente necessários para tal, o que evita custos desnecessários e otimiza a objetividade de suas ações. Dessa forma, pode-se considerar um dashboard analítico ou tático como o ponto de interseção dos *dashboards* estratégico e operacional, compartilhando atributos dos dois, de modo a permitir uma possível exploração subjacente dos dados, como característicos de *dashboards* operacionais. Isto é, não apenas para ver o que está acontecendo, mas para examinar as causas. Por exemplo: um gestor da educação que monitora o desempenho de inúmeras escolas de um município de forma estratégica, mas com base em detalhamento operacional, observa resultados menos favoráveis em um bairro específico deste município e pode promover ações para descobrir a fonte da variação.

Do ponto de vista de uso dos tipos de *dashboards* apresentados nessa seção, a depender do contexto dos dados dispostos, suas relações e a frequência com que são atualizados, todos os tipos de *dashboards* podem estar envolvidos na apresentação dos dados com uso do *dashboard* de visualização proposto neste trabalho. Por exemplo, um *dashboard* operacional é geralmente usado para representar o desempenho de algumas métricas, especialmente nas áreas ambiental e econômica. Para aplicações relacionadas à política, como governo urbano e gerenciamento urbano, o *dashboard* estratégico é capaz de apoiar a tomada de decisão e a predição, sendo aplicado tanto no governo quanto no gerenciamento urbano. O *dashboard* analítico é adequado para campos socialmente relacionados, como áreas urbanas e áreas tráfego, de modo a permitir uso de ferramentas para uma análise de desempenho e para buscar os padrões e as razões para os padrões.

2.1.2 Dashboard Geoespacial

Em sua revisão sistemática, Jing et al. (2019) descreve o um *dashboard* geoespacial como sendo uma interface interativa baseada na Web e suportada por uma plataforma que combina mapeamento, análise espacial e visualização com ferramentas de inteligência.

Para a construção de um *dashboard* geospecial, Jing et al. (2019) define três atributos necessários, são eles:

1. escalabilidade;
2. interoperabilidade; e

3. portabilidade.

A escalabilidade refere-se à capacidade de adicionar e remover hardware (incluindo computadores, dispositivos e sensores), módulos de software e componentes da interface do usuário, para usuários sem afetar a disponibilidade do sistema. A tecnologia baseada em pequenos componentes reutilizáveis (*widgets*), é um exemplo de um *design* escalável que suporta a personalização de alto nível das funcionalidades de uma aplicação e que será bastante discutida neste trabalho.

A interoperabilidade é uma característica crítica para construção de aplicações que utilizem *dashboards* geoespaciais, tendo em vista as suas características multidimensionais e espaço-temporais. Como soluções de interoperabilidade, por exemplo, pode-se citar a o uso de uma especificação de modelo de dados e plataformas de *middleware*.

A portabilidade é a capacidade de prover acesso e entendimento dos dados geoespaciais, utilizando-se de um conjunto de métricas que contribuam com a compreensão do desempenho da cidade. Dessa forma, faz necessário que interfaces de *dashboard* geoespaciais sejam capazes de tornar acessível a portabilidade de um grande volume de dados, permitindo ao usuários finais entendimento do desempenho das cidades inteligentes.

Jing et al. (2019) também analisa os *dashboards* geoespaciais existentes, tais como, *My City Dashboard* (USURELU; POP, 2017), *Dublin Dashboard* (MCARDLE; KITCHIN, 2016), *London CityDashboard*[1], entre outros, definindo uma arquitetura em três camadas para *dashboards* geoespaciais, sendo elas: camadas de dados, análise e apresentação. A Figura 3 apresenta esta arquitetura de forma conceitual.

A camada de dados gerencia os dados geoespaciais e não geoespaciais usando bancos de dados específicos e contém os modelos de dados – que são responsáveis pela organização dos dados e podem mapear um conjunto de diferentes métricas para formular indicadores. A camada de modelo inclui vários modelos de análise para os indicadores, que é o núcleo do *dashboard* geoespacial. A terceira camada é a camada de visualização, que é a interface na qual os usuários interagem, responsável pela apresentação dos dados.

Quanto ao *design* das interfaces, Jing et al. (2019) observa dois estilos principais, o estilo de uma página e o estilo de detalhamento. O estilo em uma página tem como característica apresentar todas as métricas e indicadores em uma única página, permitindo que os usuários os visualize todos de uma vez. Já o estilo de detalhamento é projetado para fornecer informações de observação, onde há muitas métricas e indicadores que não podem ser exibidos em uma tela única, geralmente são projetados como menu. A Figura 4 apresenta os tipos de estilos principais.

Dashboard geoespaciais podem reunir, visualizar, analisar e orientar sobre desempenho urbano para apoiar o desenvolvimento sustentável de cidades inteligentes. Justamente esse conceito de *dashboard* geoespacial é usado no *dashboard* de visualização deste trabalho.

[1] http://citydashboard.org/

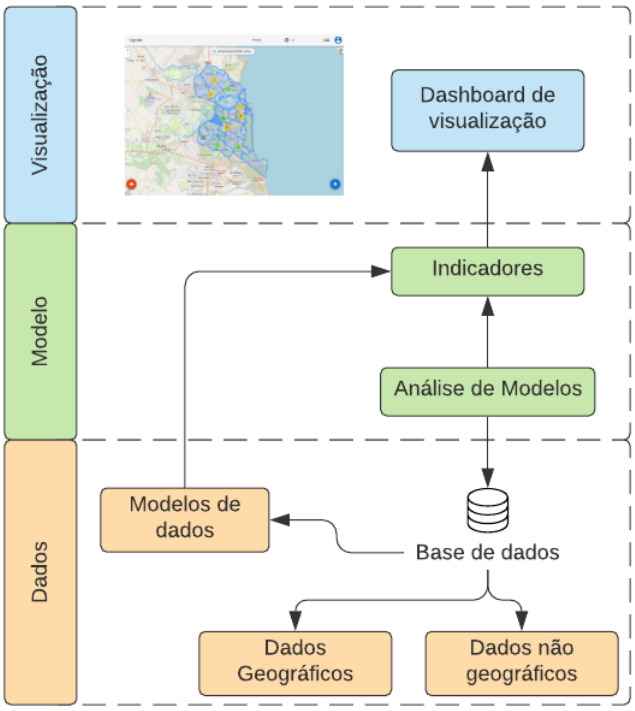

Figura 3 – Arquitetura conceitual de *dashboards* geoespaciais

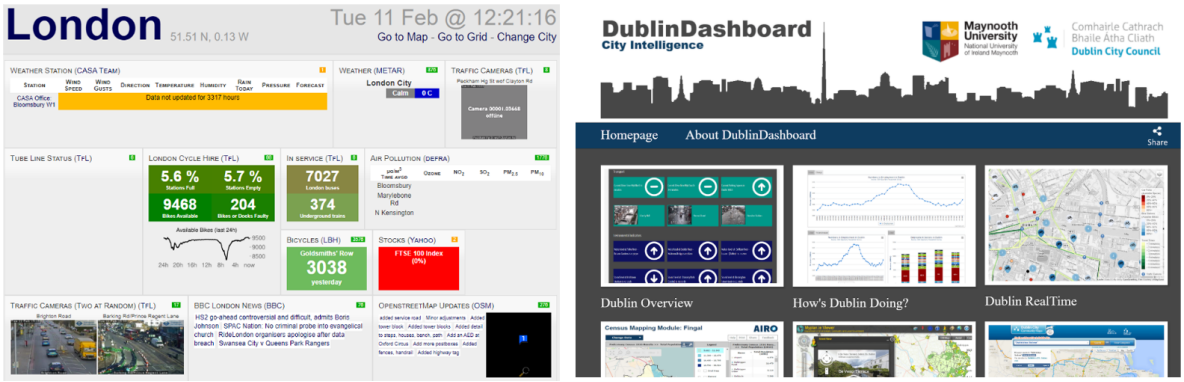

Figura 4 – Impressões de telas dos *dashboards* geoespaciais. Do lado esquerdo, *London Dashboard*, representando o estilo de uma página. Do lado direito, *Dublin Dashboard*, representando o estilo de detalhamento.

2.2 FIWARE

A FIWARE é uma plataforma de *middleware* aberta, desenvolvida com suporte da União Européia (UE), que visa prover uma infraestrutura para desenvolvimento e provisão de aplicações para a Internet do Futuro (FI). De acordo com FIWARE (2011), essa plataforma oferece um conjunto poderoso de APIs (*Application Programming Interfaces*) que facilitam o desenvolvimento de aplicações inteligentes para vários setores. Além disso, ela disponibiliza uma implementação de referência de código aberto de cada um de seus componentes, os chamados *Generic Enablers* (GEs).

Estes GEs são agrupados em capítulos técnicos de acordo com o conjunto de funcionalidades às quais estão relacionadas, como apresentado na Figura 5.

Figura 5 – Capítulos técnicos dos *Generic Enablers* (GEs)

O gerenciador de informações de contexto do FIWARE, o *Context Broker*, é o componente central da plataforma. A FIWARE também provê um conjunto de componentes complementares:

- Interface com a Internet das Coisas (IoT), robôs e sistemas de terceiros: que possibilita capturar atualizações sobre informações de contexto e traduzir as atuações necessárias.
- Gerenciamento e monetização de dados de contexto/API: fornece suporte ao controle de uso e a oportunidade de publicar e monetizar parte dos dados de contexto gerenciados.
- Processamento, análise e visualização de informações de contexto: provê mecanismos para lidar com o comportamento inteligente esperado das aplicações e/ou auxiliando os usuários finais na tomada de decisões.

Por sua vez, estes GEs podem ser montados juntamente com outros componentes de plataformas de terceiros para acelerar o desenvolvimento de aplicações inteligentes.

A versão atual do *Context Broker* do FIWARE é implementada sob o padrão estabelecido pela *Open Mobile Alliance*, o *Next Generation Service Interfaces* (NGSI[2]). Essa implementação permite a interoperabilidade na troca de informações, e conta com uma API de gerenciamento de informações de contexto com base nos recentes avanços nos

[2] <https://knowage.readthedocs.io/en/6.1.1/user/NGSI/README/index.html>

dados vinculados (LD), chamado NGSI-LD[3] (*Next Generation Services Interfaces Linked Data*), definido no RDF (*Resource Description Framework*) e tem como base o JSON-LD[4].

Além de organizar os dados em entidades, propriedades e relacionamentos, o NGSI-LD define como estes dados relacionam-se, conforme ilustra a Figura 6.

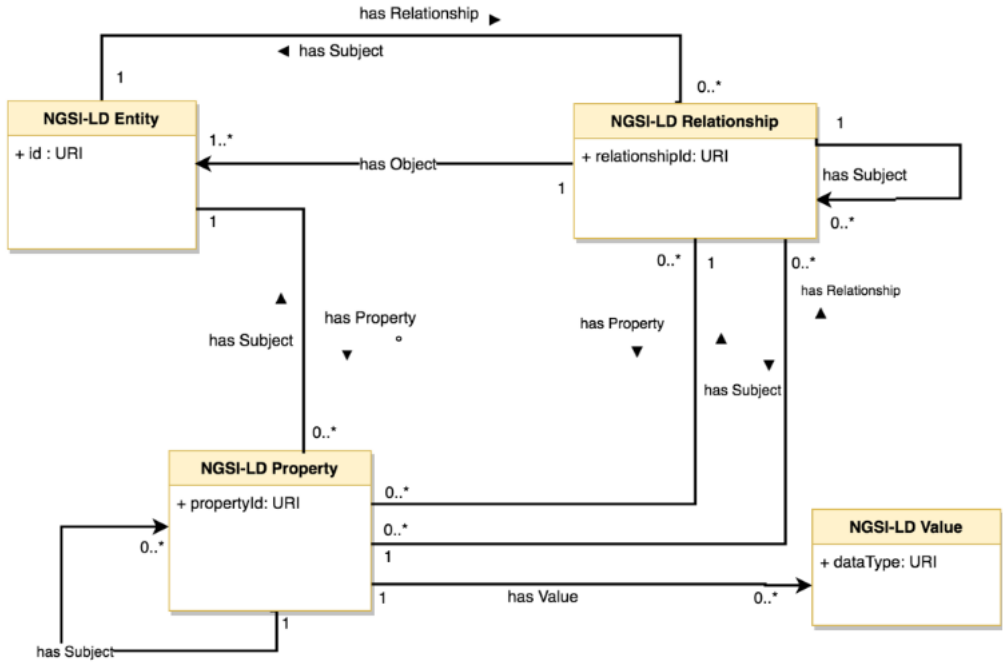

Figura 6 – Dados no modelo NGSI-LD e seus relacionamentos

Além disso, o NGSI-LD define como a API deve ser usada na manipulação e consulta de dados. Por sua vez, a API proporciona: (i) gerenciamento de entidades (criação, consulta, atualização e destruição de dados); (ii) consultas em dados geográficos; e (iii) consultas baseadas em séries de atributos temporais.

Uma outra especificação refere-se a comunicação baseada em *Publish/Subscribe*, que possibilita ao usuário se subscrever para ser notificado sobre as alterações feitas nas entidades selecionadas.

O NGSI-LD também possui uma estrutura comum de contexto que permite a reutilização e o compartilhamento de informações entre as aplicações, sendo possível, dessa forma, criar uma massa crítica de serviços avançados. A Figura 7 ilustra esse aspecto, onde é definido um contexto comum para vários serviços (à direita da figura), e onde é contrastada com a inflexibilidade do contexto definido de forma independente (à esquerda).

As informações de contexto são consideradas informações relevantes sobre entidades, suas propriedades (temperatura, localização ou qualquer outro parâmetro) e seus relacionamentos com outras entidades. As entidades podem ser representações de objetos

[3] <https://fiware-datamodels.readthedocs.io/en/latest/ngsi-ld_howto/index.html>
[4] <https://json-ld.org/

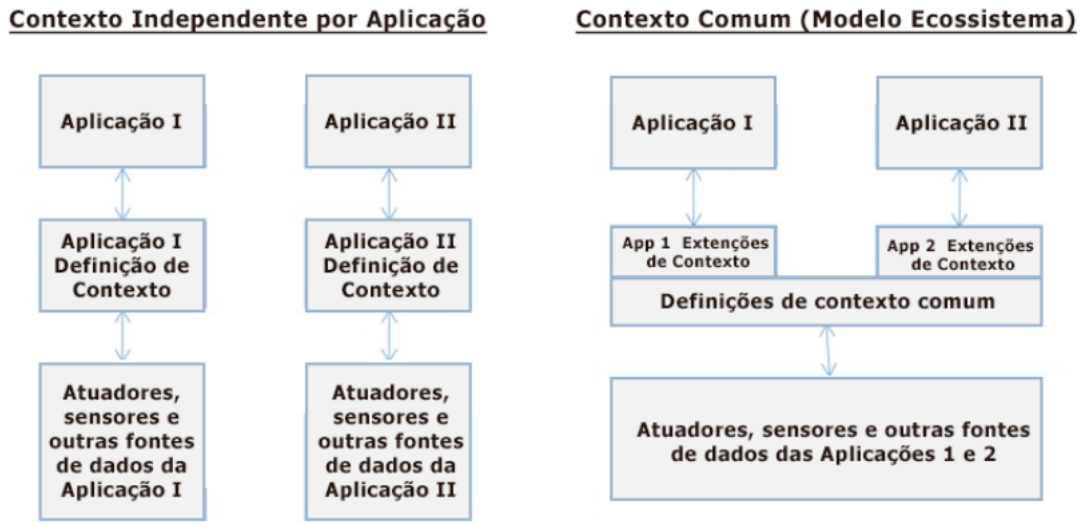

Figura 7 – Contexto Independente x Contexto Comum (Modelo Ecossistema)

com palavras reais, mas também podem ser noções mais abstratas, como uma entidade legal, corporação ou grupos de entidades.

Em resumo, as informações de contexto fornecem uma base teórica para o raciocínio automatizado sobre as características dos sistemas que eles representam. Dessa maneira, é possível comunicar-se em um formato legível onde um determinado recurso pode ser idêntico ou não a um outro recurso. Estes recursos possam estar em servidores diferentes e publicados por diferentes autores.

O uso do modelo NGSI-LD é ideal para promover interoperabilidade entre aplicações e dispositivos IoT, como também facilitar a localização e a troca de informações com bancos de dados abertos, aplicativos móveis e plataformas IoT. Entretanto, criar novas entidades usando dados de contextos ou ainda criar novos contextos pode exigir esforço demasiado do desenvolvedor. Por isso, são necessários meios para facilitar esse aspecto, com a finalidade de acelerar a criação de novas aplicações e a integração de dados.

2.3 SGeoL

O Smart Geo Layers (SGeoL) é uma plataforma de *middleware* voltada para o desenvolvimento de aplicações para cidades inteligentes, construída usando a FIWARE como infraestrutura subjacente. De acordo com Souza et al. (2018), o SGeoL tem a capacidade de integrar os mais diversos tipos de fontes de dados governamentais e dados colaborativos, fornecidos por indivíduos e empresas, reunindo uma série de camadas geográficas que fornecem um vasto conjunto de informações.

O SGeoL permite que os desenvolvedores possam criar aplicações com requisitos de: i) segurança; ii) escalabilidade; iii) privacidade; iv) suporte à processamento geográfico; v) suporte semântico; vi) gerenciamento de séries temporais de dados; vii) processamento

de eventos complexos; viii) suporte à conexão de dispositivos de IoT; ix) gerenciamento de dados de contexto, e; x) gerenciamento de grandes volumes de dados. Estas funcionalidades são providas por meio de APIs RESTful de alto nível que podem ser utilizadas para o desenvolvimento de uma infinidade de aplicações.

No SGeoL, as informações são agrupadas sob a forma de camadas. Cada camada oferece uma visão acerca da cidade. Por exemplo, a camada Escola agrupa informações referentes às escolas distribuídas no espaço urbano da cidade. A camada Saúde agrupa os dados referentes aos hospitais e postos de saúde, e assim por diante. Além disso, uma camada pode conter diversas entidades de contexto associadas a si, por exemplo, a camada Escola pode conter diversas entidades do tipo "escola" associadas a si, cada uma delas representando uma escola da cidade. Adicionalmente, os usuários interessados podem efetuar a composição de dados, criar suas próprias camadas e realizar consultas geoespaciais que envolvam dois ou mais tipos de entidade.

Cada camada oferece uma visão específica acerca da cidade e essas podem ser divididas em dois grupos principais, são elas:

1. camadas geográficas; e
2. camadas de domínio.

A primeira refere-se às camadas geográficas que representam as informações do território urbano e são responsáveis pelas entidades comumente presentes em modelos de mapeamento urbano, tais como: bairros, lotes, quadras, ruas, etc. Além disso, as camadas geográficas podem estar relacionadas com outras camadas geográficas, e elementos de camada entre si, assim como associar elementos de camada, ou camada, com entidades. Estes relacionamentos são descritos e apresentados da seguinte forma:

- *pertencente* a: representa a sobreposição de camadas. Por exemplo, um bairro pertence a um distrito, que pertence a um município.
- *parte de*: associar elementos de camadas geográficas a outros elementos de camadas distintas. Por exemplo, um elemento município pertencente a camada município faz parte individualmente de um elemento da camada estado.
- *vizinho de*: associa elementos da própria camada uns com os outros, representando a associação existente entre elementos vizinhos. Por exemplo, uma elemento da camada bairro é vizinho de um outro elemento de sua mesma camada, para representar que um bairro é vizinho de outro.

Em sua maioria, camadas geográficas são compostas de dados estáticos ou sem atualização constante, porém, existem camadas geográficas que podem ser atualizadas de forma dinâmica e frequente, por exemplo, distritos compõem cidades, cidades compõem estados e estados compõem países.

O outro nível de camadas recebe o nome de camadas de domínio e representam qualquer informação de domínio em uma cidade inteligente, como um conjunto de entidades da camada escolas, ou de saúde, ou de segurança, etc. Já as associações de uma camada de domínio a uma entidade específica de gerenciamento da cidade, permitem expressar a natureza dos domínios relacionados a uma cidade.

Ambas, camadas geográficas e de domínio, possuem elementos georreferenciados que permitem a localização em um determinado ponto do território urbano.

Em resumo, as camadas podem ser definidas como um conjunto de elementos (entidades) que possuem atributos geoespaciais, por exemplo, a camada escola pode conter diversas entidades do tipo "escola" associadas a si, cada uma delas representando uma escola da cidade. Estas entidades possuem atributos de relevância para os usuários que podem também ser usados para compor consultas relacionadas com outras entidades.

2.3.1 Arquitetura

A arquitetura do SGEOL é composta por diversos componentes que desempenham tarefas bem específicas. A Figura 8 apresenta a arquitetura do SGeoL com seus diversos módulos.

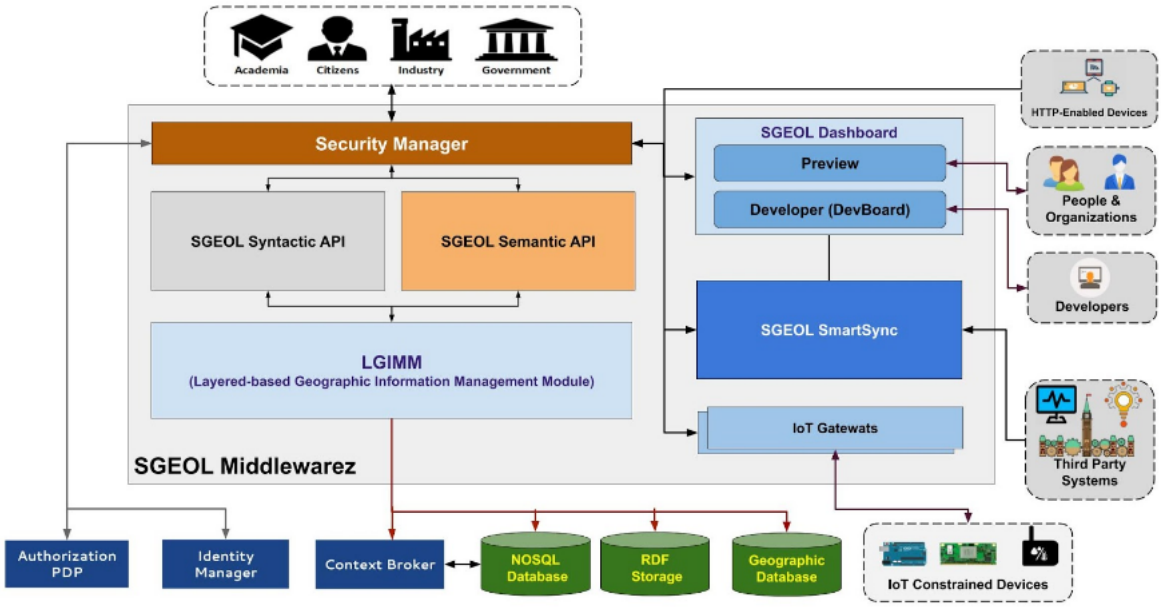

Figura 8 – Arquitetura do SGeoL com seus diversos módulos

Os módulos e componentes da arquitetura do SGeoL que estão envolvidos de forma direta com a proposta deste trabalho, são:

- *APIs (Syntactic API e Semantic API)*: provê um conjunto de serviços que permite a manipulação das camadas e entidades.

- *LGIMM (Layered-based Geographic Information Management Module)*: realiza a validação e persistência dos dados enviados pelos desenvolvedores e usuários na plataforma.
- *Security Manager*: fornece a comunicação entre as aplicações e a camada de segurança.
- *SmartSync*: permite a integração e sincronização de dados oriundos de sistemas de terceiros com a plataforma.

Estes componentes são apresentados nas subseções a seguir.

2.3.1.1 *Syntactic* API e *Semantic* API

O SGeoL dispõe de duas APIs para manipulação de camadas e entidades de contexto, sendo elas: *Syntactic* API e *Semantic* API. A *Syntactic* API adota o modelo de dados NGSI-LD, por meio dessa API os usuários podem manipular dados (inserir, atualizar, remover) e realizar consultas sintáticas, isto é, sem suporte semântico. Em geral as operações disponibilizadas por meio dessa API são processadas rapidamente. Já a *Semantic* API trabalha com o modelo de dados LGeoSIM (*Layered-based Georeferenced Semantic Information Model*) que estende o modelo de dados NGSI-LD, com o diferencial de que o mesmo oferece um melhor suporte semântico. Essa API permite manipular dados no SGeoL (inserir, atualizar, remover) além de realizar consultas semânticas. As operações disponibilizadas por meio dessa API são um pouco mais custosas se comparadas com as operações disponibilizadas por meio da *Syntatic* API, pois requer processamento adicional dos dados, a fim de garantir a consistência semântica dos dados, isto é, garantir que as informações obtidas através do usuário satisfaçam o modelo de dados semântico. Além disso, essas operações demandam a utilização de bases de dados baseadas em RDF que, em geral, têm desempenho inferior às bases de dados NoSQL.

2.3.1.2 LGIMM

O LGIMM é responsável pelas validações dos dados enviados para APIs do SGeoL, além da persistência e consultas de dados nas diversos Sistemas de Gerenciamento de Banco de Dados (SGBDs) utilizados. O LGIMM é composto internamente por diversos componentes com funcionalidades específicas, como por exemplo: (i) componentes responsáveis por gerenciar a persistência de dados de contexto e dados geográficos; (ii) componente para a realização de buscas sintáticas; (iii) componente a persistência e buscas de dados temporais. Esses componentes garantem uma melhor performance das APIs, como também possibilitam a realização de consultas envolvendo múltiplas camadas e suas entidades e um histórico dos dados de cada ação executada através das APIs.

2.3.1.3 *Security Manager*

O componente *Security Manager* é responsável por intermediar, de forma transparente, a comunicação entre os usuários e aplicações aos GEs do FIWARE responsáveis pela segurança para *Identity Manager* (IM) e o *Policy Decision Point* (PDP), o Keyrock e Authzforce, respectivamente.

O Keyrock é responsável por manter o registro dos usuários e aplicações conectadas ao SGEOL. Além disso, esse componente distribui e valida *tokens* de acesso concedidos aos usuários e aplicações conectados fazendo uso do protocolo OAuth 2.0. Já o Authzforce é responsável por gerenciar as políticas de acesso atribuídas aos usuários e aplicações conectadas ao SGeoL. Estas políticas especificam quais recursos (camadas) e operações HTTP (*GET*, *POST*, *PUT* e *DELETE*) os usuários e aplicações têm acesso.

Para prover esta comunicação, o componente *Security Manager* recupera os *tokens* de acesso do usuário e da aplicação, informados no cabeçalho HTTP da requisição, e os envia ao Keyrock para serem validados. Caso a requisição seja válida a requisição segue para a fase de autorização. Nessa fase, é verificado junto ao Authzforce se o usuário ou aplicação possui acesso à operação requisitada. Caso a requisição não seja validada ou o usuário não tenha permissão para realizar a operação, o *Security Manager* retornará uma resposta com código não autorizado.

2.3.1.4 SmartSync

O componente SmartSync é responsável por permitir a integração de dados oriundos de sistemas de terceiros com o SGeoL. Esse componente integra dados disponibilizados através de arquivos em formato de planilhas providos por aplicações ou banco de dados, como também de APIs providas por sistemas existentes em uma cidade. Estes dados disponibilizados, por sua vez, são recuperados, convertidos para o formato NGSI-LD e registrados no SGeoL, tornando-os disponíveis para outras aplicações.

Por exemplo, na educação, alguns dados podem receber atualizações continuamente como frequência, notas ou qualquer outros dados sensíveis as progressões dos alunos nas etapas de ensino. Dessa forma, é de extrema importância que estes dados estejam atualizados para que gestores realizem ações precisas, caso seja necessário.

O SmartSync é capaz de produzir e armazenar, em formato JSON, um arquivo que guarda um modelo de predefinição dos dados que devem ser importados. Esse modelo de predefinição tem como objetivo servir de base para as importações de dados, como para atualizações que podem ser agendadas e realizadas automaticamente pelo SmartSync. Sua principal função é guardar as informações da fonte de dados que será utilizada, como endereço da API ou arquivo que deve ser importado, e especificar quais dados providos por esta fonte devem ser importados.

Uma vez que um modelo de predefinição foi definido e uma primeira importação

foi efetuada, o SmartSync trata, dependendo do tipo da fonte provedora configurada no modelo, a manutenção desses dados através de duas formas: (i) atualização através de APIs; e (ii) atualização através do arquivos.

A primeira foca em realizar solicitações às APIs conforme periodicidade configurada também no modelo de predefinição. Para ter sempre os últimos dados da fonte, esta frequência deve ser menor que a frequência de atualização média dos dados na própria fonte. Por exemplo, pode-se pré-definir um modelo que acessará uma API através do verbo HTTP *GET* e que importe de todos os objetos retornados, porém selecionando somente o ID, nome, descrição e a localização de cada objeto.

Na segunda, o desenvolvedor re-utiliza o molde de importação que foi utilizado na primeira importação para a atualizações manuais futuras. O SmartSync guarda as chaves dos dados dos arquivos juntos com as chaves dos dados importados no *middleware*, possibilitando a atualização dos dados. Com isso, os arquivos que geralmente são planilhas ou arquivos CSV de grandes tamanhos, são submetidas pelo desenvolvedor, que são enviadas para o *middleware* que, através de uma fila, processa os arquivos usando balanceamento de cargas e realiza a integração dos dados. Um exemplo desta forma de importação ocorre quando, anualmente, o desenvolvedor submete as planilhas disponibilizadas pelo governo brasileiro referente ao desempenho escolar. Estas planilhas que tem a mesma estrutura todos os anos, por sua vez, são processadas pelo SmartSync de acordo com o modelo predefinido, que em seguida atualiza os dados de desempenho dos alunos no *middleware*.

3 Dashboard de desenvolvimento e *dashboard* de visualização

O *dashboard* de desenvolvimento e o *dashboard* de visualização são interfaces desenvolvidas para plataformas de *middleware* que trabalham com dados georreferenciados de diferentes domínios e organizadas em camadas.

O *dashboard* de desenvolvimento permite aos usuários desenvolvedores, — que gostariam de construir aplicações, sem serem obrigados a se tornar especialistas das plataformas de *middleware* subjacentes —, construam aplicações para cidades inteligentes de forma mais rápida e fácil. O *dashboard* de visualização, oferece aos usuários finais (agentes públicos, gestores urbanos ou os cidadãos) um ambiente simplificado que dispõe de um conjunto de mecanismos que facilitam a exibição dos dados, possibilitando que tais interessados possam usufruir dos benefícios das aplicações desenvolvidas usando a plataforma de *middleware*.

Este capítulo apresenta uma visão geral do dashboard de desenvolvimento e do *dashboard* de visualização. A seção 3.1 apresenta as funcionalidades que as interfaces devem oferecer. A seção 3.2 apresenta a arquitetura proposta. A seção 3.3 apresenta alguns detalhes relacionados com a implementação dos *dashboards* de desenvolvimento e visualização. A seção 3.4 apresenta aplicações reais que usam os dashboards fornecidos nesse trabalho.

3.1 Funcionalidades

O *dashboard* de desenvolvimento e o *dashboard* de visualização tem propósitos específicos que são discutidos nas subseções apresentadas a seguir.

3.1.1 Dashboard de desenvolvimento

Um *dashboard* de desenvolvimento abstrai funcionalidades da plataforma *middleware* de dados geográficos, como viabilizar a criação de novas aplicações, camadas e entidades. As suas principais funcionalidades são:

- **gerenciar a segurança das aplicações e dos usuários**: possibilita gerenciar a segurança das aplicações e dos usuários, como papéis dos usuários, permissões para leitura, escrita, visualização de camadas e acesso às aplicações;
- **gerenciar camadas e entidades**: provê uma interface capaz de gerenciar camadas e entidades, tais como criar, editar ou remover camadas e entidades no modelo de

dados NGSI-LD, que suporta dados vinculados (relacionamentos das entidades), propriedade de grafos e às semânticas (explorando os recursos oferecidos pelo JSON-LD);

- **suporte de importação e sincronização dos dados de diversos tipos de arquivos e fontes**: dispõe de uma interface que usa a API de integração e sincronização de dados, que facilita a integração de dados de empresas, governanças, organizações ou mesmo dispositivos da IoT, diminuindo a ocorrência de problemas relacionados a uso de sistemas legados, uso de softwares proprietários, entre outros problemas que impossibilitem a capacidade de prover serviços que permitam a integração; e
- **definição de *dashboards* por camada**: oferece aos desenvolvedores a capacidade de montar *dashboards* para as camadas, usando os variados *widgets* disponíveis. Por exemplo, para montar um *dashboard* para a camada Escola que apresente estatísticas do desempenho dos alunos no decorrer dos períodos letivos sucedidos, um desenvolvedor pode usar um *widget* de gráfico de linha para cada dados de desempenho do alunos, como número de aprovados, reprovados e evadidos, relacionando-os com os períodos letivos.

A Figura 9 descreve as funcionalidades propostas para o *dashboard* de desenvolvimento.

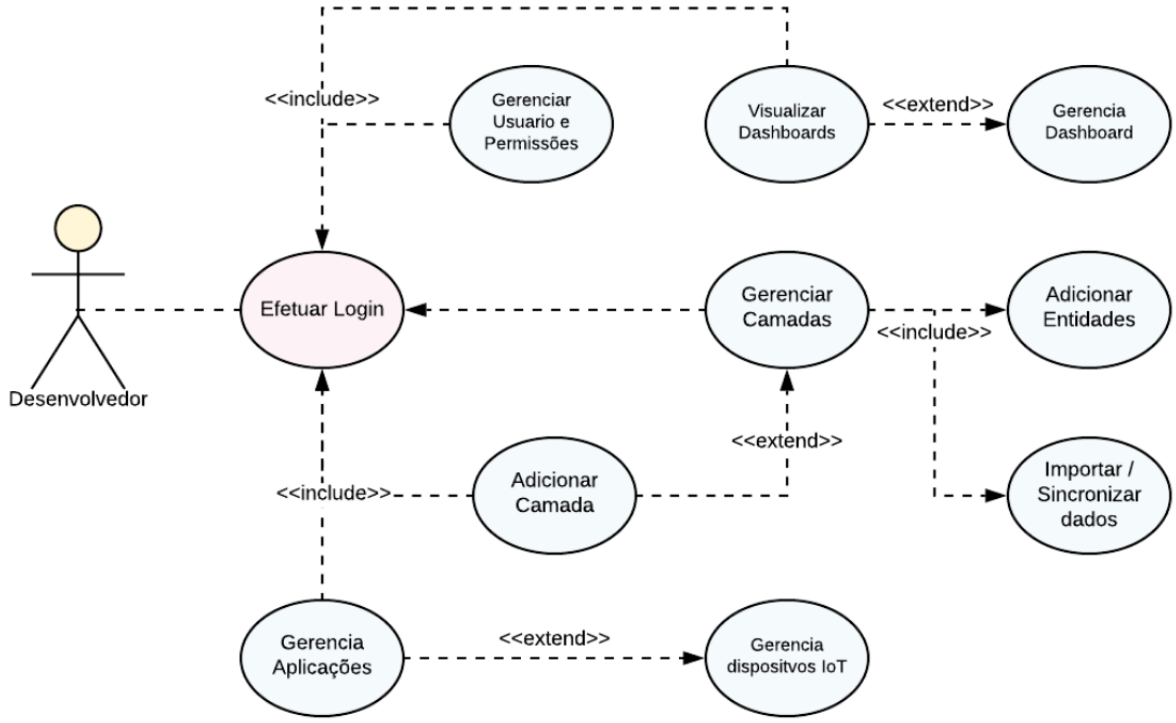

Figura 9 – Diagrama de caso de uso das funcionalidades da *dashboard* de desenvolvimento

Como visto na Figura 9, a funcionalidade de *Gerenciar dispositivos IoT* é uma extensão da Gerência de Aplicações, tendo como finalidade facilitar o uso dos recursos da plataforma *middleware* para comunicação dos dispositivos com IoT. Por exemplo, é possível realizar o cadastro, edição e remoção das configurações de dispositivos ou sensores, verificação do estado de conexão e a configuração da publicação de dados em camadas.

3.1.2 *Dashboard* de visualização

Um *dashboard* para visualização pode facilitar a compreensão dos dados das aplicações desenvolvidas usando as plataformas de *middleware* que trabalhem com dados georreferenciados por meio de *dashboards*, de mapas através da exibição em múltiplas camadas dos dados georreferenciados, através consultas de dados amigável e exibição de gráficos. Tais funcionalidades são descritas a seguir.

- **visualização em múltiplas camadas de dados georreferenciados**: viabiliza a análise integrada, multidimensional e territorializada considerando a correlação dos dados de diversas fontes, trazendo à tona informações que ficariam obscuras em uma visão clássica (unidimensional);
- **edição de dados geográficos**: oferece um componente capaz de editar pontos, polígonos ou segmentos de linhas das entidades presentes nas camadas de dados georreferenciados;
- **consulta de dados amigável**: simplifica a maneira do usuário realizar consultas sobre entidades de diferentes camadas através de uma interface amigável, eliminando a necessidade de conhecimento de linguagem de consulta;
- **geração de relatórios, tabelas ou gráficos**: dispõe de exibição de relatórios de dados em forma de texto, linguagem de marcação ou, ainda, gráficos gerados a partir de uma sequência histórica da informação;
- **exibição de *widgets* do *dashboard***: exibe *widgets* preestabelecido pelo desenvolvedor no momento da criação ou edição da camada, possibilitando aos usuários a visualização, monitoramento ou correlacionados dos dados das camadas de seu interesse de modo intuitivo e direto;
- **personalização e exibição de *dashboard* pessoal do usuário**: permite ao usuário compor um *dashboard* pessoal à medida que favorita *widgets* que apresentem dados de seu interesse. Esses *widgets* favoritados aparecem em um *dashboard* pessoal do usuário logado e podem ser também organizados a sua escolha;
- **salvar os dados exibidos no mapa**: permite ao usuário salvar a exibição atual do mapa para recupera-la em um uso posterior do *dashboard* de visualização. Esta

exibição inclui camadas, resultados de consultas, entidades, posicionamento da tela e zoom; e

- **API para iframe embutidos**: permite que desenvolvedores reutilizarem o *dashboard* de visualização e suas funcionalidades ao embuti-lo em suas aplicações ou páginas da *web* através de iframes do HTML5.

A Figura 10 descreve as funcionalidades propostas para o *dashboard* de visualização.

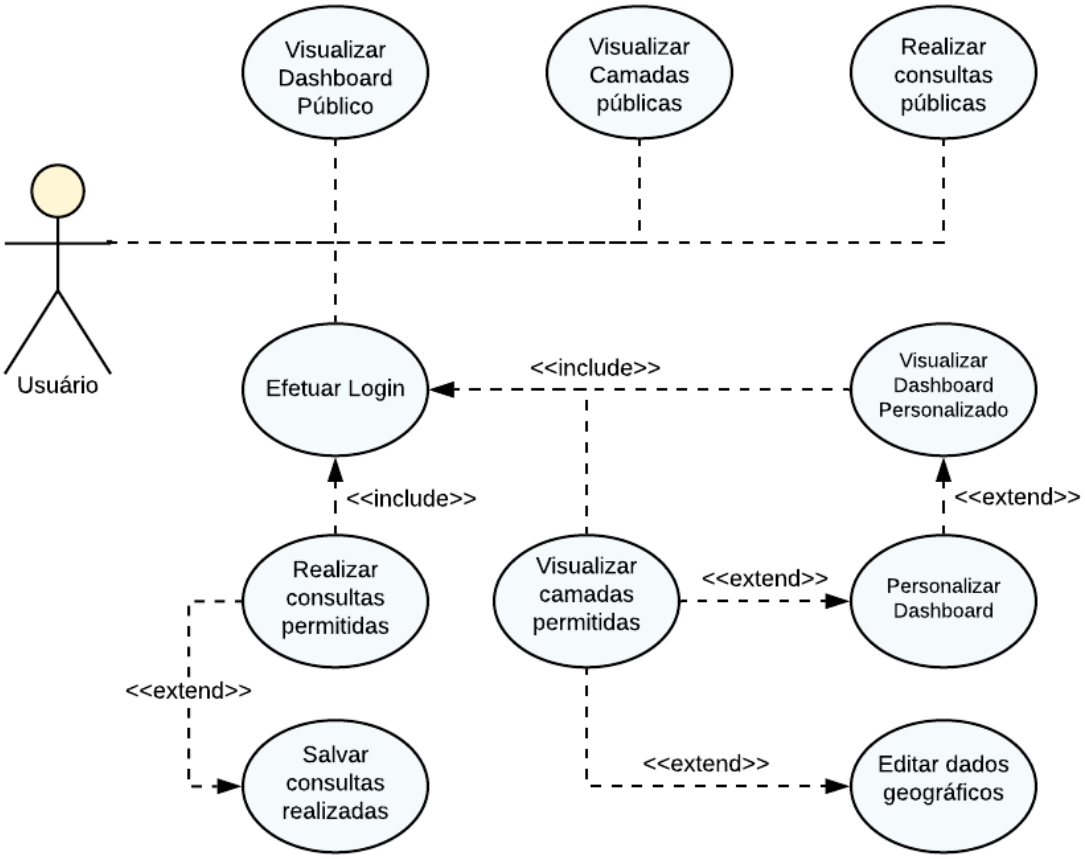

Figura 10 – Diagrama de caso de uso das funcionalidades do *dashboard* de visualização

3.2 Arquitetura

Para fornecer as funcionalidades citadas anteriormente, propomos um modelo arquitetural conceitual do *dashboard* de desenvolvimento e do *dashboards* de visualização, incluindo suas interações com a plataforma de middleware subjacente, representado pela Figura 11.

Essa arquitetura provê componentes que são base para o desenvolvimento concreto para os *dashboards* de desenvolvimento e visualização, materializados no contexto da plataforma de *middleware* SGeoL. Esta arquitetura é composta por quatro componentes principais: *gerenciador de segurança, autenticadores, provedores de recursos* e *consumidores*

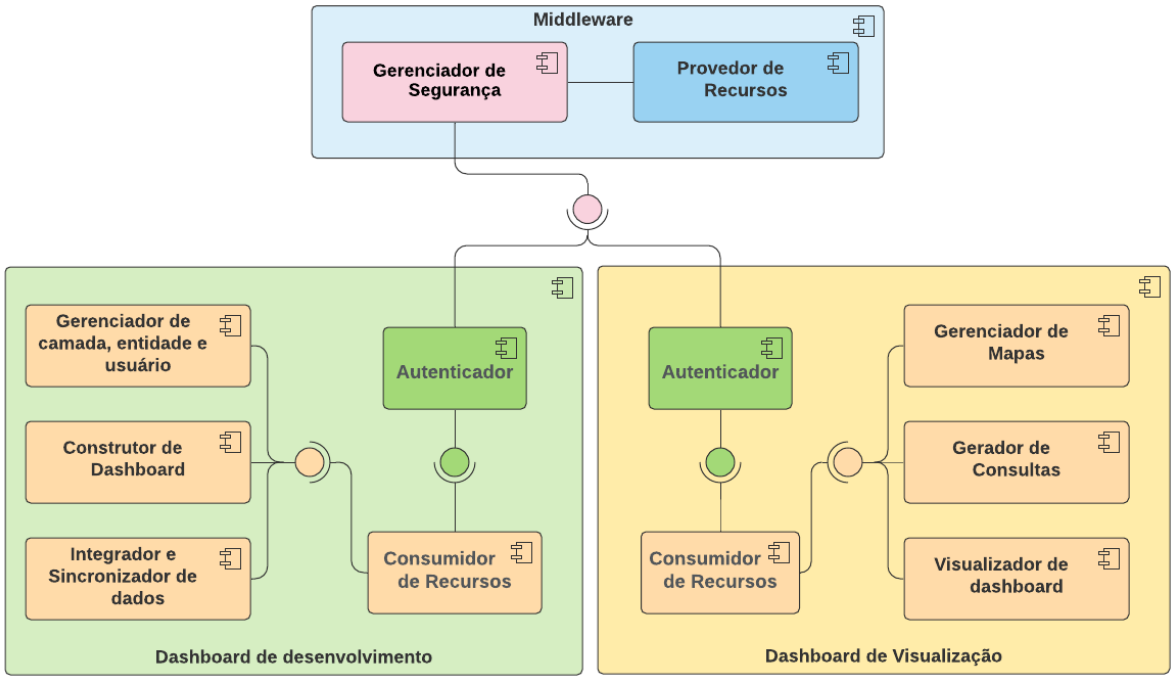

Figura 11 – Arquitetura conceitual da *dashboard* de desenvolvimento e do *dashboard* de visualização

de recursos. Essa organização possibilita a criação flexível de uma infraestrutura para plataformas de *middleware* que consideram dados geográficos, que também podem ser adaptados a possíveis singularidades, como o modelo de dados específico usado.

O *Provedor de recursos* representa os recursos fornecidos pela plataforma de *middleware*. Por exemplo, no caso do *middleware* SGeoL, os recursos são *camadas* e *entidades de cidades inteligentes* (escolas, áreas verdes, praças, hospitais, etc.).

Consumidores de recursos são componentes que usam recursos fornecidos pelo componente *Provedor de recursos*. Por exemplo, no caso do *dashboard* de visualização e desenvolvimento, os consumidores são: componente de integração e sincronizador de dados, componente do gerenciador de mapas, componente do gerador de consultas etc.

O *Gerenciador de segurança* é um elemento de *middleware* que pode autenticar os usuários para verificar se eles estão autorizados a acessar um aplicação ou consumir recursos do componente *Provedor de recursos*. Depois que um usuário é autorizado pelo *componente do Gerenciador de Segurança*, o componente *Autenticador* dos dois *dashboards* armazena as informações da sessão do usuário autenticado. Essas informações da sessão são usadas por todos os outros componentes do *dashboard* de desenvolvimento para consumir os recursos fornecidos pelo componente *Provedor de Recursos*, bem como os componentes do *dashboard* de visualização, além de exibir elementos visuais da interface do usuário que o usuário tem permissão para acessar. Por exemplo, um usuário do *dashboard* de visualização que possui permissão somente leitura para a *camada de áreas verdes* pode visualizar os dados da camada usando suas credenciais, mas não pode ver o botão de

edição da camada, porque não possui a permissão para atualizar dados nesta camada.

O *dashboard* de desenvolvimento é composto por três componentes: (i) *Gerenciador de camada, entidade e usuários*, (ii) *Construtor de dashboard* e (iii) *Integrador e sincronizador de dados*.

(i) *Gerenciador de camada, entidade e usuários*: usa os recursos fornecidos pelas APIs responsáveis por listar, criar, exibir, editar e excluir camadas, entidades e usuários. Além de gerenciar estes elementos, esse componente também gerencia a política de acesso de aplicações, usuários e camadas.

(ii) *Construtor de dashboard*: oferece a capacidade de criar *dashboards* para cada camada, individualmente. Por exemplo, para criar um *dashboard* para a camada da Escola que exiba estatísticas de desempenho dos alunos ao longo de um semestre, um desenvolvedor pode selecionar um quadro que mostre um gráfico de linhas para os dados de desempenho do aluno (número de alunos aprovados, reprovados e evadidos, etc.) em comparação a diferentes semestres. A união dessas e de outras painéis constitui um *dashboard* da camada da Escola.

(iii) *Integrador e sincronizador de dados*: consome os recursos fornecidos pela API do *Integrador e sincronizador de dados*. Esse componente facilita a integração de dados de empresas, governos, organizações ou mesmo dispositivos de IoT, reduzindo a ocorrência de problemas relacionados ao uso de sistemas legados, uso de software proprietário ou outros problemas que impossibilitam o fornecimento de serviços de integração.

O *Dashboard* de Visualização é composto por três componentes: (i) *Gerenciador de Mapas*, (ii) *Gerador de Consultas* e (iii) *Visualizador de dashboard*.

(i) *Gerenciador de Mapas*: permite uma análise de dados de forma integrada, multidimensional e territorializada, considerando a correlação de dados de diferentes camadas e trazendo informações que seriam obscurecidas em uma exibição clássica (unidimensional). Ele também fornece recursos que permitem editar dados geográficos, em forma de exibição, como pontos (marcadores), polígonos ou linhas das entidades pertencentes às camadas.

(ii) *Gerador de consulta*: simplifica a maneira como os usuários executam consultas sobre diferentes entidades disponíveis por meio de uma interface amigável, eliminando a necessidade de um conhecimento da linguagem de consulta do banco de dados. O resultado dessas consultas pode ser armazenado e reutilizado posteriormente. Por exemplo, um usuário pode definir uma consulta para visualizar no mapa quais escolas da camada Escola têm desempenho "excelente"em um determinado índice de

classificação escolar. O resultado dessa consulta pode ser nomeado, salvo e usado posteriormente, eliminando a necessidade de reescrever a consulta.

(iii) *Visualizador de dashboard*: exibe *dashboards* de camadas predefinidos pelos desenvolvedores no momento de sua criação ou edição, permitindo que os usuários visualizem, monitorem ou correlacionem diferentes camadas de dados, de forma intuitiva e direta. Além disso, permite ao usuário compor um *dashboard* pessoal ao selecionar painéis que compõem os variados *dashboards* de camadas como favoritos. Por exemplo, um usuário que visualiza as diferentes camadas disponíveis pode, ao favoritar um dos painéis que apresenta dados de interesse, compor um *dashboard* pessoal com painéis de diferentes camadas. Esse recurso elimina a necessidade de acessar, individualmente, cada camada de interesse para a coleta de informações desejadas e permite uma visão agregada sobre vários dados. Além disso, em seu *dashboard* pessoal, o usuário pode organizar a disposição destes painéis, alinhando por exemplo, os painéis que estabelecem possíveis relações casuais.

3.3 Implementação

Esta seção apresenta detalhes de implementação e decisões para cada funcionalidade da arquitetura apresentada.

Para uma melhor organização, esta seção foi dividida em quatro subseções, a primeira refere-se a escolha do *framework* e a implementação da arquitetura conceitual apresentada na Figura 11. Na segunda seção são discutidas as funcionalidades compartilhadas entre as interfaces (*dashboard* de desenvolvimento e *dashboard* de visualização). A terceira seção e a quarta seção abordam, respectivamente, as funcionalidades específicas do *dashboard* de desenvolvimento e do *dashboard* de visualização.

3.3.1 *Tecnologias Utilizadas*

Para a implementação da arquitetura conceitual apresentada na Figura 11, foi usado o Vue, um *framework* para interfaces de linguagem JavaScript. De acordo com Filipova (2016), o Vue pode ser usado de forma tão flexível que definitivamente pode ser considerado como uma estrutura capaz de suportar o desenvolvimento de ponta a ponta de aplicações *web* complexas, permitindo assim que os modelos de dados estejam vinculados diretamente na camada de visão e a sua reusabilidade em outras partes da aplicação.

A escolha deste *Framework* deve-se a experiência dos envolvidos em tal tecnologia. Além disso, o uso do Vue é bem justificada na literatura. Wohlgethan (2018), em um trabalho que compara os três principais *framework* JavaScript, aponta o Vue como o *framework* com melhor processo de aprendizado inicial. Tal processo de aprendizagem é relevante, pois permite que novos pesquisadores e desenvolvedores assimilem mais

rapidamente o conhecimento necessário para manter e aprimorar as plataformas em desenvolvimento.

De acordo com as estatísticas fornecidas pela pesquisa da Greif, Benitte e Rambeau (2019), o Vue obteve uma pontuação maior do que outros *Frameworks* quando perguntados aos desenvolvedores entrevistados: "Ouvi falar, gostaria de aprender". Esse dado é um claro indicador que reforça a curva de aprendizado para este *framework* de JavaScript. Além disso, a curva de aprendizado é um item relevante, uma vez que a medida em que o projeto cresce, novos alunos passam a integrar o grupo de desenvolvimento e desenvolver usando o *framework*.

Um outro ponto, tendo em vista a necessidade de desempenho que uma aplicação que trabalha com *dashboards* e dados geográficos deve ter, um fator importante foi revelado por Incau (2017). Em sua obra, Incau (2017) revela que o *framework* chamado React[1], concorrente do Vue, tem algumas vantagens, como seu ecossistema muito mais desenvolvido e uma comunidade maior. Entretanto, o Vue é melhor que o React em termos de desempenho.

Uma vez selecionado o *framework* que auxiliaria no desenvolvimento e implementação dos *dashboards*, foi definida a arquitetura usando o Vue, apresentada na Figura 12.

Tal arquitetura foi dividida em módulos, onde cada módulo é estruturado da seguinte forma:

- componentes: inclui todos os componentes pertencentes ao módulo específico, os quais podem ser reutilizados usando o nome do componente ou através de eventos;
- serviços: inclui todos os arquivos de configuração, endereços e métodos para o consumo das API dos serviços utilizados pelo módulo;
- *stores*: armazena os dados únicos de estado - ou seja, esse único objeto contém todo o estado do seu nível de aplicação e tem como objetivo disponibilizar a informação em seu estado atual para qualquer outro módulo; e
- rotas: possui a configuração das rotas internas de cada módulo. Por exemplo, o módulo do *dashboard* tem uma rota principal que apresenta todos os *widgets* (*/dashboard/widgets*) e tem também uma rota para apresentação de apena um *widget* (*/dashboard/widget/id*).

Essa forma de organização evita problemas de desenvolvimento a medida em que o projeto ganha extensão. A seguir, são apresentados e discutidos os módulos.

O *Core* é um módulo onde são incluídos todos os arquivos comuns a todos os módulos. Como por exemplo, seleção de idiomas, barras e menus de navegação, botões de saída e etc.

[1] <https://pt-br.reactjs.org/>

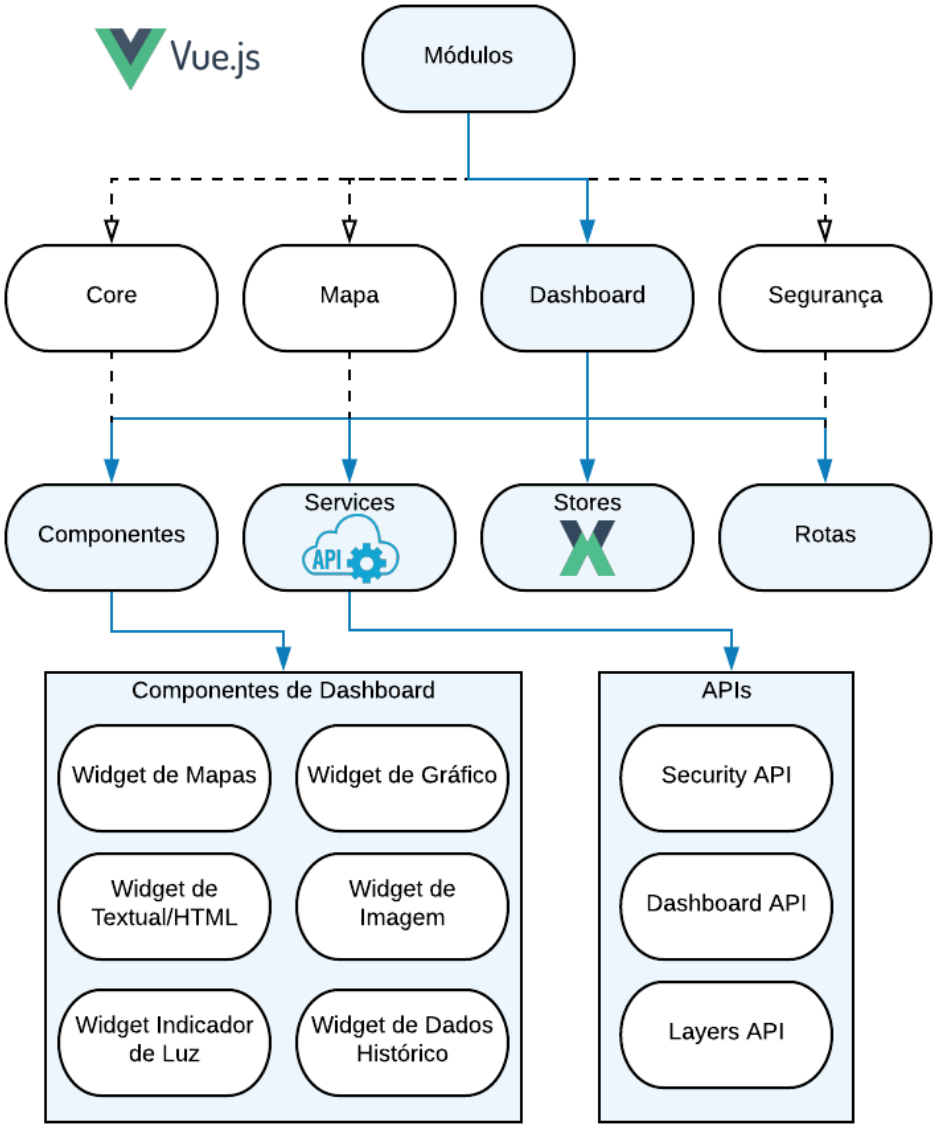

Figura 12 – Arquitetura modular

O módulo de *Mapa* é responsável pela apresentação dos dados geográficos. Ele usa a biblioteca LeafletJs[2] como base para exibição de mapa e possui vários componentes próprios, como por exemplo: Componente de seleção de camada, componente de controle de visibilidade de camada, componente de consulta geográfica, componente de edição de entidades, entre outros. Também guarda o estado de todas as entidades plotadas no mapa e dos objetos selecionados no mapa. Seus serviços incluem uso das APIs que fornecem recursos para camadas, entidades e consultas geográficas.

O módulo de *Dashboard* engloba os componentes de *widgets* como: *Widget* de Mapas, gráficos, textual e HTML, imagem, indicador de luz e dados históricos. Esses componentes de *widgets* são responsáveis pela apresentação dos dados e pelas interfaces

[2] https://leafletjs.com/

para preenchimento com os dados.

O módulo de Segurança é encarregado por fazer a comunicação entre as aplicações e o *middleware* de segurança. Esse módulo possui os componentes, rotas e telas para entrada do usuário, recuperação de senha, cadastramento de novos usuários, definição de papéis, entre outros. Além disso, esse módulo também guarda o estado dos dados de *token* do usuário e permite que outros módulos façam uso destes para o consumo das APIs do *middleware*.

3.3.2 *Dashboard* de desenvolvimento

O *dashboard* de desenvolvimento fornece ao desenvolvedor facilidades de gerenciamento de suas aplicações, usuários e dados. Em sua tela inicial, são apresentados dados da aplicação do usuário como, por exemplo, a lista das camadas mais acessadas e quantidade de entidades disponíveis nas camadas de maior interesse dos usuários (Figura 13).

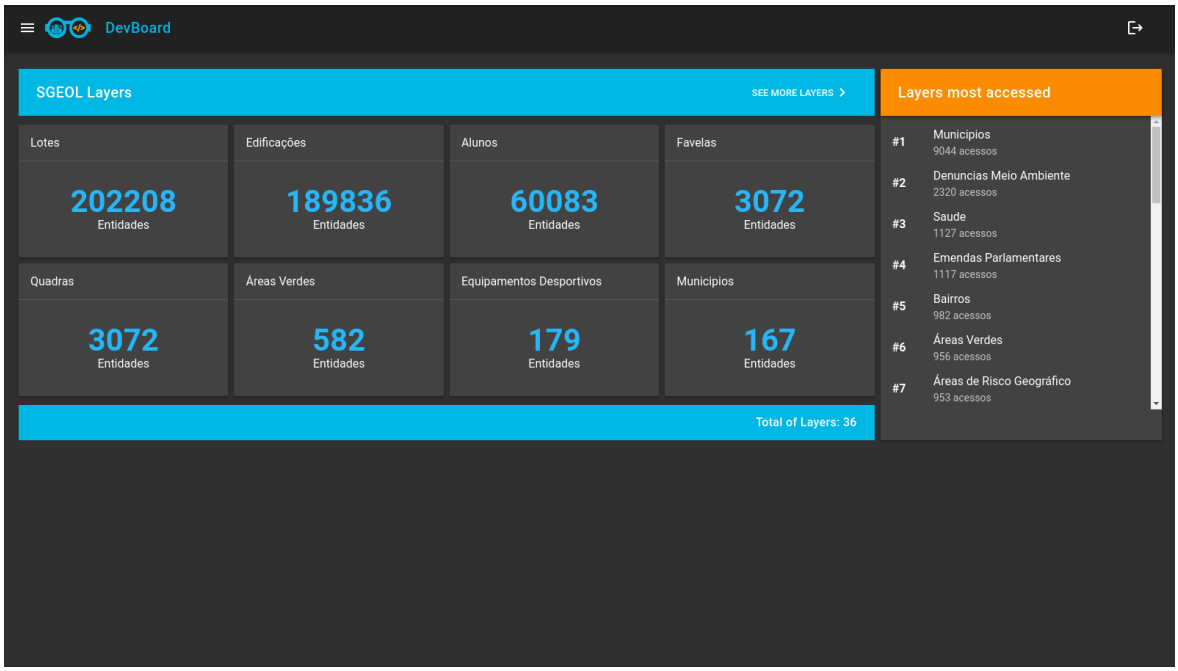

Figura 13 – *Dashboard* de desenvolvimento

A Figura 13 apresenta um *dashboard* com alguns dados como: mais a esquerda, painéis numéricos com quantidade de entidades presente em cada camadas públicas — que tem seus dados abertos e disponíveis para qualquer usuário —, como a camada de lotes, edificações, alunos, favelas, quadras, áreas verdes, equipamentos esportivos e municípios; e, mais a direita; painel de lista das camadas mais acessadas, onde é apresentada que a mais usada é a camada de municípios, seguida de denuncias do meio ambiente e saúde.

Ainda na Figura 13, na barra superior, são exibidos os botões para abertura da barra navegação — que fornece ao usuário acesso as todas as demais funções da aplicação — e botão para controle de autenticação (logar ou deslogar). Através da barra de

navegação, é possível navegar facilmente entre diferentes funcionalidades do *dashboard* de desenvolvimento, permitindo ao desenvolvedor encontrar o que procura apenas com alguns cliques. Por meio do botão para controle de autenticação, o usuário pode entrar com suas credenciais para administrar usuários, camadas, entidades ou importar dados, ou caso já esteja logado, é possível deslogar e voltar a tela inicial do *dashboard* de desenvolvimento.

A seguir são apresentadas as funcionalidades do *dashboard* de desenvolvimento e o estado atual de desenvolvimento destas funcionalidades.

Gerenciar a segurança das aplicações e dos usuários

Para controle das aplicações e dos usuários, foi adicionado um menu na barra de navegação lateral, onde é possível listar e adicionar usuários. A Figura 14 apresenta a tela onde são listados os usuários cadastrados na plataforma SGeoL. Nessa tela é possível ainda que administradores da aplicação desativem ou ativem usuários, como também definam as permissões para as camadas individualmente (Figura 15).

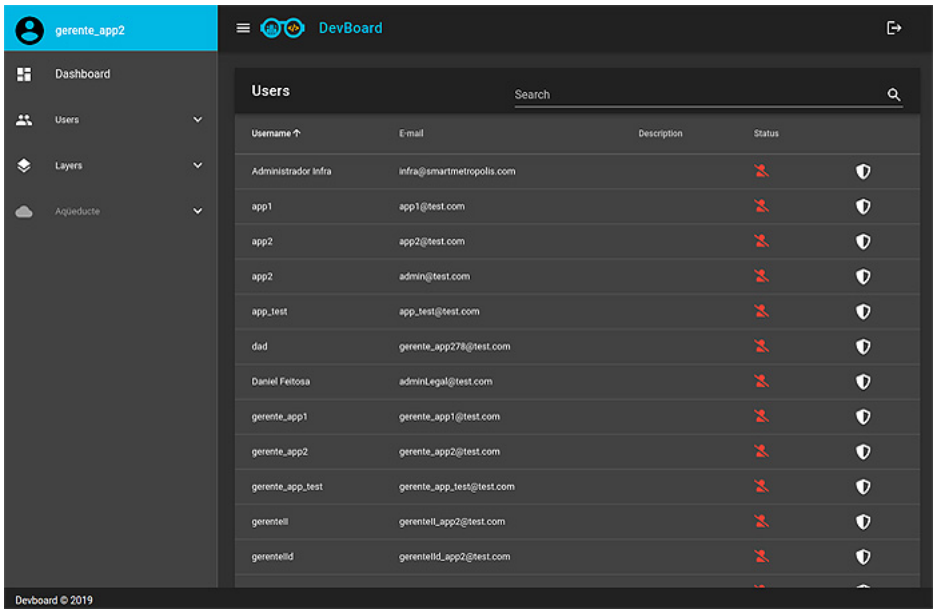

Figura 14 – Tela de listagem de usuários no *dashboard* de desenvolvimento

Na definição de permissões de usuário, é possível selecionar o que o usuário poderá ler, escrever, atualizar, apagar ou gerenciar uma camada. Um usuário que gerencia uma camada pode, além de realizar as outras permissões, definir permissões para outros usuários.

Gerenciar camadas e entidades

Para gestão das camadas e entidades foi adicionado um menu na barra de navegação lateral, onde é possível criar uma nova camada ou gerenciar as camadas. A Figura 16

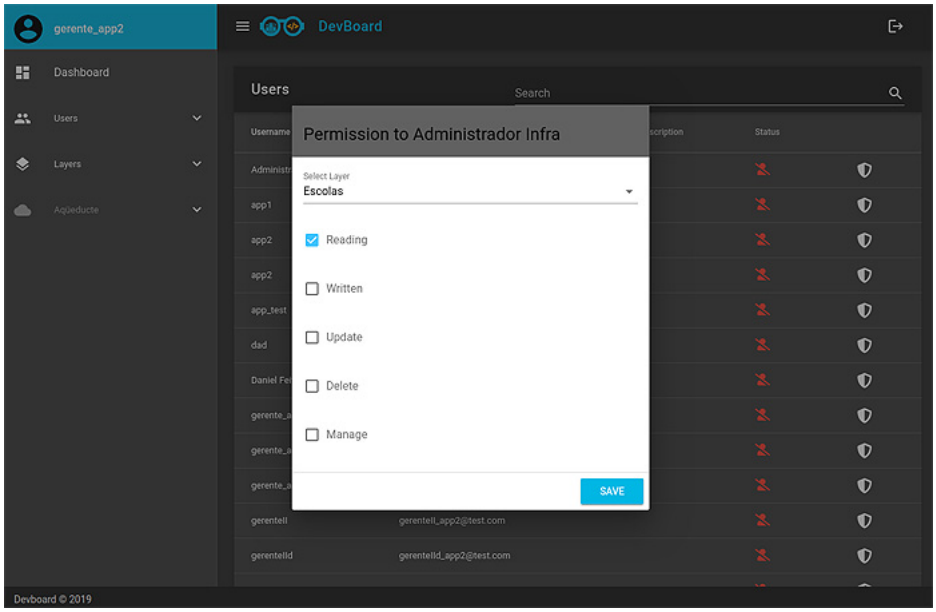

Figura 15 – Definição de permissões na tela de listagem de usuários

apresenta a tela onde são listadas as camadas da plataforma SGeoL.

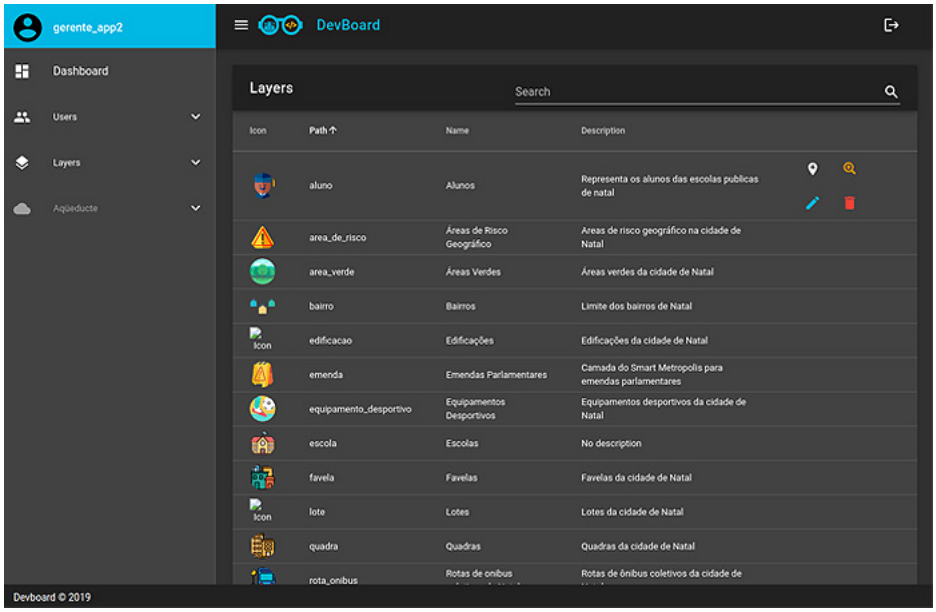

Figura 16 – Tela de listagem de camadas

A tela criada para listagem de camadas apresenta detalhes das camadas, permitindo ainda adição de nova entidade na camada, listagem das entidades da camada e remoção da camada.

Para a adição e edição de novas camadas ou entidades, foi construído um componente de formulário reutilizável – que atende às especificações do NGSI-LD e permite o gestão das propriedades deste modelo em formato JSON – a fim de evitar qualquer duplicidade de código e facilitar sua manutenção, conforme é apresentada na Figuras 17.

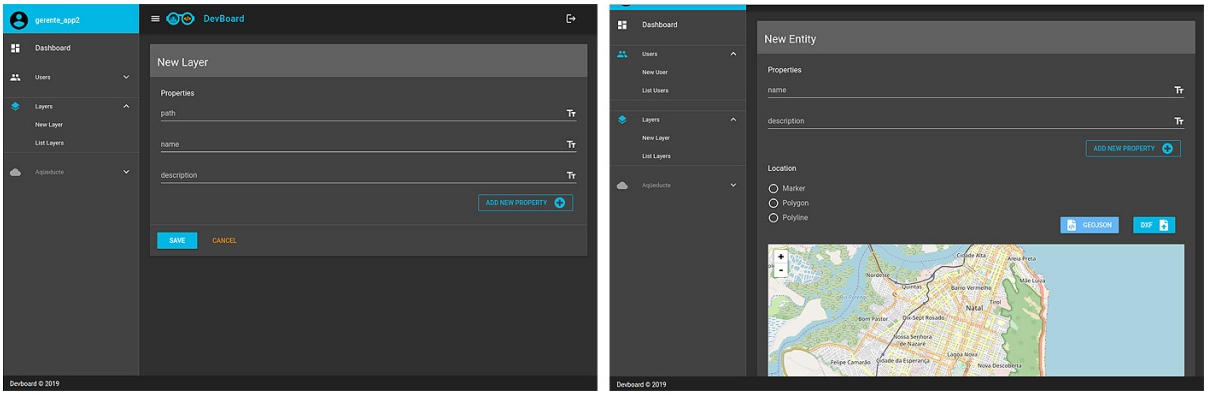

Figura 17 – Adição de uma nova camada (à esquerda) e de uma nova entidade (à direita)

Suporte à camada de importação e sincronização dos dados de diversos tipos de arquivos e fontes

Uma interface que forneça suporte às camadas de importação e sincronização é de grande importância, uma vez que pode ajudar a solucionar problemas de importação de informações de difícil acesso. Por exemplo: dados de sistemas de governo, conforme descreve Kim, Trimi e Chung (2014), ao afirmarem que, geralmente, estes mantém seus dados isolados de outros sistemas, complicando a tentativa de integrar dados complementares entre órgãos e departamentos do governo. Além disso, estas fontes geralmente não possuem uma API e, muitas vezes, acabam sendo exportações diretas de dados originados de tipos variados de banco de dados ou sistemas de arquivo, disponibilizados através de arquivos CSV, planilhas ou linguagem de notação XML, JSON, entre outros.

Ademais, a manutenção destes dados pode ser comprometida se não houver um armazenamento externo das chaves estrangeiras que identifique os dados entre as aplicações provedoras e no *middleware* após suas inserções. Assim, uma interface pode solucionar o esforço manual substancial que um desenvolvedor teria para tratar os dados e, de agora em diante, usar as APIs dos SGeoL para integrar os dados. Pensando nisso, foi desenvolvida uma interface que permite ao desenvolvedor selecionar de que forma deseja proceder diante à importação (Figura 18).

Nessa tela é possível selecionar se as importação usará arquivos CSV, JSON ou XLS, ou se dará através de APIs. A seguir, o desenvolvedor terá auxílio de um componente guia, que lhe auxiliará nas configurações necessárias, seja para submeter seus arquivos ou selecionar a API desejada, ao passo que, serão selecionados quais dados devem ser importados.

Definição de *dashboards* por camada

O *dashboard* de desenvolvimento permite ao desenvolvedor criar *dashboard* para cada

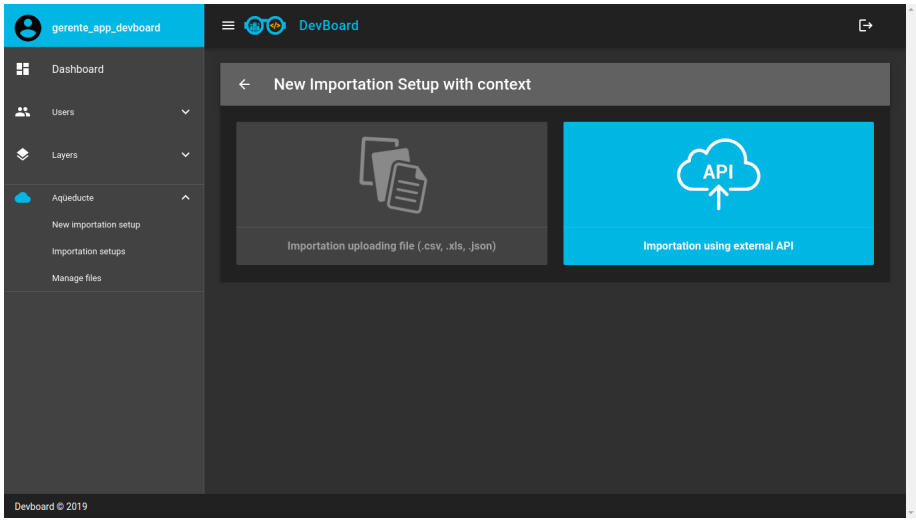

Figura 18 – Tela de suporte a importação e sincronização de dados

camada individualmente, fazendo uso dos *widgets*. Além disso, é possível usar *widgets* que cruzem dados de múltiplas camadas, que neste caso, necessitará que o usuário visualizador, possua permissão de acesso de visualização nas camadas envolvidas.

Três *widgets* iniciais foram desenvolvidos até o momento, são eles: *widget* numérico, lista e mapa. O primeiro é um *widget* que exibe um valor numérico e exibe uma descrição para o valor. O segundo é um *widget* que gera uma lista, com um título e uma descrição. O último refere-se ao mapa, que suporta a exibição de pontos, polígonos e retas e é capaz de determinar um trajeto de uma entidade baseada em sua série histórico.

O *widget* numérico e *widget* de lista são apresentado na figura 13, onde um conjunto de *widgets* numérico são usados para apresentar o número total de entidades para cada camada, enquanto um *widget* de lista é usado para apresentar quais são as camadas mais usadas. O *widget* do mapa é usado para apresentar os dados geográficos das camadas e pode ser visualizado na tela mais à direita da Figura 17.

O estado atual dessa funcionalidade tem foco em adicionar outros componentes de *widgets* mais comuns entre as aplicações pesquisadas, como, por exemplo, *widget* de gráficos, textuais e de calibre. Ao mesmo tempo que serão implementadas as funções de arraste e solte, tanto para adição de novos *widgets*, quanto para a organização destes *widgets* na formação os relacionamentos subjacentes entre as métricas.

3.3.3 *Dashboard* de visualização

O *dashboard* visualização fornece uma interface inspirada nas aplicações de mapas, como Google Maps[3] e OpenStreetMaps[4] (Figura 19).

Nessa tela é possível alterar o modo de visualização entre satélite e mapa, logar

[3] <https://maps.google.com/>
[4] <openstreetmap.org/>

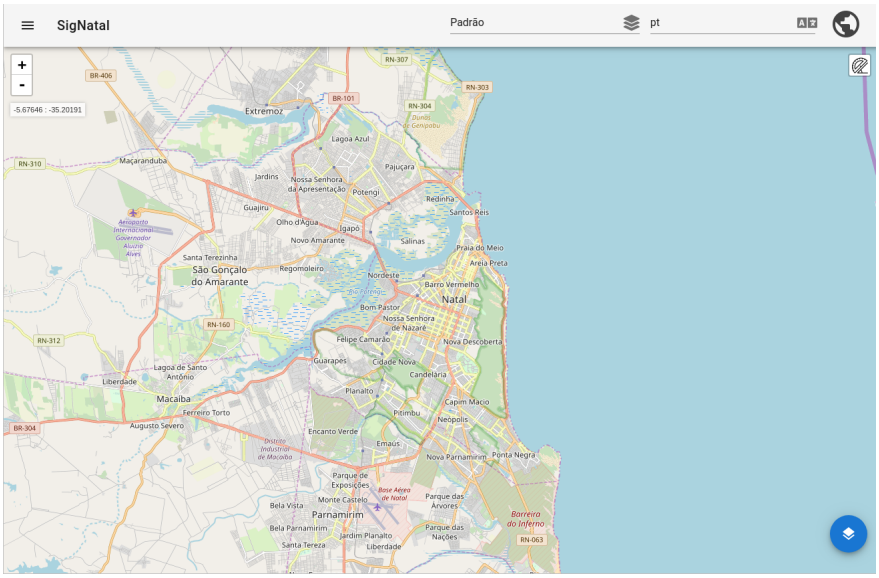

Figura 19 – Interface inicial do *dashboard* de visualização

na aplicação usando as credenciais, ou ainda, usar o botão de ação no canto inferior para usar as funcionalidades de seleção de múltiplas camadas de dados geográficos ou realizar consultas de dados.

Visualização em múltiplas camadas de dados georreferenciados

Usando os botões de navegação no canto inferior direito do mapa, o usuário pode selecionar as camadas públicas (Figura 20), podendo ainda, usar suas credenciais para logar na aplicação e ter acesso às camadas privadas de acordo com as permissões definidas no *dashboard* de desenvolvimento.

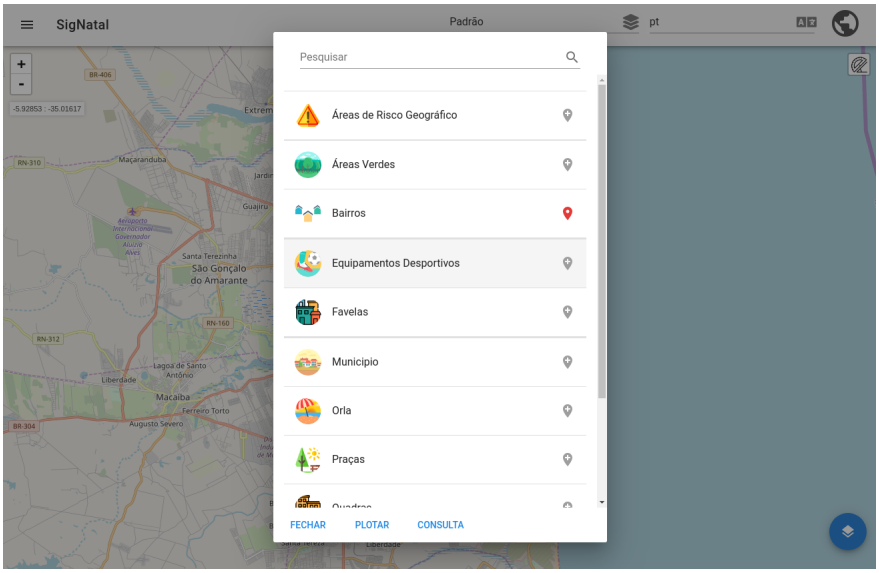

Figura 20 – Seleção de camadas no *dashboard* de visualização

Como mostra a Figura 21 é apresentada uma lista com todas as camadas públicas,

onde o usuário pode selecionar e plotar no mapa para uma visualização multidimensional e territorializada.

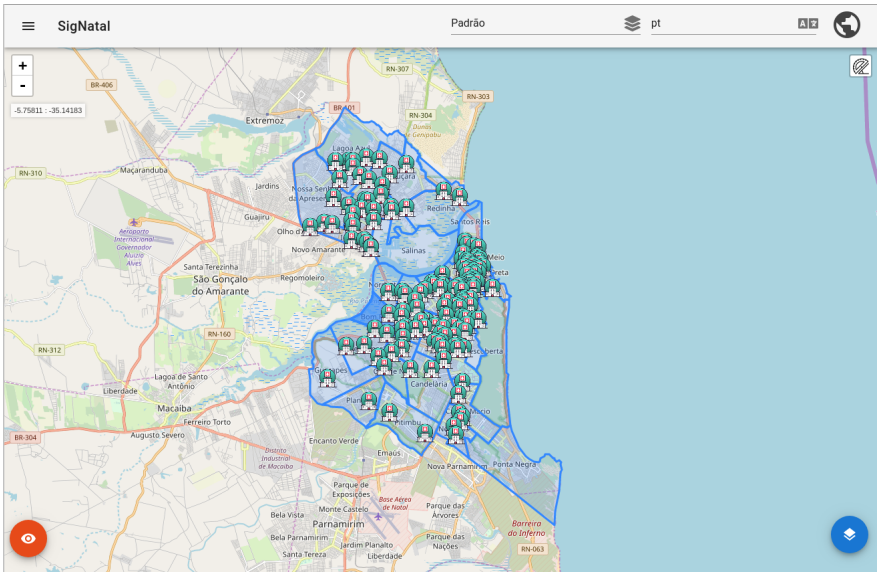

Figura 21 – Visualização de múltiplas camadas no *dashboard* de visualização

A Figura 21 apresenta a distribuição dos postos de saúde e hospitais dentro dos bairros da cidade de Natal. Essa análise é possível realizando o cruzamento das entidades da camada *Bairro* com as entidades da camada *Saúde*.

Usando o controle de visibilidade das camadas, localizado no canto inferior esquerdo, é possível ocultar/desocultar, remover camadas, ativar/desativar a exibição de mapas de calor e controlar a intensidade dos níveis de calor. Tais mapas de calor são importantes pois, de acordo com Isbister e Schaffer (2008), eles mostram visualmente onde um evento acontece. Ou seja, quanto maior a incidência de um evento, maior 'calor' é apresentado no local. A Figura 22 apresenta o uso do controle de visibilidade das camadas e dos dados em mapa de calor.

Edição de dados geográficos

Com a finalidade de facilitar e tornar edições de pontos, polígonos ou linhas mais precisas, foi desenvolvido uma componente de mapa reutilizável para permitir edição de dados geográficos (Figura 23). Usando os botões de controle na lateral do componente de mapa (Figura 23), é possível adicionar e editar diferentes tipos geométricos no mapa.

Além disso, também é possível importar dados georreferenciados de código GeoJSON (formato de intercâmbio de dados geoespaciais), tal qual demonstrado na Figura 24, como também importar geometrias usando arquivos DXF, que são arquivos de intercâmbio para modelos de aplicações desenho assistido por computador (CAD). O uso de arquivos DXF e GeoJSON permite que usuários possam usar ferramentas CAD de suas preferências ou ainda integrar seus dados geográficos de outras plataformas.

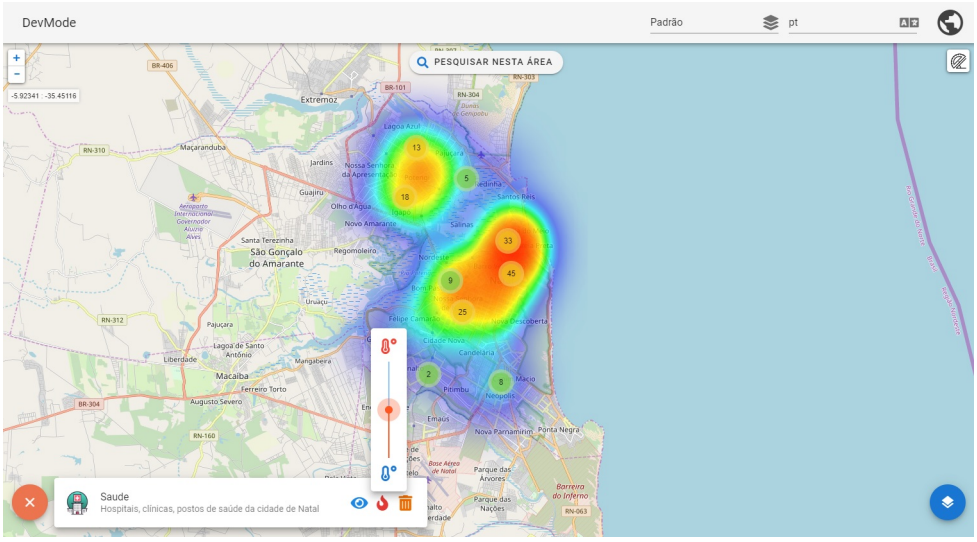

Figura 22 – Controle de visibilidade e uso dos mapas de calor no *dashboard* de visualização

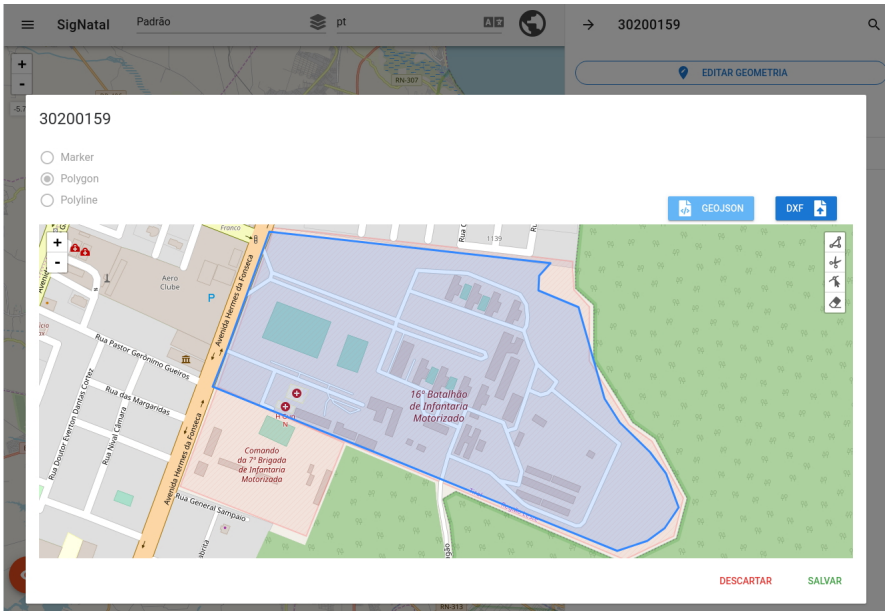

Figura 23 – Componente de mapa

Consulta de dados amigável

Além da funcionalidade de visualização de camadas, a aplicação deve permitir a criação de consultas dinâmicas, onde o usuário pode selecionar quais dados das entidades das camadas desejam incluir na consulta, realizando uma filtragem dos dados de acordo com o valor de determinados atributos dessas camadas (Figura 25).

A Figura 25 apresenta a ferramenta de consultas usando os filtros para a camada de Educação. É possível selecionar um nó “Integrado”, por exemplo, e pesquisar em todos os cursos técnicos do tipo integrados, ou mesmo selecionar “Ensino Técnico” e incluir a pesquisa para todas as modalidades de ensino técnico. Essa ferramenta tem uma interface

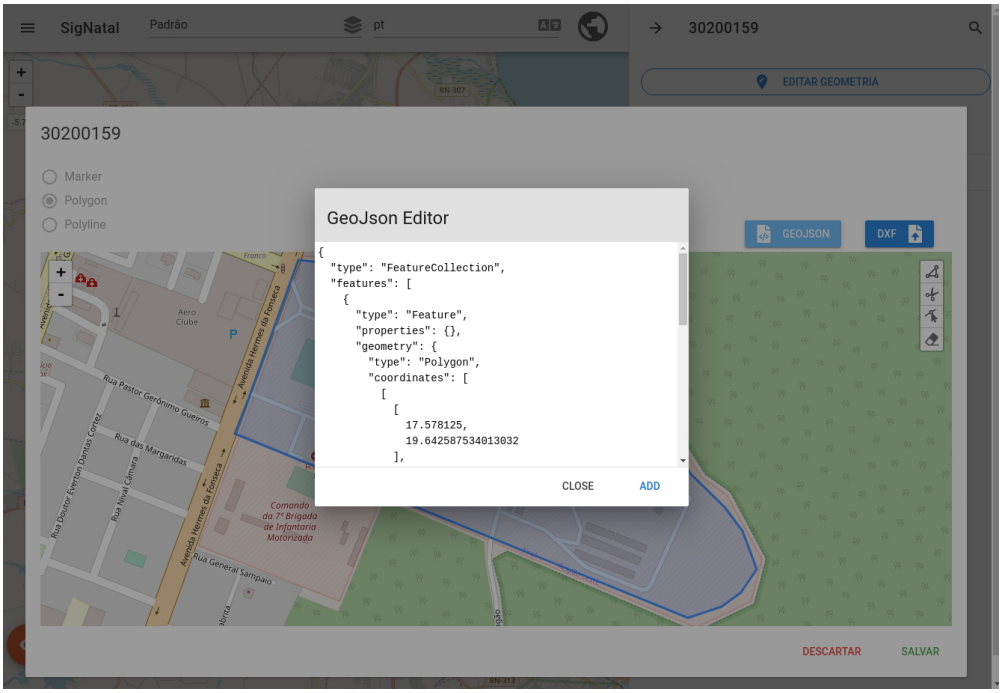

Figura 24 – Importação de dados GeoJSON

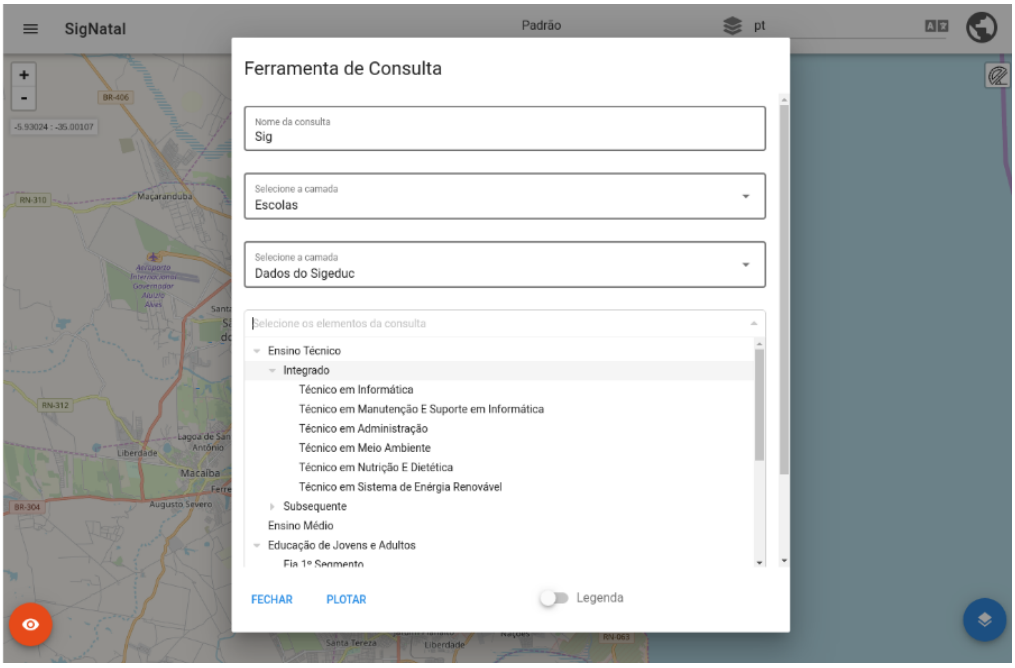

Figura 25 – Ferramenta de consulta amigável

amigável e transforma a interação do usuário em código de banco de dados para a realização de consultas no *middleware* (Figura 26).

Como resultado dessa consulta apresentada na Figura 26, a plataforma plota as escolas no mapa e elimina a necessidade que o usuário tenha conhecimento de linguagem de consulta de banco de dados.

```
{"$or":[{"$and":[{"properties.sigeduc.value.tecnico_integrado_sistema_energia_renovavel.students_amount":{"$gt":"140"}}]},{"$and":[{"properties.sigeduc.value.tecnico_integrado_nutricao_dietetica.students_amount":{"$gt":"140"}}]},{"$and":[{"properties.sigeduc.value.tecnico_integrado_meio_ambiente.students_amount":{"$gt":"140"}}]},{"$and":[{"properties.sigeduc.value.tecnico_integrado_administracao.students_amount":{"$gt":"140"}}]},{"$and":[{"properties.sigeduc.value.tecnico_integrado_manut_sup_info.students_amount":{"$gt":"140"}}]},{"$and":[{"properties.sigeduc.value.tecnico_integrado_informatica.students_amount":{"$gt":"140"}}]}]}
```


Figura 26 – Consulta de banco de dados gerado pela ferramenta de consulta (Figura 25)

Salvar os dados exibidos no mapa

Realizar consultas complexas ou correlações de dados de várias camadas pode ser entediante, caso seja necessário repetir todos os carregamentos das camadas sempre que seja necessário verificar resultados ou fenômenos destas relações. Para resolver este problema, o *dashboard* de visualização permite ao usuário salvar a exibição do mapa atual, sendo possível recarregar estas exibições com todas as camadas que estavam plotadas no mapa, consultas e legendas utilizadas, enquadramento do mapa e zoom em que se encontrava a exibição. A Figura 27 apresenta a tela que possui essa funcionalidade.

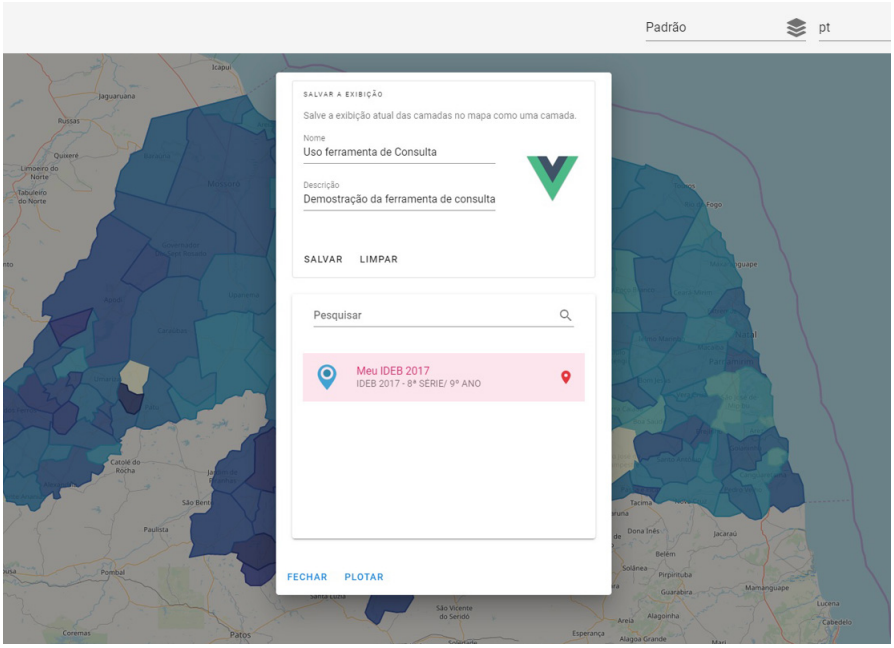

Figura 27 – Ferramenta para salvar a exibição dos dados apresentados e o carregamento dos dados salvos

É necessário atribuir um nome e uma descrição para salvar uma exibição. Para facilitar a identificação de uma exibição salva foi desenvolvido um componente que permite atribuir uma imagem a exibição salva. Para atribuir tal imagem, basta clicar no botão de carregando ou imagem já carregada anteriormente, que fica ao lado dos campos de nome

e descrição. Esse componente permite rotacionar ou selecionar uma parte específica da imagem, como mostra a Figura 28.

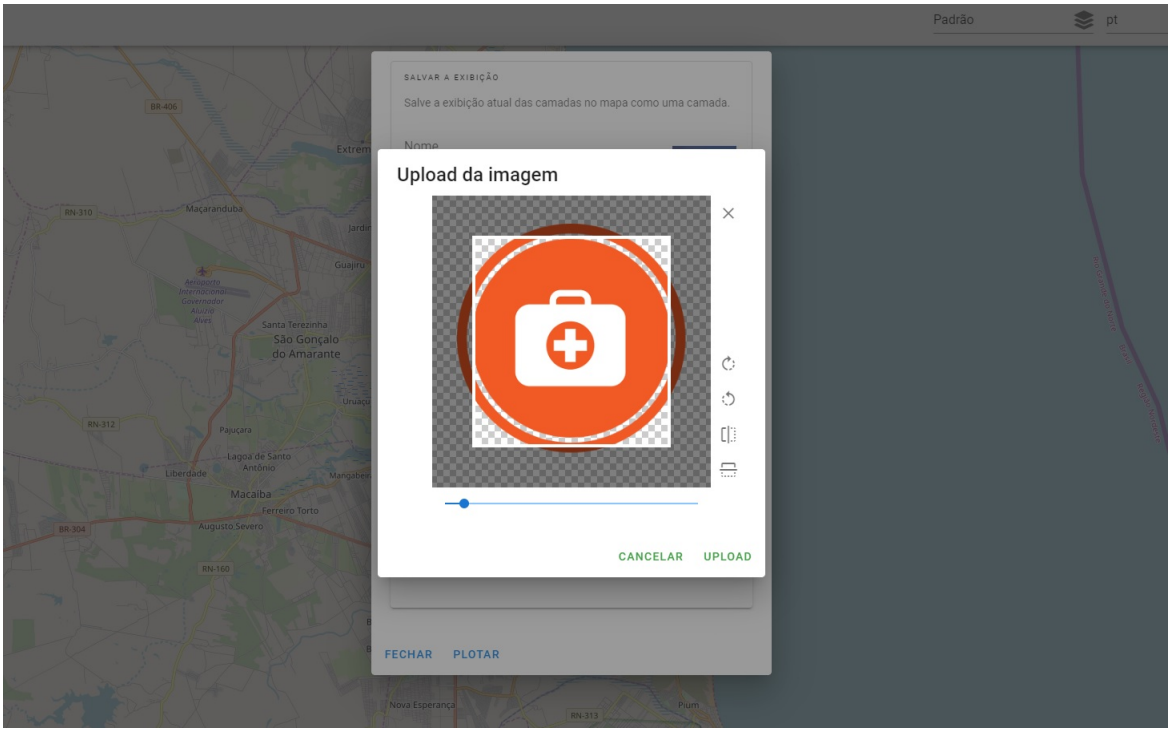

Figura 28 – Carregamento de imagem para identificar a exibição salva.

Usando o texto "Hospitais x Saúde"para o nome, o texto "Distribuição dos hospitais nos bairros de Natal"como uma descrição e a imagem carregada na Figura 28, um salvamento de uma exibição que mostra as unidades hospitalares sobre os bairros de natal pode ser visualizado na tela da Figura 29.

API para iframe embutidos

Através de uma API, desenvolvedores podem reutilizar o *dashboard* de visualização e suas funcionalidades ao embuti-lo em suas aplicações ou páginas da *web* através de iframes do HTML5. Em suma, o uso dessa API baseia-se em iframes ou janelas que são usadas como objetos intermediários para estabelecer a comunicação entre a página ou aplicação pai e o *dashboard* de visualização (aplicação filha), como mostrado na Figura 30.

Conforme apresentado pela Figura 30, o fluxo das chamadas e execução das funcionalidades ocorrem da seguinte maneira:

1. A aplicação pai incorpora um iframe que aponta o *dashboard* de visualização. O dashboard de visualização e a aplicação pai declaram um ouvinte de eventos para receber os dados emitidos por suas contrapartes.

2. A página pai envia uma mensagem, em formato JSON com a funcionalidade desejada para o *dashboard* de visualização usando o método *postMessage* do objeto iframe.

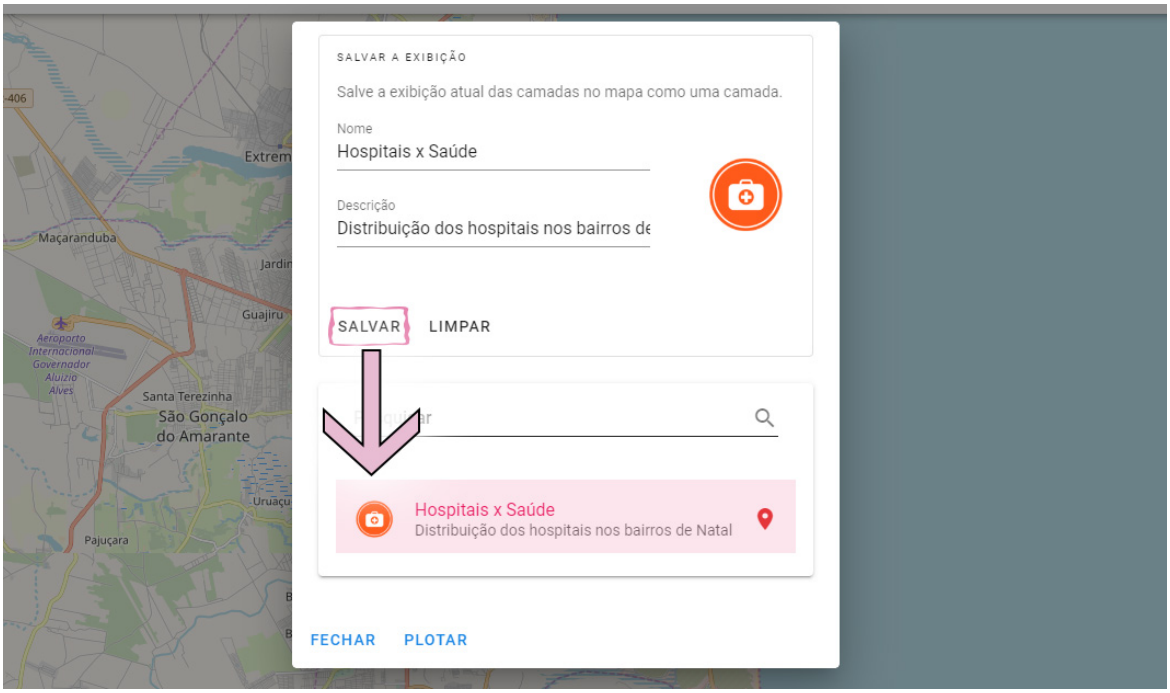

Figura 29 – Exemplo de salvamento de uma exibição que utiliza uma imagem.

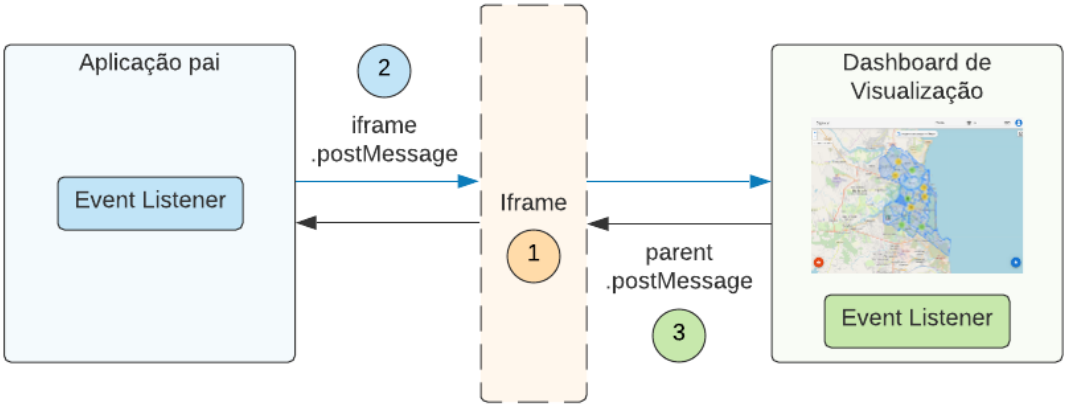

Figura 30 – Comunicação entre a aplicação pai e o *dashboard* de visualização (aplicação filha)

3. O *dashboard* de visualização executa a funcionalidade requisitada e devolve os dados resultantes.

Entre as funcionalidades disponíveis estão: adicionar ou remover camadas e entidades no mapa, realizar consultas, definir zoom, posição do mapa, emitir mensagens de alerta, ocultar barras a navegação, realizar autenticação e salvar exibição atual do mapa.

3.4 Aplicações Reais

O SGeoL tem sido aplicado como solução integradora de diversas fontes de dados heterogênicos e geográficos no cenário urbano da cidade de Natal. Por agregar uma gama de diferentes dados, o SGeoL tem atraído a atenção de desenvolvedores que demandam por mecanismos que facilitem o desenvolvimento de aplicações, bem como tem atraído a atenção da população, governos e gestores, devido as várias aplicações que executam nessa plataforma. Neste cenário em que o desenvolvimento de aplicações demandam interfaces de alto nível, fazendo uso dos *dashboards* propostos nesse trabalho, algumas aplicações foram e vem sendo desenvolvidas em parceria com o projeto *SmartMetropolis* e apresentadas a seguir:

3.4.1 Prefeitura Municipal de Natal

Uma das competências da Secretaria Municipal de Meio Ambiente e Urbanismo (SEMURB) da Prefeitura Municipal de Natal é a manutenção, armazenamento e fornecimento, para outros órgãos municipais, da base cartográfica oficial do Município de Natal.

Ao longo de vários anos, a SEMURB tem enfrentado problemas pontuais, como o processo de recebimento dos arquivos de modelos CAD fornecidos pela população, na atualização da base cartográfica oficial, ou ainda no compartilhamento de extensos arquivos cartográficos com a Secretaria Municipal de Tributação (SEMUT), que compartilham destes dados para realização de operações em campo, cobrança do Imposto Predial e Territorial Urbano (IPTU) e outros impostos e taxas.

A SEMURB, buscando manter uma base cartográfica atualizada, facilitar a comunicação entre as secretarias e melhor gerir o espaço territorial da cidade do Natal, firmou uma parceria com o projeto *SmartMetropolis*, para o desenvolvimento da plataforma SGeoL e dos *dashboards* de desenvolvimento e visualização. Neste projeto, dentre as facilidades proporcionadas pela utilização dos *dashboards* de desenvolvimento e visualização podemos citar:

- Uso do *dashboard* de desenvolvimento na importação de dados das camadas licenciamento, parcelamento, lotes e edificações, em utilização de arquivos de intercâmbio para arquivos CAD (DXF).
- Uso do *dashboard* de desenvolvimento para adicionar novas entidades nas camadas de licenciamento e parcelamento que inicia-se no processo de recebimento dos arquivos em modelo CAD fornecidos pela população na SEMURB.
- Uso do *dashboard* de visualização para visualizar, monitorar e auditar as ações realizadas sob as camadas de licenciamento e parcelamento realizadas pelas SEMUT

e SEMURB.

- Uso dos componentes de edição de dados geográficos do *dashboard* de visualização para editar dados das camadas de licenciamento e parcelamento durante ações em campo da SEMUT.

A Figura 31 mostra a tela do *dashboard* de visualização com a camada de lotes inseridos no o mapa da cidade de Natal com visualização modo satélite.

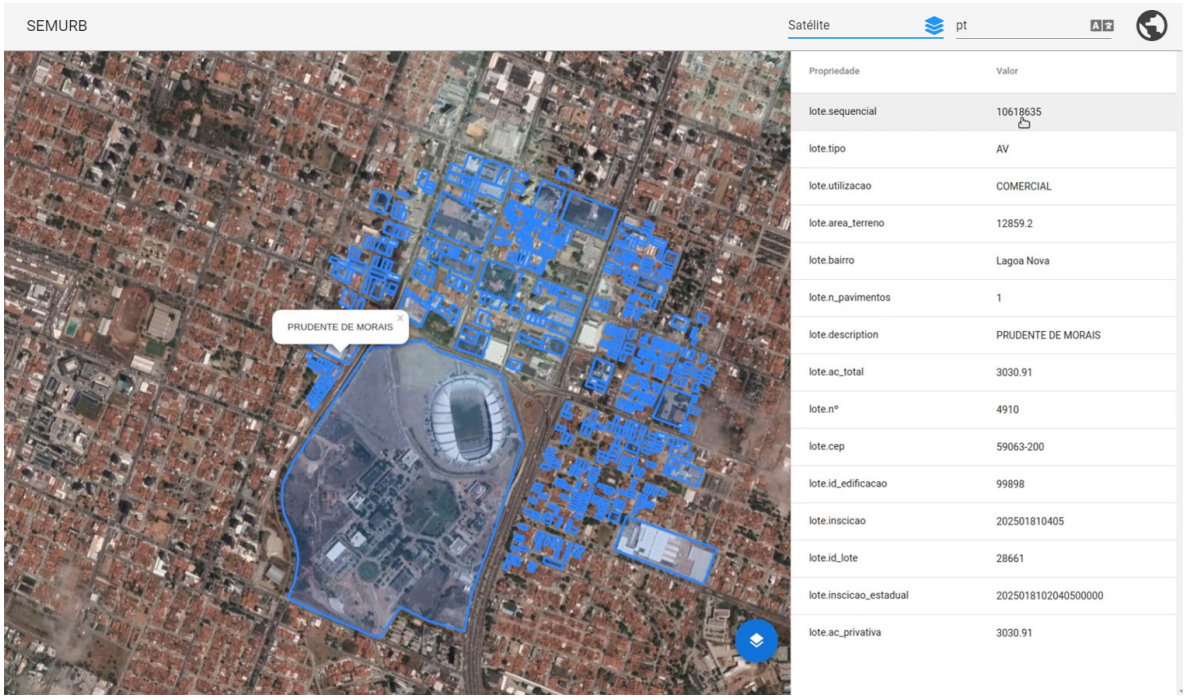

Figura 31 – Visualização da camada lotes e dos dados da entidade de nome Prudente de Morais através *dashboard* de visualização

Como visualizado na tela da Figura 31, ao clicar em um lote no mapa, são exibidos suas informações em um menu lateral à direita. Cada lote possui informações únicas, como número sequencial de inscrição, utilização (comercial, residencial e etc), área do terreno, bairro pertencente, número de pavimentos, entre outros. Estes dados foram importados usando o *dashboard* de desenvolvimento e podem também ser editados usando o *dashboard* de visualização.

3.4.2 Secretaria de Educação

A gestão pública da educação básica em seus diversos níveis (infantil, fundamental e médio) requer a ampla identificação de problemas e suas causas, para que se possa intervir sobre eles com sucesso. Para alcançar esse objetivo, pode-se fazer uso de soluções que correlacionem os diversos fatores envolvidos nesse contexto, ao invés de se ter iniciativas pontuais, cada uma abordando isoladamente um determinado tipo de problema ou fator.

Os dados educacionais — que fornecem informações quantitativas referentes ao ensino e à aprendizagem — são valiosos, porém eles não são suficientes se vistos de forma isolada.

Nesse sentido, a secretaria de educação do Estado, em parceria com *SmartMetropolis*, desenvolve um projeto para criar a camada de Educação, agregando dados do Sistema Integrado de Gestão da Educação (SigEduc[5]) aos dados fornecidos pelo Sistema Nacional de Avaliação da Educação Básica (SAEB), como os da Prova Brasil[6]. Esse projeto tem como objetivo abordar a problemática das vulnerabilidades educativas, sociais, econômicas e demográficas, dentre outras, que podem influenciar no impacto da educação escolar na formação discente nos ambientes escolares públicos brasileiros.

Do lado do SigEduc foram disponibilizados os dados georreferenciados das escolas de município de Natal e a localização aproximada dos alunos. Já os dados da Prova Brasil — que tem como objetivo avaliar, a partir de testes padronizados e questionários socioeconômicos, a qualidade do ensino oferecido pelo sistema educacional brasileiro —, foram usados para adicionar informações relacionadas com as escolas.

Como resultado do cruzamento de uma camada educacional com os dados de outras diversas camadas, foi possível produzir uma série de dados acerca de fatores de vulnerabilidades socioeducativa, vitais para a gestão da educação.

Neste projeto, algumas facilidades foram proporcionadas pela utilização dos *dashboards* de desenvolvimento e visualização, podemos citar:

- Uso do *dashboard* de desenvolvimento na importação e sincronização de dados das escolas e dos alunos através das APIs disponibilizadas pelo sistema SigEduc.
- Uso do *dashboard* de desenvolvimento na importação e sincronização dos microdados da Prova Brasil na composição dos dados das camadas das escolas e dos alunos.
- Uso do *dashboard* de visualização na identificação de fenômenos em que diferentes camadas de informação (incluindo educacionais), conectam-se, relacionam-se e compõem o cenário dos fatores que fragilizam a trajetória escolar de grande parte dos estudantes das escolas públicas.

A Figura 31 mostra a tela do *dashboard* de visualização com a camada educacionais (escola e seu respectivos alunos) sobrepostos sobre os bairros no o mapa da cidade de Natal.

Na tela da Figura 31, as esferas com referencias numéricas representam os aglomerados de alunos da escola Estadual Professor Varela Barca (representada pelo ícone da escola mais ao centro do mapa), inseridas sobre os bairros de Natal. A legenda no canto superior esquerdo representa, em uma escala de tom azul, a quantidade populacional dos bairros onde estão inseridos a escola e alunos. No canto inferior esquerdo é apresentado o

[5] https://sigeduc.rn.gov.br/sigeduc/
[6] http://portal.mec.gov.br/prova-brasil

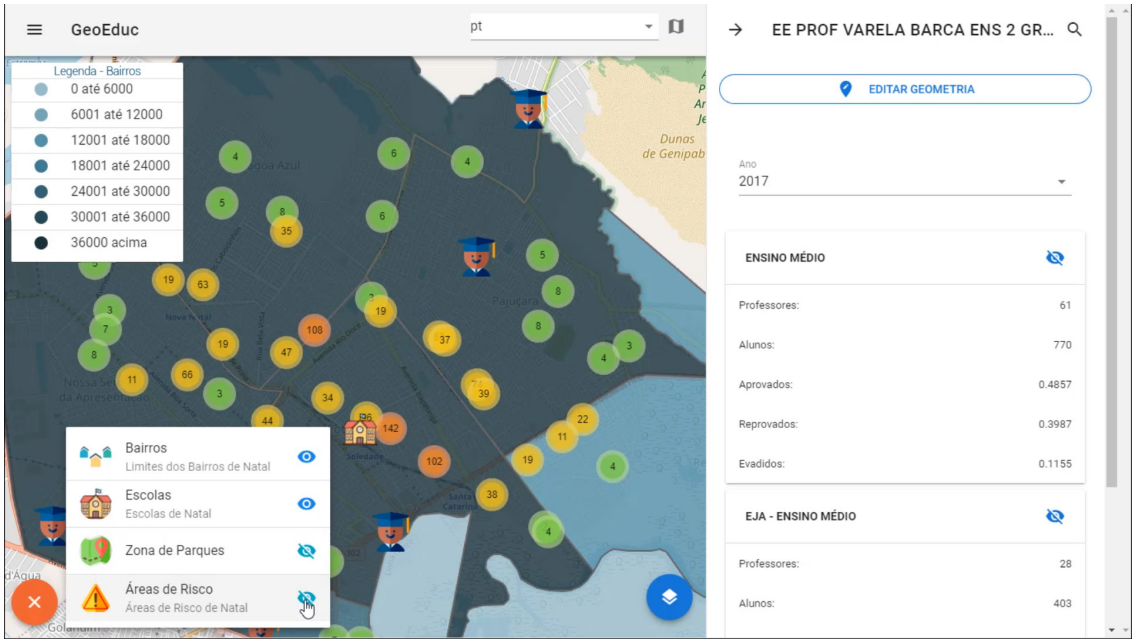

Figura 32 – Visualização da camada educacionais através *dashboard* de visualização

controle de visibilidade das camadas. Do lado direito é apresentada a barra lateral com as informações da escola selecionada (Estadual Professor Varela Barca), onde pode ser verificado o número de professores, total alunos, alunos aprovados, alunos reprovados e alunos evadidos para etapa do ensino médio desta escola.

3.4.3 Ministério Público do Rio Grande do Norte

Dentre as muitas atividades realizadas pelo Ministério Público Federal do Rio Grande do Norte (MPRN) está a realização de investigações contra os crimes de patrimônio público, pedofilia, macrocriminalidade, dentre outros crimes que prejudicam a sociedade. Com recente avanços na área da inteligência computacional no que se refere ao processamento de Big Data e Inteligência Artificial, tornou-se de interesse para o MPRN o desenvolvimento de ferramentas computacionais que os auxiliem nas mais diversas tarefas realizadas por ele.

Nesse contexto, o projeto INSIDE propõe-se a criar um sistema composto por diversas soluções que visam melhorar o fluxo de trabalho de um investigador, tornando o processo investigativo mais automatizado. Este sistema é composto por pequenas partes responsáveis por determinadas funções que inclui processamento de texto, imagem e dados geográficos, entre estes estão o *dashboard* de desenvolvimento e o *dashboard* de visualização.

Neste projeto, algumas facilidades foram proporcionadas pela utilização dos *dashboards* de desenvolvimento e visualização:

- Uso do *dashboard* de desenvolvimento na importação e sincronização de dados como IDEB, IDH, estabelecimentos, mortalidade, morbidade, arborização de vias públicas,

esgotamento sanitário, internações, população ocupada, receitas de fontes externas, rendimento domiciliar, salário médio mensal, taxa de escolarização, urbanização de vias públicas, entre outros. Grande parte destes dados vem de fontes como Informações de Saúde (TABNET[7]) do Ministério da Saúde, do Instituto Brasileiro de Geografia e Estatística (IBGE[8]), entre outras fontes de dados abertos governamentais.

- O *dashboard* de visualização foi embutido na aplicação do MPRN em Dados — que faz uso das API para iframe embutidos —, como componente de visualização para dados geográficos. Nessa aplicação, os dados geográficos originários de processamento de dados ou importações são relacionados e suas representações são salvas em formas de cartões. Estes cartões salvos na aplicação MPRN em Dados servem como mecanismos de acesso rápido aos de interesse aos procuradores ou a população, permitindo promover discussões em forma de comentários (semelhante a uma rede social), atribuição de tags, compartilhamento, entre outras funcionalidades.

A Figura 33 mostra a tela da aplicação do MPRN em dados onde o *dashboard* de visualização está embutido.

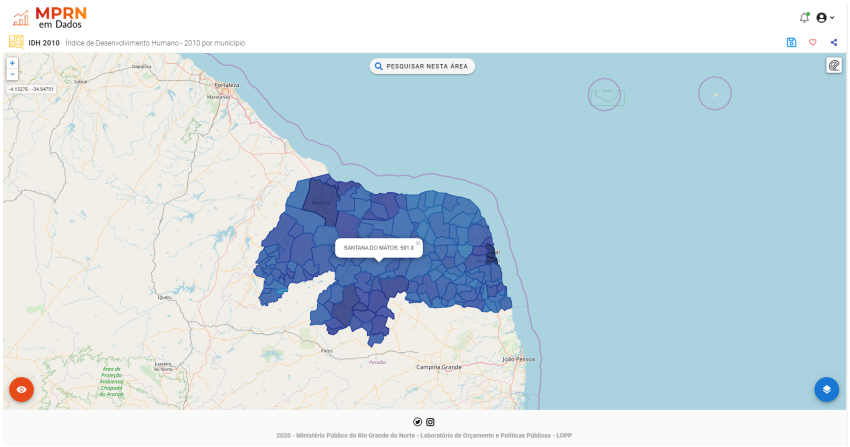

Figura 33 – *Dashboard* de visualização embutido na aplicação MPRN em Dados como componente de visualização para dados geográficos.

Como visto na Figura 33, através das APIs para iframe embutidos, a aplicação MPRN em dados incorpora as funcionalidades do *dashboard* de visualização. Ao carregar um cartão, a aplicação MPRN em dados inicia o *dashboard* de visualização solicitando que seja carregado a exibição salva. No canto superior direito é possível salvar um novo cartão, favorita-lo ou compartilhá-lo. Também é possível que o usuário adicione novas camadas ou use o controle de visibilidade.

[7] http://www2.datasus.gov.br/DATASUS/index.php?area=02
[8] http://www.econ.puc-rio.br/datazoom/censoMicro.html

4 Trabalhos relacionados

Algumas plataformas da atualidade têm aplicado conceitos de *dashboard* para o desenvolvimento de novas aplicações e visualização de dados. Nessa perspectiva, esse capítulo tem como objetivo apresentar uma visão geral de algumas plataformas para o desenvolvimento de aplicações que dão suporte ao desenvolvimento e visualização usando *dashboards*. As plataformas apresentadas nesse capítulo foram selecionadas a partir dos seguintes requisitos: Uso de componentes para construção de *dashboads*, suporte a dados geográficos e gerenciamento de usuários. A seção 4.1 apresenta a plataforma Freeboard; a seção 4.2 discorre sobre a plataforma ThingsBoard; a seção 4.3 discute sobre a plataforma Kaa Iot; a seção 4.4 apresenta a plataforma glue.things; a seção 4.5 resume algumas funcionalidades compartilhadas pelas plataformas discutidas nestes capítulo e, por fim, apresenta como os passos para a construção de *dashboard* foram envolvidos no desenvolvimento de *dashboards* nestas plataformas.

4.1 Freeboard

O Freeboard[1] funciona como uma ferramenta utilizada para visualizar dados em tempo real que alia seus componentes a outras APIs e possibilita a construção e o desenvolvimento de *dashboards* de controle que permitam uma melhor visualização dos dados que são produzidos pelos sistemas baseados na loT, o que os aproxima da subcategoria dos navegadores.

"(...) o Freeboard, uma plataforma que oportuniza a criação de interfaces gráficas possuindo informações sobre localização, temperatura, velocidade ou simplesmente o estado do dispositivo (TALBOT, 2014)"

O Freeboard dispõe de variedade de elementos de interfaces chamado de *widgets*. *Widgets* são componentes que permitem a interação do usuário com a interface e são responsáveis por facilitar a compreensão das informações do *dashboard*.

Alem disto, estes *widgets* podem ser configurados especificando fontes de dados, unidades de medida, legendas, intervalos de atualização de dados, entre outros. A partir da configuração realizada e do tipo de *widget* selecionado, o *widget* passará a exibir um elemento visual da informação, permitindo fácil e rápida interpretação ao usuário. Dentre as variedades destes componentes no Freeboard podemos citar, *widgets* de Mapas, HTML, Imagens, Texto, Medidor, Gráfico de linha, Ponteiro e Indicador de luz.

A Figura 34 apresenta vários *widgets* que podem apresentar dados de diferentes fontes e variados dispositivos, e ainda, receber dados atualizados, sob diferentes óticas, de

[1] <https://freeboard.io/>

Figura 34 – Dashboard formado por vários *widgets* no Freeboard

como está operando, permitindo a obtenção de informações mais gerais e consolidadas a respeito do ambiente monitorado.

O Freeboard conecta-se, em especial, com o Dweet[2] – uma plataforma de *middleware* para Internet das Coisas que tem a ideia de que qualquer produto, dispositivo ou máquina que possua conexão com a Internet possa compartilhar informações – oferece meios que expandem as potencialidades da loT, o que facilita a interconexão dos fluxos de dados para diversas utilizações. O uso do Dweet no Freeboard possibilita o uso de um componente exclusivo dessa interação, o *widget* de gráfico de série histórica do Dweet. Esse *widget* usa como fonte de dados um lista de dados e traça um gráfico de linha baseado em sua série histórica.

O funcionamento do Freeboard é bastante simples, resumindo-se a: adicionar dispositivos e fontes de dados, acrescentar (e movimentar) *widgets* e compartilhar (e duplicar/ clonar) as informações instantaneamente, tendo total controle do acesso. Devido à sua praticidade, o Freeboard atende, inclusive, a cenários como o Jornalismo, visto que cria painéis onde a rotatividade de informações se dá ao passo que são feitas atualizações, através dos dispositivos e/ ou máquinas que a ele estiverem conectadas. O Freeboard é uma plataforma de código aberto, que dispõe de um serviço de hospedagem, onde é possível publicar *dashboards* públicos gratuitamente. Para utilização privada, necessita ser

[2] <https://dweet.io/>

realizada a adesão a um plano.

O Freeboard tem uma interface intuitiva e de fácil configuração. Porém, faltam algumas funcionalidades como: gerenciador de dispositivos IoT, gerenciador de usuários e permissões, mecanismos para integração de dados. Estas funcionalidade geralmente presentes em plataformas de *middleware*, como por exemplo o SGeoL e FIWARE, como também nas plataformas ThingsBoard e Kaa Iot que apresentadas mais adiante. Desta maneira, pode-se considerar o Freeboard como uma plataforma exclusivamente de suporte a visualização de dados.

4.2 ThingsBoard

A ThingsBoard[3] é uma plataforma loT de código aberto para coleta, processamento, visualização e gerenciamento de dados. A plataforma possibilita que dispositivos loT conectem-se, usando os protocolos MQTT, CoAP ou HTTP, para enviar, monitorar e coletar dados, exibir dados de telemetria em seu painel, acionar algo ou permitir que os dados sejam consumidos por outra aplicação.

As regras para lidar com os dados recebidos de dispositivos IoT ou de outras plataformas podem ser definidas individualmente, e estes dados podem ser visualizados com *widgets* internos ou personalizados, viabilizando dashboards flexíveis. Pode-se iniciar por definir cadeias de regras de processamento de dados, bem como: de que forma esses dados serão visualizados ou acessados por outras aplicações, se serão exibidos em forma de gráficos ou apenas números, se serão utilizados os *widgets* disponíveis ou se será criada uma visualização própria destes dados. Facilidades como o recebimento de alarmes sobre eventos de telemetria, atualizações de atributos, inatividade do dispositivo e ações do usuário são outras vantagens da plataforma. A Figura 35 apresenta alguns dos *widgets* disponibilizados pelo ThingsBoard.

Como visto na Figura 35, o ThingsBoard oferece a possibilidade de criar *dashboards* avançados que podem ser atualizados em tempo real, garantindo a visualização de dados que podem ser personalizados, como: *widgets* de alarmes, medidores analógicos e digitais, cartões, gráficos, controle, datas, administradores de entidades, *gateways widgets*, portas de entrada e saída, mapas, entre outros.

O ThingsBoard possui um *token* de acesso exclusivo, o que possibilita que cada dispositivo que envia dados seja organizado de acordo com seu perfil individual, permitindo, ainda, que sejam definidas as relações entre os dispositivos conectados, entre os ativos, os clientes ou quaisquer outras entidades. Isso permite que seja feita a identificação da propriedade dos dados usados, além de que os protocolos utilizados para conectar os dispositivos são suportados por muitas linguagens de programação. Para além disso, o

[3] <https://thingsboard.io/>

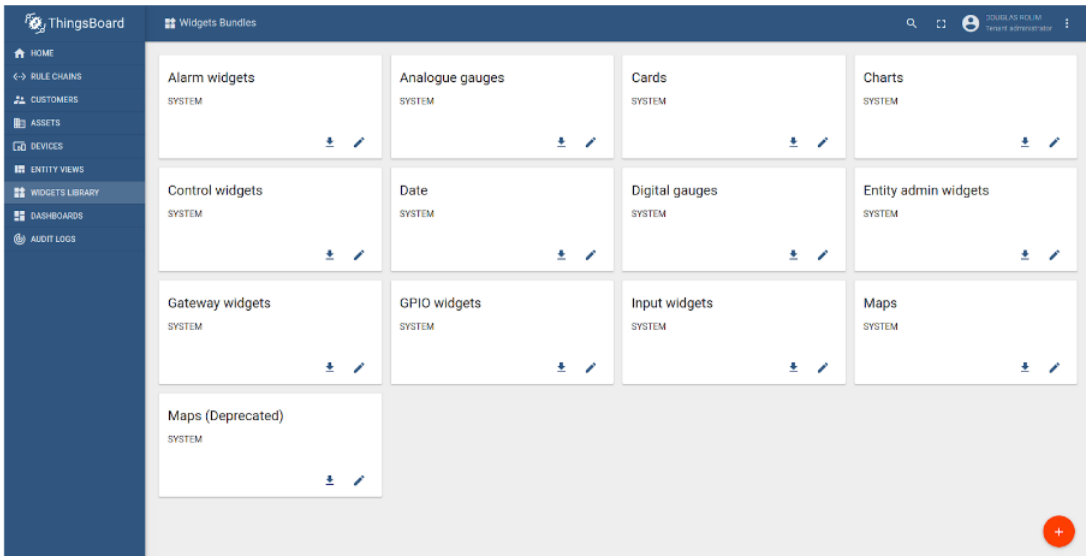

Figura 35 – Biblioteca de *widgets* para montagem de *dashboards* no ThingsBoard

ThingsBoard combina escalabilidade, tolerância a falhas e bom desempenho para que os dados que nele foram armazenados nunca sejam perdidos.

Devido ao seu painel, que fornece a possibilidade de diversas visualizações dos dados – como são os casos dos gráficos de linhas, que mostram dados de séries temporais, e a barra de medida, que exibe um único valor –, o ThingsBoard é adequado, inclusive, para suprir a necessidade de estações de trabalho médico que necessitam exibir dados em tempo real e detecção precoce.

A Figura 36 apresenta o exemplo de um *dashboard* para monitoramento de temperatura na plataforma em questão.

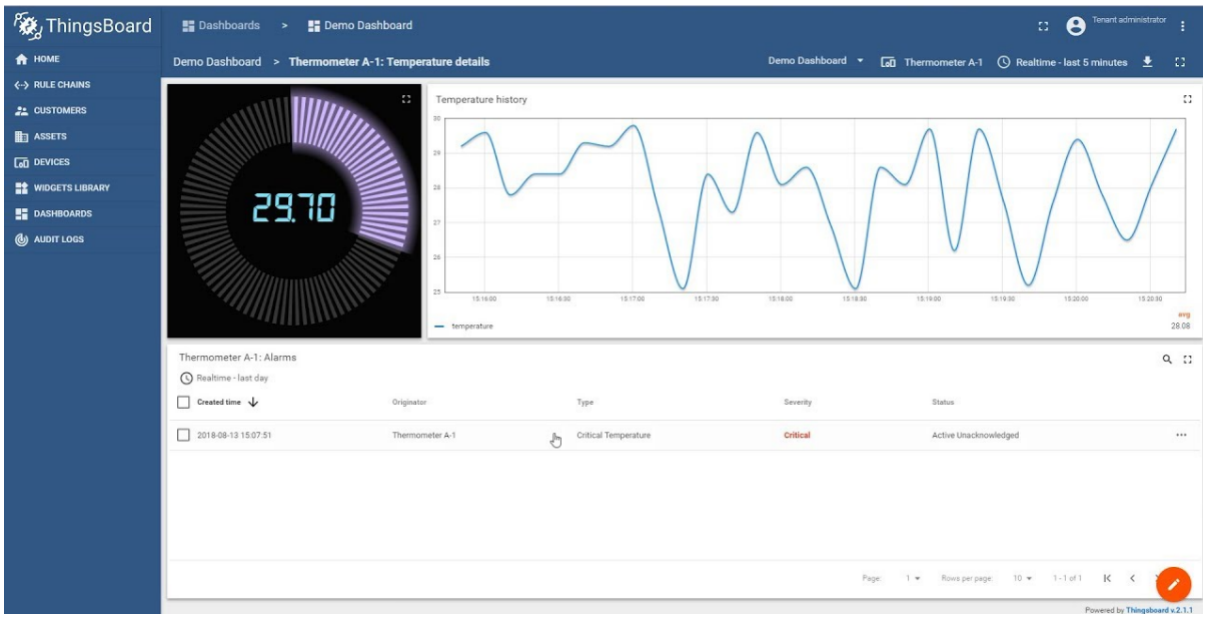

Figura 36 – Visualização de dados no *dashboard* do ThingsBoard

Para formular um *dashboard* para monitoramento de temperatura (Figura 36) são usados dois tipos de *widgets*. O primeiro é um componente de calibre, que recebe como fonte de dado a última temperatura registrada e mostra a este dado em uma variação de 0 até 100. Já o segundo é um componente de gráfico de linha (geralmente usado para demonstrar uma sequência numérica de um determinado dado ao longo do tempo), que neste caso, é utilizado para demonstrar uma sequência de temperaturas nos seus horários em que ocorreram as medições.

O ThingsBoard, diferentemente da plataforma Freeboard.io é uma plataforma mais completa. Nessa plataforma é possívelutilizar gerenciador de dispositivos IoT, gerenciador de usuários e permissões, porém não há mecanismos para integração de dados. Além disso, para usar os *widgets* do ThingsBoard e montar *dashboards*, é necessário que os dispositivos estejam cadastrados na plataforma e enviem seus dados diretamente para a esta plataforma. Isso impossibilita a construção de *dashboard* que usem dados externos de outras plataformas, deixando de fora ainda, fonte de dados que possam ser acessadas para importação ou sincronização dos dados.

4.3 Kaa Iot

A Kaa IoT[4] é uma plataforma de *middleware* de código aberto para implementação de aplicações para IoT. De acordo com Kaa (2010), os principais recursos da plataforma Kaa IoT são: o monitoramento de dispositivos em tempo real, a implantação e configuração de dispositivos remotos, a coleta e análise de dados de sensores, dentre outros.

O Kaa IoT também permite a coleta de dados de dispositivos que usam protocolos leves de IoT. Como exemplo, podemos citar MQTT e CoAP. Além disso, o Kaa emprega uma arquitetura de *gateway* que se comunica com o dispositivo através de protocolo de rede local ou através de um protocolo de proximidade, além de realizar a conversão de mensagens no nível de transporte ou até mesmo representa-os para torná-los acessíveis na Internet.

Assim como o Freeboard e ThingsBoard, o componente de visualização de dados do Kaa compreende um rico conjunto de *widgets*, com algumas particularidades adicionais, como por exemplo, *widget* de configuração de um valor (onde o usuário pode aumentar ou diminuir um valor usando botões), formulário de configuração (permite visualizar e editar várias métricas de configuração), configuração de JSON (permite visualizar e editar a configuração como um JSON bruto), controle (permite ao usuário executar comandos em um terminal e exibir seus resultados). Estes *widgets* possibilitam a visualização de diferentes tipos de dados, sejam eles em forma de telemetria, estatística, geolocalização, metadados, filtros, atualizações de software ou dados históricos e atuais. Além disso, alguns

[4] <https://www.kaaproject.org/>

destes *widgets* permitem que o usuário interaja com os dispositivos enviando comandos, alterando configurações e metadados, dentre outras funções.

Todos os *widgets* são configuráveis e permitem alterar as fontes de dados, bem como a representação visual. Para resolver casos de usos especiais, o componente de visualização Kaa permite a personalização dos *widgets*.

Um exemplo de *dashboard* no Kaa Iot para monitoramento e controle da temperatura de uma sala é apresentado a seguir (Figura 37).

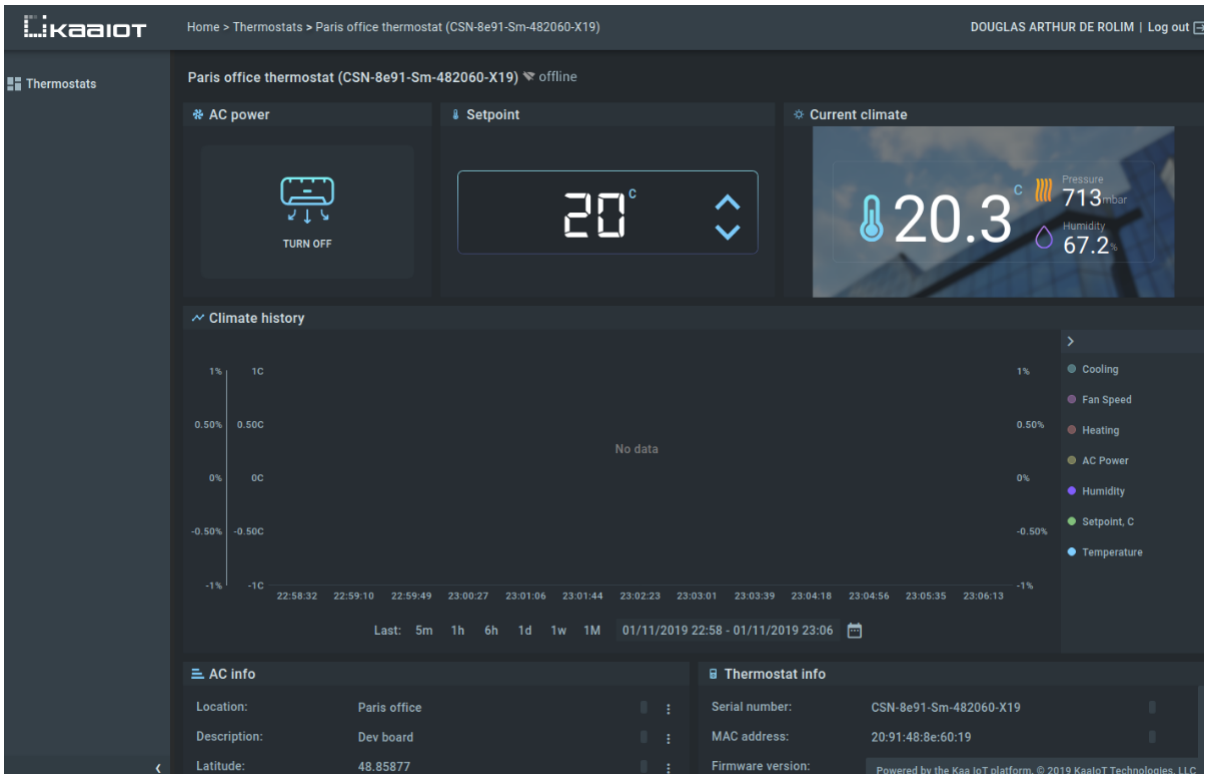

Figura 37 – Visualização de dados no *dashboard* do Kaa IoT

Para a construção do *dashboard* da Figura 37, foram usados quatro diferentes tipos de *widgets*. O primeiro um *widget* é um atuador de controle de estado responsável por enviar um uma requisição para ligar ou desligar o resfriador. O segundo também é um *widget* atuador, porém este envia requisições para que o dispositivo defina o valor da temperatura do resfriador desejada. O terceiro é um *widget* de HTML, onde é exibida usando imagem de fundo e os valores de temperatura, umidade e pressão recebidos pelo dispositivo que monitora estas informações. Já o último, é um *widget* de gráfico de linha, que assim como o *widget* apresentado na Figura 36, também é utilizado para demonstrar uma sequência de temperaturas nos seus horários em que ocorreram as medições.

Enquanto ajudam a organizar os *widgets* em grupos lógicos e na definição do seu *layout*, os painéis, podem, ainda, possuir *hiperlink* que simplifica a navegação nos conjuntos de dados complexos para vários dispositivos. Para além disso, o Kaa suporta modelos de *dashboard*, o que permite reutilizar uma configuração para vários *dashboards* de dispositivos

diferentes.

Devido às suas APIs abertas para integração de sistemas de terceiros, o Kaa pode ser conectado a ferramentas de visualização e exploração de dados de aplicações externas.

O Kaa, assim como a ThingsBoard, permite gerenciador de dispositivos IoT, gerenciador de usuários e permissões, mas não possui mecanismos para integração de dados. Para usar os *widgets* e montar *dashboard*, também é preciso que as fontes de dados sejam configuráveis para enviar suas informações para a plataforma.

4.4 Glue.things

Outra plataforma para gerenciar dispositivos de IoT e desenvolvimento de aplicações é glue.things (KLEINFELD et al., 2014). O glue.things é uma plataforma de *mashup* coerente e robusta, cobrindo aspectos de entrega e gerenciamento de fluxos de dados de dispositivos, aplicações e sua integração. Nesse sentido, o glue.things se baseia em redes de comunicação comprovadas em tempo real para facilitar a integração de dispositivos e o gerenciamento de fluxo de dados.

O glue.things permite conectar dispositivos através de protocolos (HTTP, CoAP, MQTT) e através de *socket web* e gerenciar o controle de acesso aos dados produzidos pelos dispositivos.

Como diferencial em relação ao Freeboard, ThingsBoard e KaaIoT — que oferecem a configurações de *widgets* através de interfaces específicas — , esta plataforma permite a configuração dos *widgets* através do uso de *on marshups*. A Figura 38 apresenta o *dashboard* gerencial do glue.things (à esquerda) e o desenvolvimento de aplicações através de *marshups*.

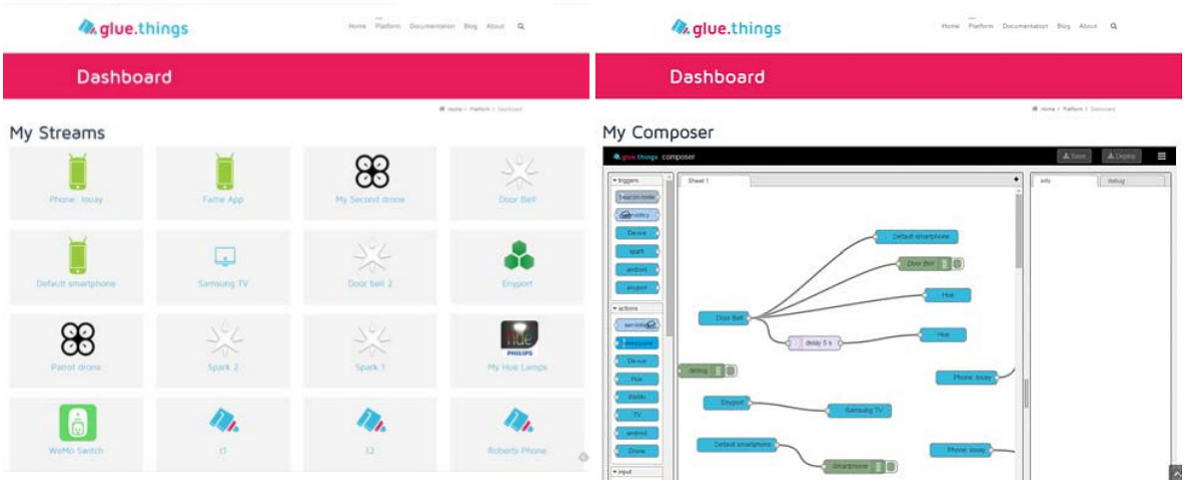

Figura 38 – Dashboard do glue.thigs (a esquerda) e desenvolvimento aplicações usando *marshups* (a direita)

Através das *marshups*, desenvolvedores podem desenvolver aplicações de forma

simplificada, uma vez que estes componentes são facilmente configurados e podem ser combinados para compor novos componentes. Por exemplo, a plataforma fornece componentes que podem ser configurados para receber um fluxo de dados de um sensor de presença, que pode ser conectado a um *widget* que exibirá os dados históricos coletados em um gráfico. Esse componente também pode ser conectado a um atuador que executará alguma ação com base nos dados, como acender ou apagar uma lâmpada.

A plataforma glue.things não pode ser avaliada por não haver uma versão disponível em um repositório, nem tão pouco online para o uso. Entretanto, o uso de *mashups* se mostra funcional para configuração de fonte de dados para exibição de informações através de *widgets*, uma vez que o processo de configuração é simplificado e intuitivo.

4.5 Comparativo das plataformas analisadas

Nesse capítulo foram apresentadas algumas plataformas para IoT e *middleware* com foco no desenvolvimento de aplicações através do uso de *dashboards*. Para proporcionar um efeito comparativo das plataformas apresentadas, a Tabela 1 reúne e exibe os recursos comuns existentes em pelo menos em duas destas plataformas.

Tabela 1 – Tabela comparativa dos recursos encontrados nas plataformas relacionadas

Plataforma	Protocolos	Gerência de dispositivos	Widgets	Gateway	Gerência de Usuários
Freeboard	MQTT, HTTP	Não	Mapas, HTML, Imagens, Texto, Medidor, Gráfico de linha, Ponteiro e Indicador de luz	Não	Não
ThingsBoard	MQTT, HTTP, CoAP	Sim	Alarmes, Medidores Analógicos e Digitais, Cartões, Gráficos, Controle, Datas, Administradores de Entidades, Gateways widgets, Portas de entrada e saída	Sim	Sim
KaaIoT	MQTT, HTTP, CoAP	Sim	Mapas, HTML, Imagens, Texto, Mediddores, Gráficos, Valor númerico, Formulário de Configuração, Configuração de JSON , Controle de Atuadores	Sim	Sim
glue.things	MQTT, HTTP, CoAP	Sim	Não especificados	Não	Sim
SGeoL	MQTT, HTTP, CoAP	Sim	Numérico, Lista, Mapas	Sim	Sim

A coluna 1 dispõe dos nomes da plataforma; a coluna 2 exibe os protocolos

suportados pelas plataformas; a coluna 3 apresenta quais plataformas são capazes de gerenciar dispositivos IoT; a coluna 4 retrata os *widgets* suportados pelas plataformas para a elaboração de *dashboards*; a coluna 5 apresenta quais plataformas tem a capacidade de atuar como *gateway*, comunicando-se com o dispositivo ou aplicações através de protocolo de rede local ou através de um protocolo de proximidade; e, a coluna 5, retrata quais plataformas dispõe de quais plataformas fazem o gerenciamento de usuários e permissões.

Para a análise comparativa apresentada na Tabela 1, foram incluídos para a plataforma SGeoL as funcionalidades proporcionadas pelo *dashboards* propostos por este trabalho.

Como visto anteriormente, na Figura 2, para o desenvolvimento de um *Dashboard* devemos seguir alguns passos, consistindo, em linhas gerais: (i) selecionar as métricas a serem utilizadas; (ii) preencher o *dashboard* com dados; e (iii) estabelecer relações causais entre os itens.

O primeiro passo consiste em definir as métricas a serem utilizadas, isto é, estabelecer os itens avaliadores de desempenho das entidades presentes nas mais variadas camadas e seus relacionamentos. Porém, tendo em vista as particularidades dos dados, como por exemplo, os tipos de dados (caracteres, inteiros, flutuantes, etc), as grandezas (tempo, temperatura, corrente elétrica, etc.), as unidades de medidas (metro, quilograma, ampere, etc.) ou ainda os formatos (data, horários, etc), definir métricas pode não ser trivial. A depender do dado de interesse, definir métricas pode ser fácil ou complexo.

Em vista disso, analisando as plataformas apresentadas neste capítulo, constata-se que as mesmas convergem em usar componentes chamados *widget* para definição das métricas. Cada *widget* tem a capacidade de representar uma ou mais particularidade do dado, formalizando a informação em um elemento de interação para o usuário. Por exemplo, o uso de um *widget* comum entre as plataformas, como *widget* de indicador de luz, representa o estado atual de uma informação, como uma luz acesa ou apagada, uma porta aberta ou fechada, ou ainda se há presença de um indivíduo em uma sala.

O segundo passo consiste em preencher o *dashboard* com dados. Neste passo, nas plataformas Freeboard, ThingsBoard, Kaa Iot e glue.things, foi verificado que após selecionar um *widget* para representação de métrica, é necessário selecionar pelo menos uma fonte de dado para, em seguida, selecionar um ou mais os dados presentes nas fontes. Por sua vez, esses dados comporão os *widgets*, que os apresentarão de maneira clara e representativa.

A glue.things, como foi apresentada na seção 4.4 deste capítulo, faz uso de *marshups*, permitindo aos desenvolvedores realizarem composições de recursos (dispositivos, fontes de dados ou serviços) sem a necessidade de conhecimento mais aprofundado na plataforma. Esse recurso facilita a configuração dos dados apresentados através dos *widgets*.

O Freeboard, ThingsBoard, Kaa IoT permitem que o desenvolvedor crie funções em linguagem JavaScript para tratar os dados. Essas funções podem, por exemplo, fazer

conversão de medida do dado, transformação em porcentagem, operações matemáticas, cruzamento de dados, e etc. Uma função desse tipo é apresentada na Figura 39.

```
(datasources["000019b2b25f"]["cpuT"] + datasources["00001bd224ff"]["cpuT"]) / 2
```

Figura 39 – Função que calcula a média de processamento entre dois dispositivos

A Figura 39 demonstra uma simples função que calcula a média de processamento entre dois dispositivos de IoT (referenciados pelos nomes "00019b2b25f"e "0001bd224ff"e o dado "cpuT", presente em ambos os dispositivos).

Exemplificando, podemos usar a função da Figura 39 para criar uma métrica que apresenta o uso médio processamento entre estes dois dispositivo. Para tal, selecionamos um *widget* de calibre no Freeboard, em seguida, escolhemos as duas fontes de dados descritas, definimos a função e temos como resultado, um *widget* conforme apresentado pela Figura 40.

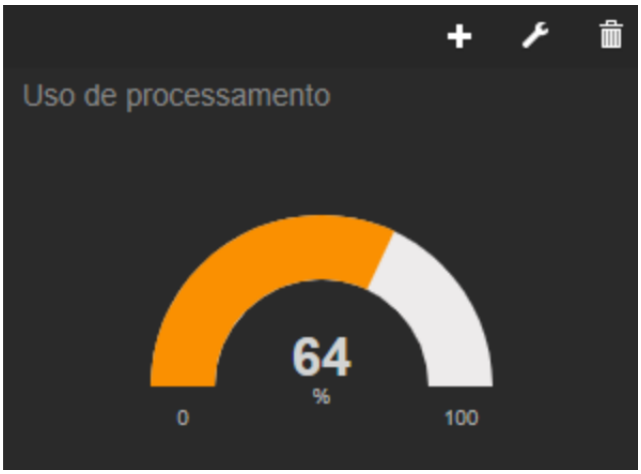

Figura 40 – *Widget* de calibre no Freeboard

Um outro exemplo, usando os dados do *Middleware* SGeoL, para definir uma métrica que apresenta a porcentagem de alunos reprovados em um determinado semestre, deve-se utilizar um *widget* que use gráfico de linha (apresenta dados de séries temporais, e a barra de medida, que exibe um único valor), selecionando como fonte de dados a camada escola e, em seguida, a propriedade que possua este dado diretamente através do semestre. Caso o dado não esteja totalizado, é possível criar uma função que calcule essa média ao usar a propriedade de situação escolar para somar todas as ocorrência de reprovação naquele semestre e dividi-la pela quantidade total de escolas.

Uma das características também presentes nas plataformas Freeboard, ThingsBoard, KaaIot e glue.things, é a capacidade de trabalhar com os dados de forma instantânea. Essa aptidão mantém em exibição no *widget*, sempre a última versão dado disponível na fonte.

Pauwels et al. (2009) afirma que um problema na seleção das fontes de dados, trata-se da periodicidade com que os dados são atualizados, uma vez que alguns dados

podem ser coletados diariamente, enquanto outros são coletados anualmente, ou até com menos frequência. Assim, para que gestores estejam aptos a realizarem ações precisas, é importante que os dados estejam também atualizados. Dessa maneira, faz-se necessário também que estas plataformas facilitem a manutenção de dados.

O terceiro passo envolve a determinação dos relacionamentos subjacentes entre as métricas. Essa etapa difere os *dashboard* de uma apresentação simples de informações (o que consideramos um requisito mínimo para um painel) para um entendimento profundo das informações para o apoio de decisão. De acordo com Kamakura Schneider et al. (2005), Kamakura et al. (2002), os métodos para estabelecer relações métricas incluem modelos de equações estruturais e análise de ligação.

Neste passo, as plataformas Freeboard, ThingsBoard e KaaIoT permitem que o desenvolvedor e o usuário possa arrastar os *widgets* e aninha-los, de forma que a informação fique organizada, e para que sejam criadas estas relações métricas. No caso do FreeBoard, é possível ainda criar caixas onde pode-se inserir vários *widgets* dentro, o que melhora ainda mais essa organização, conforme apresentado pela Figura 41.

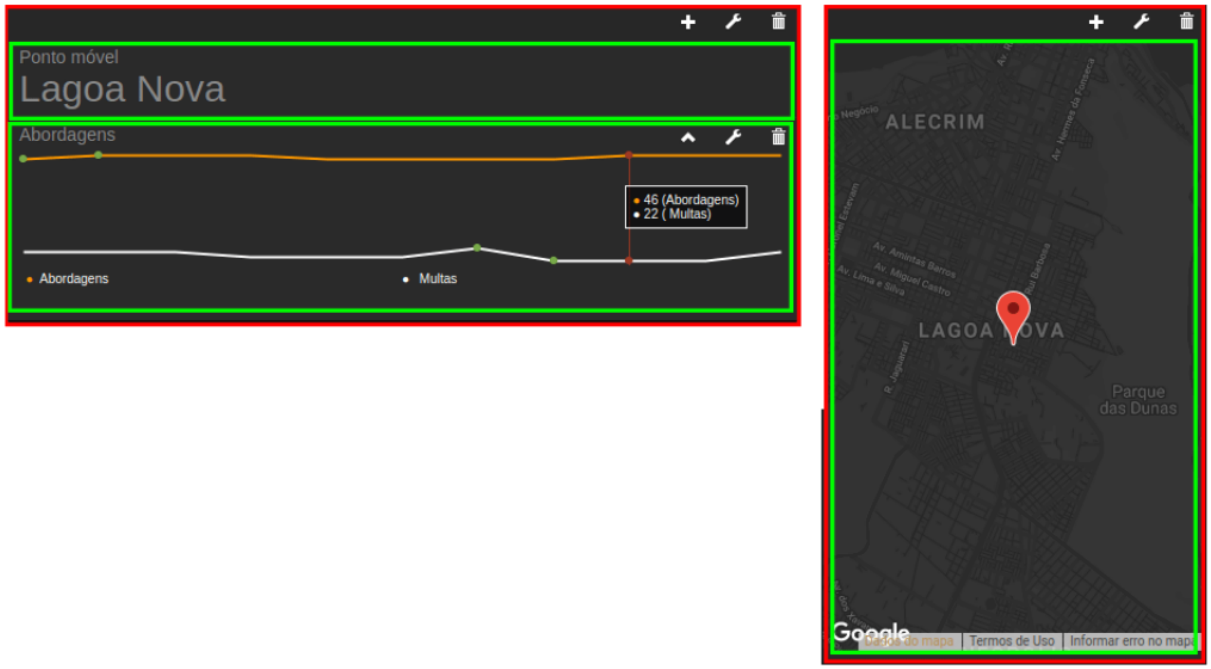

Figura 41 – Organização das caixas de dos *widgets* no Freeboard

A Figura 41 demostra de que maneira é dada a organização dos *widgets* (em retângulos na verde) dentro de caixas (retângulos em vermelhos). Sendo na primeira caixa com dois *widgets* e a segunda com apenas um. Dessa maneira, pode-se por exemplo, dois ou mais *widgets* que estejam ligados, sintetizando informações e exibindo informações de maneira uniforme.

Já o Kaa Iot possui um *widget* de mapa onde é possível selecionar entidades e visualizar um outro *dashboard* previamente montado. Da mesma maneira, o Kaa IoT

também possui um *widget* específico para listagem de dispositivos, onde é possível definir um *dashboard* para cada tipo de dispositivo, conforme apresentado pela Figura 42.

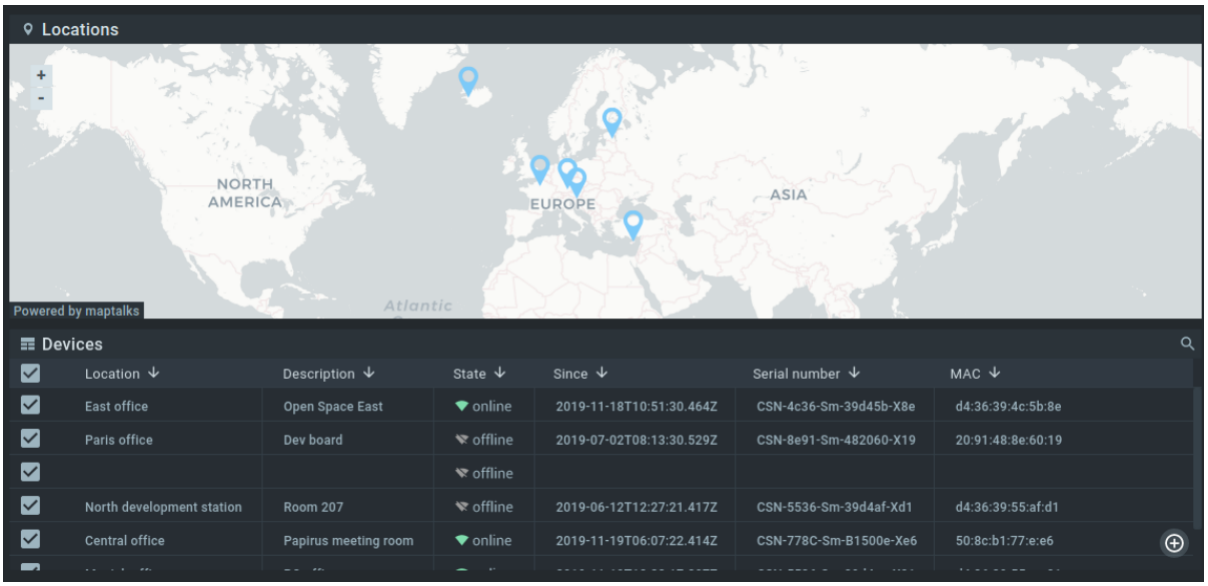

Figura 42 – *Widget* para seleção de dispositivos para exibição de *dashboard*

A Figura 42 mostra o *widget* de seleção de dispositivo no mapa e o *widget* de seleção de dispositivo em tabela. Selecionar o dispositivo no mapa pode ser interessante quando se quer selecionar baseado na localização, enquanto selecionar através da tabela pode ser mais prático para seleção baseada em atributos, como descrição do dispositivo, estado atual, entre outros.

É importante observar que estas plataformas convergem em uso de *widgets* para apresentação dos dados, em configurações para definição das fontes de dados, legendas e frequência de atualização da informação. Dessa forma, pode-se concluir que os *dashboard* e seus *widgets* devem ser configuráveis ao ponto que, os próprios usuários ou desenvolvedores definam as métricas considerando estes tipos de dados, em suas particularidades e relações. Estas características ajudam a moldar as funcionalidades que os *dashboards* de desenvolvimento e visualização devem apresentar.

5 Avaliação

Este Capítulo descreve o processo de avaliação dos *dashboards* de desenvolvimento e visualização, bem como apresenta e discute os resultados. Para avaliá-los, foi realizado um experimento controlado para verificar se eles facilitam o desenvolvimento de aplicações de cidades inteligentes e a visualização de dados relacionados. Esse estudo foi realizado usando o processo sistemático proposto por Wohlin et al. (2012).

A seção 5.1 discute os objetivos da avaliação; a seção 5.2 apresenta as questões de pesquisas envolvidas na avaliação; a seção 5.3 introduz as métricas usadas para responder as questões de pesquisas envolvidas nesta avaliação; a seção 5.4 expõe os questionários usados para avaliação; a seção 5.5 mostra o planejamento da experimentação; a seção 5.6 retrata as tarefas desempenhadas pelos participantes do experimento; A seção 5.7 apresenta as hipóteses para os objetivos desse experimento, e por fim, a seção 5.8 relata os resultados obtidos e apresenta uma análise sobre eles.

5.1 Objetivos da avaliação

Os objetivos da avaliação partiram dos objetivos estabelecidos para este trabalho e foram definidos com base na metodologia *Goal-Question-Metric (GQM)* proposta por Caldiera e Rombach (1994). O paradigma GQM é um mecanismo para avaliar produtos e processos de *software*, usando medição, bastante utilizado na Engenharia de Software. De acordo com Basili (1992), essa abordagem sistemática adapta e integra objetivos com modelos do processo de software, produtos e perspectivas de qualidade de interesse, com base nas necessidades específicas do projeto e da organização. Seguindo o método GQM, o passo inicial consiste em definir os objetivos que serão alcançados no processo de avaliação.

Como apresentado no capítulo 1, os objetivos deste trabalho são: *(i) Propor uma arquitetura conceitual para os dashboards de visualização e desenvolvimento; (ii) Concretizar a arquitetura proposta através de uma implementação da mesma para uma plataforma de desenvolvimento de aplicações para cidades inteligentes; (iii) Realizar uma avaliação dos dashboards propostos e verificar os benefícios trazidos em termos de diminuição do esforço de desenvolvimento de aplicações e melhoria da usabilidade.* Estes objetivos foram otimizados para dar origem ao objetivo geral (*Goals*) do experimento controlado.

G1: Analisar os *dashboards* de desenvolvimento e visualização com o propósito de verificar se eles facilitam o desenvolvimento de aplicações de cidades inteligentes e a visualização de dados relacionados.

A partir do objetivo da avaliação foram formadas as questões (*Questions*) especificadas na seção 5.2.

5.2 Questões

Duas questões de pesquisa foram definidas a partir do objetivo da avaliação:

QP1: O uso do *Dashboard* de desenvolvimento facilita o desenvolvimento de aplicações?

QP2: Qual é o nível de satisfação do usuário com a usabilidade do *Dashboard* de visualização?

A investigação do QP1 considerou o uso de *Dashboard* de desenvolvimento em comparação com o uso direto de APIs fornecidas pelo SGeoL (consulte a Seção 3.3.2). Por meio das APIs do SGeoL, os desenvolvedores podem gerenciar dados (inserir, atualizar, remover) de camadas e entidades, além de gerenciar usuários e permissões. O uso de *Dashboard* associado ao SGeoL pode facilitar o uso desses recursos, pois fornece uma interface que permite ao desenvolvedor validar e manipular dados e informações geográficas, enquanto abstrai as preocupações com os tokens de segurança.

Por sua vez, a investigação do QP2 consistiu em avaliar o próprio *Dashboard* de acordo com um conjunto de questões de usabilidade, dado que um *dashboard* bem projetado e com bons métodos de visualização pode ajudar na mineração de conhecimento (JING et al., 2019).

5.3 Métricas

Duas métricas foram escolhidas para responder a QP1, a saber, a facilidade e a satisfação percebidas do uso (*métrica M1*) e o esforço de desenvolvimento (*métrica M2*).

A métrica M1 abordou a avaliação de *Dashboard* em uma perspectiva qualitativa, uma vez que os participantes foram solicitados a responder a um questionário baseado na teoria de Davis (1989), que avalia o nível de concordância do usuário com algumas declarações.

A métrica M2 avaliou o esforço de trabalho medindo o tempo gasto por um usuário para executar um conjunto de tarefas, considerando o uso do *Dashboard* em relação ao uso direto das APIs do SGeoL (ou seja, sem o uso do *dashboard* de desenvolvimento).

Para responder ao QP2 sobre o uso de *Dashboard*, uma análise exploratória foi realizada com base na *System Usability Scale*, (SUS) proposta por Brooke (1996). O objetivo dessa avaliação é analisar o nível de satisfação do usuário em relação à usabilidade e identificar elementos que possam interferir na avaliação da satisfação.

Os questionários usados na avaliação das métricas apresentadas nessa seção são apresentados a seguir.

5.4 Questionários

A métrica M1 foi avaliada através dos questionários respondidos pelos participantes para avaliar o *Dashboard* de desenvolvimento em comparação com o uso direto de APIs fornecidas pelo SGeoL. Cada abordagem possui 3 questões que abordam as facilidades de uso e 3 questões para satisfação de forma intercalada, totalizando 12 questões apresentadas a seguir na Tabela 2.

Tabela 2 – Questionário de facilidade e satisfação

Item	Questão
1	O quão fácil ou difícil você considera realizar a autenticação (API de Segurança da SGeoL) para utilização dos recursos de camadas e entidades?
2	O quão satisfeito você ficou ao utilizar a API de Segurança do SGeoL?
3	O quão fácil ou difícil você considera criar utilizar as APIs de Camadas para a criar camadas usando a SGeoL?
4	O quão satisfeito você ficou ao utilizar as APIs de Camada da SGeoL?
5	O quão fácil ou difícil você considera utilizar as APIs de Entidades para a criar entidades para camadas usando a SGeoL?
6	O quão satisfeito você ficou ao utilizar as APIs de Entidades da SGeoL?
7	O quão fácil ou difícil você considera realizar autenticação usando o *Dashboard* de Desenvolvimento para utilização dos recursos de camadas e entidades?
8	O quanto satisfeito você ficou após usar a autenticação através do *Dashboard* de Desenvolvimento?
9	O quão fácil ou difícil você considera criar camadas usando o *Dashboard* de Desenvolvimento?
10	O quanto satisfeito você ficou após criar uma camada utilizando o *Dashboard* de Desenvolvimento?
11	O quão fácil ou difícil você considera criar entidades para as camadas usando o *Dashboard* de Desenvolvimento?
12	O quanto satisfeito você ficou após criar entidades utilizando o *Dashboard* de Desenvolvimento?

A partir dos valores de M1 é possível identificar a opinião dos participantes em relação a facilidade de uso e a satisfação provida pelo *dashboard* de desenvolvimento em reação ao uso direto das APIs do SGeoL. Caso os valores obtidos para M1 sejam notáveis, isso demonstra que os participantes consideram fácil e estão satisfeitos em utilizar as facilidades providas pelo *dashboard* de desenvolvimento. Caso os valores sejam insuficientes, isso significa que o *dashboard* não cumpriu com o seu papel facilitador.

Para obter o resultado uma análise exploratória realizada em resposta a QP2, o modelo de questionário SUS apresentado na tabela 3 foi aplicado.

O SUS abrange a aplicação de um questionário que alterna intencionalmente entre itens positivos e negativos, coagindo os participantes a refletir após a leitura e evitando vieses nas respostas. No SUS, os itens ímpares refletem preocupações positivas, enquanto os itens pares refletem os negativos.

Tabela 3 – Modelo de questionário do SUS

Item	Afirmação
1	Eu acho que gostaria de usar este sistema frequentemente.
2	Achei o sistema desnecessariamente complexo.
3	Eu pensei que o sistema era fácil de usar.
4	Eu acho que precisaria do apoio de uma pessoa técnica para poder usar esse sistema.
5	Eu descobri que as várias funções deste sistema estavam bem integradas.
6	Eu pensei que havia muita inconsistência neste sistema.
7	Eu imagino que a maioria das pessoas aprenderia a usar esse sistema muito rapidamente.
8	Achei o sistema muito complicado de usar.
9	Eu me senti muito confiante usando o sistema.
10	Eu precisava aprender muitas coisas antes de começar esse sistema.

O modelo SUS gera um número único, representando uma medida composta da usabilidade geral do sistema. A pontuação total $\mathcal{S}$ pode ser dada pela seguinte equação:

$$\mathcal{S} = 2.5 \times \left(\sum_{x=1}^{5} (S_{2x-1} - 1) + \sum_{x=1}^{5} (5 - S_{2x}) \right)$$

em que S_{2x-1} é a pontuação (1–5) atribuída a itens ímpares e S_{2x} é a pontuação (1–5) atribuída a itens pares.

A pontuação $\mathcal{S}$ varia de 0 a 100 e classifica o sistema em estudo como *pior imaginável* (0–20,5), *pobre* (21–38,5), *mediano* (39–52,5), *bom* (53–73,5), *excelente* (74–85,5) e *melhor imaginável* (86–100) (BANGOR; KORTUM; MILLER, 2009). A partir desta pontuação, é possível reconhecer o nível de satisfação do usuário em relação à usabilidade proporcionada pelo *dashboard* de visualização.

5.5 Planejamento

O experimento controlado foi realizado em ambientes acadêmicos e envolveu sete estudantes de Ciência da Computação. Todos os estudantes são desenvolvedores e possuíam experiência de um a dois anos com desenvolvimento de aplicações usando plataformas para Cidades Inteligentes. Quando perguntados sobre sua experiência anterior no uso do SGeoL, dois participantes já usaram o SGeoL em aplicações de cidades inteligentes, enquanto os outros não. O experimento foi dividido em três etapas. Na primeira etapa, os participantes receberam treinamento para aprender como criar entidades e camadas por meio das APIs do SGeoL e do *Dashboard*. Um *script* do experimento foi disponibilizado aos participantes em um repositório do Github (*https://github.com/douglasrolim/exp-dashboards*), contendo todas as instruções necessárias para a execução de tarefas, além de documentação da

API, exemplos de solicitações da API e manuais para os *Dashboard* de desenvolvimento e visualização.

O segundo estágio envolveu a criação de camadas e entidades usando as APIs do SGeoL ou de *Dashboard*. Foi feito um primeiro sorteio para dividir os participantes em dois grupos (#1 e #2) e um segundo sorteio para definir a abordagem a ser usada pela primeira vez por cada grupo. O Grupo #1 foi selecionado para usar primeiro as APIs do SGeoL e, em seguida, o *Dashboard* de desenvolvimento, enquanto o Grupo #2 é o oposto.

A terceira e última etapa consistiu na visualização de camadas e entidades criadas durante as etapas anteriores através do *Dashboard* de visualização. Nesta etapa, os participantes também usaram mapas de calor para as entidades das camadas criadas e dados relacionados de camadas e entidades com outras camadas predefinidas, como as camadas vizinhança e de saúde.

No final do experimento, os participantes responderam aos questionários apresentados na seção 5.4. O questionário que aborda as facilidades de uso e a satisfação do usuário tinha 12 perguntas, enquanto no questionário que aborda a satisfação do usuário em relação à usabilidade havia 10 questões.

5.6 Tarefas

Um conjunto de tarefas[1] foram configuradas para ser executada pelos participantes para permitir a coleta de métricas. As tarefas foram realizadas por todos os participantes na mesma ordem, uma vez que algumas tarefas dependem dos resultados da anterior. As tarefas estabelecidas foram:

T1 – Realize o login do usuário na aplicação. As informações foram disponibilizadas para cada participante, onde ambos faziam uso das mesmas credenciais.

T2 – Crie uma camada. A camada deve ter o caminho, o nome, a descrição e uma propriedade extra das propriedades.

T3 – Crie três entidades. Cada entidade deve ter um nome, descrição e localização. Foram disponibilizados arquivos de dados geográficos dispondo de pontos ou polígonos, para cada uma das três entidades.

T4 – Use o *Dashboard* de visualização para visualizar camadas e entidades criadas nas tarefas anteriores, relacioná-las com camadas pré-estabelecidas, ativar visualização em mapa de calor.

A Figura 43 descreve as etapas das tarefas citadas, realizadas pelos participantes.

[1] A descrição completa das tarefas está disponível no repositório do experimento: <https://github.com/douglasrolim/exp-dashboards/>.

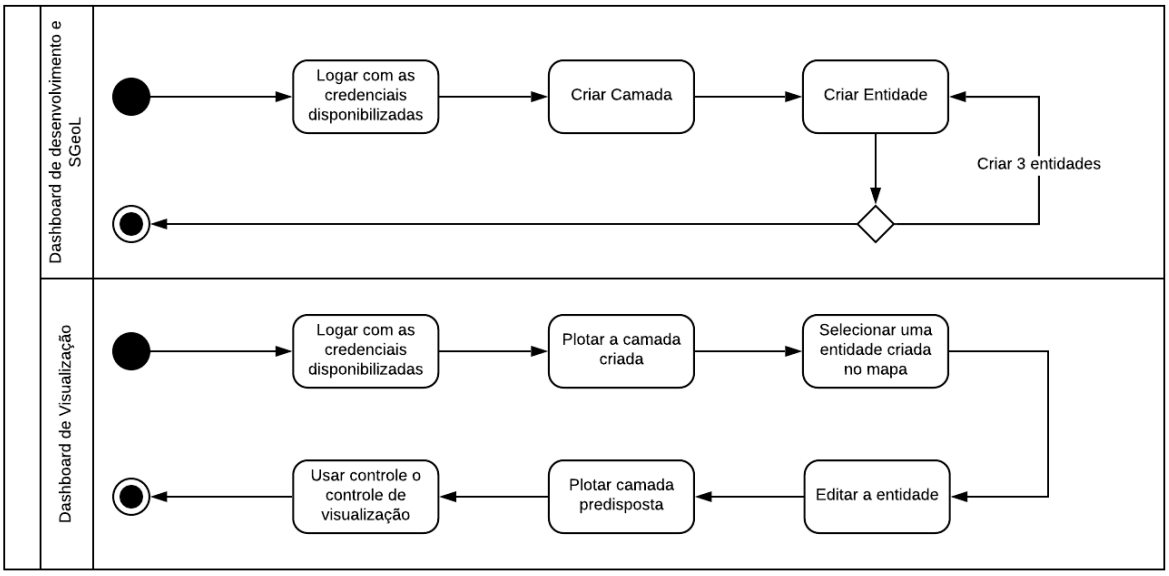

Figura 43 – Etapas das tarefas desenvolvidas pelos participantes.

5.7 Hipóteses

Para o objetivo deste experimento foram definidas as hipóteses nulas (H_0) e hipóteses alternativas (H_1). A hipótese nula definida para este experimento é:

- Hipótese nula (H_0) - A diferença em termos de tempo para realização das atividades com e sem o usando o *Dashboard* de desenvolvimento não é significativa, possuindo, nas duas, tempo de configuração semelhante.

A hipótese alternativa definida para este experimento é:

- Hipótese alternativa (H_1) A diferença em termos de tempo para realização das atividades com e sem o usando o *Dashboard* de desenvolvimento é significativa.

5.8 Análises do resultados

Como mencionado anteriormente, o QP1 foi definida com base em duas métricas: M1 estava relacionado à facilidade e satisfação percebidas do uso, enquanto a métrica M2 estava relacionada ao esforço de desenvolvimento. A métrica M1 foi avaliada através dos questionários respondidos pelos participantes para avaliar o *Dashboard* de desenvolvimento. A métrica M2 fo i avaliada analisando o tempo gasto por cada participante para executar as tarefas estabelecidas.

As Fig. 44 e Fig. 45 apresentam, respectivamente, as respostas fornecidas pelos participantes em relação à facilidade de uso e ao nível de satisfação ao executar as tarefas

de autenticação, criação de camadas e criação de entidades sem o uso do *Dashboard* de desenvolvimento (I) e seu uso (II).

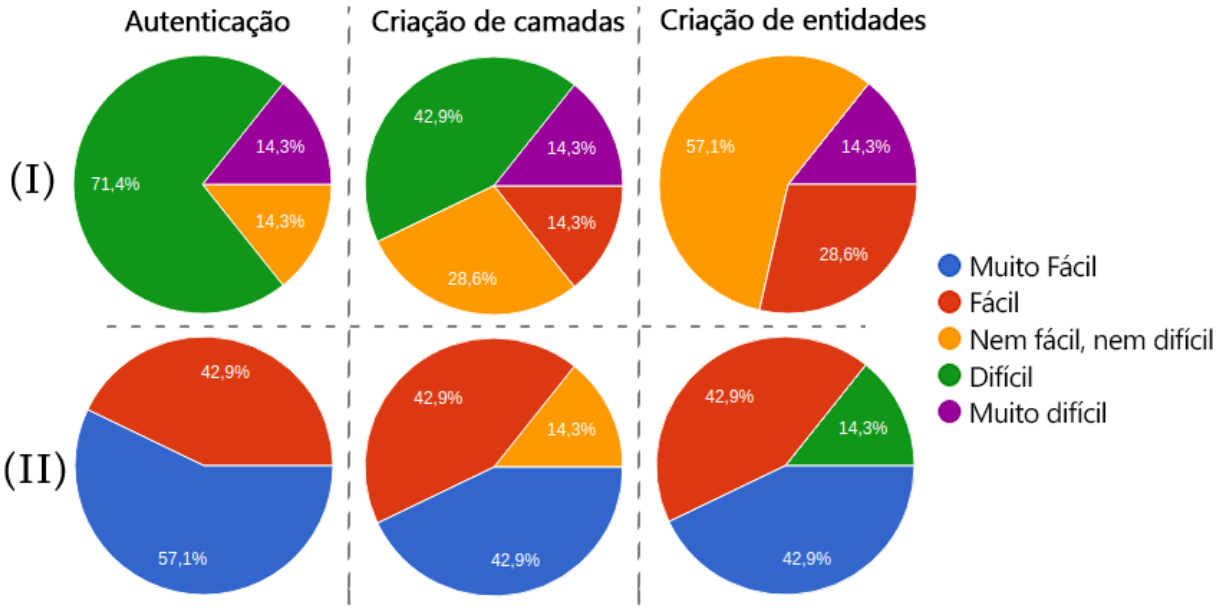

Figura 44 – Opiniões dos participantes sobre facilidade de uso sem o *Dashboard* de desenvolvimento (I) e com ele (II) para autenticação, criação de camada e criação de entidade.

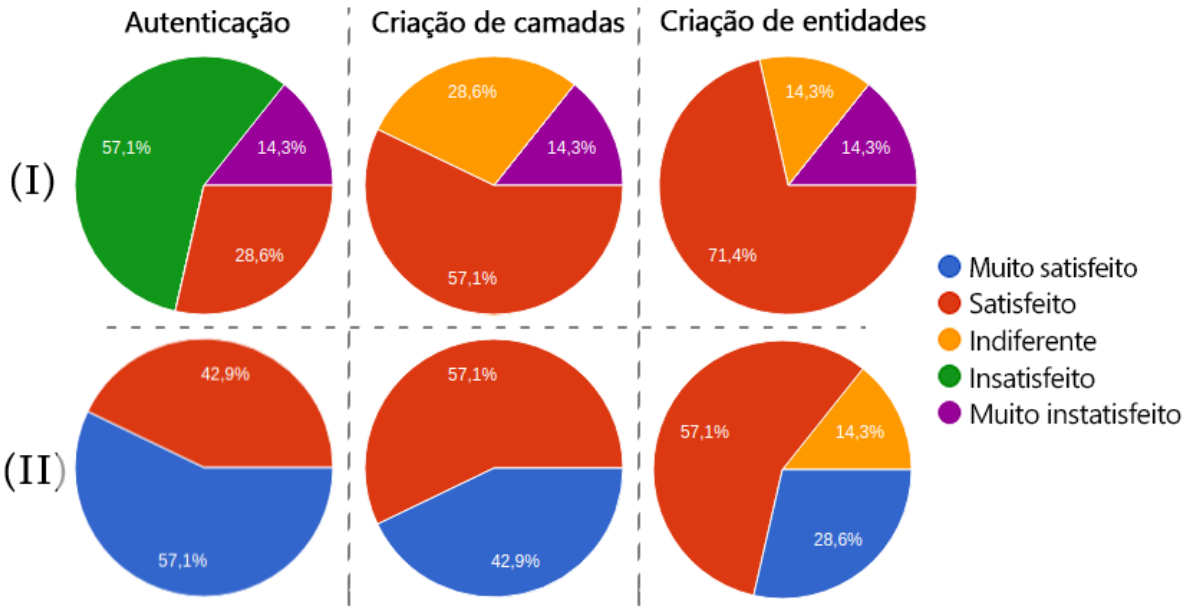

Figura 45 – Opiniões dos participantes sobre o nível de satisfação sem o *Dashboard* de desenvolvimento (I) e com ele (II) para autenticação, criação de camada e criação de entidade.

Os resultados referentes à facilidade de uso (consulte a Fig. 44) mostraram que 71% dos participantes acreditavam que é difícil lidar com a autenticação sem usar o *Dashboard*

de desenvolvimento, enquanto 51% dos participantes acreditavam que é muito fácil com o seu uso. Para criar camadas, 42,9% dos participantes acharam essa tarefa difícil sem usar o *Dashboard* de desenvolvimento, contra 42,9% que acreditavam ser fácil com o uso. Em termos de dificuldade para a criação de entidades, 57,1% dos participantes ficaram indiferentes a isso, enquanto 42,9% consideraram o *Dashboard* de desenvolvimento fácil de executar esta tarefa.

Os resultados referentes ao nível de satisfação (ver Fig. 45) mostraram que 57,1% dos participantes mostraram-se insatisfeitos com o processo de autenticação sem o *Dashboard* de desenvolvimento, a mesma proporção observada na satisfação do usuário com o uso. Para criar camadas, 51,7% dos participantes disseram estar satisfeitos com as duas abordagens. Para a criação de entidades, 71,4% dos participantes disseram estar satisfeitos com a criação de entidades sem o uso do *Dashboard* de desenvolvimento, enquanto 85,7% disseram estar satisfeitos ao criar entidades através do *Dashboard* de desenvolvimento.

A métrica M2 foi obtida medindo o tempo gasto por cada participante para executar as tarefas do experimento. A tabela 4 mostra o tempo que cada participante levou para executar as tarefas nas duas abordagens. Os resultados obtidos mostram que a abordagem usando o *Dashboard* de desenvolvimento reduziu o tempo médio em 798% na realização de todas as tarefas (autenticação, criação de camada e criação de entidade) em comparação com a abordagem sem ela. Um dos fatores mais significativos para essa diferença é a dificuldade enfrentada pelo desenvolvedor para usar a API de autenticação, pois a execução de operações na camada e nas entidades exige o gerenciamento de dados de segurança em solicitações, por exemplo, *token* de informações, identificadores de aplicações e usuários, etc. O *Dashboard* de desenvolvimento ajusta as informações em um modelo de dados apropriado, tornando a criação de camadas e entidades mais rápida e fácil.

Tabela 4 – Tempo gasto pelos participantes para executar as tarefas do experimento

Grupo	Participante	Abordagem	
		Dashboard de desenvolvimento e APIs SGeoL	
#1	P1	00:06:59	00:42:12
	P2	00:07:11	01:02:12
	P3	00:07:26	01:02:41
	P4	00:07:51	01:03:27
#2	P5	00:07:11	01:05:15
	P6	00:09:37	01:04:11
	P7	00:07:51	01:05:19

Durante a análise exploratória do *Dashboard* de visualização, os participantes responderam ao questionário do SUS com dez perguntas fechadas, além de uma pergunta aberta opcional. Os dados obtidos através do questionário SUS foram analisados e o escore total foi calculado para cada participante como métrica M3. A pontuação média $\mathcal{S}$ foi 70, que classifica a usabilidade do *Dashboard* de visualização como boa (consulte a Seção

5.4). Apesar do resultado ser bem satisfatório, ainda é necessário evoluir para melhorar a experiência dos usuários no uso do *Dashboard* de visualização.

5.9 Ameaças à Validade

A validade de um experimento refere-se a confiabilidade do experimento por um todo. Em outras palavras, demonstra qual a credibilidade dos elementos envolvidos no processo de avaliação, que vão desde a base teórica adotada até os resultados, bem como a forma como esses resultados são apresentados (LIMA; NETO; EMER, 2014).

São três fatores que determinam a ameaça a validade do experimento realizado: *(i) validade interna, (ii) validade externa e (iii) validade de construção*. De acordo com Travassos, Gurov e Amaral (2002), a validade interna define se a relação analisada durante o tratamento e o resultado não teve influencia de um outro fator não controlado. As ameaças a validade identificadas são apresentadas a seguir:

1. *Limitação de conhecimento dos participantes*: A limitação de conhecimento e experiência dos participantes necessária para realização das tarefas usando o modelo de dados e as APIs da plataforma SGeoL para comparação pode afetar os resultados do experimento. Para reduzir tal ameaça, foi realizado um treinamento antes das tarefas, reduzindo a disparidade de conhecimento dos participantes.

2. *Expectativa dos participantes ou do experimentador*: a perspectiva dos participantes ou o ponto de vista do experimentador por um resultado positivo ou negativo pode formar resultados tendenciosos. Para evitar essa eventualidade, os participantes só foram instruídos sobre as tarefas e suas etapas durante a execução do experimento.

3. *Desmotivação e cansaço*: Como a execução das tarefas pode levar ter uma duração elevada, os participantes podem sentir cansados ou deixarem de estar motivados. Dessa maneira, cientes da possibilidade que poderiam abandonar a qualquer momento caso desejassem, foi mantido o monitoramento constante durante a execução do experimento e não foram observados sinais de fadiga ou desinteresse.

4. *Aprendizado*: a forma de como o experimento foi planejado poderia possibilitar que os participantes aprendessem a medida em que eram executadas as tarefas em ambas as abordagem, isto poderia afetar a opinião dos participantes. Para tratar desta ameaça e evitar as fontes de viés, de acordo com as abordagens utilizadas, foi definida uma aleatoriedade na distribuição dos participantes e as estações experimentais.

A validade externa define condições que limitam a habilidade de generalizar os resultados de um experimento para a prática industrial (TRAVASSOS; GUROV; AMARAL, 2002). O experimento foi descrito com informações suficientes para permitir sua

replicabilidade em outros ambientes, embora ainda seja necessário experimento exibe as informações para ser reproduzido em outros ambientes para legitimar os resultados.

6 Considerações Finais

Plataformas para cidades inteligentes são, em particular, caracterizadas por integrarem uma diversidade de dados de fontes heterogêneas e pela necessidade de considerar a existência de informações geográficas do espaço urbano e outros aspectos relacionados ao contexto no qual as cidades estão inseridas. Embora algumas plataformas atuais já disponham de mecanismos para o desenvolvimento de aplicações, deem suporte a dados geográficos e permitam a integração dos dados, dispositivos e pessoas, o desenvolvimento de aplicações para cidades inteligentes ainda é uma tarefa complexa e custosa. Além disso, a visualização dos dados envolvidos nas aplicações desenvolvidas e inerentes ao contexto geoespacial é de extrema importância para que gestores e a população entendam a situação da cidade.

Nessa dissertação, apresentamos a arquitetura de *dashboards* de desenvolvimento e visualização baseados na Web para aplicações de cidades inteligentes que visam fornecer interfaces amigáveis para desenvolvedores e usuários. Enquanto o *dashboard* de visualização pode reunir, organizar e exibir dados de diversos domínios no contexto de uma cidade, inclusive correlacionar dados de diferentes domínios, os painéis de desenvolvimento oferecem suporte ao desenvolvimento de aplicações de cidades inteligentes. Tal arquitetura proposta foi instanciada sob a plataforma para cidades inteligentes, SGeoL, fornecendo um conjunto de facilidades para desenvolvedores e usuários, que vem sendo usado em cenários reais na cidade de Natal.

6.1 Contribuições

Entre as contribuições desse trabalho destacam-se:

(i) a proposição, especificação, e implementação do *dashboard* de desenvolvimento, que possibilita o desenvolvedor de aplicações gerenciar a segurança das aplicações, usuários, camadas e entidades, facilitando ainda, a integração e sincronização dos dados de diversos tipos de arquivos e fontes.

(ii) a proposição, especificação, e implementação do *dashboard* de visualização, que permite a visualização das múltiplas camadas de dados georreferenciados, edição de dados geográficos, consulta de dados através de uma interface amigável, geração de relatórios, tabelas ou gráficos e a possibilidade de reuso do *dashboard* de visualização através de uma API que permite embuti-lo em outras aplicações ou em páginas da web.

(iii) a proposição de uma arquitetura conceitual que possibilita a criação flexível de uma infraestrutura para plataformas de *middleware* que considerem dados geográficos, que também podem ser adaptados a possíveis singularidades, como o modelo de dados específico usado.

(iv) a concretização da arquitetura em uma plataforma para cidades inteligentes, o SGeoL, usando tecnologias bem consolidadas em ambientes Web, como JavaScript, Vue.js, GeoJson, Leaflet, Open Street Map. O uso de tecnologias e padrões consolidados permite que eles sejam estendidos com mais facilidade.

(v) a realização de uma avaliação dos dashboards em dois processos: Para o *dashboard* de desenvolvimento foi realizado um experimento controlado que o avaliou, em termos de esforços de desenvolvimento, bem como facilidade e satisfação proporcionada; Para o *dashboard* de visualização, a avaliação focou na usabilidade — que é um critério subjetivo que pode ser considerado segundo diferentes fatores, como produtividade, satisfação e acessibilidade, igualmente importantes. Os resultados mostraram que o *dashboard* de desenvolvimento ajuda o desenvolvedor e é capaz reduz o esforço de desenvolvimento e, em paralelo, o *dashboard* de visualização demostrou ter boa usabilidade e facilitar a compreensão dos dados.

(vi) a viabilização do uso do SGeoL em contextos reais da cidade de Natal, que demandam interfaces de alto nível, como providas pelos dashboards propostos nesse trabalho. Em parceria com o projeto *SmartMetropolis*, o uso dos *dashboards* de desenvolvimento e visualização nas seguintes aplicações:

- Prefeitura Municipal de Natal: o *dashboard* de desenvolvimento foi usado para gerenciar camadas de lotes, edificações, licenciamento e parcelamento da Secretaria Municipal de Meio Ambiente e Urbanismo (SEMURB) Urbanismo e da Secretaria Municipal de Tributação (SEMUT); e o *dashboard* de visualização usado para exibir dados geográficos, bem como permitir que a SEMURB edite e importe dados geográficos a partir de arquivos CAD fornecidos nos processos de licenciamento, e a SEMUT edite dados geográficos a partir de imagens e levantamentos em campo.
- Secretaria de Educação: o *dashboard* de desenvolvimento foi usado para criar a camada de Educação, agregando dados do Sistema Integrado de Gestão da Educação (SigEduc[1]) e da prova Brasil; e o *dashboard* de visualização usado para realizar consultas que envolvem atores em torno dos dados educacionais, possibilitando uma análise quantitativa e qualitativa de forma ampla e contextualizada dos fatores não educacionais que muitas vezes interferem no processo de ensino e aprendizagem.

[1] https://sigeduc.rn.gov.br/sigeduc/

- Ministério Público do Rio Grande do Norte: o *dashboard* de desenvolvimento é usado para criar algumas camadas como, IDEB, IDH, estabelecimentos, mortalidade, morbidade, arborização de vias públicas, esgotamento sanitário, internações, população ocupada, receitas de fontes externas, rendimento domiciliar, salário médio mensal, Taxa de escolarização, Urbanização de vias públicas, etc; o *dashboard* de visualização é embutido em uma de suas aplicações como ferramenta de mapas para exibição de dados geográficos e consultas relacionando os dados das camadas importadas.

Revisitando as questões de pesquisa estabelecidas no Capítulo 1 deste trabalho, foram obtidas as seguintes respostas.

Para a questão de pesquisa (QP1), que questiona quais elementos são necessários para a construção *dashboards* de desenvolvimento e visualização para plataformas de Cidades Inteligentes centradas em dados geográficos, temos que *Dashboards* de desenvolvimento e visualização são interfaces que devem prover funcionalidades específicas, sendo elas:

- Um *dashboard* de desenvolvimento deve considerar dados geográficos, ser capaz proporcionar facilidades para o desenvolvimento de novas aplicações para cidades inteligentes. Dentre estas facilidades, podemos citar: gerenciar camadas, entidades e usuários e o controle permissões das aplicações desenvolvidas e suas e camadas.

- Um *dashboard* de visualização deve prover facilidades para a compreensão de dados através de consulta amigável de dados via *widgets* e do uso de mapas que permita uma visão em múltiplas camadas de dados georreferenciados.

Com objetivo de mostrar como criar *dashboards* de desenvolvimento e visualização, foi proposta uma arquitetura conceitual (consultar 3.2). Tal arquitetura provê componentes que são base para o desenvolvimento concreto para os *dashboards* de desenvolvimento e visualização e mostrou-se flexível na criação de uma infraestrutura para plataformas de *middleware* que consideram dados geográficos.

Para questão de pesquisa (QP2), que questiona se uso do *dashboard* de desenvolvimento facilita o desenvolvimento de aplicações, o experimento controlado avaliou a satisfação percebida no uso do *dashboard* de desenvolvimento. Como resultado, verificou-se que o *dashboard* de desenvolvimento provê facilidade para o desenvolvimento de aplicações.

Por último, para responder a questão de pequisa (QP3), que pergunta se uso *dashboard* de visualização facilita acesso e a compreensão dos dados, foi realizada uma análise exploratória do *dashboard* de visualização com a aplicação de um questionário. A análise dos dados obtidos através deste questionário demostraram que o *dashboard* de visualização tem boa usabilidade e que trabalhos futuros devem realizados para melhoria da experiência dos usuários.

6.2 Limitações

Em seu estado atual de desenvolvimento, o *dashboard* de desenvolvimento e o de visualização apresentam algumas limitações, sendo as principais:

- Para o *dashboard* de desenvolvimento:

(i) a interface não provê suporte para a criação de arquivos de informações de contexto. Tais arquivos são importantes para adicionar aspectos semânticos relativos a *Linked Data* e ontologias.

(ii) foi realizada apenas uma instanciação da arquitetura, sob a plataforma de *middleware* SGeoL. Para assegurar o caráter genérico da proposta, é importante que sejam feitas validações da arquitetura sob diferentes plataformas de *middleware* e que adotem diferentes especificações de modelo de dados.

- Para o *dashboard* de visualização:

(i) a interface ainda é limitada no sentido de não fornecer mecanismos que facilitem ainda mais a visualização das formas (polígonos, pontos ou linhas) das entidades plotadas no mapa, como por exemplo: destacar as formas ao posicionar o cursor sobre as entidades plotadas no mapa, destacar as formas ao posicionar o cursor sobre a legenda da camada, ou ainda falta de controle de visibilidade por entidade (ocultar e exibir entidades).

(ii) O componente de consulta amigável não dá suporte a consulta semântica. Consultas semânticas tem como objetivo aprimorar a precisão dos resultados das pesquisas utilizando ontologias para atingir os objetivos dos usuários e o significado contextual dos termos por ele utilizados. Tendo em vista a quantidade de dados heterogênios existentes no contexto de cidade inteligentes, torna-se fundamental que os *dashboards* sejam capazes capazes de realizar consultas semânticas da mesma forma que são realizadas consultas relacionais (apoiado em palavras chaves).

(iii) A apresentação no mapa através dos mapas de calor é suportada somente para exibição de entidades do tipo pontos (marcadores). Tendo em vista a importância do uso dos mapas de calor discutidas na seção 3.3.3, é importante que essa funcionalidade seja capaz de englobar quaisquer tipos de exibição no mapa, incluindo polígonos e linhas, como já suportado na exibição por pontos.

(iv) Não há uma maneira intuitiva de apresentar os dados temporais. A notoriedade dos dados geográficos podem ser afetados caso não estejam associados diretamente a um período de tempo, em especial dados que se referem a geopolítica de uma cidade, que sofrem constantes mudanças ao passar do tempo. Atualmente, não há uma forma

simples e rápida de alternar entre os dados temporais. Desta maneira, é necessário que os *dashboards* disponham de mecanismos que permitam exploração interativa de dados espaciais temporais.

6.3 Trabalhos em andamento e futuros

Diante das limitações elencadas na seção 6.2, diversos trabalhos futuros poderão ser desenvolvidos a partir deste trabalho, e alguns deles já estão em andamento. Por exemplo, para o *dashboard* de visualização, já se encontra em desenvolvimento as melhorias de interfaces, como função de exibição flutuante das legendas e mecanismos de arraste-e-solte aos botões de ação. Estas funções poderão ser usadas para reposicionar as legendas, evitando que sua exibição se sobreponham aos dados exibidos no mapa.

Outro trabalho futuro a ser desenvolvido é a implementação de novos recursos para API que permite embutir o *dashboard* de visualização como parte de aplicações ou páginas da *Web*, como personalização da interface, ocultação de barras de navegação ou reposicionamento de botões. Estas funcionalidades proporcionarão aos desenvolvedores reusarem os *dashboard* de visualização de forma mais adequada as suas necessidades.

A arquitetura proposta para os *dashboards* não impõe necessariamente o uso de dados geográficos, sendo possível criação de camadas e entidades sem este tipo de dado. Entretanto, trabalhos futuros devem ser realizados para que os *dashboards* forneçam mecanismos para visualização e analise de dados sem informação geográfica.

Além disso, as aplicações operacionais no contexto da Prefeitura e do Ministério Público estão continuamente contribuindo, com suas experiências de uso, para adição de novos recursos e melhorias da usabilidade dos *dashboards*.

Referências

BANGOR, A.; KORTUM, P.; MILLER, J. Determining what individual sus scores mean: Adding an adjective rating scale. *J. Usability Studies*, Usability Professionals' Association, 2009. Citado na página 72.

BASILI, V. R. *Software modeling and measurement: the Goal/Question/Metric paradigm*. [S.l.], 1992. Citado na página 69.

BRATH, R.; PETERS, M. Dashboard design: Why design is important. *DM Direct*, v. 85, p. 1011285–1, 2004. Citado na página 14.

BROOKE, J. *SUS: A quick and dirty usability scale*. 1996. Citado na página 70.

CALDIERA, V. R. B. G.; ROMBACH, H. D. The goal question metric approach. *Encyclopedia of software engineering*, p. 528–532, 1994. Citado na página 69.

DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, JSTOR, p. 319–340, 1989. Citado na página 70.

DOBRAJA, I.; KRAAK, M.-J.; ENGELHARDTA, Y. Facilitating insights with a user adaptable dashboard, illustrated by airport connectivity data. In: *Proceedings of the ICA*. [S.l.: s.n.], 2018. v. 1, p. 30. Citado na página 19.

FEW, S. Information dashboard design. O'Reilly Sebastopol, CA, 2006. Citado na página 19.

FILIPOVA, O. *Learning Vue.js 2*. Packt Publishing, 2016. ISBN 9781786461131. Disponível em: <https://books.google.com.br/books?id=nszcDgAAQBAJ>. Citado na página 37.

GREIF, S.; BENITTE, R.; RAMBEAU, M. *The State of JavaScript 2018: Front-end Frameworks - Overview*. 2019. Disponível em: <https://2018.stateofjs.com/front-end-frameworks/overview/>. Citado na página 38.

INCAU, C. *Vue.js: Construa aplicações incríveis*. Casa do Código, 2017. ISBN 9788555192685. Disponível em: <https://books.google.com.br/books?id=Ft-8DgAAQBAJ>. Citado na página 38.

ISBISTER, K.; SCHAFFER, N. *Game usability: Advancing the player experience*. [S.l.]: CRC press, 2008. Citado na página 46.

JING, C. et al. Geospatial dashboards for monitoring smart city performance. *Sustainability*, Multidisciplinary Digital Publishing Institute, v. 11, n. 20, p. 5648, 2019. Citado 4 vezes nas páginas 12, 20, 21 e 70.

KAA. *Kaa features overview: multi-purpose IoT technology*. 2010. Disponível em: <https://www.kaaproject.org/overview>. Citado na página 61.

KAMAKURA, W. A. et al. Assessing the service-profit chain. *Marketing science*, INFORMS, v. 21, n. 3, p. 294–317, 2002. Citado na página 67.

KIM, G.-H.; TRIMI, S.; CHUNG, J.-H. Big-data applications in the government sector. *Communications of the ACM*, ACM, v. 57, n. 3, p. 78–85, 2014. Citado na página 43.

KLEINFELD, R. et al. glue.things: a mashup platform for wiring the internet of things with the internet of services. In: *WoT '14*. [S.l.: s.n.], 2014. Citado 2 vezes nas páginas 13 e 63.

LIMA, V. C. M.; NETO, A. G. S. S.; EMER, M. C. F. P. Investigação experimental e práticas ágeis: Ameaças à validade de experimentos envolvendo a prática ágil programação em par/experimental investigation and agile practices: Threats to the validity of experiments involving the pair programming agile practice. *Revista Electronica de Sistemas de Informaçao*, Faculdade Cenecista de Campo Largo-FACECLA, v. 13, n. 1, p. 1, 2014. Citado na página 77.

MCARDLE, G.; KITCHIN, R. The dublin dashboard: Design and development of a real-time analytical urban dashboard. 2016. Citado na página 21.

PAPPAS, L.; WHITMAN, L. Riding the technology wave: Effective dashboard data visualization. In: SPRINGER. *Symposium on Human Interface*. [S.l.], 2011. p. 249–258. Citado na página 19.

PAUWELS, K. et al. Dashboards as a service: why, what, how, and what research is needed? *Journal of service research*, Sage Publications Sage CA: Los Angeles, CA, v. 12, n. 2, p. 175–189, 2009. Citado 2 vezes nas páginas 18 e 66.

RASMUSSEN, N. H.; BANSAL, M.; CHEN, C. Y. *Business dashboards: a visual catalog for design and deployment*. [S.l.]: John Wiley & Sons, 2009. Citado 2 vezes nas páginas 18 e 19.

ROLIM, D. et al. Uma plataforma integradora da educação no território inteligente. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2018. v. 29, n. 1, p. 208. Citado na página 15.

SANTANA, E. F. Z. et al. Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. *ACM Computing Surveys (CSUR)*, ACM, v. 50, n. 6, p. 78, 2018. Citado na página 13.

SANTANA, E. F. Z. et al. Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. *ACM Computing Surveys*, v. 50, 09 2016. Citado na página 12.

SCHNEIDER, B. et al. Understanding organization-customer links in service settings. *Academy of Management Journal*, Academy of Management Briarcliff Manor, NY 10510, v. 48, n. 6, p. 1017–1032, 2005. Citado na página 67.

SOUZA, A. et al. A geographic-layered data middleware for smart cities. In: *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*. [S.l.: s.n.], 2018. p. 411–414. Citado 2 vezes nas páginas 15 e 25.

TALBOT, D. *An Easy Way to Program the Internet of Things*. MIT Technology Review, 2014. Disponível em: <https://www.technologyreview.com/s/526006/an-easy-interface-for-the-internet-of-things/#comments>. Citado na página 57.

TEIXEIRA, T. et al. Service oriented middleware for the internet of things: a perspective. In: SPRINGER. *European Conference on a Service-Based Internet*. [S.l.], 2011. p. 220–229. Citado na página 12.

TRAVASSOS, G. H.; GUROV, D.; AMARAL, E. Introdução à engenharia de software experimental. UFRJ, 2002. Citado na página 77.

USURELU, C.-C.; POP, C. My city dashboard: Real-time data processing platform for smart cities. *Journal of Telecommunications and Information Technology*, 2017. Citado na página 21.

WOHLGETHAN, E. *SupportingWeb Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue. js*. Tese (Doutorado) — Hochschule für Angewandte Wissenschaften Hamburg, 2018. Citado na página 37.

WOHLIN, C. et al. *Experimentation in Software Engineering*. Germany: Springer Berlin Heidelberg, 2012. Citado na página 69.