

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Pablo Henrique Penha Silva

**Prototipação Rápida de Aplicações Orientadas
à Contexto em Cidades Inteligentes - O caso
BuzApp**

Uberlândia, Brasil

2017

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Pablo Henrique Penha Silva

**Prototipação Rápida de Aplicações Orientadas à
Contexto em Cidades Inteligentes - O caso BuzApp**

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, Minas Gerais, como
requisito exigido parcial à obtenção do grau
de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Flávio de Oliveira Silva

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2017

Pablo Henrique Penha Silva

Prototipação Rápida de Aplicações Orientadas à Contexto em Cidades Inteligentes - O caso BuzApp

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, Minas Gerais, como
requisito exigido parcial à obtenção do grau
de Bacharel em Sistemas de Informação.

Trabalho aprovado. Uberlândia, Brasil, 28 de julho de 2017:

Prof. Dr. Flávio de Oliveira Silva
Orientador

Prof. Dr. Pedro Frosi Rosa
Convidado

**Prof. Dr. João Henrique de Souza
Pereira**
Convidado

Uberlândia, Brasil
2017

Dedico este trabalho primeiramente aos meus pais Ézio Walter da Silva e Flávia Penha de Freitas Silva que, mesmo quando não compreendiam o que eu escolhi ser como profissional sempre estavam lá para me ajudar e incentivar. Desta mesma forma, agradeço aos meus avós paternos Maria Eunice de Gois Silva e José Augusto da Silva e avós maternos Cleyde Penha de Freitas e Francisco Angelo de Freitas. Também dedico este trabalho aos meus tios e tias e parentes próximos, em especial Ary Neto Penha de Freitas, Roberto Penha de Freitas e Marco Azevedo de Gois, pois foram, para mim, os pioneiros na área de exatas da minha família, assim como incentivaram meus estudos iniciais para a área da computação. Por fim, dedico este trabalho também aos meus amigos, colegas, namorada e em especial ao corpo docente da faculdade de computação da Universidade Federal de Uberlândia e da Morgan State University que testaram os meus protótipos, leram meus rascunhos e incentivaram minhas pesquisas.

—

'Innovation distinguishes between a leader and a follower'. Steve Jobs.

Resumo

Este estudo objetivou em explorar o uso de técnicas de prototipação rápida para a criação de aplicações sensíveis ao contexto. Para isso, pesquisou-se, de acordo com os teóricos e documentos oficiais, sobre meios de prototipação rápida, *middlewares*, *context-aware applications* e ferramentas de desenvolvimento de aplicações móveis. Para realizar as avaliações neste estudo, utilizou-se de comparações entre *middlewares* e o estudo de ferramentas e técnicas de desenvolvimento de prototipação rápida. Desta forma, os objetivos foram alcançados e, como resultado, uma aplicação foi desenvolvida e publicada online para que qualquer pessoa com acesso à internet possa acessá-la e testá-la. Por fim, as técnicas aqui estudadas e usadas, assim como as formas de aplicação, foi uma grande contribuição para o desenvolvimento web de protótipos e do uso de aplicações *context-aware*.

Palavras-chave: Prototipação, Orion, PWA, Context-Aware, FIWARE.

Lista de ilustrações

Figura 1 – Exemplo em alto nível da aplicação BuzApp.	24
Figura 2 – Demonstração de dados sendo enviados do ônibus para o back-end. A nuvem é composta pelo Orion (símbolo de pontos ligados) onde este, por sua vez, usa o MongoDB (símbolo de uma folha) para armazenar os dados.	26
Figura 3 – Tecnologias que compõem a aplicação web: Vue.js, Webpack e PWA. .	26
Figura 4 – Demonstração de uso, prompt de adição à Home e ícone criado na Home do celular.	34

Lista de tabelas

Tabela 1 – Tabela de comparação de Middlewares	20
--	----

Nomenclature

API Application Program Interface

CORBA Common Object Request Broker Architecture

CSS3 Cascade Style Sheets 3

GPS Global Positioning System

HTML5 Hypertext Markup Language 5

HTTP Hypertext Transfer Protocol

HTTPS HTTP Secure

I/O Input/Output

ICMM Integrated Crisis Management Middleware

ID Identifier

IOT Internet of Things

IP Internet Protocol

JSON JavaScript Object Notation

PSCBBB Publish Subscribe Context Broker Building Block

PWA Progressive Web Application

REST Enterprise JavaBeans

REST Representational State Transfer

RMI Remote Method Invocation

SPA Single Page Application

TCP Transmission Control Protocol

URL Uniform Resource Locator

Sumário

1	INTRODUÇÃO	10
1.1	Objetivos	11
1.2	Método	11
1.3	Resultados	11
1.4	Organização do Trabalho	12
2	REVISÃO BIBLIOGRÁFICA	13
2.1	Middleware	13
2.2	Aplicações Context-Aware	14
2.3	Middleware para aplicações Context-Aware	15
2.4	Critérios de comparação entre Middlewares	16
2.5	<i>Middlewares</i> escolhidos para comparação	17
2.5.1	Publish Subscribe Context Broker Building Block	17
2.5.2	Mires	18
2.5.3	Orion	19
2.6	Comparações entre <i>Middlewares</i>	20
2.7	Ferramentas de desenvolvimento de aplicações móveis	20
2.7.1	Frameworks JavaScript	21
2.7.2	Progressive Web App (PWA)	22
3	DESENVOLVIMENTO DO PROTÓTIPO FUNCIONAL BUZAPP	24
3.1	Infraestrutura BuzApp	25
3.2	Front-end Buzapp	25
3.3	Back-end BuzApp	31
3.4	Funcionamento do BuzApp	33
4	CONCLUSÃO	35
	REFERÊNCIAS	36

1 Introdução

Estima-se que existirá mais de 20 bilhões de dispositivos eletrônicos conectados em 2020. Esta é a previsão para os fabricantes de semicondutores para o que chamamos de IoT (*Internet of Things*), conhecida como Internet das coisas, no qual é uma variedade de dispositivos eletrônicos que se comunicam e que fornecem dados importantes para o dia-a-dia, como tomar decisões com base nas preferências e hábitos, realizar pagamento, entre outros (LIMA, 2017).

Existem várias razões pelas quais as empresas optam por desenvolver um produto conectado. Sabe-se a premissa de que o melhor produto é aquele que o cliente quer comprar, mas hoje já existe um segmento de mercado que quer um produto com recursos prontos para enviar dados usando rede sem fio e integrados em vários sistemas, softwares ou aplicativos.

Com a IoT as novas aplicações criadas tendem a usar dados do mundo real. Com isso, surgem middlewares que gerenciam e disponibilizam dados de um contexto.

Aplicações modernas de uso geral possuem alguns requerimentos básicos para serem desenvolvidas, como por exemplo, pela existência de diversos dispositivos como computadores, tablets e smartphones, estas devem possuir uma interface adaptada e funcionar em todos esses meios. Com isso, técnicas de desenvolvimento são aplicadas para que se consiga ter estes e outros requerimentos alcançados. Aplicações com este perfil, geralmente, são aplicações móveis e, ou até, aplicações para internet (web).

O tempo para se desenvolver uma aplicação em cada plataforma e tipo de dispositivo pode ser custoso, visto que existem de duas a três grandes plataformas móveis e web. Sendo assim, técnicas e padrões de desenvolvimento são usados para encurtar esse abismo existente, desenvolvendo uma única aplicação capaz de funcionar em todas as plataformas.

Nesta conjuntura, uma outra característica que pode ser apresentada são as de aplicações *context-aware*. Essas aplicações são super interessantes e poderosas, pois conseguem entender o contexto na qual estão sendo executadas e gerar informações valiosas para um determinado fim. Exemplos significantes para essas aplicações são as responsáveis por entender como o trânsito funciona e poder fazer previsões de melhor roteamento de tráfego durante o dia ou, ainda no contexto trânsito, uma aplicação que consiga capturar informações do transporte público para conseguir prover um melhor serviço para os passageiros que o usam.

1.1 Objetivos

O objetivo deste trabalho é explorar o uso de técnicas de prototipação rápida para a criação de aplicações sensíveis ao contexto. Para tanto, o trabalho explorará o uso da técnica PWA (*Progressive Web Application*) e como estudo de caso será criada uma aplicação móvel baseada em contexto.

Para a criação do aplicativo será realizado um estudo comparativo de *middlewares* com capacidade para a criação de aplicações sensíveis ao contexto. Um desses *middlewares* será responsável por fornecer dados reais em tempo real.

Este estudo de caso envolve uma aplicação para mobilidade urbana, onde os dados consumidos são do posicionamento GPS (*Global Positioning System*) em tempo real dos ônibus urbanos da cidade de Uberlândia.

1.2 Método

Neste trabalho, foi aprofundado o meu conhecimento sobre prototipação rápida de aplicações web através do uso de uma nova técnica no mercado chamada *Progressive Web App*. Essa técnica provê uma alternativa sólida frente ao desenvolvimento nativo ou híbrido até então usado.

Um estudo foi realizado para conhecer mais sobre *middlewares* e aplicações orientadas à contexto. Ambos assuntos foram necessários para entender como aplicações desenvolvida para Internet das Coisas funcionasse e criar o serviço que faria a aplicação funcionar. Um estudo também foi realizado para comparar e escolher o melhor *middleware* disponível no mercado.

Após levantar todos os estudos e ferramentas necessárias para o desenvolvimento do protótipo rápido, foi criada uma aplicação que faz uso de dados reais do sistema público de ônibus da cidade de Uberlândia.

1.3 Resultados

Uma avaliação comparativa de *middlewares* para aplicações *context-aware* foi feita para que o melhor *middleware* possa ser usado para gerenciar e disponibilizar os dados de posicionamento dos ônibus, no contexto da cidade de Uberlândia.

Explorou-se o uso de novas técnicas de prototipação rápida para a criação de aplicações em dispositivos móveis. Uma aplicação web foi criada e utilizou os recursos providos pelos *middlewares* supracitado. Com isso, a aplicação deverá ser demonstrada como ferramenta de uso popular para usuários de ônibus da cidade de Uberlândia.

Uma URL de acesso direto à aplicação está disponível em uma (*Uniform Resource Locator*), visto que a aplicação web está hospedada em servidores gratuitos providos pelo github, um serviço web de repositório de códigos. A URL de acesso à aplicação é <https://pablohpsilva.github.io/busapp-pwa> e a URL para ver o código fonte é <https://github.com/pablohpsilva/busapp-pwa>.

1.4 Organização do Trabalho

O restante desse trabalho está dividido da seguinte forma: O segundo capítulo realiza uma revisão sobre a área de *middlewares* e aplicações *context-aware*. Também é feita uma comparação entre *middlewares* e há uma revisão de ferramentas de desenvolvimento de aplicações móveis.

No terceiro capítulo há o desenvolvimento de todo o estudo até então revisado e estudado para o desenvolvimento proposto por este trabalho. Este capítulo está dividido em infraestrutura, back-end, front-end e como essas partes se encaixam.

Por fim, o quarto capítulo contém a conclusão deste trabalho.

2 Revisão Bibliográfica

Este capítulo apresenta um estudo das partes responsáveis para que este trabalho pudesse ser realizado. O capítulo apresenta uma visão sobre *middlewares*, cujo propósito é discutido em detalhe nas subseções e que é fundamental para entender como aplicações *context-aware* funcionam. Faço também uma comparação entre *middlewares*, sendo a escolha de um dos apresentados, tornar-se-á o responsável por gerenciar o contexto em que a prototipação proposta neste trabalho funcionará. Após a comparação, há um breve estudo de ferramentas de desenvolvimento de aplicação móvel, como frameworks JavaScript e *Progressive Web App*.

2.1 Middlewares

Nesta seção, revisaremos o que são *middlewares*, como funcionam e como são importantes para este trabalho.

Em ambientes como o de IoT, não é viável impor padrões que todos devem seguir. Milhares de sensores e redes de larga escala podem emitir eventos que são gerados por todas as coisas que encontram-se na IoT, englobando dispositivos, tecnologias e aplicações de IoT, deixando a computação ubíqua muito difícil de ser usada (GAVRAS et al., 2007; PARIDEL et al., 2010). Neste contexto, um *middleware* pode oferecer um serviço comum para aplicações e desenvolvimento destas, através de uma integração heterogênea, comunicação entre dispositivos e suportar a interoperabilidade dentre esses dispositivos e serviços. Vários sistemas operacionais (CONTIKI, 2009 (acessado February 3, 2017); TINYOS, 2009 (acessado February 3, 2017)) foram desenvolvidos para suportar esses *middlewares*, assim como linguagens de programação (KATASONOV et al., 2008; NI et al., 2005), porém, estas, em sua maioria, falham ao dar suporte a serviços *context-aware*, assim como na escalabilidade (RAZZAQUE et al., 2016).

Middleware é um software que provê serviços para outras aplicações além daquelas oferecidas pelo próprio sistema operacional. Usando-os, torna-se mais fácil o desenvolvimento e implementação de comunicação e I/O (*Input/Output*) para desenvolvedores, deixando estes com mais tempo e foco para resolver problemas mais específicos das aplicações em que estes trabalham (WIKIPEDIA, 2009 (acessado February 20, 2017)).

O termo é mais usado para softwares que permitem a comunicação e gerenciamento de dados em aplicações distribuídas, ou seja, são serviços que ficam acima da camada de transporte (TCP/IP) - respectivamente, *Transmission Control Protocol* e *IP Internet Protocol* -, porém abaixo do ambiente de aplicação, mas que são facilitadores em oferecer

serviços complementares à uma aplicação (WIKIPEDIA, 2009 (acessado February 20, 2017)).

De acordo com (OBJECTWEB, 2016), um *middleware* é: "*The software layer that lies between the operating system and applications on each side of a distributed computing system in a network*" (WIKIPEDIA, 2009 (acessado February 20, 2017)).

2.2 Aplicações Context-Aware

Nesta seção, revisaremos o que são aplicações *context-aware*, como estes funcionam e como eles são importantes para este trabalho.

No ambiente, na natureza, contexto é qualquer informação que pode ser usada para caracterizar uma entidade. De acordo com (PRIYANTHA; CHAKRABORTY; BALAKRISHNAN, 2000; COUTAZ et al., 2005), pessoas e lugares são considerados entidades e são relevantes para uma interação usuário/aplicação (YU; TSAI; PERRONE, 2016).

Desde o início dos anos 90 o termo *Context Awareness*, primeiramente usado por Schilit e Theimer, existe e tem sido usado. A definição que conhecemos de *context* e *context-aware* aceita pela comunidade acadêmica e de pesquisa hoje é a proposta por (DEY; ABOWD; SALBER, 2001). Desde então, aplicações *context-aware* desktop, transformaram-se em aplicações web, móveis, pervasiva/ubíqua até ser aplicada em IoT durante a última década (BROWN; BOVEY; CHEN, 1997).

A composição de um *middleware* é uma coleção de serviços que se adaptam automaticamente de acordo com a mudança de ambiente através de sensores, dá-se o nome de aplicações *context-aware* (YU; TSAI; PERRONE, 2016).

O *middleware* fica responsável por coletar informações do contexto via sensores, assim como fazer a análise e selecionar ou compor serviços que reagem a mudança do contexto. O serviço geralmente produz um efeito referente a mudança do contexto. Exemplos de reações são: enviar mensagem a um usuário deste serviço, ativar um controle ou mudar seu estado interno, dado uma mudança de contexto (YU; TSAI; PERRONE, 2016).

Em geral, *context-aware application* é uma aplicação que se adapta ao contexto em que esta está sendo usada, sendo este contexto objetos que o compõem e até mesmo circunstâncias e situações que exigem a computação de dados para que o contexto possa ser definido (JOHNSON; MONTAGUE; STEKETEE, 2003). Um exemplo seria o contexto de casas inteligentes (*smart-homes*), onde o contexto é definido através do uso de sensores que enviam informações sobre várias partes da mesma, seja ela o ar, a temperatura, até mesmo a pressão no solo ou câmeras de segurança (HUEBSCHER; MCCANN, 2004).

Outro exemplo seria as cidades inteligentes (*smart cities*), sendo estas, fortemente baseada em sensores que proveêm informações sobre temperatura, humidade, poluição,

condições de trânsito, alocação de pessoas em transportes públicos, calorimetria de regiões mais e menos adaptadas da cidade, entre outros. Esses sensores podem prever e melhorar o entedimento de como é a vida de cada cidadão a qualquer momento, afinal, todos esses dados capturados pelos sensores e aplicados a uma aplicação que os entende, torna-se viável tais previsões e entendimentos. Esses conhecimentos poderiam ser aplicados para melhores estratégias de contenção e prevenção de acidentes e doenças, melhoria no trânsito e melhor disponibilização e disponibilidade de ferramentas públicas, como por exemplo, o transporte público (SOLANAS et al., 2014).

2.3 Middleware para aplicações Context-Aware

Nesta seção, revisaremos o que são *middlewares* para aplicações *context-aware*, como estes funcionam e como eles são importantes para este trabalho.

Os dados gerados com a mudança de contexto do ambiente em que vivemos e que controlamos via sensores podem ser usadas para gerar novas soluções de problemas que temos em nossos dias a dia, graças ao rápido desenvolvimento e inserção da computação ubíqua e IoT (FRIESS, 2013). Essas soluções e aplicações podem ser usadas para medir e compreender comportamentos em nosso ambiente com o mínimo possível de intervenção humana, graças a aplicações context-aware. Com isso em mente, nasce um novo desafio que é resolver novos desafios e melhorar a usabilidade do contexto em que vivemos (LI et al., 2015).

Outras abordagens, consideradas menos efetivas do que o *middleware context-aware* (HU; INDULSKA; ROBINSON, 2008), são:

- Contruir uma aplicação que interage, obtém, processa e usa o contexto para o que for preciso somente para essa aplicação;
- Outras bibliotecas almejam obter e processar o contexto para que se possa reusar e adicionar o mesmo na construção de aplicações que são *context-aware*.

Context-aware middlewares devem prover gerenciamento de contexto, como a modelagem, processamento entre outros serviços que tenham como objetivo libertar os desenvolvedores de toda a responsabilidade de ter que gerenciar o contexto em que estão trabalhando, assim como, permitir que o foco destes em desenvolver a aplicação e aplicar as regras de negócio (HU; INDULSKA; ROBINSON, 2008).

O conceito de *context-aware middleware* surgiu no início dos anos 90, quando se tinha a necessidade de explorar dados locais. Os *middlewares* dessa época são os *The Active Badge System* (WANT et al., 1992) e *Cricket Compass* (PRIYANTHA et al., 2001), seguidos por outros como *Context Toolkit* (DEY; ABOWD; SALBER, 2001), Gaia

(ROMAN; CAMPBELL, 2000), Cobra (CHEN et al., 2004) e SOCAM (GU; PUNG; ZHANG, 2005), sendo esses últimos os primeiros a serem nomeados como plataformas *middlewares* de *context-aware*.

Os primeiros *middlewares* para *context-aware* não possuíam várias características que ajudam no gerenciamento de contexto e poderiam ser usadas para um número restrito de aplicações, geralmente sendo essas aplicações bem simples (LEE; CHANG; LEE, 2011). As características que antes não eram encontradas e que fariam diferença se fossem implementadas variam entre usar computação distribuída, inteligência artificial, melhor tolerância de falhas, entre outras (HU; INDULSKA; ROBINSON, 2008).

2.4 Critérios de comparação entre Middlewares

Nesta seção veremos os critérios de comparação entre *middlewares* e a importância do mesmo para este trabalho.

Para avaliar o melhor *middleware* é preciso definir critérios de comparação para que o melhor seja escolhido. A necessidade para tal vem para suprir a necessidade de escolha de um *middleware* que seja *context-aware*.

Para que um desses *middlewares* fosse escolhido, ele deve satisfazer os seguintes critérios:

1. Possuir documentação e esta ser fácil de ler e compreender;
2. Ser utilizado e suportado por uma(s) empresa(s) ou organização(ões);
3. Ser um *Docker Containers* e ter suporte para este;
4. Não depender de outros sistemas, aplicações e/ou *middlewares* para funcionar;
5. Implantação fácil de ser feita, de preferência *plug n' play*;
6. Ser uma aplicação ou serviço *context-aware*;
7. Suportar *Publish/Subscribe* nativamente, sem configurações extras;
8. Ter uma API (*Application Program Interface*) REST (*Representational State Transfer*) bem documentada;
9. Suportar comunicações HTTP (*Hypertext Transfer Protocol*) e HTTPS (*HTTP Secure*) usando JSON (*JavaScript Object Notation*);
10. Suportar HTTP2.

Dessa forma, os *middlewares* selecionados foram estudados e o que foi encontrado sobre cada um pode ser visto nas subseções a seguir.

2.5 *Middlewares* escolhidos para comparação

Nesta seção estão descritos os *middlewares* escolhidos para a comparação mencionada anteriormente. Após pesquisas na área, três *middlewares* foram escolhidos, sendo estes usados no mercado:

- Publish/Subscribe Context Broker Building Block ([CRISMA, 2017](#))
- Mire ([SOUTO et al., 2006](#))
- Orion Context Broker ([FIWARE, 2017](#))

2.5.1 Publish Subscribe Context Broker Building Block

O PSCBBB (*Publish Subscribe Context Broker Building Block*) é uma combinação entre um *Broker* que aceita e recebe eventos de *subscribers* e um serviço de acesso que provê informações de um contexto específico. Dessa forma, o PSCBBB é usado para realizar a inscrição de eventos e delegá-los a funcionalidades do ICM (Integrated Crisis Management Middleware), *middleware* desenvolvido pela empresa Crisma que é usado em conjunto com o presente *Context Broker*. Este *Broker* possui uma estrutura organizada e desenvolvida pelos padrões usados pela própria empresa que a construiu. Sendo assim, a integração deste com outros *middlewares* não é onerosa e quase instantânea ([CRISMA, 2017](#)).

O PSCBBB trabalha com elementos de contexto que são estruturas de dados que descrevem o estado de uma determinada entidade. O conjunto de dados de entidades semelhantes dá-se o nome de contexto. Essas estruturas são armazenadas e servidas via uma API REST onde os dados podem ser estigmatizados com uma data de expiração, para que o dado seja expirado quando a data seja alcançada, mantendo somente dados novos e em tempo real ([CRISMA, 2017](#)).

As principais funcionalidades do PSCBBB podem ser resumidas em:

- Suporte a uma ampla gama de dados e meta-informações;
- Permitir a gestão de elementos/eventos de contexto de forma uniforme, transparente e de fácil utilização, independentemente do tipo de entidade concreta;
- Fornece uma API fácil de usar e fácil de entender baseado em HTTP REST;
- Mecanismos de suporte para subscrever eventos/elementos de contexto (por exemplo, alteração do estado da entidade (simulação iniciada, resultado do modelo disponível, entre outras);

- Permitir o acesso a elementos de contexto válidos (versão mais recente, a menos que tenha expirado).

2.5.2 Mires

Mires é um *middleware* que provê uma implementação de comunicação *publish/-subscribe* para sensores e aplicações. Este foi proposto para tornar-se uma ferramenta melhor e mais robustas comparadas aos outros que já estão no mercado, apesar de estarem em processo de depreciação, sendo esses: Java RMI (WOLLRATH; RIGGS; WALDO, 1996), EJB (THOMAS, 1998) e CORBA (OTTE; PATRICK; ROY, 1996).

O que este *middleware* propõe é uma forma de entregar continuamente dados coletados e gerados de uma grande quantidade e variedade de sensores de forma íntegra. A sua implementação *publish/subscribe* tem o funcionamento parecido com os outros *middlewares*, sendo a diferença deste, uma forma de enviar as notificações e fazer o buffer de mensagens de forma diferente das outras do mercado, visto que o Mires aproveita o modelo proposto por *Sensor Network Application* (YONEKI, 2003; CUGOLA; JACOBSEN et al., 2002) e estende este modelo. Os passos nas camadas entre os nós são definidas dessa forma:

- Os nós avisam sobre uma alteração no contexto em que estão;
- Após a coleta destas alterações (mensagens), estas são roteadas para um nó especial chamado sink, através de um algoritmo de *multi-hop*;
- Uma aplicação que está conectada ao nó sink pode buscar qualquer informação disponibilizada pelos sensores e monitorá-las.
- Por último, as mensagens são enviadas via broadcast para todos os nós da rede.

O *sink node* (nó sink) é um nó responsável por implementar uma agregação de informações vindas dos sensores. Sendo assim, dado um conjunto de grupos de sensores, sendo estes de tipos e propósitos homogêneos e heterogêneos, o nó conseguirá agrupar e coletar os dados. A combinação dos dados de diferentes fontes são feitas através de funções de supressão, min, max e média e a disponibilização destes é feita por um outro serviço que computa e disponibiliza as agregações (MADDEN et al., 2002). A implementação dessas técnicas que compõem o nó sink foram usadas para que fosse reduzido o número de mensagens transmitidas, assim como a latência e consumo de energia (LINDSEY; RAGHAVENDRA, 2002; KRISHNAMACHARI; ESTRIN; WICKER, 2002; YAO; GEHRKE, 2002), de acordo com (SOUTO et al., 2006).

Por fim, Mires é apenas uma proposta e o seu contexto pode estar distribuído em várias aplicações de grande porte hoje. Contudo, por mais que este *middleware* seja

bom, ainda não houve ninguém que o implementasse por completo e que usasse o nome do mesmo para criar uma aplicação, produto ou serviço. Como dito, as idéias que o Mires provê, podem estar disseminadas em outros middlewares novos no mercado (SOUTO et al., 2006).

2.5.3 Orion

O Orion é um *Context Broker* concebido e desenvolvido no âmbito do projeto FIWARE (FIWARE, 2017; WIKI, 2017). Além de ser um middleware de *publish/subscribe*, este é um *Context Broker*, capaz de gerenciar todo o ciclo de vida de um contexto, incluindo: registros de novos sensores, atualizações de informações, subscribe de aplicações e provê uma interface para realizar consultas (*queries*) no contexto gerenciado (FERNÁNDEZ et al., 2016).

A interface provida pelo Orion é composta por implementações de APIs REST. A escolha de usar APIs REST baseia-se na idéia de que a informação pode chegar de várias fontes diferentes e estas precisam coletadas e filtradas para que o gerenciamento possa ser feito. Atrrelado a estas APIs, o acesso da informação independente da fonte de onde foi gerada é de extrema necessidade, vide o controle e disponibilização dessas informações de forma estruturada pelo Orion. As consultas mencionadas anteriormente estão disponíveis na requisição HTTP feita a API via a própria URL feita para chamar o serviço (FIWARE, 2017; MALLO, 2015; FERNÁNDEZ et al., 2016).

Para que o Orion funcione ele deve transmitir e receber dados de sensores e aplicações. Em outras palavras, ele deve agir como *Context Producer* (armazena novas ou atualiza informações disponíveis) e como *Context Consumer* (exportar as informações para algum lugar) quando necessário (FIWARE, 2017). A coleção e emissão de dados são feitas por interfaces *NGSI9* e *10*, sendo que estas implementam:

- *Queries* para buscar informações do contexto;
- *Subscriptions* para o contexto de informações contendo atualizações (notificações);
- Atualizações não solicitadas (invocadas pelos *context providers*).

A vantagem que o Orion tem em agir como *Context Producer* e *Context Consumer* está na facilidade de interação para aplicações através do uso da própria API, o que difere dos outros middlewares onde existe uma forma de interface para cada uma das fontes do contexto. Sendo assim, a facilidade de acesso de informações, assim como a segurança e escalabilidade que o Orion provê, é um grande diferencial com outros middlewares do mercado (MALLO, 2015; FERNÁNDEZ et al., 2016).

2.6 Comparações entre *Middlewares*

Após o estudo aprofundado de cada um dos *middlewares* presentes na seção anterior, o Orion foi o escolhido, pois possui todos os requisitos apresentados na lista de requisitos de um *middleware* para o uso e desenvolvimento deste trabalho. Sendo assim, para que houvesse maior facilidade em analisar o que cada um dos *middlewares* propostos oferecem, de acordo com a lista de requerimentos.

	Mires	PSCBBB	Orion
1 - Documentação	O	O	O
2 - Empresa/Organização		O	O
3 - Docker			O
4 - Independente de outros sistemas	O		O
5 - <i>Plug n' Play</i>		O	O
6 - Publish/Subscribe	O	O	O
7 - API Rest	O	O	O
8 - JSON	O	O	O
9 - Suporte a HTTP2			O

Tabela 1 – Tabela de comparação de Middlewares

2.7 Ferramentas de desenvolvimento de aplicações móveis

Nesta seção reviso algumas ferramentas de desenvolvimento de aplicações móveis, como frameworks e um tipo de desenvolvimento de aplicações web recente.

O *middleware* escolhido neste trabalho funciona por si só, contudo, de nada adianta ter somente um sistema que capta e gerencia contextos se não houver uma forma de disponibilizar essas informações de forma amigável ao usuário, este sendo, neste trabalho, os usuários de ônibus da cidade de Uberlândia.

De acordo com (CONSTANTINIDES; FOUNTAIN, 2008), Web 2.0 é uma coleção de aplicações *open-source*, interativas e controladas por usuários, onde estas estão expandindo a experiências, conhecimento de mercado e poder dos usuários na participação de negócios e processos sociais. Aplicações que suportam Web 2.0 facilitam a criação de rede informal de usuários e fluxo de conhecimento e idéias, permitindo a geração, disseminação, compartilhamento, edição e refinamento eficiente de conteúdo informal.

Sendo assim, as tecnologias de desenvolvimento móvel escolhidas são as mais atuais e pontas de linha. O objetivo final da escolha dessas são facilitar o desenvolvimento e trazer uma boa experiência ao usuário através de uma aplicação móvel.

Visto que os dias do desenvolvimento nativo de aplicações móveis estão tentando a ser deixada de lado em favor de tecnologias mais novas (ELLIOTT, 2016a, 2016a),

não há como desenvolver um trabalho no presente momento sem deixar de falar sobre frameworks JavaScript e PWA, sendo este último, a tecnologia que poderá substituir o desenvolvimento nativo nos próximos anos (ELLIOTT, 2016a; ELLIOTT, 2016b).

2.7.1 Frameworks JavaScript

Nos últimos anos, a inovação em aplicações web e móveis tem crescido estrondosamente. Essas aplicações, como Facebook (FACEBOOK, 2017), Twitter (TWITTER, 2017) como redes sociais e Medium (MEDIUM, 2017) como sendo a evolução dos blogs, estão muito mais interativas e atrativas ao usuário comparadas com o que era disponível a dez anos atrás. Isso só foi possível graças a chegada da Web 2.0 e constante melhoria da plataforma e tecnologias Web, ou seja, estamos falando sobre HTML5, CSS3, JavaScript, componentes web, Engines robustas de navegadores como a V8 do Google, melhoria da camada de transporte com a chegada do HTTP2, como atores e exemplos principais (MARIANO, 2017).

JavaScript é uma das linguagens de programação mais usadas e conhecidas por desenvolvedores web front-end. Primordialmente esta era usada somente no ambiente web, porém, por ser uma linguagem poderosa e fácil de aprender, pode-se encontrar JavaScript em ambientes de serviços com o NodeJS (<https://nodejs.org/>) e até ligada a hardware com a plataforma Johnny-Five (<http://johnny-five.io/>). Com essa expansão e desenvolvimento de aplicações e serviços complexos, frameworks são criados para facilitar e melhorar a forma de gerenciar aplicações, assim como reduzir a complexidade destas (ENDERS; WANG; LIEBICH, 2009).

Um framework JavaScript é uma estrutura criada para que código seja usado e cria abstrações e generaliza operações complexas que são usadas comumente. Além disso, o framework também provê funções e ferramentas que facilitam o desenvolvimento e geram compatibilidade entre navegadores (MARIANO, 2017).

Desenvolvedores atuais usam vários frameworks e bibliotecas para conseguir desenvolver produtos e atender as demandas de usuários. As aplicações web atuais são baseadas em *Single Page Application* (SPA), onde a aplicação não tem troca de contexto (não atualiza), sendo sempre fixa a uma única janela de aplicação. Framework JavaScripts podem fazer variados tipos de aplicação, porém, o foco destes é a criação de SPAs (MARIANO, 2017).

Com esses frameworks, conseguir desenvolver aplicações escaláveis e com um tempo menor de desenvolvimento é algo totalmente tangível. Desta forma, pode-se usar técnicas mais avançadas e futurísticas de aplicações, como, por exemplo, desenvolver *Progressive Web Applications*.

2.7.2 Progressive Web App (PWA)

PWA é um tipo especial de aplicação web que usa ferramentas e APIs web modernas para entregar uma aplicação que aparente ser uma aplicação nativa em qualquer dispositivo (RUSSELL, 2015; DEVELOPERS, 2016a).

Essas aplicações conseguem funcionar com precária ou nenhuma conexão com a internet, assim como processar informações em background, tem suporte a push notifications, segurança e muitas outras características que seriam encontradas em uma aplicação nativa (DEVELOPERS, 2016a).

A criação e disponibilidade de PWA's tem como um pré-requisito funcionar exclusivamente via um serviço HTTPS, visto que esta faz uso de *Service Workers* que torna disponível fazer intercepções de requests/response, assim como gerenciar todos os dados de cache e acessos da aplicação.

Em geral, *Service Worker* é um script que o browser roda em *background*. Este script tem como finalidade liberar funcionalidades que não exigem interações do usuário nem da página web em si. Exemplos de *Service Workers* são o envio de *push notifications*, sincronização de dados em background e, como mencionado anteriormente, interceptar requisições e fazer cache de dados (GAUNT, 2017).

Aplicações PWA devem ser seguras a ponto de garantir que esses dados e controle de informações não fazem para uma outra aplicação ou fuja para um usuário malicioso (DEVELOPERS, 2016a).

Em linhas gerais, os requerimentos para se criar e ter um PWA são (RUSSELL, 2015; DEVELOPERS, 2016a):

- Confiável - Deve carregar instantaneamente e nunca estar indisponível, mesmo quando a conexão à internet for precária;
- Rápida - Responder de forma rápida e natural a interações com o usuário, fazendo uso de animações;
- Atraente - Deve parecer uma aplicação nativa em qualquer dispositivo, visando a experiência do usuário.

Para cada um dos requerimentos acima, existem técnicas e tecnologias que são os principais atores, ou seja, são aqueles que fazem esses três requerimentos serem alcançados.

O ator principal para que o PWA possa ser confiável, técnicas de *pre-caching* em conjunto com *service workers* são usados. Mais detalhadamente, um *service worker* funciona como um proxy do lado do cliente. Ele é escrito em JavaScript e coloca o desenvolver em controle de cache da aplicação e como recursos desta respondem a uma requisição.

Quando uma aplicação é previamente *cached* (*pre-cached*), a aplicação pode funcionar sem depender de conexão com a internet (DEVELOPERS, 2016b; DEVELOPERS, 2016a).

Para que o PWA possa ser rápido, técnica de minificação de código, pré-renderização e animações JavaScript/CSS. Mais detalhadamente:

- Minificação e geração de *bundle* JavaScript: todo o código JavaScript responsável por fazer a aplicação funcionar é concatenada em um arquivo único, onde este, após concatenação, é comprimido e minificado tornando-o, assim, mais rápido para ser transportado para o navegador cliente;
- *Pre-render*: técnica que renderiza "elementos" que servem como *placeholders* para futuros componentes. São usados para dar a impressão que a página está sendo carregada e que ali tem conteúdo;
- CSS na raiz da aplicação: todo o CSS necessário para renderizar a aplicação é posto na página raiz da aplicação, onde este evita uma chamada a mais no servidor para carregar um arquivo CSS.

Por último, para que o PWA possa ser atraente, deve-se, em grande parte, ao *Web App Manifest*. Mais detalhadamente, a aplicação deve estar em uma versão em tela cheia que imerse e atraia o usuário para ela, assim como usar recursos que uma aplicação nativa usaria. Para isso, o Web App Manifest permite que o desenvolvedor controle como a aplicação será carregada, assim como ícones que aparecerão na tela principal do dispositivo, telas de apresentação e ainda é responsável por dizer ao browser que o PWA pode ser "instalado" no dispositivo. O termo "instalado" aqui não é o mesmo que temos tradicionalmente, de que o código fonte é copiado para a máquina ou dispositivo do usuário, mas sim, instalado aqui significa a criação de um atalho da aplicação, onde este atalho possui o cache e código necessário para o funcionamento desta (DEVELOPERS, 2016b; DEVELOPERS, 2016a).

Atualmente, todos os browsers e dispositivos aceitam e estão suportando PWA. O único que não provê 100% de suporte a essas aplicações é o Safari, navegador da Apple. Para que o suporte chegue a quase 100% e PWA's possam ficar disponíveis em dispositivos desta empresa, *meta-tags* ou técnicas de wrapping para aplicações híbridas podem ser usadas (ELLIOTT, 2016a; ELLIOTT, 2016b).

3 Desenvolvimento do protótipo funcional BuzApp

Como objetivo aqui apresentado, o BuzApp seria desenvolvido usando tecnologias aqui descritas para que fosse usado com o intuito de verificar a aplicabilidade da tecnologia PWA em um caso real de uma aplicação baseada em contexto. Para o desenvolvimento da aplicação, a solução foi desenvolvida utilizando três diferentes camadas de desenvolvimento: infraestrutura, back-end e front-end. Essas três camadas foram necessárias para desenvolver uma aplicação simples e fácil de usar para os usuários de ônibus e que pudesse, também, dar suporte aos mesmos, através de um back-end robusto e rápido ao gerenciar as informações vindas do contexto. A parte de infraestrutura refere-se aos dispositivos usados dentro dos ônibus, sendo estes responsáveis por criar os contextos pelos quais são gerados os dados e informações pertinentes ao nosso domínio de aplicação.

A figura abaixo representa, de maneira resumida, como a aplicação funciona: o ônibus emite dados dos seus sensores para uma nuvem, onde esta, por sua vez, possui um serviço de contexto chamado Orion. Este processa e armazena os dados e os dispõe na API REST da aplicação. Essa API é consumida pela aplicação móvel.

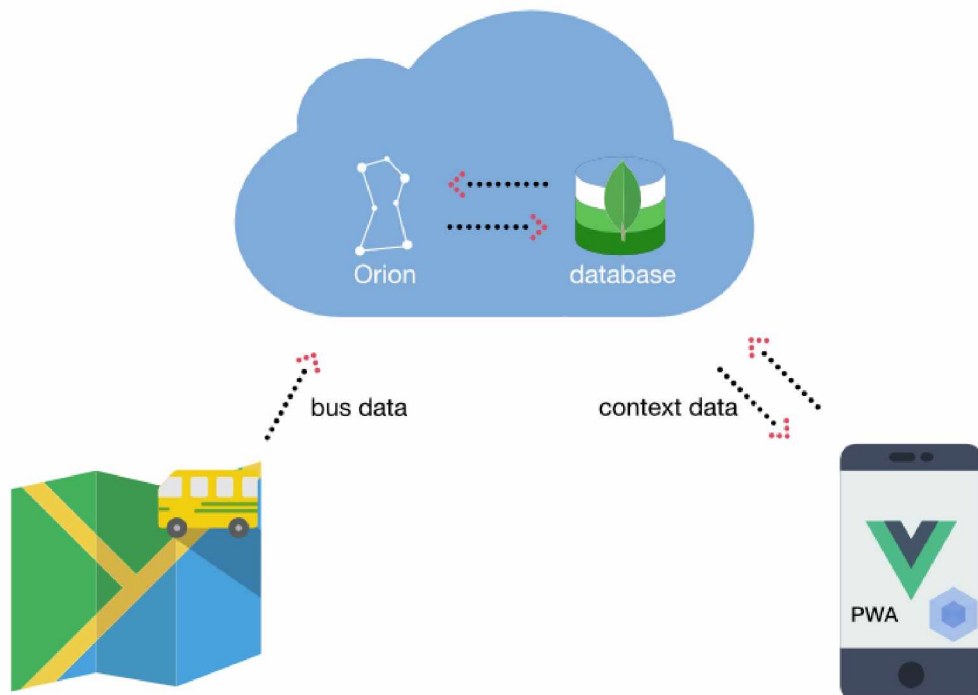


Figura 1 – Exemplo em alto nível da aplicação BuzApp.

As próximas subseções mostrarão cada parte da diagrama demonstrado na figura

acima. Foram criadas três subseções para melhor explicar todo o processo da arquitetura e uma última que descreve como tudo funciona.

3.1 Infraestrutura BuzApp

Após diversas reuniões com as empresas de ônibus da cidade e prefeitura, os ônibus da linha principal (T131) foram disponibilizados para teste. Sendo assim, uma aplicação Android foi criada e instalada em dispositivos que tinham a única finalidade de servir dados referentes a posições geográficas. Esses dispositivos são dispositivos de telemetria, criados e implantados por uma empresa chamada Inlog (INLOG, 2017), empresa responsável por toda captura de dados de transporte público em Uberlândia.

Basicamente, a cada 15 (quinze) segundos a aplicação envia a posição geográfica atual do ônibus para o Orion via uma rede comum 3G, assim como o ID do ônibus e linha em que ele está. Os dispositivos que enviam informações dos ônibus são da empresa Inlog, responsável por todos os dados de telemetria gerados pelos transportes. Os dados são enviados e persistidos no Orion. O Orion, por sua vez, capta esses dados do contexto e trabalha com eles.

O Orion, por sua vez, foi instalado em uma máquina na nuvem do Digital Ocean, visto que eles aceitam Dockers e o serviço é confiável. A porta principal para o serviço é a 1026 e a API REST fica no aguardo de requisições, sem quaisquer restrições de acesso. A Máquina que contém o Orion e o back-end é a única que possui acesso aos serviços da empresa Inlog.

A figura abaixo demonstra como o que foi dito ocorre e qual é o sentido pelo qual os dados trafegam do ônibus para o back-end que, por sua vez, é composto pelo Orion e o MongoDB.

3.2 Front-end Buzapp

Como comentado antes, o BuzApp é um PWA. Ele foi construído usando as guidelines aqui apresentadas, assim como, fez uso da ferramenta Lighthouse da Google, sendo esta especializada em analisar aplicações web para ver se elas são ou não consideradas PWA.

O framework usado foi o Vue.js, um framework especializado na criação de componentes e aplicações web progressivas. Devido ao seu tamanho e sua excelente configuração padrão webpack, além de desenvolver os componentes que montam a aplicação, somente o service worker e o web manifest tiveram que ser criado. Todo o processo de bundle, registro de cache, minificação, otimização, referencias, abstração de códigos semelhantes

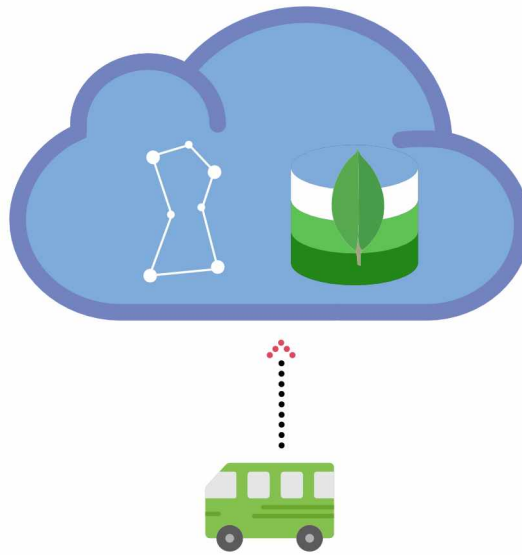


Figura 2 – Demonstração de dados sendo enviados do ônibus para o back-end. A nuvem é composta pelo Orion (símbolo de pontos ligados) onde este, por sua vez, usa o MongoDB (símbolo de uma folha) para armazenar os dados.

e disponibilização de versão foi feito pelo webpack. Para mostrar o mapa onde os ônibus aparecerão, foi usado um componente Vue.js do Google Maps.

Na figura abaixo, pode-se ver as tecnologias usadas para compor a aplicação e como a comunicação ocorre entre ela e o serviço back-end.

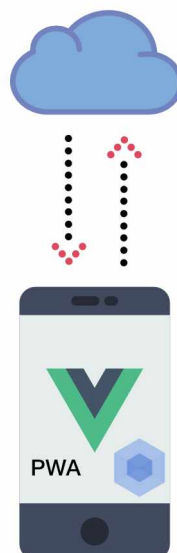


Figura 3 – Tecnologias que compõem a aplicação web: Vue.js, Webpack e PWA.

A base da aplicação foi criada usando uma ferramenta chamada vue-cli. Para iniciar o projeto, executei o seguinte comando, após instalar a ferramenta vue-cli via NPM:

```
$ vue init pablohpsilva/vue-material-boilerplate busapp-pwa
```

Com este comando, foi criado um projeto base baseado em outro projeto que possuo. O projeto usa o framework vue.js, assim como o framework CSS chamado vue-material. As configurações deste projeto, usam o Webpack para gerar o bundle da aplicação, assim como criar um servidor de desenvolvimento local. Os comandos abaixo, respectivamente, iniciam o servidor de desenvolvimento local e o outro cria um bundle da aplicação para ser publicada em qualquer serviço de hospedagem de aplicações.

```
$ cd busapp-pwa && yarn dev
```

```
$ cd busapp-pwa && yarn build
```

A estrutura de diretórios e estruturação do projeto pode ser vista em detalhes abaixo:

- **build**: Diretório onde encontra-se os arquivos de build de desenvolvimento e de produção do webpack;
- **config**: Diretório onde possui configurações básicas do webpack, como variáveis de ambiente;
- **src/common**: Diretório onde arquivos que são compartilhados por toda aplicação;
- **src/common/directives**: Diretivas Vue que podem ser usadas na aplicação;
- **src/common/functions**: Funções usadas na aplicação;
- **src/common/resources**: Resources para consumo de API Rest;
- **src/common/validations**: Validações customizadas que atendam às necessidades da aplicação;
- **src/shared-components**: Onde todos os componentes que são frequentemente usados na aplicação deverão ser criados e colocados aqui;
- **src/spa**: Onde cada uma das páginas da SPA serão criadas;
- **src/vuex**: Todos os arquivos necessários para que o vuex funcione. Contempla actions, mutations, getters e a store;
- **src/vuex/modules**: Módulos do vuex que serão usados na aplicação.
- **router-config.js**. Arquivo que configura um objeto VueRouter para ser usada na aplicação. Todas as rotas encontram-se aqui. Este arquivo é o arquivo raiz de rotas, sendo assim, arquivos externos poderão ser importados e usados aqui;
- **src/App.vue**: Componente raiz da aplicação já com vue-router iniciado, assim como, com transição slide-fade adicionada;

- **src/main.js**. Arquivo de configuração raiz da aplicação. Onde tudo inicia.
- **static**: Diretório onde, após o comando de build ser executado, armazenará a aplicação inteira, pronta para ser exportada;
- **test**: Diretório configurado para executar testes.

Para que uma aplicação possa ser considerada um PWA, ele deve seguir todas os requerimentos citados anteriormente, na subseção que explica sobre este termo. O Service Worker responsável por fazer cache da aplicação foi gerado a partir de uma configuração do webpack, onde esta, após executado o comando de build para a produção, gerou automaticamente este script. A configuração do webpack é esta (dentro do arquivo build/webpack.prod.config.js:

```
//...
new SWPrecacheWebpackPlugin(
  {
    cacheId: 'buzapp-pwa-v3',
    filename: 'buzapp-pwa-worker.js',
    maximumFileSizeToCacheInBytes: 4194304,
    minify: true,
    runtimeCaching: [{
      handler: 'cacheFirst',
      urlPattern: /\.mp3$/,
    }],
  }
),
// ...
```

Além do Service Worker, precisa-se também do Web Manifest. O arquivo JSON que o compõe é copiado para a raiz da aplicação quando esta é gerada para produção. O Web Manifest do BuzApp copia alguns recursos como as imagens para ícones, o nome da aplicação, a URL pelo qual o PWA será iniciado entre outras configurações. O código pode ser visto abaixo:

```
{
  "name": "BuzApp_PWA",
  "short_name": "BAPWA",
  "icons": [
    {
      "src": "/buzapp-pwa/static/img/icon-48.png",
```

```

    "sizes": "48x48",
    "type": "image/png"
  },
  {
    "src": "/buzapp-pwa/static/img/icon-96.png",
    "sizes": "96x96",
    "type": "image/png"
  },
  {
    "src": "/buzapp-pwa/static/img/icon-144.png",
    "sizes": "144x144",
    "type": "image/png"
  },
  {
    "src": "/buzapp-pwa/static/img/icon-196.png",
    "sizes": "196x196",
    "type": "image/png"
  },
  {
    "src": "/buzapp-pwa/static/img/icon-384.png",
    "sizes": "384x384",
    "type": "image/png"
  }
],
"start_url": "/buzapp-pwa/",
"display": "standalone",
"orientation": "portrait",
"background_color": "#ffffff",
"theme_color": "#2196f3"
}

```

Com o Web Manifest e o Service Worker prontos, foi necessário chamá-los no arquivo `index.html`. Com isso, o browser consegue encontrar e carregar os arquivos corretamente, fazendo funcionar a aplicação. As chamadas dentro do `index.html` podem ser vistas abaixo:

```

<html>
  <head>
    <!-- ... -->
    <link rel="manifest" href="./manifest.json">

```

```

    <!-- ... -->
</head>
<body>
    <!-- ... -->
    <script>
        (() => {
            if ( 'serviceWorker' in window.navigator ) {
                window.navigator
                    .serviceWorker
                    .register ( './busapp-pwa-worker.js ' );
            }
        })();
    </script>
</body>
</html>

```

A aplicação onde mostra-se o mapa e o ônibus movendo sobre este, foi feito com muita simplicidade. O middleware de context-aware Orion está online e pronto para prover dados dos ônibus. A aplicação, através de requisições GET, busca os dados da linha T131 dos ônibus corrente naquele momento e repete essa requisição a cada cinco (5) segundos. Todas as posições encontradas são guardadas no Local Storage do browser, pois podem ser usadas para calcular em quanto tempo o ônibus estará em um dado ponto de ônibus, mesmo que a aplicação fique offline.

A aplicação guarda um conjunto de informações via cache da aplicação. Esses dados são: posições dos ônibus, imagens, fatias de imagens do mapa já percorrido e quaisquer outras informações que forem necessárias. Caso o Orion esteja offline, por qualquer razão, e a aplicação não possua dados consistentes em cache, uma lista de rotas pré-definidas é acionada, para que o usuário consiga ver algo funcionando, pois esta é umas das diretrizes de PWA.

O mapa e toda a lógica da aplicação pode ser vista no componente Map.vue, no link <https://github.com/pablohpsilva/busapp-pwa/blob/master/src/spa/map/Map.vue> do github da aplicação.

A aplicação foi hospedada em um repositório Github, tanto o código quanto a aplicação. Github provê uma forma gratuita de servir páginas web e, sendo o BuzApp uma aplicação Web, ela foi hospedada gratuitamente neste serviço. A escolha deste serviço como hóspede da aplicação não foi a toa. Foi escolhido este serviço para, justamente, deixar uma única fonte que dá acesso ao público à aplicação, assim como, deixa público e no alcance de qualquer interessado, ver como ela foi criada.

Por fim, o front-end do BuzApp foi uma junção das seguintes tecnologias e ferramentas:

- vue.js: framework JavaScript baseado em componentes web (YOU, 2016). Para o desenvolvimento da aplicação, este framework foi usado pela facilidade de se criar componentes web, sendo assim, tornando a aplicação mais modular e fácil de desenvolver;
- vue-material: framework para vue.js que facilita o desenvolvimento de aplicações com padrão Material Design (MOURA, 2016). Este framework de interface gráfica foi usado por ser fácil de usar e por ser fiel ao padrão de interface gráfica usado pelo Google;
- vue-google-maps: biblioteca para vue.js do Google Maps (SIM; LECLERC, 2016). Como a aplicação usa um mapa, esta biblioteca foi ótima para o desenvolvimento, visto que provê o Google Maps e tem integração com o framework vue.js;
- ECMAScript 6 (Babel): *transpilers* e *transducers* para JavaScript (MCKENZIE, 2016). Foi necessário para escrever uma aplicação que funcione em todos os navegadores;
- SASS: pré-compilador CSS feito em Ruby (CATLIN, 2016). O framework de interface gráfica vue-material usa este pré-compilador. Usar pré compiladores torna o desenvolvimento mais simples e rápido e, também, garante que o CSS gerado funcionará na maioria dos navegadores;
- webpack: módulo que gera *bundles* (arquivos estáticos) da aplicação onde é usado (KOPPERS, 2016). Além disso, webpack é responsável por otimizações nos arquivos javascript, CSS, imagens, fontes, entre outros arquivos estáticos;
- yarn: gerenciador de pacotes Node.js, substituto do NPM (FACEBOOK, 2016). Um gerenciador de pacotes é essencial para se usar código de ferramentas desenvolvidas por terceiros, facilitando o desenvolvimento.

3.3 Back-end BuzApp

Devido a facilidade de uso, foi usado a versão Docker do Orion. O docker foi publicado no servidor de hospedagem de dockers DigitalOcean, devido a sua fácil utilização e a sua excelente integração com docker containers. Após subir um DockerFile contendo as informações do docker do Orion, o DigitalOcean cuidou para que o container fosse carregado, configurado e disponível em um IP privado.

Após ter o Orion disponível e online, todos os serviços que ele provê também estão disponíveis, sem nenhuma necessidade de configuração extra, o que torna este middleware muito fácil de usar.

BuzApp usa o Orion como back-end e o Orion usa o MongoDB para armazenar as informações vindas do contexto. Como o Orion consegue servir os dados do contexto em que ele está trabalhando, a API está sempre exposta com os dados obtidos da linha e dos ônibus. O contexto sempre estará organizado e disposto, inclusive se o BuzApp fosse um Subscriber do Orion.

Para que a aplicação consiga acessar o Orion, uma função chamada resource foi criada e pode ser vista abaixo, sendo o mesmo código encontrado em no github da aplicação, no arquivo `src/common/resources/orion.js`:

```
const basePath = (process &&
  process.env &&
  process.env.NODE_ENV === 'development') ?
  '/v2/entities' :
  'http://67.205.174.57:1026/v2/entities';

const orionActions = {
  list: { method: 'GET', url: `${basePath}` },
  post: { method: 'POST', url: `${basePath}` },
  get: { method: 'GET', url: `${basePath}/{/id}` },
  update: { method: 'PUT', url: `${basePath}/{/id}` },
};

export const orionResource = resource =>
  resource(`${basePath}`, {}, orionActions);

export default orionResource;
```

A função acima, pode variar a URL chamada dependendo da forma que o ambiente da aplicação esteja disponível. Se estiver em ambiente de produção, a URL que acessa o Orion no DigitalOcean responde o acesso. Um exemplo de uso desta função pode ser visto logo abaixo e, para um exemplo real, o arquivo `src/spa/map/Map.vue` pode ser lido no github da aplicação.

```
import orionResource from '../common/resources/orion';
// ...
orionResource(this.$resource).list()
  .then((response) => {
    // ...
```

```
    })  
    .catch((error) => {  
      // ...  
    });
```

3.4 Funcionamento do BuzApp

Nesta seção explico como o protótipo funcional desenvolvido funciona com todas as partes até este ponto escritas neste trabalho. Após a explicação, uma imagem ilustrativa demonstra como é a aplicação, assim como as funções que a tornam uma aplicação do tipo *Progressive Web App*.

Para usar o BuzApp, o usuário deverá acessar a aplicação via uma URL específica (<https://pablohpsilva.github.io/busapp-pwa>). Para ver o código fonte da aplicação, basta acessar a URL <https://github.com/pablohpsilva/buzapp-pwa>.

Assim que a aplicação carregar, os services workers trabalharão para criar o cache de tudo que for necessário para que a aplicação funcione offline. A primeira visão do usuário para com o sistema é uma tela onde há uma barra de busca por linhas de ônibus e um mapa da cidade de Uberlândia. De acordo com a disponibilidade do serviço da Inlog, todas as linhas de ônibus estão disponíveis para serem buscadas. Uma vez que uma linha tenha sido pesquisada, a aplicação atualizará a posição de todos os ônibus de acordo com a sua própria mudança real geográfica em tempo real.

O BuzApp guarda o caminho percorrido por todos os ônibus por cada linha presente na mesma. Isso foi implementado e pensado para facilitar o uso da aplicação quando não houver conexão com a internet. Quando há falha na conexão, a aplicação pega a última posição conhecida dos ônibus e compara com a última volta dos mesmos na mesma rota, para que seja encontrada o ponto mais próximo daquele atual. Sendo assim, por mais que os dados não sejam reais, há uma forma de simular por onde o ônibus passaria se estivesse em um dado local. Assim que a conexão voltar, os dados reais e atuais dos ônibus voltam a aparecer.

A imagem abaixo mostra como a aplicação funciona. Uma vez tendo acessado a aplicação via o próprio navegador (tela à esquerda), após alguns minutos de uso da aplicação, o Web Manifest entrará em ação e perguntará se o usuário gostaria de ter a aplicação em sua Home (tela do centro). Se o usuário aceitar, um ícone será criado na Home do usuário (tela à direita).

Em casos extremos onde os dados de GPS não possam ser coletados em tempo real dos ônibus ou quando não haja como os dados chegarem até o Orion, a aplicação BuzApp conseguirá repassar os dados obtidos de um outro dia, no mesmo horário em que

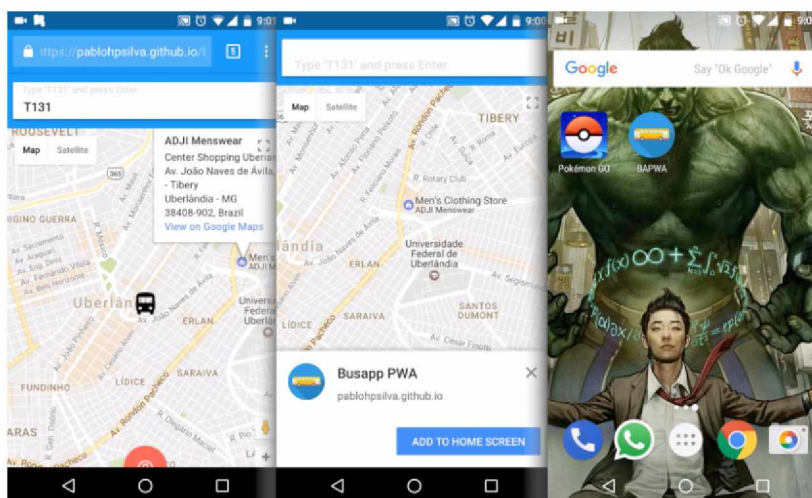


Figura 4 – Demonstração de uso, prompt de adição à Home e ícone criado na Home do celular.

a aplicação é acessada. Esta medida foi tomada para que a aplicação pudesse ser usada mesmo sem dados reais sendo obtidos em tempo real. A rota de ônibus que o BuzApp está usando (T131) possui quantidade de ônibus e horários consistentes ao decorrer dos dias, sendo assim, mesmo quando o BuzApp esteja somente replicando os dados armazenados, a aplicação ainda é útil.

4 Conclusão

O objetivo proposto nesse trabalho foi alcançado, a saber, foi explorado técnicas de prototipação rápida utilizando PWA e Orion *Context-Aware* para a criação de aplicações sensíveis ao contexto. Como resultado da exploração, foi criada uma aplicação que encontra-se disponível para testes em <https://pablohpsilva.github.io/busapp-pwa> e o código fonte em <https://github.com/pablohpsilva/buzapp-pwa>, sendo que esta usa todos as ferramentas, técnicas, serviços de terceiros (por exemplo, Google Maps e Inlog) e contextos apresentados neste trabalho.

Neste trabalho, também foi possível entender e conhecer mais sobre aplicações *context-aware* e usar estas para conseguir entender e gerir um contexto. A aplicação BuzzApp desenvolvida neste trabalho não existiria sem o uso do Orion, visto que gerir o contexto de forma manual, não seria possível de ser feita em tempo para se escrever este trabalho e, também, fugiria muito do objetivo geral deste.

Quanto às recomendações para estudos futuros, é valido indicar a realização de estudos em novas práticas de desenvolvimento de prototipação rápida, assim como a forma que desenvolvemos aplicações modernas, onde o desenvolvimento para várias plataformas possam surgir de um único código base e este, por si, seja o suficiente para que quaisquer dispositivo consiga entendê-lo e executá-lo. Para estudos relacionados a aplicações *context-aware*, percebo que há um grande campo de estudos para aplicações de Internet das Coisas, assim como, no campo de realidade virtual e computação ubíqua. Todos estes campos usam ou podem fazer uso de aplicações voltadas ao contexto, formando assim, novos trabalhos e pesquisas.

Por fim, acredita-se que o presente estudo tenha contribuído de forma teórica e prática para o estudo e desenvolvimento de prototipação rápida aplicações, assim como para aplicações *textitcontext-aware*, estimulado o interesse por pesquisas sobre o tema. Segundo (DEMO, 2004), citado por (BEDANI, 2012), nenhuma pesquisa representa um ponto final ou tem a capacidade de esgotar um tema. Pelo contrário, uma das principais funções do trabalho científico é suscitar o interesse pelo tema pesquisado, provocando novas investigações que confirmem, contestem ou refutem os resultados e modelos anteriormente propostos.

Referências

- BEDANI, M. O impacto dos valores organizacionais na percepção de estímulos e barreiras à criatividade no ambiente de trabalho. *Revista de Administração Mackenzie*, Mackenzie Presbyterian University, v. 13, n. 3, p. 150, 2012. Citado na página 35.
- BROWN, P. J.; BOVEY, J. D.; CHEN, X. Context-aware applications: from the laboratory to the marketplace. *IEEE personal communications*, IEEE, v. 4, n. 5, p. 58–64, 1997. Citado na página 14.
- CATLIN, H. *SASS language*. 2016. Ferramenta de desenvolvimento. Disponível em: <<http://sass-lang.com/>>. Acesso em: 20 apr. 2017. Citado na página 31.
- CHEN, H. et al. Intelligent agents meet the semantic web in smart spaces. *IEEE Internet Computing*, IEEE, v. 8, n. 6, p. 69–79, 2004. Citado na página 16.
- CONSTANTINIDES, E.; FOUNTAIN, S. J. Web 2.0: Conceptual foundations and marketing issues. *Journal of direct, data and digital marketing practice*, Springer, v. 9, n. 3, p. 231–244, 2008. Citado na página 20.
- CONTIKI. *Contiki*. [S.l.], 2009 (acessado February 3, 2017). Disponível em: <<http://www.contiki-os.org/>>. Citado na página 13.
- COUTAZ, J. et al. Context is key. *Communications of the ACM*, ACM, v. 48, n. 3, p. 49–53, 2005. Citado na página 14.
- CRISMA. *Publis Subscribe Context Broker BB*. 2017. Artigo online. Disponível em: <<https://crisma-cat.ait.ac.at/bb/Publish-Subscribe-Context-Broker-BB>>. Acesso em: 25 mar. 2017. Citado na página 17.
- CUGOLA, G.; JACOBSEN, H. et al. Using publish/subscribe middleware for mobile systems. *ACM SIGMOBILE Mobile Computing and Communications Review*, ACM, v. 6, n. 4, p. 25–33, 2002. Citado na página 18.
- DEMO, G. *Percepção de políticas de gestão de pessoas: o papel dos valores pessoais e da percepção de justiça organizacional*. Tese (Doutorado) — Tese de Doutorado. Programa de Pós-graduação em Psicologia. Universidade de Brasília, Brasília. D'Iribarne, P.(1983). La Logique d'Honneur. Gestion des entreprises et traditions nationales. France: Editions du Seuil, 2004. Citado na página 35.
- DEVELOPERS, G. *Progressive Web App*. 2016. Artigo online. Disponível em: <<https://developers.google.com/web/progressive-web-apps>>. Acesso em: 24 apr. 2017. Citado 2 vezes nas páginas 22 e 23.
- DEVELOPERS, G. *Your First Progressive Web App*. 2016. Artigo online. Disponível em: <<https://developers.google.com/web/fundamentals/getting-started/your-first-progressive-web-app/?hl=en>>. Acesso em: 24 apr. 2017. Citado na página 23.
- DEY, A. K.; ABOWD, G. D.; SALBER, D. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-computer*

interaction, L. Erlbaum Associates Inc., v. 16, n. 2, p. 97–166, 2001. Citado 2 vezes nas páginas 14 e 15.

ELLIOTT, E. *Native Apps are Doomed*. 2016. Artigo online. Disponível em: <<https://medium.com/javascript-scene/native-apps-are-doomed-ac397148a2c0>>. Acesso em: 20 apr. 2017. Citado 3 vezes nas páginas 20, 21 e 23.

ELLIOTT, E. *Why Native Apps Really are Doomed: Native Apps are Doomed pt 2*. 2016. Artigo online. Disponível em: <<https://medium.com/javascript-scene/why-native-apps-really-are-doomed-native-apps-are-doomed-pt-2-e035b43170e9>>. Acesso em: 20 apr. 2017. Citado 2 vezes nas páginas 21 e 23.

ENDERS, D.; WANG, C.; LIEBICH, J. X. Organocatalytic asymmetric aza-michael additions. *Chemistry—A European Journal*, Wiley Online Library, v. 15, n. 42, p. 11058–11076, 2009. Citado na página 21.

FACEBOOK. *Yarn*. 2016. Ferramenta de desenvolvimento. Disponível em: <<https://yarnpkg.com/en/>>. Acesso em: 20 apr. 2017. Citado na página 31.

FACEBOOK. *Facebook*. 2017. Página online. Disponível em: <<https://facebook.com>>. Acesso em: 23 jun. 2017. Citado na página 21.

FERNÁNDEZ, P. et al. Smartport: a platform for sensor data monitoring in a seaport based on fiware. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 16, n. 3, p. 417, 2016. Citado na página 19.

FIWARE. *Fiware*. 2017. Página online. Disponível em: <<https://fiware.org>>. Acesso em: 23 jun. 2017. Citado na página 19.

FIWARE. *FIWARE Orion Documentation*. 2017. Artigo online. Disponível em: <<https://fiware-orion.readthedocs.io/en/master/>>. Acesso em: 20 mar. 2017. Citado 2 vezes nas páginas 17 e 19.

FRIESS, P. *Internet of things: converging technologies for smart environments and integrated ecosystems*. [S.l.]: River Publishers, 2013. Citado na página 15.

GAUNT, M. *Service Workers: an Introduction*. 2017. Artigo online. Disponível em: <<https://developers.google.com/web/fundamentals/getting-started/primers/service-workers>>. Acesso em: 20 apr. 2017. Citado na página 22.

GAVRAS, A. et al. Future internet research and experimentation: The fire initiative. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 37, n. 3, p. 89–92, jul. 2007. ISSN 0146-4833. Disponível em: <<http://doi.acm.org/10.1145/1273445.1273460>>. Citado na página 13.

GU, T.; PUNG, H. K.; ZHANG, D. Q. A service-oriented middleware for building context-aware services. *Journal of Network and computer applications*, Elsevier, v. 28, n. 1, p. 1–18, 2005. Citado na página 16.

HU, P.; INDULSKA, J.; ROBINSON, R. An autonomic context management system for pervasive computing. In: IEEE. *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*. [S.l.], 2008. p. 213–223. Citado 2 vezes nas páginas 15 e 16.

- HUEBSCHER, M. C.; MCCANN, J. A. Adaptive middleware for context-aware applications in smart-homes. In: ACM. *Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*. [S.l.], 2004. p. 111–116. Citado na página 14.
- INLOG. *Inlog*. 2017. Página online. Disponível em: <<http://www.inlog.com.br/>>. Acesso em: 23 jun. 2017. Citado na página 25.
- JOHNSON, C.; MONTAGUE, P.; STEKETEE, C. (Ed.). *ACSW Frontiers '03: Proceedings of the Australasian Information Security Workshop Conference on ACSW Frontiers 2003 - Volume 21*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2003. ISBN 1-920682-00-7. ISSN 1445-1336. Citado na página 14.
- KATASONOV, A. et al. Smart semantic middleware for the internet of things. *ICINCO-ICSO*, v. 8, p. 169–178, 2008. Citado na página 13.
- KOPPERS, T. *Webpack*. 2016. Ferramenta de desenvolvimento. Disponível em: <<https://webpack.github.io/>>. Acesso em: 20 apr. 2017. Citado na página 31.
- KRISHNAMACHARI, L.; ESTRIN, D.; WICKER, S. The impact of data aggregation in wireless sensor networks. In: IEEE. *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*. [S.l.], 2002. p. 575–578. Citado na página 18.
- LEE, S.; CHANG, J.; LEE, S.-g. Survey and trend analysis of context-aware systems. *Information-An International Interdisciplinary Journal*, v. 14, n. 2, p. 527–548, 2011. Citado na página 16.
- LI, X. et al. Context aware middleware architectures: survey and challenges. *Sensors, Multidisciplinary Digital Publishing Institute*, v. 15, n. 8, p. 20570–20607, 2015. Citado na página 15.
- LIMA, A. *Internet das Coisas é destaque na Campus Party Brasil*. 2017. Disponível em: <<http://economia.estadao.com.br/noticias/releases-ae,internet-das-coisas-e-destaque-na-campus-party-brasil-cpbr10,70001648184>>. Acesso em: 31 jan. 2017. Citado na página 10.
- LINDSEY, S.; RAGHAVENDRA, C. S. Pegasus: Power-efficient gathering in sensor information systems. In: IEEE. *Aerospace conference proceedings, 2002. IEEE*. [S.l.], 2002. v. 3, p. 3–3. Citado na página 18.
- MADDEN, S. et al. Tag: A tiny aggregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review*, ACM, v. 36, n. SI, p. 131–146, 2002. Citado na página 18.
- MALLO, A. G. *Development of an IOT front-end with the Fiware platform for Smart City solutions*. Dissertação (B.S. thesis), 2015. Citado na página 19.
- MARIANO, C. L. Benchmarking javascript frameworks. Dublin Institute of Technology, 2017. Citado na página 21.
- MCKENZIE, S. *Babel JS*. 2016. Ferramenta de desenvolvimento. Disponível em: <<https://babeljs.io/>>. Acesso em: 20 apr. 2017. Citado na página 31.

MEDIUM. *Medium*. 2017. Página online. Disponível em: <<https://medium.com>>. Acesso em: 23 jun. 2017. Citado na página 21.

MOURA, M. *Vue-Material: Material Design framework for vue.js*. 2016. Ferramenta de desenvolvimento. Disponível em: <<https://vuematerial.github.io/>>. Acesso em: 20 apr. 2017. Citado na página 31.

NI, Y. et al. Programming ad-hoc networks of mobile and resource-constrained devices. *SIGPLAN Not.*, ACM, New York, NY, USA, v. 40, n. 6, p. 249–260, jun. 2005. ISSN 0362-1340. Disponível em: <<http://doi.acm.org/10.1145/1064978.1065040>>. Citado na página 13.

OBJECTWEB. *What is Middleware*. 2016. Artigo online. Disponível em: <<http://middleware.objectweb.org/index.html>>. Acesso em: 20 apr. 2017. Citado na página 14.

OTTE, R.; PATRICK, P.; ROY, M. *Understanding CORBA: Common Object Request Broker Architecture*. [S.l.]: Prentice Hall PTR, 1996. Citado na página 18.

PARIDEL, K. et al. Middleware for the internet of things, design goals and challenges. *Electronic Communications of the EASST*, v. 28, 2010. Citado na página 13.

PRIYANTHA, N. B.; CHAKRABORTY, A.; BALAKRISHNAN, H. The cricket location-support system. In: ACM. *Proceedings of the 6th annual international conference on Mobile computing and networking*. [S.l.], 2000. p. 32–43. Citado na página 14.

PRIYANTHA, N. B. et al. The cricket compass for context-aware mobile applications. In: ACM. *Proceedings of the 7th annual international conference on Mobile computing and networking*. [S.l.], 2001. p. 1–14. Citado na página 15.

RAZZAQUE, M. A. et al. Middleware for internet of things: A survey. *IEEE Internet of Things Journal*, v. 3, n. 1, p. 70–95, Feb 2016. ISSN 2327-4662. Citado na página 13.

ROMAN, M.; CAMPBELL, R. H. Gaia: enabling active spaces. In: ACM. *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*. [S.l.], 2000. p. 229–234. Citado na página 16.

RUSSELL, A. *Progressive Web Apps: Escaping Tabs Without Losing Our Soul*. 2015. Artigo online. Disponível em: <<https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/>>. Acesso em: 24 apr. 2017. Citado na página 22.

SIM, D.; LECLERC, G. *vue2-google-maps NPM package*. 2016. Ferramenta de desenvolvimento. Disponível em: <<https://www.npmjs.com/package/vue2-google-maps>>. Acesso em: 20 apr. 2017. Citado na página 31.

SOLANAS, A. et al. Smart health: a context-aware health paradigm within smart cities. *IEEE Communications Magazine*, IEEE, v. 52, n. 8, p. 74–81, 2014. Citado na página 15.

SOUTO, E. et al. Mires: a publish/subscribe middleware for sensor networks. *Personal and Ubiquitous Computing*, Springer, v. 10, n. 1, p. 37–44, 2006. Citado 3 vezes nas páginas 17, 18 e 19.

- THOMAS, A. Selecting enterprise javabeans technology. *WebLogic, Inc., Boston, MA*, 1998. Citado na página 18.
- TINYOS. *TinyOS*. [S.l.], 2009 (acessado February 3, 2017). Disponível em: <http://www.tinyos.net/>. Citado na página 13.
- TWITTER. *Twitter*. 2017. Página online. Disponível em: <https://twitter.com>. Acesso em: 23 jun. 2017. Citado na página 21.
- WANT, R. et al. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, ACM, v. 10, n. 1, p. 91–102, 1992. Citado na página 15.
- WIKI, F. *FIWARE*. 2017. Artigo online. Disponível em: <https://en.wikipedia.org/wiki/FIWARE>. Acesso em: 25 fev. 2017. Citado na página 19.
- WIKIPEDIA. *Middleware*. [S.l.], 2009 (acessado February 20, 2017). Disponível em: <https://en.wikipedia.org/wiki/Middleware>. Citado 2 vezes nas páginas 13 e 14.
- WOLLRATH, A.; RIGGS, R.; WALDO, J. A distributed object model for the java^ t^ m system. *Computing Systems*, University of California Press, v. 9, p. 265–290, 1996. Citado na página 18.
- YAO, Y.; GEHRKE, J. The cougar approach to in-network query processing in sensor networks. *ACM Sigmod record*, ACM, v. 31, n. 3, p. 9–18, 2002. Citado na página 18.
- YONEKI, E. Mobile applications with a middleware system in publish-subscribe paradigm. In: *the 3rd Workshop on applications and services in wireless networks, Bern, Switzerland, July*. [S.l.: s.n.], 2003. Citado na página 18.
- YOU, E. *Vue.js*. 2016. Ferramenta de desenvolvimento. Disponível em: <https://vuejs.org/>. Acesso em: 20 apr. 2017. Citado na página 31.
- YU, L.; TSAI, W. T.; PERRONE, G. Testing context-aware applications based on bigraphical modeling. *IEEE Transactions on Reliability*, v. 65, n. 3, p. 1584–1611, Sept 2016. ISSN 0018-9529. Citado na página 14.