UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

MARCOS KINTSCHNER

Parada de Ônibus Inteligente: Uma Aplicação para Cidades Inteligentes Baseado no Conceito de Web das Coisas

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

Orientador: Prof. Dr. Alexandre Carissimi

Porto Alegre
2017

AGRADECIMENTOS

Eu gostaria de expressar meus mais sinceros agradecimentos ao meu professor orientador, Alexandre Carissimi, pelos seus ensinamentos, conselhos, incentivos, por sua diligência e por ter se esforçado nobremente em me ajudar a elaborar e concluir este trabalho. Eu não tenho palavras suficientes para agradecer.

Agradeço ao meu amigo Guilherme Longoni pelo apoio e compreensão, sem as quais teria sido muito difícil concluir este projeto.

Agradeço a todos os professores e as excelentes pessoas com quem eu tive a oportunidade de conviver e aprender em todos estes anos na universidade.

Agradeço aos meus amigos e familiares que sempre estiveram ao meu lado e me apoiaram em todos os momentos, em especial à minha mãe, Maria Felomena Kintschner, por ter me criado sozinha, por ter me amado e por ter me dado todas as oportunidades para chegar até aqui.

RESUMO

A Internet das Coisas pretende interconectar uma infinidade de dispositivos heterogêneos ao nosso redor, levantando problemas relacionados a escalabilidade e compatibilidade. A Web das coisas é apontada como possível solução, visando facilitar a conectividade e o desenvolvimento de aplicações por meio do reaproveitamento das tecnologias Web. Quando preocupações, demandas e tendências globais são somadas a essas novas tecnologias, é criada a ideia de Cidade Inteligente, em que questões como sustentabilidade, qualidade de vida e mobilidade urbana são otimizadas. Baseado neste conceito será apresentado uma proposta de Parada de Ônibus para Cidades Inteligentes, implementada sob a abordagem de Web das Coisas, cujo objetivo é fornecer diversas informações em tempo real a respeito os ônibus que passam pelo local.

Palavras-chave: Internet das Coisas. Web das Coisas. Cidades Inteligentes. Mobilidade Urbana.

Smart Bus Stop: An Application for Smart Cities Based on the Concept of Web of Things

ABSTRACT

The Internet of Things aims to interconnect a multitude of heterogeneous devices around us, raising issues related to scalability and compatibility. The Web of Things is suggested as a possible solution aiming to facilitate connectivity and application development through the reuse of Web technologies. When concerns, demands and global trends are added to these new technologies, the idea of Smart City is created, in which issues such as sustainability, quality of life and urban mobility are optimized. Based on this concept, it will be presented a proposal for Bus Stop for Smart Cities, implemented under the approach of Web of Things, aimed at providing real-time informations about the buses that pass by.

Keywords: Internet of Things, Web of Things, Smart Cities, Urban Mobility.

LISTA DE ABREVIATURAS E SIGLAS

AJAX	Asynchronous Javascript and XML
CSS	Cascading Style Sheets
DOM	Document Object Model
GTFS	General Transit Feed Specification
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDC	International Data Corporation
IERC	European Research Cluster on the Internet of Things
IoT	Internet of Things
IPSO	IP for Smart Objects
ITU-T	International Telecommunication Union - Telecommunication Standardization Sector
JSON	Javascript Object Notation
NFC	Near Field Communications
ORM	Object-Relational Mapping
RDF	Resource Description Framework
REST	Representation State Transfer
RFID	Radio Frequency IDentification
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SVG	Scalable Vector Graphics
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WoT	Web of Things

WSAN Wireless Sensor and Actuator Networks

WSDL Web Services Description Language

XML eXtensible Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A Internet surgiu em laboratórios de pesquisa de universidades americanas no início da década de 1970 e, desde então, foi ampliada por diversas instituições espalhadas pelo mundo com o desenvolvimento de tecnologias, aplicações, protocolos e serviços. Hoje, extrapolando seu propósito inicial, a Internet atinge todos os continentes, interconectando computadores e pessoas por toda a parte. Com a evolução de sistemas embarcados, sensores e tecnologias de comunicação sem fio nos últimos anos, a Internet encontra-se agora a um passo de uma importante transformação.

A chamada Internet das Coisas surge como uma mudança de paradigma onde a conexão à Internet torna-se possível não apenas a qualquer momento, ou em qualquer lugar, mas também por meio de qualquer "coisa". Uma imensidão de coisas interconectadas pelo mundo deve movimentar um mercado de US$1,29 trilhão em 2020 de acordo com a projeção da IDC (*International Data Corporation*) (IDC, 2017). Um número cada vez maior de dispositivos deve ser inserido na vida cotidiana das pessoas e na indústria, alterando a forma como percebemos e interagimos com objetos ao nosso redor.

Diversas soluções para a viabilização da Internet das Coisas se encontram disponíveis atualmente. No entanto, muitas dessas soluções se apresentam de forma heterogênea. Como consequência, muitos dispositivos tendem a ser incompatíveis entre si, tornando complicado o acesso e a identificação uniforme na Internet das Coisas. À luz desse problema, a Web das Coisas emerge implantando tecnologias Web extensamente utilizadas juntamente com a aplicação do estilo REST às coisas (GUINARD et al., 2011). Identificadas unicamente por URIs (*Uniform Resource Identifier*), essas podem ser acessadas por um navegador, ou mesmo interagir com outras coisas integradas à Web, utilizando para isso uma interface uniforme padrão.

As mudanças trazidas pela Internet e suas novas tecnologias, juntamente com novas tendências globais, tem feito cidades ao redor mundo debater questões relacionadas à sustentabilidade, utilização de recursos, poluição e qualidade de vida da população em geral. Esses assuntos, quando associados às novas tecnologias de comunicação e informação tem dado origem ao termo *Cidades Inteligentes*, amplamente discutidos atualmente. Gestão de recursos, mobilidade, saúde, segurança e áreas públicas inteligentes são alguns dos tópicos de maior interesse nessas cidades.

1.1 Objetivos do trabalho

Neste trabalho, será realizado um estudo sobre o conceito de Web das Coisas e será proposto e implmentado um sistema para uma *Parada de Ônibus Inteligente*. O sistema será constituído de dispositivos acoplados a monitores pensados para ficarem em áreas públicas (paradas de ônibus) de uma Cidade Inteligente. Esses dispositivos fornecerão informações em tempo real sobre os ônibus que passam pelo local com o objetivo de informar as pessoas sobre os horários de chegada. As informações mostradas podem incluir o tempo de chegada dos ônibus (à partir de dados de GPS enviados por esses), além de suas posições atualizadas em tempo real em um mapa. Por fim, uma aplicação cliente Android será desenvolvida com o objetivo de demonstrar a interação entre o dispositivos e a parada.

1.2 Organização do trabalho

Este texto é composto de 6 capítulos, incluindo esta introdução. No capítulo 2, será apresentada uma visão geral dos conceitos de Internet das Coisas e será visto como a Web das Coisas é proposta como solução de alguns problemas enfrentados na atual Internet das Coisas. Também será mencionado a ideia por trás de Cidades Inteligentes e mobilidade urbana. No capítulo 3, será especificado a aplicação para as paradas de ônibus e os requisitos obtidos para realizar o projeto. A implementação do sistema será detalhada no capítulo 4, descrevendo as decisões tomadas, arquitetura do sistema e tecnologias utilizadas. No capítulo 5, será mostrado a metodologia utilizada para testar e avaliar o sistema. Por fim, no capítulo 6, o texto será feita uma conclusão do trabalho, descrevendo as contribuições do trabalho e algumas melhorias necessárias.

2 CONTEXTUALIZAÇÃO

A Web, inicialmente um conjunto de documentos interligados, cresceu muito nas últimas décadas, influenciando a vida e a cultura das pessoas em todo mundo. Com esse crescimento, a Web definiu uma série de padrões abertos que facilitam o desenvolvimento de diversas aplicações. Não só a Web mudou, mas as cidades também. Cidades mais tecnológicas, eficientes e que proporcionam melhor qualidade de vida para as pessoas tem surgido cada vez mais ao redor do mundo. A Internet das Coisas está muito relacionada a essas cidades e tem atraído cada vez mais interesse de grandes corporações com estudos apontando o impulsionamento de um mercado bilionário nos próximos anos. No entanto, há muito a ser feito para se implementar uma Internet das Coisas real.

2.1 Internet das Coisas

A Internet das Coisas, ou *Internet of Things* (IoT), é um paradigma de comunicação relativamente recente que antecipa um mundo de coisas interconectadas ao nosso redor através da Internet e por meio de tecnologias diversas (ZANELLA et al., 2014). Apesar de esse ser um pensamento geral quando se fala em Internet das Coisas (dado o seu próprio nome), não há uma definição única para ela, e as diversas definições existentes na literatura podem acabar tornando o seu real significado confuso devido às diferenças que tendem a surgir de autor para autor. Tais divergências se devem à forma como as partes interessadas escolhem abordar o tema, em geral guiadas por interesses particulares, ou motivadas pela área de conhecimento, ou tecnologias que dominam, dependendo também, por vezes, do contexto da época em que o assunto foi tratado.

O termo “Internet das Coisas” foi usado pela primeira vez em 1999 pelo britânico Kevin Ashton que, então, trabalhava na *Auto-ID Labs*, uma rede de laboratórios de pesquisa acadêmica focada no desenvolvimento de redes RFID (*Radio Frequency IDentification*), uma tecnologia que usa campos magnéticos para identificação e rastreamento de pequenos dispositivos conhecidos como *tags*. O nome foi utilizado para fazer uma ligação entre dois termos de interesse crescente na época: a tecnologia RFID e a Internet ‘’(ASHTON, 2009).

Desde então, o termo tem sido amplamente usado por pesquisadores para descrever a interação do mundo real com o mundo virtual através de tecnologias de identificação, sistemas de localização em tempo real, sensores e atuadores (UCKELMANN; HARRI-

Figura 2.1: Diferentes visões do Internet das Coisas

Fonte: Adaptado de (ATZORI; IERA; MORABITO, 2010)

SON; MICHAHELLES, 2011).

Alguns centros de pesquisa têm se empenhado em tratar de maneira mais geral o conceito de Internet das Coisas. O IERC (*European Research Cluster on the Internet of Things*) afirma que a Internet das Coisas é "um conceito e um paradigma que considera que coisas possam se conectar a qualquer momento, em qualquer lugar, a qualquer coisa, utilizando, idealmente, qualquer rede ou serviço, e que sejam capazes de interagir e colaborar uns com os outros afim de criar novos serviços e aplicações." (VERMESAN et al., 2014). Já a ITU-T (*International Telecommunication Union - Telecommunication Standardization Sector*) apresenta o conceito de Internet das Coisas como: "Uma infraestrutura global que fornece serviços avançados baseados na interconexão de objetos físicos ou virtuais que possuam a capacidade de serem identificados e integrados em redes de comunicação."

Dada a forma decentralizada como o tema Internet das Coisas foi inicialmente tratado, esta pode ser melhor compreendida através da análise de três diferentes perspectivas que colaboraram para o surgimento da Internet das Coisas. Essas perspectivas são: orientada a coisas, orientada a Internet e orientada a semântica. A Internet das Coisas resulta da composição das tecnologias abordadas nas três visões, tornando-se um conceito mais abrangente, conforme mostrado na Figura 2.1 (ATZORI; IERA; MORABITO, 2010).

Na visão orientada a coisas, o foco é no desenvolvimento de dispositivos *endpoints*,

sensores, atuadores, tecnologias de identificação e rastreamento. A ideia inicial de Internet das Coisas foi criada sob essa perspectiva, em uma época em que os proprietários estavam interessados em incluir seus dispositivos (coisas) à Internet. Entre as tecnologias de maior destaque estão as de identificação e comunicação sem fio, como RFID, NFC (*Near Field Communications*) e o WSAN (*Wireless Sensor and Actuator Networks*).

Na visão orientada a Internet, o objetivo é desenvolver soluções para que as diversas "coisas" se interconectem à Internet de forma eficiente e segura. A IPSO (*IP for Smart Objects*) Alliance defende o uso do protocolo IP como meio de conectar os objetos inteligentes (*Smart Objects*) ao redor do mundo (DUNKELS; VASSEUR, 2008). Além dessa, foi proposta por um centro de pesquisa da MIT, a Internet 0, um protocolo de roteamento da camada física cujo objetivo é apresentar uma complexidade inferior a do IP. É também sob essa perspectiva que surge a Web das Coisas, uma importante abordagem da Internet das Coisas e tema de estudo deste trabalho cujas características serão descritas na próximas seções.

Finalmente, na visão orientada a semântica é defendido que tecnologias semânticas possam ser uma solução para as consequências que a quantidade enorme de coisas conectadas na Internet trará quanto a representação, armazenamento, interconexão, busca e organização de informação (TOMA; SIMPERL; HENCH, 2009).

Especula-se que o mercado da Internet das Coisas movimentará mais de US$1 trilhão de dólares até 2020. Inúmeras aplicações para casas inteligentes, cidades inteligentes, *wearables*, cuidados com a saúde, controle de eficiência de energia e muitas outras já estão sendo desenvolvidas.

No entanto, a Internet das Coisas tem um problema devido a justamente esse grande interesse e competitividade do mercado. O mesmo fator que incentiva as empresas a quererem criar novos padrões e tecnologias as afastam, e faz com que a Internet das Coisas atualmente esteja vivenciando uma verdadeira corrida armamentista em busca da dominação do mercado com suas tecnologias, tornando-as padrão *de facto* antes dos seus competidores. O chamado problema da fragmentação da Internet das Coisas, impede que muitos dispositivos que, por terem tecnologias desenvolvidas por proprietários diferentes, fiquem impossibilitados de se comunicar com outros dispositivos IoT por incompatibilidade de padrões e protocolos.

A próxima seção tem como intuito apresentar a Web das Coisas e como ela se propõe a resolver esse e outros problemas atuais da Internet das Coisas.

2.2 Web das Coisas

O furor em torno do desenvolvimento de novas tecnologias, e a vontade dos proprietários de tecnologias IoT em dominar o mercado, talvez tenha criado um dos maiores problemas da Internet das Coisas no momento: a sua fragmentação. A interconexão de dispositivos IoT não apenas é uma premissa, mas também um dos aspectos mais interessantes e promissores da Internet das Coisas. Dificultar a comunicação entre os dispositivos restringe o verdadeiro potencial da IoT e é um problema que a Web das Coisas se propõe a resolver. A Web das Coisas, ou *Web of Things* (WoT), é uma área de pesquisa nascida com o intuito em focar em soluções para facilitar o desenvolvimento de projetos IoT, assim como buscar resolver problemas relacionados principalmente a interoperabilidade devida à heterogeneidade de dispositivos e protocolos associados à Internet das Coisas através da aplicação de padrões Web abertos.

Uma de suas vantagens principais está no fato de que um desenvolvedor que segue essa abordagem não tem de se preocupar com detalhes específicos de protocolos ou conhecer diversos padrões diferentes da IoT. A Web das Coisas, assim como a Web, lida essencialmente com a camada de aplicação do modelo OSI (em contraste com a Internet das Coisas que tem potencial para lidar com todas elas). Sendo assim, a Web das Coisa efetivamente quebra o padrão "um protocolo, um dispositivo, uma aplicação", encontrado com frequência no estado atual da Internet das Coisas, ao integrar as "coisas" na Web, explorando padrões e tecnologias já bem estabelecidos como os protocolos de comunicação HTTP (*Hypertext Transfer Protocol*) e WebSockets, além de tipos de mídia como HTML (*Hypertext Markup Language*) e JSON (*Javascript Object Notation*). Ao transformar "coisas" em recursos web, a conexão entre dispositivos heterogêneos se torna muito mais simples e o desenvolvedor pode focar na lógica das aplicações em vez de ter se de preocupar com questões que fogem de seu escopo ou área de conhecimento. (GUINARD; TRIFA, 2016).

Uma das maneiras sugeridas para se implementar a Web das Coisas é através da integração das coisas em uma arquitetura de *Web Service* padronizado (GUINARD et al., 2011). No entanto, *Web Services* que implementam padrões como SOAP (*Simple Object Access Protocol*) ou WSDL (*Web Services Description Language*) podem ser custosos demais em termos de processamento e consumo de memória. Considerando que muitos dispositivos que formam a Internet das Coisas possam conter sistemas de hardware e software muito simples, essa opção pode não ser a mais adequada.

Uma alternativa recente para a solução baseada em *Web Services* tem sido o uso de servidores Web embarcados (EWS) que podem ser executados em dispositivos com apenas alguns kB de memória RAM ou EEPROM (DUQUENNOY; GRIMAUD; VANDEWALLE, 2009). Nesse contexto de Web das Coisas, o sistema de um cliente web geralmente é mais poderoso que o de um servidor embarcado. Ao contrário de um servidor web comum, porém, ele não precisa lidar com milhares de conexões e requisições. Existem conjuntos de técnicas bem conhecidas, como AJAX *(Asynchronous Javascript and XML)* e Comet, que possibilitam que o cliente realize o processamento que, de outra forma, seria feito pelo servidor.

Talvez um dos aspectos que mais se destacam nos dispositivos da Web das Coisas em relação a Internet das Coisas, é a forma como uma Coisa Web é referenciada. Na Web das Coisas, cada Coisa Web é um recurso Web. Um recurso no contexto da Web, estritamente falando, é qualquer informação que possa ser identificada por uma URI (BERNERS-LEE; FIELDING; MASINTER, 2005). Um recurso pode ser um documento, uma imagem ou mesmo uma coleção de outros recursos. Sendo assim, um dispositivo WoT deve ser identificado por uma URI única ou, mais especificamente, uma URL semelhante a *http://example.web.thing:8000*. Além de ser um recurso por si, um dispositivo WoT geralmente terá vários outros recursos que representam propriedades, eventos e ações, ou serviços, que são expostos para que outras coisas possam interagir com ela formando, de fato, uma "Web"de Coisas.

Juntamente aos servidores embarcados, é desejável aplicar o estilo REST para assegurar diversas características importantes para a Web das Coisas conforme será mostrado a seguir.

2.3 Estilo arquitetural REST

O REST (*REpresentation State Transfer*) é um estilo arquitetural da Web que visa melhorar propriedades de sistema como desempenho, escalabilidade e segurança (FIELDING, 2000). Esse estilo destina-se a guiar a criação de serviços para que sejam fracamente acoplados e de fácil reusabilidade ao impor um conjunto de restrições sobre elementos que a compõe. Um sistema que aplica as restrições exigidas pelo estilo arquitetural REST pode ser chamado de *RESTful*.

Um recurso é qualquer informação que pode ser nomeada e identificada por uma URI (BERNERS-LEE; FIELDING; MASINTER, 2005). É a abstração de informação

principal no REST. Pode-se citar como exemplo documentos eletrônicos, imagens ou uma coleção de outros recursos. Recursos tem um estado interno que pode ser estático ou alterado dinamicamente. A captura, ou a atualização, do estado de um recurso é feito pela transmissão de uma representação a partir de um componente. Uma representação é constituída por dados descritos por metadados. Recursos também podem ter metadados que são independentes da representação. Além desses, existem metadados de controle, usados para indicar o propósito, ou o significado, ou alterar um comportamento padrão de um elemento. Um identificador de recursos é utilizado para identificar unicamente um recurso na transmissão entre componentes. É importante notar que um recurso é um mapeamento para uma entidade e não a entidade referida. Essas entidades podem ser um conjunto de representações ou identificadores (FIELDING, 2000). Considere o exemplo onde um recurso pode ser identificado pela URL *http://www.example.com:3000/temperature* e requisitado pelo verbo GET do protocolo HTTP. Considerando que esse recurso guarde a informação sobre a temperatura atual de um dispositivo, uma possível representação enviada para outro componente poderia ser um documento JSON que define *"temperature": "25"*.

Algum tempo depois, o estado interno pode ter sido alterado (a temperatura pode ter aumentado ou diminuído) e representação será algo do tipo: *"temperature": "18"*. Apesar de a entidade ter sido alterada, o mapeamento e a semântica continuam o mesmo e, portanto, o recurso é o mesmo (apesar de seu estado ter sido alterado em algum momento).

Um dos aspectos mais importantes do REST é a definição de uma interface uniforme entre os componentes da arquitetura. Baseado na identificação e manipulação de recursos por meio de URIs e através da aplicação de métodos HTTP (GET, PUT, POST, DELETE, etc.), ela garante o desacoplamento entre a semântica da aplicação e os serviços fornecidos por esta, além de uma melhoria na visibilidade das interações entre os componentes.

Com isso, o acesso aos mais diversos dispositivos pertencentes a Web das Coisas é simplificado. Pessoas poderiam facilmente, através de um navegador Web, por exemplo, procurar, acessar, interagir e compartilhar informações sobre os dispositivos, da mesma forma que hoje é feito com qualquer recurso Web identificável.

Além da exigência de uma interface uniforme, duas restrições são de grande importância para tornar um sistema RESTful: a separação explícita das responsabilidades entre cliente e servidor, e a condição de que o servidor não guarde qualquer informação sobre o estado atual de seus clientes - isso é, uma requisição do cliente deve conter toda

a informação necessária para que o servidor possa tomar a decisão. A primeira restrição é necessária para que os componentes do sistema possam evoluir de forma independente. Já a segunda, melhora conceitos de visibilidade, confiabilidade e escalabilidade (FIELDING, 2000).

Um servidor também precisa declarar (implícita ou explicitamente), na resposta de uma requisição, se os dados requisitados devem, ou não, ser mantidos em cache. Essa requisição tem por objetivo melhorar a eficiência, escalabilidade e o desempenho percebido pelo usuário ao ter o tempo médio das respostas reduzido.

Por fim, a última restrição do REST define a implementação de um sistema em camadas, de forma que cada componente (cliente ou servidor) não tenha conhecimento de outras camadas, além daquelas que estão interagindo. Dessa forma, é possível um cenário onde hajam servidores intermediários ao longo do caminho sem que o cliente tenha como distingui-los do servidor final. Através do balanceamento de carga e da utilização de caches compartilhadas, esses servidores intermediários podem melhorar a escalabilidade e o desempenho do sistema. Além disso, podem aplicar políticas de segurança ao agirem como *firewalls*, por exemplo.

2.4 Arquitetura W3C

A W3C (*World Wide Web Consortium*) é a principal organização internacional de desenvolvimento de padrões para a Web. Foi fundada pelo inventor da *World Wide Web*, Tim Berners-Lee, e tem como objetivo primário desenvolver protocolos e diretrizes para assegurar o crescimento da Web (W3C, 2017b). Todos os padrões desenvolvidos pela W3C são abertos. Alguns exemplos de padrões conhecidos são: HTML, XML, CSS, DOM *(*Document Object Model), HTTP, *Linked Data*, SOAP e SVG (*Scalable Vector Graphics*) (W3C, 2017a).

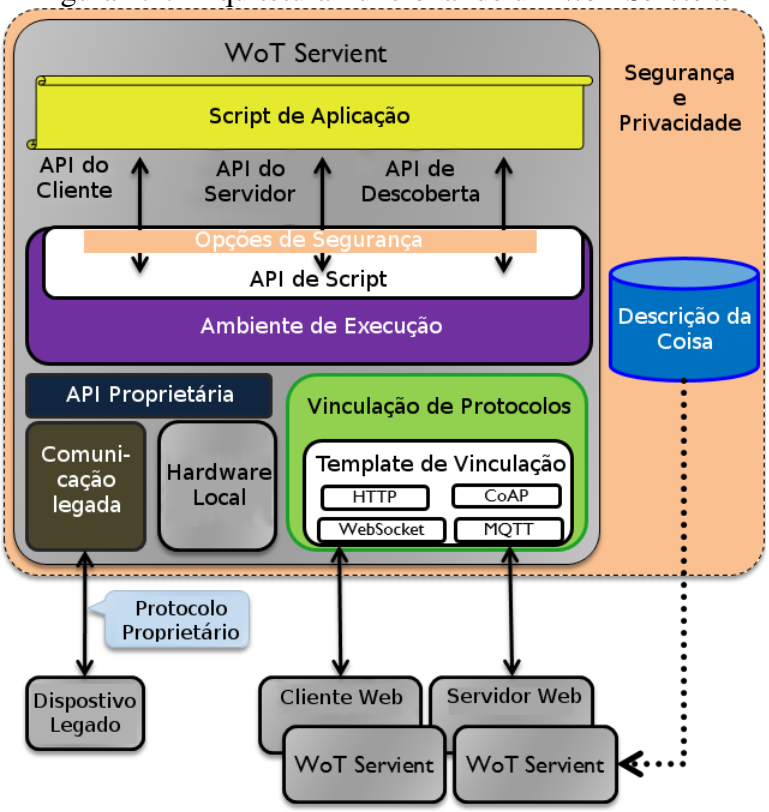
Em feveiro de 2017, a W3C lançou um grupo de trabalho com o objetivo de padronizar a Web das Coisas (W3C, 2017c). A finalidade do grupo é desenvolver uma arquitetura para WoT tendo em mente questões relacionadas a flexibilidade, compatibilidade, segurança e privacidade. A arquitetura está sendo projetada levando em consideração inúmeros dispositivos físicos que já existem atualmente na Internet das Coisas, servindo como ponte de ligação entre sistemas legados IoT e tecnologias baseadas na Web.

Nessa arquitetura, dispositivos WoT são representados por entidades chamadas *WoT Servients*. Um *Servient* é um modelo virtual responsável por expor o dispositivo

na Web e fornecer as suas interfaces de acesso e controle. Ele é constituído de vários módulos conforme mostrado na Figura 2.2.

Os módulos que formam o Servient são: Descrição da Coisa, Vinculação de Protocolos, Ambiente de Execução, API de *Script*, *Script* da Aplicação e Segurança e Privacidade. Além desses, Hardware Local e Comunicação Legada são módulos opcionais para *Servients* hospedados em dispositivos físicos que se comunicam por protocolos proprietários não padronizados pela WoT. A seguir será explicado a finalidade de cada módulo.

Figura 2.2: Arquitetura Funcional de um *WoT Servient*



Fonte: Adaptado de <http://w3c.github.io/wot/architecture/wot-architecture.html>

2.4.1 Descrição da Coisa

Este módulo apresenta metadados que descrevem o dispositivo WoT, suas propriedades, que serviços oferece e a forma como um cliente deve proceder para se comunicar com o dispositivo. Para ser capaz de se comunicar, é necessário conhecer os protocolos que o dispositivo reconhece e a forma que sua interface mapeia as requisições para os seus recursos. Para dispor essas informações, a Descrição da Coisa se baseia em mecanismos de Web Semântica, em especial no RDF (*Resource Description Framework*) que é usado como modelo de dados padrão. JSON-LD é utilizado como notação oficial do

RDF. A Descrição da Coisa não necessariamente necessita estar armazenada no dispositivo, podendo estar hospedada em outro lugar. Isto pode ocorrer quando um dispositivo IoT legado está sendo incorporado a um dispositivo WoT.

Um vocabulário semântico foi criado para a Web das Coisas onde são descritos os padrões de interação de um dispositivo WoT através do conceito de propriedades, eventos e ações.

As propriedades representam dados ou variáveis de uma Coisa Web, que podem ser estáticos ou dinâmicos. Para melhor visualizar as propriedades, será fornecido um exemplo baseado em um dispositivo que possui um sensor de temperatura.

Na Figura 2.3 é possível ver a descrição de um dispositivo WoT chamado *MyTemperatureThing* cuja única interação disponível é uma propriedade chamada *temperature*. Na propriedade é definido um link <coap://mytemp.example.com:5683/temp> e

Figura 2.3: Exemplo de uso de propriedades.

```
{
  "@context": ["http://w3c.github.io/wot/w3c-wot-td-context.jsonld"],
  "@type": "Thing",
  "name": "MyTemperatureThing",
  "interactions": [
    {
      "@type": ["Property"],
      "name": "temperature",
      "outputData": {"valueType": { "type": "number" }},
      "writable": false,
      "links": [{
        "href" : "coap://mytemp.example.com:5683/temp",
        "mediaType": "application/json"
        }]
    }
  ]
}
```

Fonte: adaptado de <https://w3c.github.io/wot-thing-description/>

uma *media-type* "application/json". Além disso, é informado que a propriedade é do tipo *number* e que ela não pode ser escrita. Com essas informações, quem quiser acessar o valor medido pelo sensor de temperatura do dispositivo sabe que precisa fazê-lo por meio da URL informada e que receberá como resposta um arquivo JSON contendo um valor que deve ser interpretado como *number*. As chaves precedidas por '@' são palavras reservadas do JSON-LD. A palavra *context* define o vocabulário que está sendo usado. Todo dispositivo WoT será do tipo *Thing* que é declarado no vocabulário definido em <http://w3c.github.io/wot/w3c-wot-td-context.jsonld>. É nele que é declarado a semântica dos termos *interactions*, *Property*, *outputData*, e muitos outros termos associados a WoT.

As propriedades representam dados que podem ser lidos e escritos em tempo real. Mas o dispositivo WoT também pode ser capaz de realizar algum tipo de processamento,

ou serviço, como acender uma lâmpada, desligar um motor ou aumentar o volume. As ações servem justamente para definir o que um dispositivo WoT é capaz de fazer e como deve ser feita a requisição. Um exemplo de uma Descrição da Coisa de um dispositivo chamado *MyLedThing* que pode controlar um LED é mostrado na Figura 2.4. Nele é destacado uma ação chamada *fadeIn* que é invocada pela URL <http://mytemp.example.com:8080/in> e que deve passar um dado do tipo JSON contendo um valor que deve ser do tipo inteiro.

Figura 2.4: Exemplo de uso de ações.

```
{
  "@context": [
    "http://w3c.github.io/wot/w3c-wot-td-context.jsonld"
  ],
  "@type": "Thing",
  "name": "MyLEDThing",

  "interactions": [

    ...

    {
      "@type": ["Action"],
      "name": "fadeIn",
      "inputData": {
        "valueType": { "type": "integer" }
      },
      "links": [
        {
          "href": "http://mytemp.example.com:8080/in",
          "mediaType": "application/json"
        }
      ]
    },

    ...

  ]
}
```

Fonte: autor.

O último padrão de interação, chamado de eventos, permite que um cliente se inscreva no dispositivo com a intenção de que este o informe quando um evento de determinado tipo ocorrer. Na Figura 2.5 é mostrado um exemplo de descrição de um evento chamado *criticalTemperature*. Nesse caso o protocolo CoAP é utilizado, e um cliente pode se inscrever nesse evento através da URL <coap://mytemp.example.com:8080/ev>. Quando o evento for disparado, o dispositivo WoT informará todos os clientes inscritos no evento, que devem esperar um valor do tipo string.

Figura 2.5: Exemplo de uso de eventos.

```
{
  "@type": ["Event"],
  "name": "criticalTemperature",
  "outputData": { "valueType": { "type": "string" } },
  "links": [
    {
      "href": "coap://mytemp.example.com:8080/ev",
      "mediaType": "application/json"
    }
  ]
}
```

Fonte: autor.

2.4.2 Vinculação de Protocolos

A Descrição da Coisa informa a URL e os parâmetros de entrada e saída para cada uma das três interações definidas na arquitetura WoT. No entanto, é necessário informações sobre quais métodos ou verbos do protocolo utilizar para acessar esse recurso. O módulo Vinculação de Protocolos foi pensado para fazer o mapeamento de cada protocolo e de cada método que este usa para se comunicar com os recursos que o dispositivo WoT oferece. Ele define como cada propriedade, ação e evento devem ser acessados de acordo com o protocolo usado. Até agora, dois protocolos baseados em REST são suportados: HTTP e CoAP. É previsto também o mapeamento também para MQTT, WebSockets, entre outros. A Tabela 2.1 mostra como é feito a vinculação dos protocolos HTTP e CoAP.

Tabela 2.1: Vinculação dos Protocolo HTTP e CoAP

Recurso	*Interação*	*Método HTTP*	*Método CoAP*
Property	Read	GET	GET
	Write	PUT	PUT
Action	Invoke	POST	POST
	Update task	PUT	PUT
	Cancel task	DELETE	DELETE
Event	Subscribe	POST + polling	POST + Observe
	Update subscription	PUT	PUT
	Cancel subscription	DELETE	DELETE

Fonte: adaptado de (WG, 2017).

Para HTTP e CoAP, propriedades são lidas com o verbo *GET* e são escritas (quando o valor do atributo *writable* for *true*) com o verbo *POST*. Ações são invocadas por *POST*, atualizadas por *PUT* e removidas por *DELETE*. É possível se inscrever em um evento através de *POST*, no entanto, o protocolo HTTP não tem suporte para eventos, e a aplicação deverá fazer *polling* para verificar quando o evento ocorrer. Já o CoAP pode utilizar o método *observe*.

2.4.3 Ambiente de Execução

Este módulo implementa o interpretador de *scripts* e o *loop* de eventos, além de gerenciar o ciclo de vida dos *scripts* de aplicação. Também é responsável por fornecer acesso local e remoto para recursos através de APIs de baixo nível.

2.4.4 API de Script

O módulo API de *Script* fornece APIs para comunicação entre dispositivos. Essas APIs podem ser separadas em quatro categorias: WoT, Cliente, Servidor e Opções de Segurança.

Tabela 2.2: As quatro categorias de *API de Scripts*

API de *Script*	*Exemplos de API*
Descoberta	discover retrieve createLocalThing
Cliente	invokeAction getProperty addListener
Servidor	addProperty addAction addEvent register onRetrieveProperty onObserve
Opções de segurança	autenticação autorização

A API de Descoberta apresenta funcionalidades para que seja possível descobrir e criar Coisas Web. A API do Cliente fornece métodos para interagir com um dispositivo WoT que foi acessado. Por exemplo, a API oferece o método *getProperty* que retornará uma propriedade específica dessa Coisa Web. A API do Servidor permite que um *Servient* crie, ou modifique, suas próprias propriedades, ações e eventos. Opções de Segurança é responsável por fornecer métodos que autentiquem e autorizem Coisas na WoT, bem como efetuar uma comunicação segura entre os dispositivos.

2.4.5 Script da Aplicação

Esse módulo é essencialmente o código implementado para o dispositivo com o uso das funcionalidades da API de *Script* sendo executado sobre o Ambiente de Execução.

2.4.6 Segurança e Privacidade

Não é um módulo em si, mas um mecanismo de segurança e privacidade que está relacionado com vários módulos. Por exemplo, Descrição da Coisa podem descrever o tipo de protocolo de segurança que o dispositivo usa. API de *Script* deve fornecer métodos seguros e boas práticas para assegurar a segurança do dispositivo.

2.5 Cidades Inteligentes

De acordo com o Departamento de Assuntos Sócio-Econômicos da ONU (UN, 2014), 54% da população mundial vivia em cidades no ano de 2014, e a perspectiva é que essa porcentagem tenda a aumentar ainda mais nas próximas décadas. Associados a este fenômeno urbano, são cada vez maiores os problemas relacionados a gestão de resíduos, escassez de recursos, poluição do ar, problemas de saúde, congestionamentos de trânsito, bem como problemas de natureza social e organizacional (CHOURABI et al., 2012). Cresce, portanto, o interesse em tornar mais eficientes os diversos setores de uma cidade, assim como tornar locais públicos mais inteligentes e mais agradáveis para a população em geral.

Dessas motivações surge o conceito de “Cidades Inteligentes”, termo que vem sendo amplamente debatido nos últimos anos. Diversas cidades já se auto-proclamam “inteligentes”, não havendo, no entanto, um consenso que defina formalmente o momento em que uma cidade se torna uma. De um modo geral, uma “cidade inteligente” surge quando tecnologias de informação e comunicação são aplicadas a centros urbanos visando fornecer uma infraestrutura que garanta formas de se alcançar o desenvolvimento social, econômico e urbano de maneira sustentável, melhoria na qualidade de vida da população e maior eficiência na utilização de recursos (GEA et al., 2013).

De acordo com (GIFFINGER et al., 2007), uma Cidade Inteligente é uma cidade construída através esforço de cidadãos conscientes, e que se destaca positivamente em seis

Figura 2.6: Fatores e características de uma Cidade Inteligente

Economia Inteligente (Competitividade)	População Inteligente (Capital Humano e Social)	Administração Inteligente (Participação)
Espírito inovador	Nível de qualificação	Participação na tomada de decisões
Empreendedorismo	Afinidade com aprendizagem ao longo da vida	Serviços públicos
Imagem econômica e marcas registradas	Pluralidade ética e social	Serviços sociais
Produtividade	Flexibilidade	Administração transparente
Flexibilidade do mercado de trabalho	Criatividade	
Inserção internacional	Cosmopolitanismo	Perspectivas e estratétias políticas
Capacidade de transformar	Participação na vida pública	
Mobilidade Inteligente (Transporte e TCI)	**Ambiente Inteligente (Recursos Naturais)**	**Modo de Vida Inteligente (Qualidade de Vida)**
Acessibilidade local	Atratividade das condições naturais	Centros culturais
Acessibilidade (inter)nacional	Poluição	Condições de saúde
		Segurança individual
Disponibilidade de infraestrutura de TCI	Proteção ambiental	Qualidade de moradia
		Instalações educacionais
Sistemas de transporte sustentáveis, inovadores e seguros	Gerenciamento sustentável de recursos	Atrações turísticas
		Coesão social

Fonte: adaptado de (GIFFINGER et al., 2007)

características: economia inteligente, população inteligente, administração inteligente, mobilidade inteligente, ambiente inteligente e modo de vida inteligente. A Figura 2.6 mostra as seis características e os fatores associados que as descrevem.

Como pré-requisito para uma cidade tornar-se inteligente é necessário que ela usufrua de uma infraestrutura de redes sem fio para permitir a alta demanda de conexões dos cidadãos e de organizações locais. A infraestrutura e ambientes públicos da cidade devem ser equipadas com dispositivos inteligentes, possibilitando o gerenciamento de dados em tempo real, além da emissão de avisos e processamento de informações (SCHAFFERS et al., 2011).

2.5.1 Mobilidade Inteligente

A Mobilidade Inteligente é uma característica de uma Cidade Inteligente que visa melhorar aspectos como segurança, sustentabilidade e eficiência no transporte urbano, assim como tornar mais fácil e rápido o deslocamento de pessoas ao longo da cidade (CITY, 2015a). Aplicações para Mobilidade Inteligente são de grande interesse devido aos impactos que tem sobre a cidade. Com o objetivo de contextualizar, serão apresentados brevemente quatro exemplos de aplicações.

Uso compartilhado de bicicletas. Após efetuado cadastro em um aplicativo, usuários podem utilizar bicicletas, por um determinado período de tempo, que estão estacionadas em diversas áreas espalhadas pela cidade. Exemplos de cidades que implementam: Milão, Paris, Nova Iorque, Barcelona, Porto Alegre (WIKIPEDIA, 2016a) (BIKEPOA, 2016).

Controle de tráfego inteligente. Inclui sistema de sensores e controle de semáforos com o objetivo de regular o fluxo de trânsito de acordo com a demanda (TRANSPORT, 2016). Em Barcelona, os semáforos ficam verdes quando há veículos de emergência em rota (CITY, 2015b). Exemplos de cidades que implementam: Barcelona, Amsterdã (CITY, 2016).

Paradas de ônibus inteligentes. Podem fornecer diversas informações em tempo real para os passageiros, tais como horários, tipos de ônibus (com acessibilidade, com suporte para bicicletas, etc.), itinerários, e informações gerais. Exemplos de cidades que implementam: Seul (GOVERNMENT, 2010), Moscou (TIMES, 2013).

Estacionamentos inteligentes. São aplicações que podem indicar o melhor local para o motorista estacionar. Baseiam-se no número de vagas disponíveis em estacionamentos e nos locais aonde o usuário pretende se deslocar. Exemplos de cidades que implementam: Amsterdã (WIKIPEDIA, 2016b).

3 ESPECIFICAÇÃO

Este capítulo apresenta a especificação a modelagem do sistema. Primeiramente será apresentada uma visão geral do projeto para em seguida detalhar os requisitos funcionais e não funcionais do sistema. Por fim será mostrado o protótipo de interfaces e interações do sistema.

3.1 Descrição do sistema

Neste trabalho é proposto o desenvolvimento de uma aplicação para um dispositivo eletrônico com a finalidade de ficar posicionado em uma parada de ônibus de uma Cidade Inteligente. Essa aplicação se baseará no modelo WoT descrito nos capítulos anteriores e tem como principal objetivo receber informações em tempo real sobre o tempo de chegada dos próximos ônibus que será informadas ao público. Essas informações serão mostradas em um monitor onde um dispositivo baseado em Raspberry Pi deve estar conectado. Juntamente, será implementado um aplicativo em um ambiente Android com o objetivo de demonstrar a forma como um cliente pode interagir com o dispositivo WoT através da interface Web exposta pelo dispositivo. Esse aplicativo possibilitará o usuário acompanhar o tempo em que um ônibus de uma determinada linha levará para chegar na parada selecionada pela interface. Além disso, será implementado um servidor que irá agregar em um banco de dados as mensagens recebidas dos ônibus ativos. O servidor será responsável por implementar o algoritmo que estima o tempo de chegada dos ônibus e informar os dispositivos WoT das paradas com os valores calculados. Para o desenvolvimento deste projeto, será abstraída a forma como tanto o dispositivo WoT e os ônibus se conectarão a Internet, i.e., será considerado que todos as partes envolvidas no sistema podem se conectar a Internet de alguma forma.

3.2 Requisitos funcionais

Os requisitos funcionais são utilizados para descrever as funcionalidades de um sistema e especificar o comportamento sob determinadas situações. Os requisitos funcionais mapeados para a aplicação proposta são os seguintes:

- Os ônibus devem ser capazes de transmitir suas coordenadas atuais (latitude e lon-

gitude), seu ID (que deve ser único) e o ID da sua linha para o servidor através da Internet.

- O servidor deve armazenar os dados recebidos pelos ônibus em um banco de dados para processamento futuro.
- Os dispositivos da paradas de ônibus devem poder se conectar ao servidor via Internet.
- Os dispositivos da paradas de ônibus devem mostrar as informações sobre os tempos de chegada dos ônibus em um monitor.
- As paradas devem expor sua interface WoT para serem acessadas por clientes na Web.
- Um aplicativo Android deve poder se conectar com uma parada e obter informações sobre os ônibus que passam por ela.

3.3 Requisitos não-funcionais

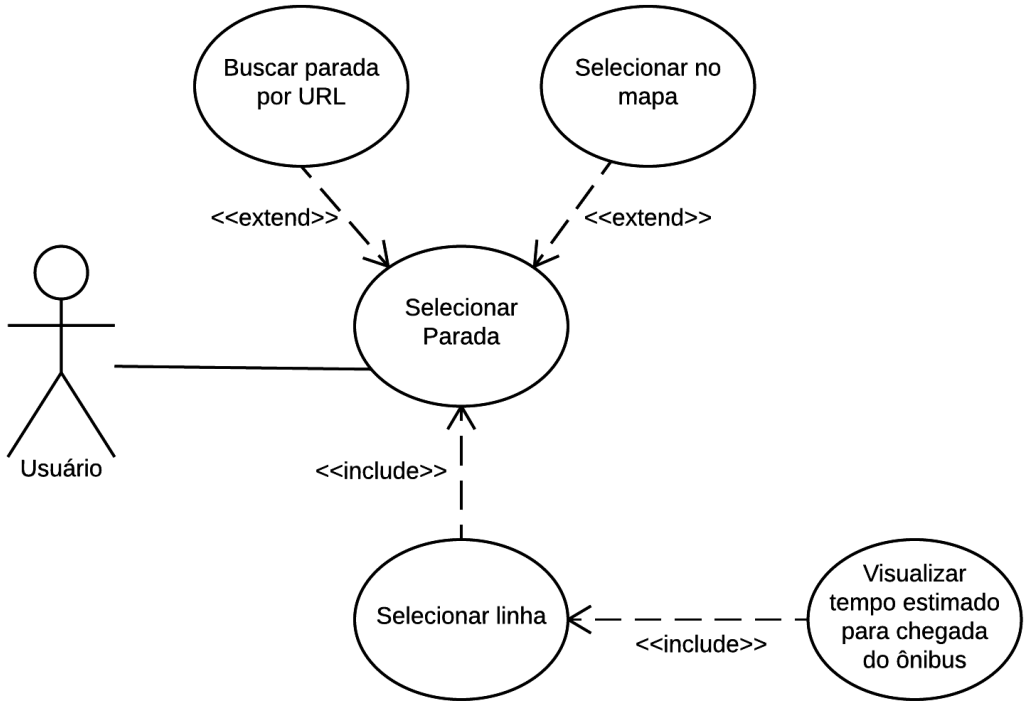
Os requisitos não funcionais, ao contrário dos funcionais, não especifica o comportamento do sistema, mas estabelece restrições que visam melhorar questões como qualidade e eficiência. Neste projeto, os requisito não-funcionais definidos são:

- O sistema deve ser desenvolvido, sempre que possível, com software de código aberto.
- O servidor deve executar no sistema operacional GNU/Linux.
- O servidor deve ser implementado em Node.js versão 6 ou superior.
- O dispositivo WoT deve executar em um Raspberry 3 com sistema operacional Raspbian.
- O dispositivo WoT deve ser capaz de executar versão 6 do Node.js.
- A aplicação cliente para *Smartphones* deve ser capaz de executar no sistema operacional Android.
- O dispositivo WoT deve utilizar padrões Web abertos, assim como explorar os novos padrões emergentes da WoT.

3.4 Diagrama de casos de uso da aplicação Android

A Figura 3.1 mostra o diagrama de casos de uso para a aplicação Android elaborado de acordo com os requisitos funcionais estabelecidos. O caso de uso descreve a interação do usuário com a aplicação, que envolve selecionar paradas e linhas e visualizar o tempo de chegada dos próximos ônibus.

Figura 3.1: Diagrama de Casos de Uso



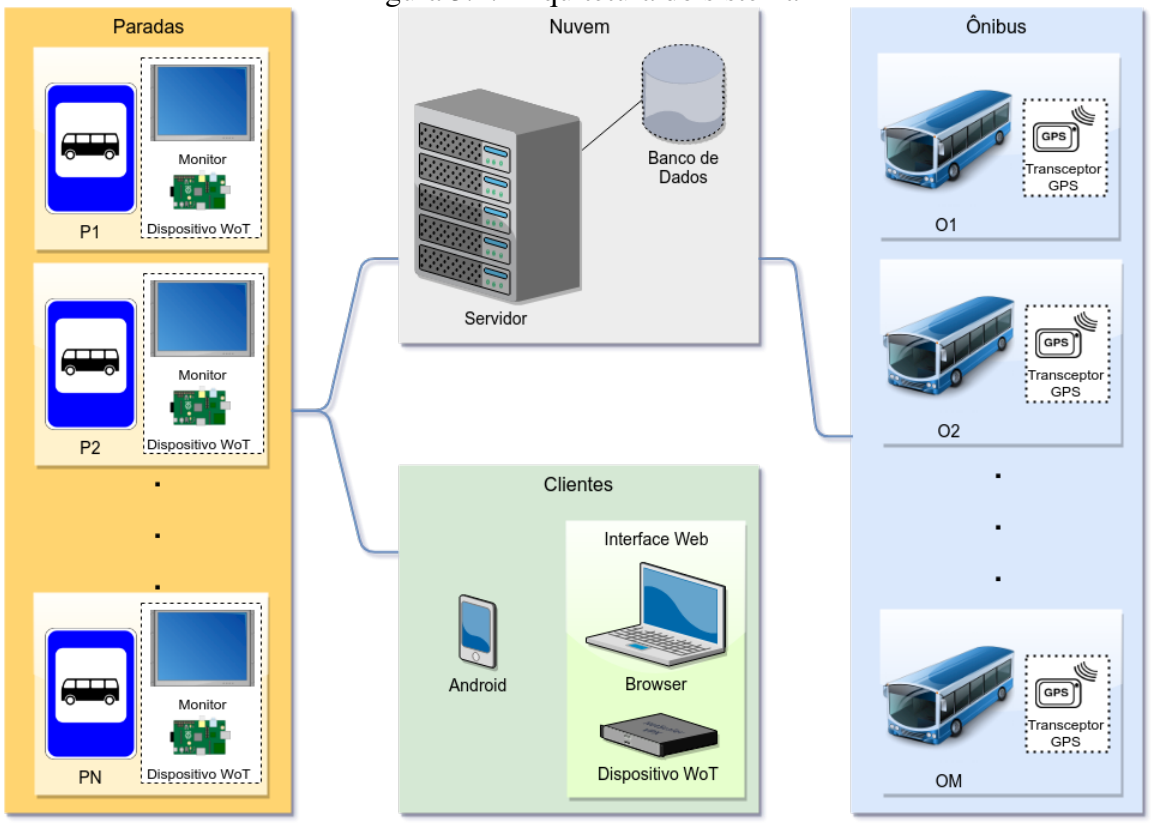
Fonte: elaborada pelo autor.

3.5 Projeto do sistema

Foram consideradas duas formas para implementar a comunicação entre as paradas e os ônibus: direta e por intermédio de um servidor. A forma direta é a maneira mais trivial de abordar o sistema, fazendo com que os ônibus abram uma conexão diretamente com as paradas. No entanto, há dois problemas nessa abordagem. Primeiramente, um ônibus teria que manter possivelmente dezenas de conexões abertas com todas as paradas às quais ele deve informar sua posição. Isto não apenas faria com que o dispositivo dos ônibus demandassem de recursos de hardware mais custosos, como implicaria em ter uma lógica de software mais complexa, fazendo com o que o ônibus necessitasse saber a quais paradas transmitir seus dados. Em segundo lugar, as paradas, além de também precisarem

manter várias conexões abertas com os ônibus, teriam que calcular por si mesmo o tempo estimado de cada ônibus. Associando ao fato de que a parada já teria que lidar com outros clientes, isso poderia fazer com que os dispositivos nas paradas fossem facilmente sobrecarregados. A segunda abordagem, inserindo um servidor no sistema, além de resolver o problema das múltiplas conexões (um servidor geralmente é capaz de receber milhares de requisições facilmente), inclui o fato de que todas as informações recebidas pelos ônibus podem ser armazenadas em um banco de dados para um possível processamento de dados das empresas responsável pelo transporte público. Esses dados podem ser usados para melhorar a distribuição da frota de ônibus, por exemplo.

Figura 3.2: Arquitetura do sistema



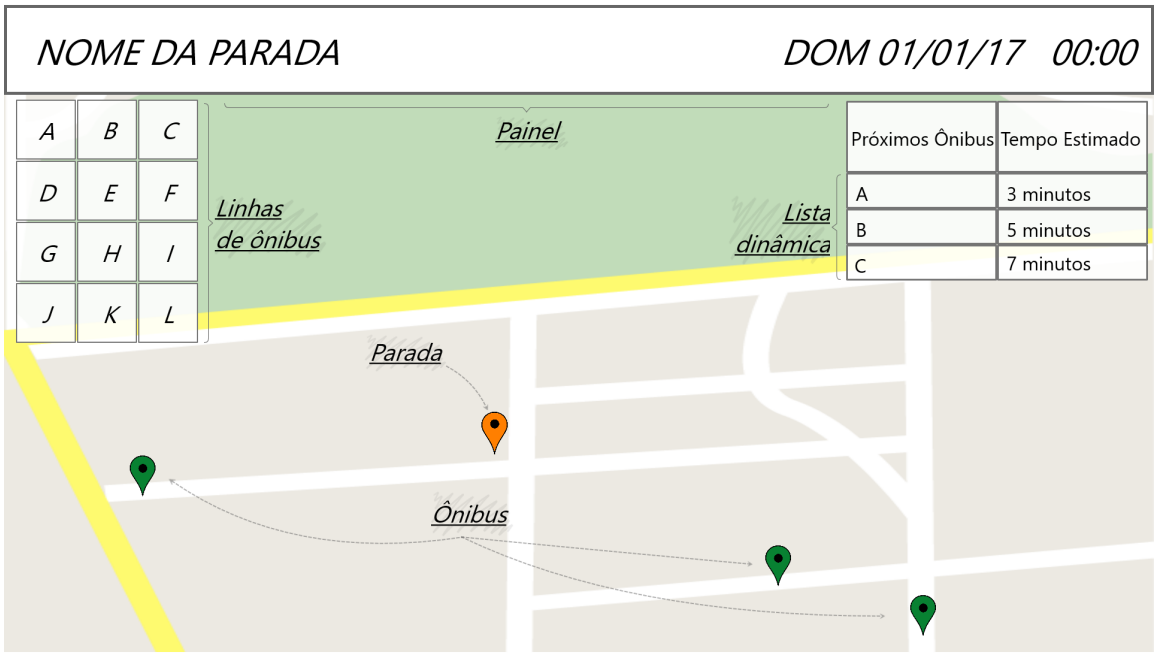
Fonte: elaborada pelo autor.

3.6 Protótipos das interfaces

Nesta seção será apresentada um esboço utilizado para planejar a criação das interfaces com o cliente. Primeiramente será apresentado a interface desenvolvida para o dispositivo WoT, contendo informações para o público. Em seguida será mostrada a interface do aplicativo Android que interage com o dispositivo WoT.

3.6.1 Dispositivo WoT

Figura 3.3: Interface da tela do dispositivo para a parada de ônibus.



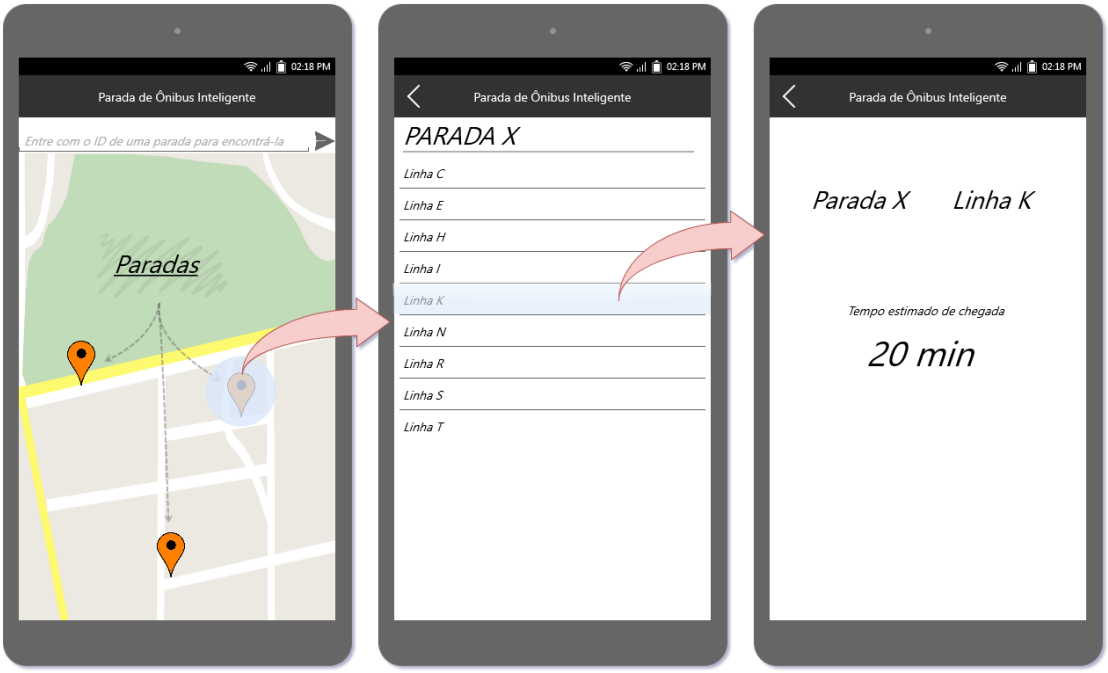
Fonte: elaborada pelo autor.

A Figura 3.3 apresenta os principais blocos que irão compor a interface. No topo, haverá uma área em destaque com o nome da parada e informações de data e hora. No canto esquerdo, será reservado um espaço para listar todas as linhas de ônibus que circulam por aquela parada, inclusive as linhas que não têm ônibus em movimento no momento. À direita, uma lista dinâmica informará os próximos ônibus que chegarão à parada, bem como o tempo que cada um levará para alcançá-la. Por fim, no centro, cobrindo quase toda interface, o mapa do local será exibido marcando a posição da parada no centro da tela e Os ônibus também serão expostos no mapa com suas posições atualizadas em tempo real.

3.6.2 Aplicativo Android

O aplicativo terá três interfaces principais como mostra a Figura 3.4. A primeira tela (à esquerda) terá um mapa onde aparecerão todas as Paradas de Ônibus Inteligente conectadas. O usuário poderá percorrer o mapa, ou digitar o ID da parada no topo da tela, para centralizar o mapa na parada. Ao tocar no ícone de alguma parada, a aplicação mudará para a segunda tela (a do meio na Figura 3.4). Na parte de cima, será mostrado

Figura 3.4: Interface das telas do aplicativo Android.



Fonte: elaborada pelo autor.

o nome da parada que o usuário acabou de selecionar. Abaixo, todas as linhas de ônibus associadas àquela parada serão listadas. O usuário pode selecionar uma linha e, ao fazê-lo, será redirecionado para a terceira tela (à direita na Figura 3.4). Nessa tela será apresentado o nome da parada e da linha de ônibus selecionada. No meio desta tela, um contador indicará o tempo estimado para o próximo ônibus daquela linha chegar na parada.

4 IMPLEMENTAÇÃO

Este capítulo irá apresentar o método de desenvolvimento do projeto, bem como as tecnologias utilizadas, seguindo a especificação do capítulo anterior e os conceitos abordados ao longo deste trabalho. Esta parte será dividida em três seções, detalhando, primeiramente, a implementação do servidor, seguindo do dispositivo WoT e, por último, do aplicativo móvel para *smartphones* Android.

4.1 Implementação do servidor

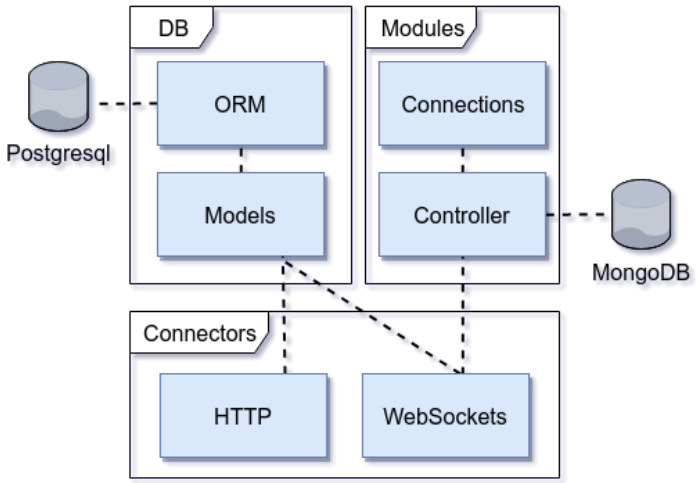
Conforme especificado, todas as partes do servidor devem ser, preferencialmente, compostas de software de código aberto. Com isso em mente, o projeto foi desenvolvido no sistema sistema operacional GNU/Linux com a distribuição Ubuntu 16.04. A plataforma escolhida foi o Node.js v6.10.3, um ambiente de execução Javascript que implementa a engine V8 da Google. Além de ser eficiente ao lidar com múltiplas requisições assíncronas, talvez a maior vantagem do Node.js seja sua comunidade ativa. Através do repositório de módulos,e do instalador de pacotes *npm*, torna-se extremamente fácil procurar e instalar bibliotecas de todos os tipos.

Para a persistência dos dados, foram utilizados dois tipos de bancos de dados, relacional e não relacional. O banco de dados relacional será responsável para armazenar dados mais ou menos estáticos como, por exemplo, característica das paradas e linhas de ônibus. Ele é necessário para que o servidor realize consultas relacionais envolvendo informações de várias entidades conforme será visto adiante. O software para este tipo de banco utilizado foi Postgresql v9.6.3. O banco de dados não relacional escolhido foi MongoDB v3.4.6. Ele tende a ser muito mais eficiente que o banco de dados relacional, e é indicado para armazenar grande quantidades de dados ao mesmo tempo. Suas desvantagens estão relacionadas ao fato de que ele não possui validações tão rigorosas quanto o SQL, por exemplo, mas como será visto, ele será utilizado para armazenar dados em grande quantidade de apenas uma entidade, não exigindo relações ou consultas complexas.

4.1.1 Organização do servidor

A Figura 4.1 apresenta uma visão geral e simplificada dos principais componentes do servidor. Os componentes mostrados em *Connectors* implementam a parte que expõe

Figura 4.1: Organização do servidor.



Fonte: elaborada pelo autor.

os servidores HTTP e WebSocket, implementados utilizando as bibliotecas *express* e *ws*, respectivamente. O servidor HTTP é utilizado para atualizar dados no banco através de uma interface RESTful. Ele também tem um único recurso público para que se possa descobrir quais paradas estão conectadas no momento. O componente *Models* representa o conjunto de classes relacionadas às entidades no banco de dados relacional. Essas classes implementam, através da biblioteca *sequelize*, uma interface ORM (*Object-Relational Mapping*) para acessar o banco de dados Postgresql, evitando a necessidade de construir consultas SQL diretamente. O módulo *Connections* é responsável por encapsular os dados do WebSocket do cliente. Ele também implementa métodos como *destroyConnection()* e *sendMessage()*, em vez de lidar diretamente com o WebSocket, prevenindo que alguns erros aconteçam, tornando o código mais robusto. Por fim, o módulo *Controller* mantém a listas de todas as conexões separadas de acordo com os dois tipos de clientes: ônibus e parada. Métodos como *subscribeBus()* e *sendToAllStops()* são implementados aqui.

4.1.2 Modelagem do banco de dados relacional

Para criar o banco de dados do servidor, deve-se pensar nas entidades do sistema e no seus relacionamentos. Pela especificação, o servidor deverá manter conexões entre paradas e ônibus. Quando um ônibus informar sua posição para o servidor, é necessário que este possa encontrar as paradas pelas quais esse ônibus passará. Para especificar essa relação entre paradas e linhas (entre outras), o modelo das tabelas do banco de dados seguirá o formato definido pelo GTFS (*General Transit Feed Specification*), um padrão que define uma série de arquivos que modelam informações associadas ao transporte público tais como paradas, linhas, viagens e horários de partida e chegada (GOOGLE, 2016). Existem 13 tipos de arquivos definidos no padrão GTFS, mas apenas 6 deles são obrigatórios. Os arquivos que devem ser obrigatoriamente definidos são mostrados na Tabela 4.1 com uma breve descrição.

Tabela 4.1: Arquivos obrigatórios do formato GTFS

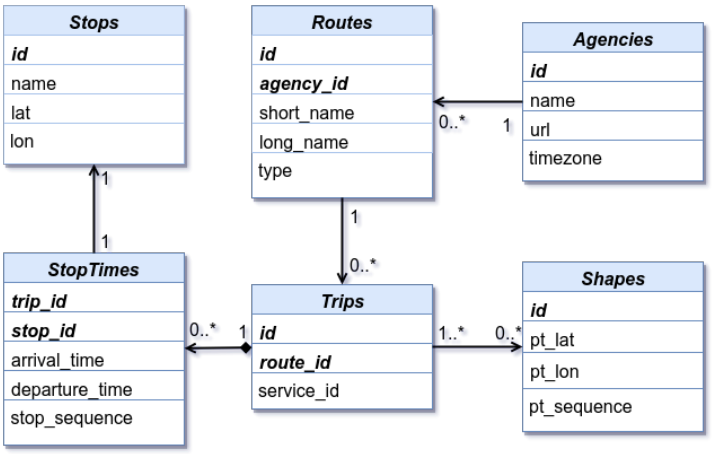
Arquivo	*O que descreve*
agency.txt	Empresas de transporte público.
stops.txt	Locais de embarque e desembarque de passageiros.
routes.txt	Linhas de transporte público.
trips.txt	Sequência de paradas percorridas por veículo de uma linha em um determinado horário.
stop_times.txt	Horários (estáticos) de partida e chegada dos veículos em uma parada.
calendar.txt	Dias da semana em que um serviço determinado é válido.

Fonte: (GOOGLE/TRANSIT, 2017)

A escolha desse padrão se deve ao fato de que o modelo GTFS é amplamente utilizado em aplicações relacionadas ao transporte público e pelo fato de haver arquivos GTFS do mundo todo disponíveis em bases de dados *online* como em <http://transitfeeds.com/feeds>. Neste projeto, o banco de dados será populado com os dados contidos nos arquivos GTFS modelados para o transporte público de Porto Alegre disponíveis no portal de dados abertos de Porto Alegre <http://datapoa.com.br/dataset/gtfs>. Nesses arquivos estão definidos 5547 paradas e 410 linhas de ônibus da cidade de Porto Alegre.

Cada tabela do banco de dados foi criada de forma análoga a um arquivo GTFS, contendo os mesmos campos e dados e mantendo os relacionamento estabelecido nos arquivos. O modelo relacional que representa o banco de dados criado é mostrado na Figura 4.2.

Figura 4.2: Modelo relacional de entidades.



Fonte: elaborada pelo autor.

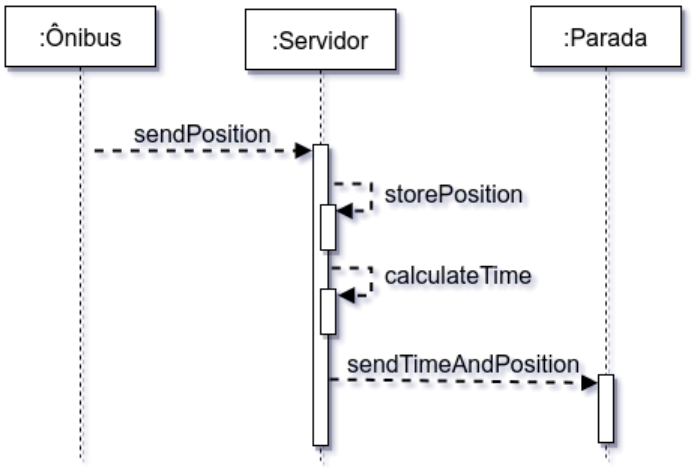
4.1.3 Modelagem do banco de dados não relacional

O banco de dados não relacional tem a única tarefa de armazenar as informações enviadas pelos ônibus. Por este motivo, ele é extremamente simples e possui apenas uma coleção chamada *BusData* contendo os campos: id do ônibus, id da rota, latitude e longitude do ônibus e *timestamp*.

4.1.4 Mensagens entre os ônibus

A trocas de mensagens entre os ônibus é feita através de WebSockets. Há basicamente dois tipos de mensagens: *subscribeBus* e *sendPosition*. A mensagem de *subscribeBus* informa o id do ônibus e o id da sua linha. Com esses dados, o servidor coloca o ônibus na lista de conexões e associa essa conexão com a linha enviada. Um diagrama de sequência para a mensagem *sendPosition* é mostrado na Figura 4.3. A mensagem *sendPosition* envia as coordenadas geográficas - latitude e longitude - atuais do ônibus que são armazenadas no servidor MongoDB. Em seguida, o servidor utiliza o algoritmo para estimar o tempo de chegada em cada uma das paradas na rota do ônibus. O servidor envia, então, uma mensagem contendo o tempo estimado (diferente para cada parada) e a posição do ônibus para os dispositivos nas paradas.

Figura 4.3: Diagrama de sequência da mensagem sendPosition.



Fonte: elaborada pelo autor.

4.2 Implementação do dispositivo WoT

Serão apresentados nesta seção os detalhes do desenvolvimento do dispositivo WoT para a parada de ônibus. Na primeira parte, será mostrado as tecnologias e a arquitetura final do dispositivo, incluindo sua interface RESTful. Na segunda parte, serão exibidos os passos e o resultado final da interface gráfica a ser exibida em um monitor na parada de ônibus.

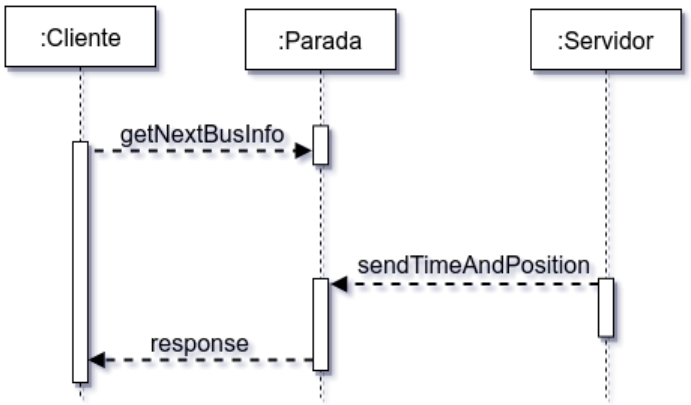
O dispositivo foi implementado em um mini computador Raspberry PI 3 Model B utilizando o sistema operacional Raspbian. O Raspberry PI preenche diversos requisitos de projeto como utilização de sistema operacional GNU/Linux, inclui *chip* para processamento gráfico necessário para a exibir as informações no monitor e pode se conectar a Internet através de WiFi, Ethernet ou mesmo Internet móvel como 3G através de um módulos disponíveis no mercado. Para a implementar a aplicação, utilizou-se, como no servidor, a plataforma Node.js v6.10.3.

O dispositivo utiliza um pequeno banco de dados Sqlite3 para guardar informações internas da parada como ID, nome e posição geográfica, assim como as linhas de ônibus obtidas através do servidor.

A aplicação expõe uma interface RESTful pública que se baseia no modelo WoT. Apesar de não ter sensores ou atuadores, a parada contém propriedades dinâmicas: as informações de posição e tempo de chegada dos ônibus que estão em rota até ela. É interessante pensar que a parada tem um "sensor virtual" de localização de ônibus. Com isto, a parada define propriedades básicas que podem ser consultadas pela interface RESTful

como nome, *id*, latitude, longitude e também uma lista fixa de linhas de ônibus que passam por ela. Também foi implementado uma interação através de um modelo *publish/subscribe* onde clientes podem, através de WebSockets, se inscreverem para obter o tempo estimado do próximo ônibus de uma determinada linha. Essa interação se dá através da mensagem *getNextBusInfo*, cujo argumento é o id de uma linha de ônibus.

Figura 4.4: Diagrama de sequência da mensagem getNextBusInfo.



Fonte: elaborada pelo autor.

O funcionamento dessa interação é dada da seguinte forma: o cliente envia a mensagem *getNextBusInfo* para se registrar no evento, conforme mostrado no diagrama de sequência da Figura 4.4. A parada espera pelo recebimento da mensagem *sendTimeAndPosition* vinda do servidor (que é mostrada na Figura 4.3). Ao receber a mensagem, o dispositivo WoT verifica quais clientes se inscreveram para aquela mensagem e envia a resposta contendo o tempo estimado para a chegada do ônibus.

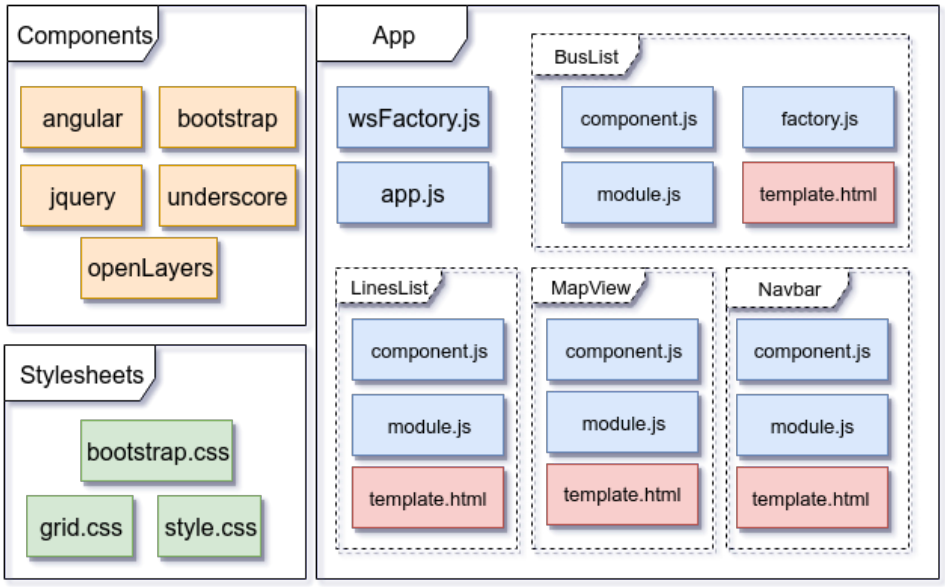
4.2.1 Interface

Para a criação da interface gráfica, foi implementado uma solução para um aplicativo Web que ficará em execução no browser Chromium na parada.

Para isto, a interface foi desenvolvida com os padrões Web HTML5 e CSS3. Foram utilizadas a bibliotecas *Angular* para criar a lógica da aplicação, *Bootstrap* que fornece diversas classes em CSS para facilitar o desenvolvimento de interfaces, como gerar linhas e colunas de forma simples, além de oferecer alguns estilos CSS pré-definidos para diversos componentes. Para implementar o mapa, foi utilizado a biblioteca *OpenLayers* que fornece uma API para manipulação de mapas utilizando dados do *OpenStreetMaps* (OPENLAYERS, 2017).

A Figura 4.5 mostra a organização dos componentes da interface. As bibliotecas

Figura 4.5: Organização dos componentes da interface gráfica.

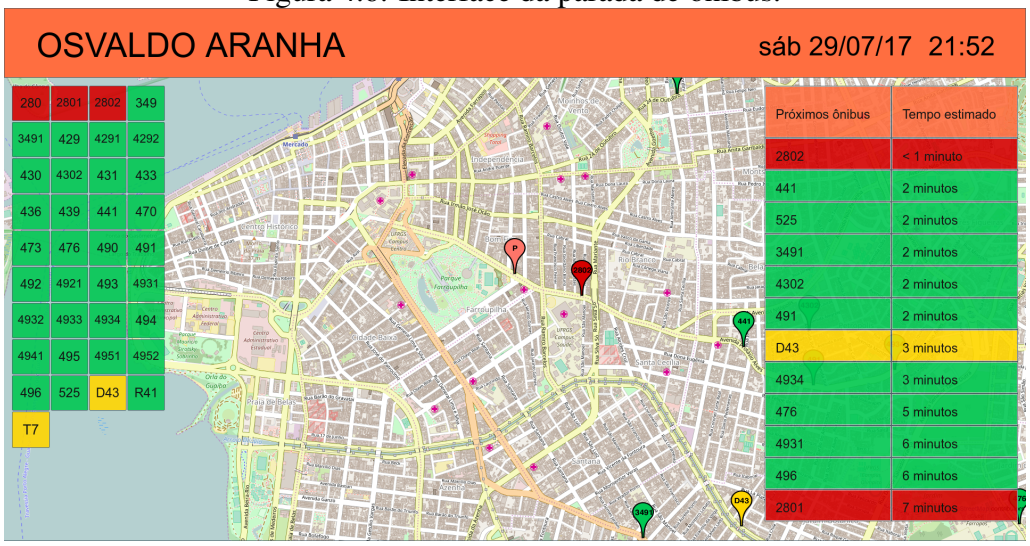


Fonte: elaborada pelo autor.

utilizadas no desenvolvimento da aplicação estão representadas no bloco *Components*. Em *Stylesheets* estão os arquivos CSS que customizam o estilo da interface como cores, formatos e posições. No bloco *App* estão os arquivos Javascript e os *templates* HTML para aplicação desenvolvida utilizando o *framework* Angular. O arquivo *app.js* é o arquivo principal que é responsável por controlar e invocar os outros módulos. O arquivo *wsFactory.js* é responsável por implementar o serviço de WebSockets. Os WebSockets são utilizados apenas para comunicação local com o servidor do próprio dispositivo. *BusList*, *LinesList*, *MapView* e *Navbar* são os blocos que geram a lógica e a interface gráfica de cada módulo. Os arquivos *template.html* são populados com dados a partir dos arquivos *module.js* de cada bloco e são inseridos dinamicamente no arquivo HTML principal. *BusList* implementa a lista dinâmica de ônibus, mostrando o tempo de chegada. *LinesList* implementa a lista estática de linhas de ônibus relacionadas à parada. *MapView* é responsável por exibir o mapa na tela, assim como controlar a posição dos marcadores da parada e dos ônibus. Por fim, *Navbar* é responsável por renderizar o painel no topo da tela com o nome da parada e informações de data e hora.

A interface final é vista na Figura 4.6. Nela, é possível visualizar à esquerda as linhas de ônibus associadas à parada. No lado direito, a lista de ônibus com as informações de tempo estimado de chegada é exibida na ordem de tempo de chegada crescente. No mapa, é possível ver os ônibus mostrados na lista. Repare que um ônibus da linha 2802

Figura 4.6: Interface da parada de ônibus.



Fonte: elaborada pelo autor.

está bem próximo da parada e o tempo estimado é para menos de 1 minuto. As cores das linhas, vermelho, amarela e verde, são provenientes do campo *color* do arquivo routes.txt.

Na Figura 4.7 é apresentada uma foto com o dispositivo WoT exibindo a interface gráfica no monitor ao fundo.

Figura 4.7: Protótipo do dispositivo WoT conectado ao monitor.

Fonte: elaborada pelo autor.

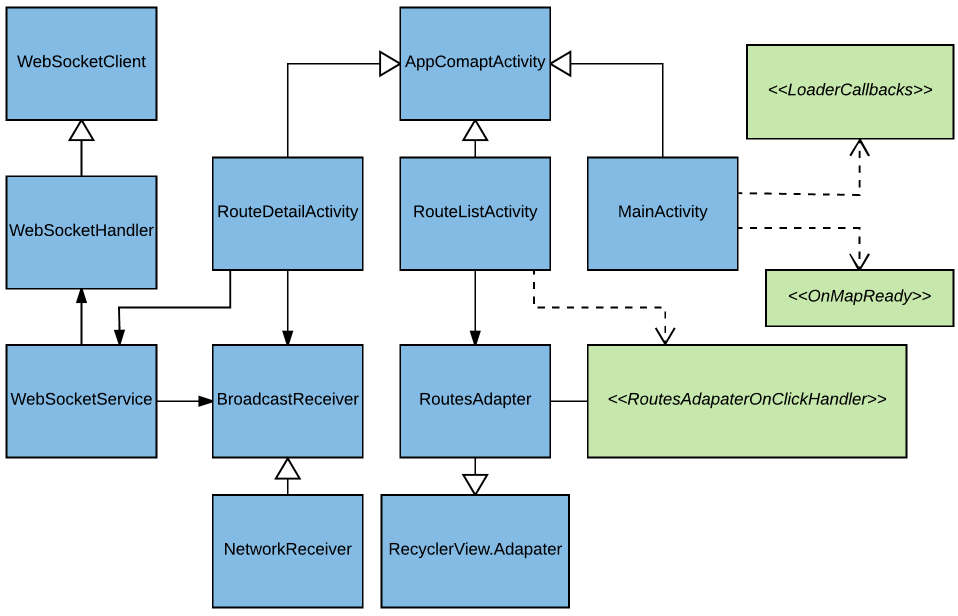
4.3 Implementação do aplicativo Android

Nesta seção, serão mostrados os passos no desenvolvimento da aplicação Android. Primeiramente será mostrada o diagrama de classes do aplicativo, explicando os principais blocos lógicos do sistema. Depois, será mostrado algumas telas exibindo a interface final do aplicativo.

4.3.1 Diagrama de classes do aplicativo Android

O diagrama de classes para o aplicativo é mostrado na Figura 4.8. Pode-se notar que exitem três *Activities* que herdam da classe *AppCompatActivity*. A classe *MainActivity* implementa a tela inicial da aplicação que apresenta o mapa com as paradas. Esta atividade utiliza *Loaders* para realizar requisições HTTP de forma assíncrona para o servidor. Estas requisições são feitas para descobrir quais paradas existem e quais estão conectadas.

Figura 4.8: Diagrama de classes do aplicativo.



Fonte: elaborada pelo autor.

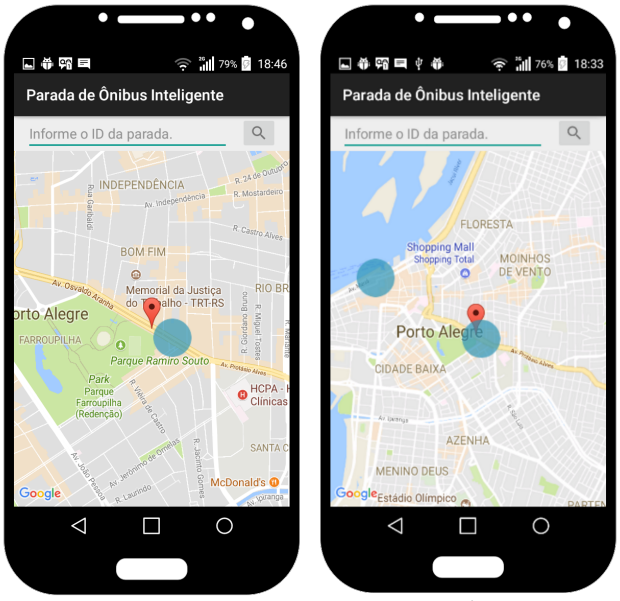
A classe *RoutesListActivity* implementa a atividade que exibe a lista de linhas de ônibus da parada selecionada. Para isso, ela implementa um componente chamado *RecyclerView* que é populado de forma dinâmica e eficiente com items definidos pela classe *RoutesAdapter*. Ao selecionar um elemento da lista, a atividade invocada é a implemen-

tada na classe *RoutesDetailActivity*. Essa classe inicia a conexão com WebSockets com as paradas de ônibus através do serviço implementado na classe *WebSocketService*, que é responsável por manter a conexão aberta em *background*. A classe *WebSocketHandler* encapsula o WebSocket proveniente da classe *WebSocketClient*.

4.3.2 Interface

A Figura 4.9 mostra a tela inicial do aplicativo com a) visão inicial e b) com visão distanciada. O ícone mostra a posição da pessoa no momento (para isto é necessário que o *smartphone* esteja com a opções de localização habilitadas). Os círculos mostram a posição das paradas com os dispositivos WoT.

Figura 4.9: Tela inicial do aplicativo.



a) b)

Fonte: elaborada pelo autor.

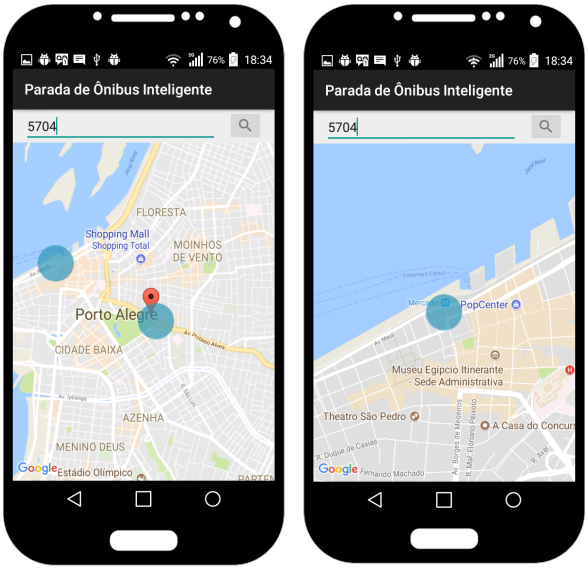
O usuário pode, também, pesquisar pelo ID de uma parada no campo de texto no topo da tela. Isto faz com que o mapa centralize na parada selecionada com o *zoom* original. Na Figura 4.10a o cliente busca pelo ID 5704 e na Figura 4.10b tem a visão da tela centralizada na parada com ID correspondente.

Ao clicar em um dos um círculo, o usuário é redirecionado a uma tela de seleção de linhas. A Figura 4.11 mostra as telas de seleção para as paradas 5704 e 2111.

Ao selecionar uma rota, a interface que mostra o tempo estimado de chegada do ônibus é exibida. A Figura 4.12 mostra duas capturas de tela. Na primeira, a imagem mostra que o tempo estimado para chegada do ônibus da linha 2802 até a parada 2111 é

de 10 minutos, enquanto que na segunda, a linha selecionada D43 deverá levar 11 minutos aproximadamente para chegar a mesma parada.

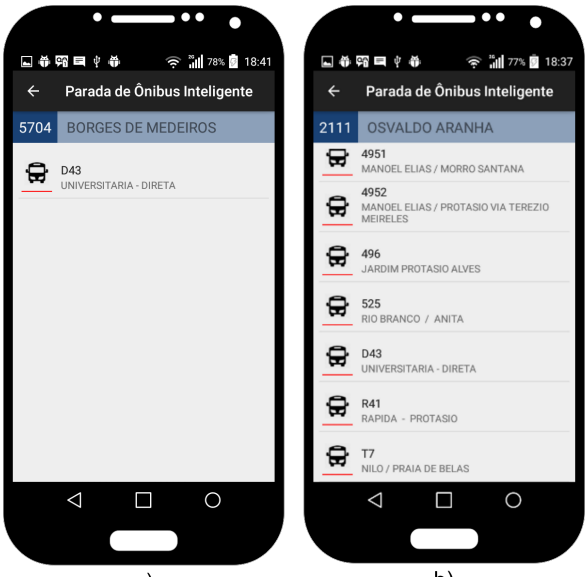
Figura 4.10: Busca de paradas por ID no mapa.



a) b)

Fonte: elaborada pelo autor.

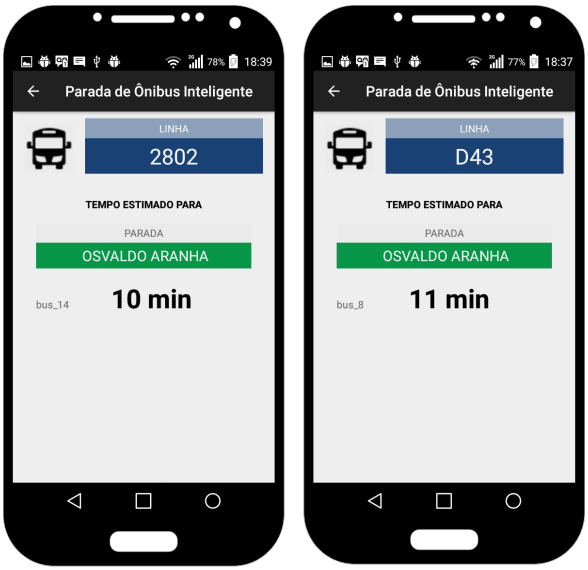
Figura 4.11: Telas de seleção de linhas.



a) b)

Fonte: elaborada pelo autor.

Figura 4.12: Telas com o tempo estimado dos ônibus.



a) b)

Fonte: elaborada pelo autor.

5 AVALIAÇÃO

Este capítulo tem por objetivo realizar a avaliação para verificar o correto comportamento do sistema desenvolvido.

5.1 Metodologia

A metodologia de testes irá envolver a interação entre o servidor, dispositivo WoT e o aplicativo Android de forma automatizada. De forma que os testes se aproximem de um cenário realista, será necessário que muitos ônibus estejam mandando mensagens em tempo real. Com isso em mente, uma aplicação que simula os dispositivos dos ônibus foi implementada e será descrita a seguir.

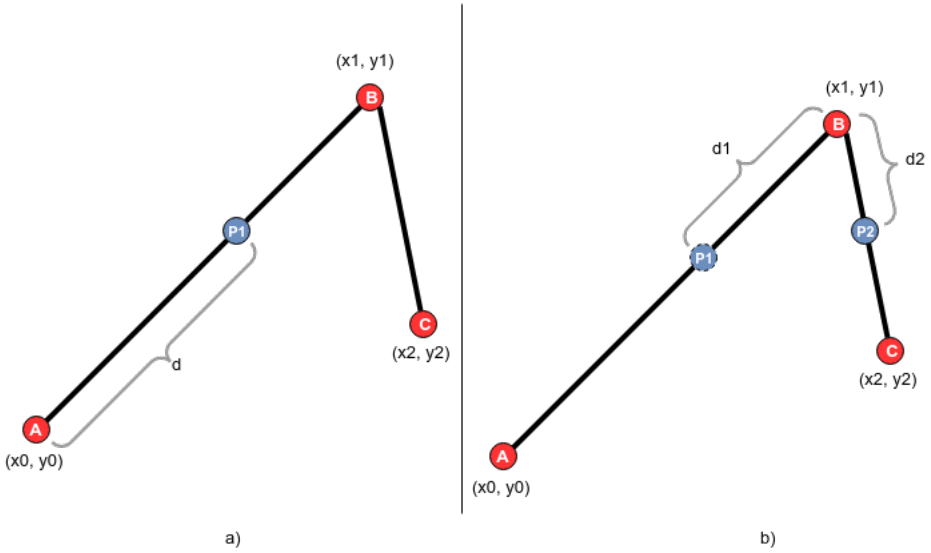
5.2 Simulador

Para simular a mensagem de um ônibus é fácil, basta abrir uma conexão WebSocket com o servidor e mandar as coordenadas do GPS. No entanto, é necessário obter essas coordenadas que devem seguir rotas específicas para cada linha de ônibus. O algoritmo descrito a seguir, simula sinais de GPS ao longo de uma rota pré-definida. Estas rotas pré-definidas podem ser encontradas na tabela do banco de dados *Shapes* que corresponde ao arquivo *shapes.txt* do modelo GTFS. Esse arquivo lista uma série de pontos geográficos ordenados que representa o caminho que um ônibus percorre. Conhecendo o caminho, é possível gerar uma série de posições aleatórias que caem exatamente entre a reta que liga dois pontos consecutivos desta lista estática. Para mostrar como isso é feito, um diagrama é mostrado na Figura 5.1. Os pontos vermelhos representam os pontos estáticos, como aqueles definidos em *shapes.txt*. As retas representam a ligação entre dois pontos consecutivos.

Partindo do ponto *A*, para fazer com que um ônibus viaje uma distância aleatória *d*, deve-se calcular para que ponto da linha *AB* ele se moverá. A distância entre dois pontos em uma esfera é dada pela fórmula de Haversine:

$$D = 2 \times \arcsin\left(\sqrt{\sin^2\left(\frac{lon_2 - lon_1}{2}\right) + (lon_1)\cos(lon_2)\sin^2\left(\frac{lat_2 - lon_1}{2}\right)}\right) \quad (5.1)$$

Figura 5.1: Ponto aleatório sobre uma reta.



Fonte: elaborada pelo autor.

Onde *lat* e *lon* representam a latitude e longitude dos pontos.

Com isto, obtém-se um valor p que representa quão próximo o ônibus está do ponto B

$$p = \frac{d}{D} \tag{5.2}$$

Se o resultado de p for menor que 1, significa que o ônibus está dentro da reta AB. Nesse caso, são obtidas as novas coordenadas para o ônibus representadas na Figura 5.1 pelo ponto $P1$:

$$P_1 = (x^{'}, y^{'}) = (x_0 + p \times (x_1 - x_0), y_0 + p \times (y_1 - y_0)) \tag{5.3}$$

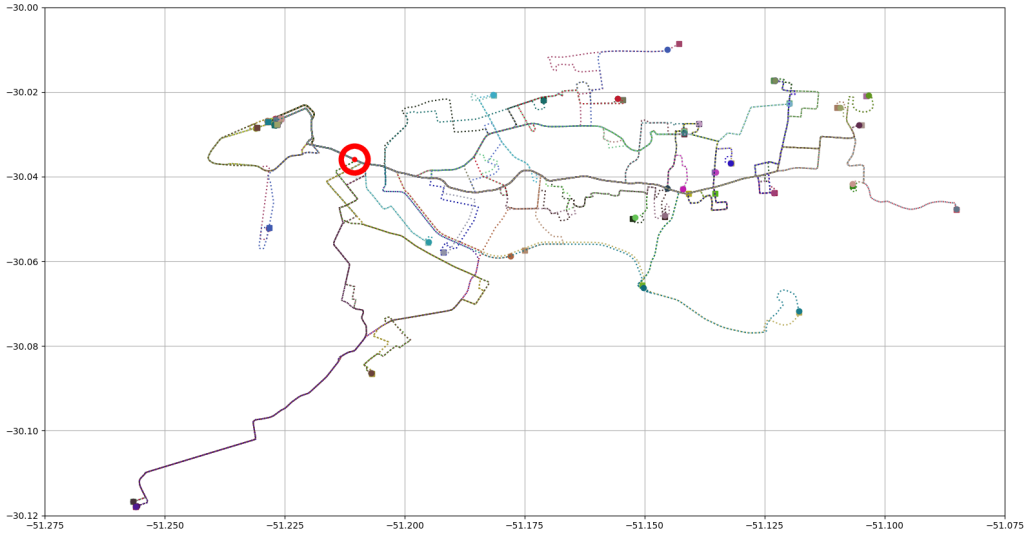
No próximo passo, exemplificado na Figura 5.1 b), o ônibus viaja uma distância $d = d1 + d2$ que é maior que a distância entre os pontos $P1$ e B, ou seja, $p > 1$. Logo, o ônibus "atravessou" o ponto B. Para corrigir isso, o ônibus é posicionado no ponto B e o algoritmo recomeça em direção ao ponto C, agora com $d = d2$. O algoritmo continua até o ônibus atingir o ponto final C.

O simulador gera ônibus com linhas aleatórias a cada 8 segundos. Cada ônibus criado calcula a posição no mapa como o algoritmo descrito e envia essa posição para o servidor. A distância que o ônibus "viaja" a cada interação depende do intervalo de envio das mensagens e da sua velocidade máxima permitida. Para este simulador, foi considerado que um ônibus pode viajar a no máximo 50 km/h = 14 m/s. O intervalo de envio de mensagens é de 5 segundos. Com isto, a distância máxima que um ônibus pode

viajar em 5 segundos é de *14 * 5 = 70 m*. Dessa forma, é gerado uma distância aleatória entre 0 e 70 a cada 5 segundos. Esse valor é inserido no algoritmo para calcular a posição atual do ônibus que é enviado para o servidor.

A Figura 5.2 mostra uma simulação para todos os ônibus das linhas que passam pela parada 2111. As linhas pontilhadas representam as rotas dos ônibus, os marcadores quadrados e redondos representam o início e o fim das linhas, respectivamente. O círculo maior representa a posição da parada no mapa.

Figura 5.2: Plotagem das posições dos ônibus simulados para a parada 2111.



Fonte: elaborada pelo autor.

5.3 Testes de interação

Com todos os componentes do sistema e o simulador desenvolvidos, será feito testes para verificar que todos os sistemas estão interagindo corretamente.

Para isto, serão escolhidas três paradas de ônibus para estarem conectadas ao mesmo tempo. O teste visa mostrar que as mensagens de ônibus estão chegando em todas as paradas ao mesmo tempo. Também será mostrado como o usuário pode escolher uma parada e se inscrever no evento para receber mensagens sobre os tempos de ônibus.

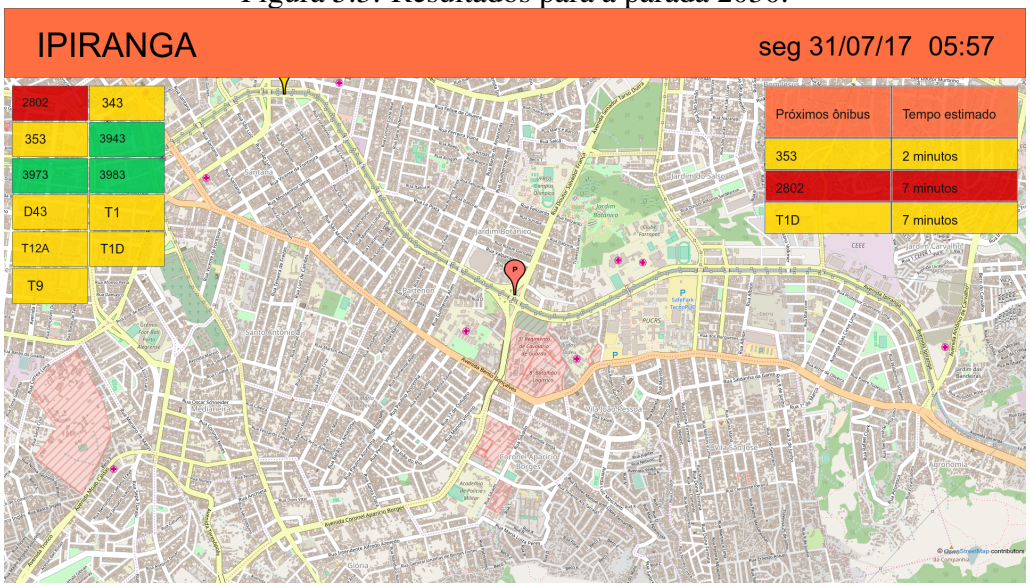
Os testes foram realizados em um computador com sistema operacional GNU/Linux, distribuição Ubuntu 16.04, com 8GB de memória e processador AMD FX-6300. O *smartphone* utilizado para os testes é um LG G2-mini como sistema operacional Android 5.0.2.

O dispositivo Raspbery PI será conectado e representará uma parada. As outras

paradas serão executadas no computador descrito. O servidor e o simulador também serão executados nesse computador.

As Figuras 5.3, 5.4 e 5.5 mostram as interfaces dos dispositivos conectados para três paradas escolhidas onde é possível ver os ônibus que estão a caminho.

Figura 5.3: Resultados para a parada 2030.



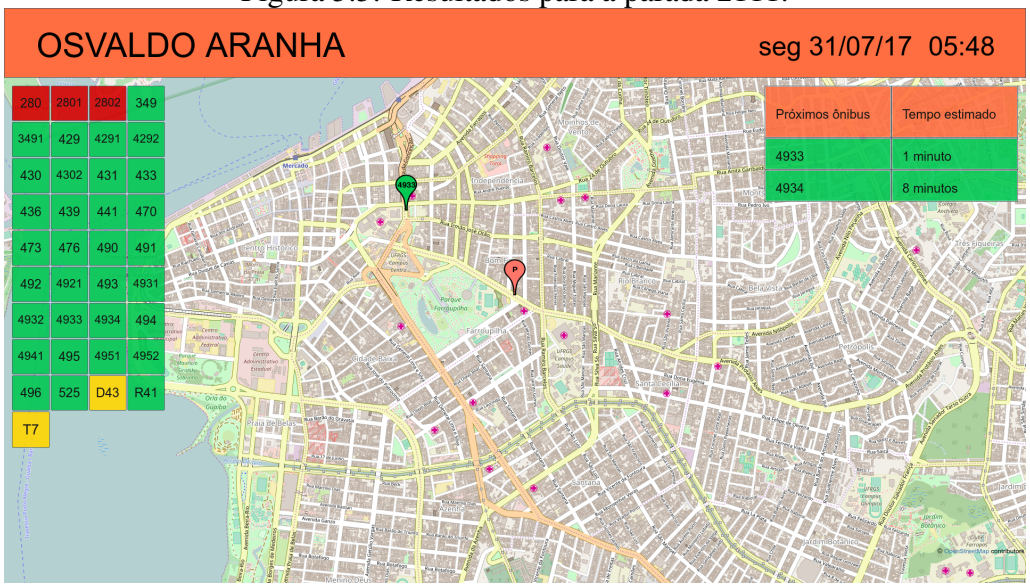
Fonte: elaborada pelo autor.

Figura 5.4: Resultados para a parada 3146.

Fonte: elaborada pelo autor.

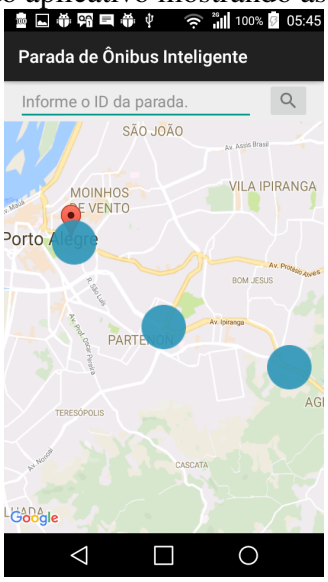
Após as paradas estarem operantes, e recebendo informações dos ônibus através do servidor, o aplicativo o Android foi iniciado para testar a conexão com as paradas. A tela inicial é mostrada na Figura 5.6. Nesta figura é possível ver as três paradas posicionadas pelo mapa. O próximo teste é selecionar linhas de ônibus e comparar o resultado

Figura 5.5: Resultados para a parada 2111.



Fonte: elaborada pelo autor.

Figura 5.6: Tela inicial do aplicativo mostrando as três paradas conectadas.



Fonte: elaborada pelo autor.

obtido na interface do aplicativo com a interface do dispositivo.

Na Figura 5.7 são mostradas as linhas escolhidas para cada uma das três paradas. Na Figura 5.7a é mostrado os detalhes para a linha 2802 da parada Ipiranga, que informa que o ônibus chegará em 7 minutos. Na Figura 5.7b, a linha escolhida, T10, é exibida para a parada Agronomia Bento Gonçalves com um tempo estimado de chegada de 3 minutos. Por último, na Figura 5.7c, é mostrado que o próximo ônibus da linha 4934 chegará em 8 minutos na parada Osvaldo Aranha.

Comparando esses resultados com aqueles das Figuras 5.3, 5.4 e 5.5, pode averiguar que os resultados obtidos foram os mesmos para cada parada e para o aplicativo.

Figura 5.7: Resultados da aplicação Android testado com três paradas.

Fonte: elaborada pelo autor.

5.4 Considerações finais

Com base apenas nos testes realizados, pode-se concluir que o sistema funciona de acordo com o esperado. A interação entre os três componentes: servidor, dispositivo WoT e aplicativo móvel, com o uso do simulador, mostrou resultados esperados pela especificação do sistema. As aplicações do dispositivo WoT conseguiu exibir corretamente as informações dos ônibus na tela e o aplicativo móvel foi capaz se conectar em qualquer parada e obter informações do tempo de chegada dos ônibus compatíveis com aqueles disponibilizados na interface do dispositivo WoT.

6 CONCLUSÃO

Neste trabalho, foi apresentado o conceito de Web das Coisas, que visa resolver alguns dos problemas enfrentados pela Internet das Coisas atual. Foi visto que padrões Web podem ser utilizados para implementar um modelo em que dispositivos podem se comunicar através de uma interface bem definida e por padrões abertos.

O desenvolvimento de Cidades Inteligentes com a integração de diversas tecnologias propicia diversos benefícios para a cidade e seus cidadãos. Entre esses benefícios, foi citado o conceito de mobilidade urbana inteligente, tratando de ideias para tornar o trânsito mais eficiente e facilitar a vida dos passageiros de transporte público e privado.

Com base e interesse nesses conceitos, neste trabalho foi especificado, implementado e avaliado uma solução baseada no modelo de Web das Coisas para uma Parada de Ônibus Inteligente, visando a integração com uma Cidade Inteligente, com o objetivo de fornecer informações úteis à população através de diversas tecnologias.

No projeto foram especificados os requisitos funcionais e não-funcionais para a implementação do sistema. Baseado nesses requisitos, foi implementado um servidor HTTP e de WebSockets, um dispositivo WoT e um aplicativo para *smartphones* Android.

O servidor foi implementado no sistema operacional GNU/Linux, distribuição Ubuntu 16.04, utilizando o ambiente de execução Javascript Node.js v6 e os bancos de dados Postgresql e MongoDB.

O dispositivo WoT foi programado para um mini computador Raspberry Pi 3 Model B com sistema operacional Raspbian Jesse, sendo executado, também, sobre o Node.js v6. As bibliotecas Angular e Bootstrap foram utilizadas para implementar a interface gráfica que exibe as informações sobre os ônibus em um browser Chromium. Para renderizar o mapa na interface, foi utilizada a biblioteca *OpenLayers* com os dados da *OpenStreetMaps*.

O aplicativo Android foi implementado com o uso do Android SDK 25 e da biblioteca Java-WebSockets, podendo ser executado em *smartphones* com a versão 4.1, ou superior, do sistema operacional Android.

Para realizar os testes, foi implementado um aplicação que simula ônibus em movimento, com base nos dados do arquivo *shapes.txt* do modelo GTFS que descrevem os itinerários das linhas de ônibus. Os dados de GPS simulados foram enviados para o servidor, onde foi calculado o tempo estimado de chegada dos ônibus às paradas, e repassado para os dispositivos WoT. Os clientes Android se conectaram, então, aos dispositivos WoT

para obter informações sobre os próximos ônibus.

A avaliação realizada demonstrou que o sistema opera conforme esperado, levando informações para paradas de ônibus e implementando um modelo Web que poder ser acessado por *smartphones* e outros dispositivos.

O projeto, no entanto, é capaz de receber várias melhorias como a implementação de um algoritmo de estimativa de tempo mais preciso, uso de protocolos de criptografia nas mensagens, um sistema de autenticação para os ônibus para tornar o projeto mais seguro e a aplicação de mecanismos relacionados à Web Semântica.

REFERÊNCIAS

ASHTON, K. That 'internet of things' thing. In: **RFID Journal**. [s.n.], 2009. Available from Internet: <http://www.rfidjournal.com/articles/view?4986>.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. In: **Computer Networks**. [S.l.: s.n.], 2010.

BERNERS-LEE, T.; FIELDING, R. T.; MASINTER, L. Uniform resource identifier (uri): Generic syntax (rfc 3986). In: . [S.l.: s.n.], 2005.

BIKEPOA. **Sobre o Bike PoA**. 2016. Available from Internet: <http://www2.portoalegre.rs.gov.br/eptc/default.php?p_secao=228>.

CHOURABI, H. et al. Understanding smart cities: An integrative framework. In: **System Science (HICSS), 2012 45th Hawaii International Conference on**. [S.l.: s.n.], 2012. p. 2289–2297. ISSN 1530-1605.

CITY, A. A. S. Smart traffic management. In: . [s.n.], 2016. Available from Internet: <https://amsterdamsmartcity.com/projects/smart-traffic-management>.

CITY, B. S. Smart mobility. In: **Smart City Areas**. [s.n.], 2015. Available from Internet: <http://smartcity.bcn.cat/en/new-bus-network.html>.

CITY, B. S. Smart traffic lights. In: **Smart City Areas**. [s.n.], 2015. Available from Internet: <http://smartcity.bcn.cat/en/smart-traffic-lights.html>.

DUNKELS, A.; VASSEUR, J. Ip for smart objects, internet protocol for smart objects (ipso) alliance, white paper #1. In: . [s.n.], 2008. Available from Internet: <http://dunkels.com/adam/dunkels08ipso.pdf>.

DUQUENNOY, S.; GRIMAUD, G.; VANDEWALLE, J. J. The web of things: Interconnecting devices with high usability and performance. In: **Embedded Software and Systems, 2009. ICESS '09. International Conference on**. [S.l.: s.n.], 2009. p. 323–330.

FIELDING, R. T. Architectural styles and the design of network-based software architectures (ph.d.). In: . University of California, 2000. Available from Internet: <http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>.

GEA, T. et al. Smart cities as an application of internet of things: Experiences and lessons learnt in barcelona. In: **Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013 Seventh International Conference on**. [S.l.: s.n.], 2013. p. 552–557.

GIFFINGER, R. et al. Smart cities: Ranking of european medium-sized cities. In: . [s.n.], 2007. Available from Internet: <http://www.smart-cities.eu/download/smart_cities_final_report.pdf>.

GOOGLE. **GTFS Static Overview**. Google, 2016. Available from Internet: <https://developers.google.com/transit/gtfs/>.

GOOGLE/TRANSIT. **General Transit Feed Specification Reference**. 2017. Available from Internet: <https://github.com/google/transit/blob/master/gtfs/spec/en/reference.md>.

GOVERNMENT, S. M. Smart bus stops in seoul. In: . [s.n.], 2010. Available from Internet: <http://english.seoul.go.kr/smart-bus-stops-in-seoul/>.

GUINARD, D. et al. From the internet of things to the web of things: Resource-oriented architecture and best practices. In: **Architecting the Internet of Things**. [S.l.]: Springer, 2011.

GUINARD, D. D.; TRIFA, V. M. Building the web of things. In: . [S.l.]: Manning, 2016.

IDC. Internet of things spending forecast to grow 17.9manufacturing, transportation, and utilities investments, according to new idc spending guide. In: . [s.n.], 2017. Available from Internet: <http://www.idc.com/getdoc.jsp?containerId=prUS42209117>.

OPENLAYERS. **Layer source for the OpenStreetMap tile server.** 2017. Available from Internet: <http://openlayers.org/en/latest/apidoc/ol.source.OSM.html>.

SCHAFFERS, H. et al. Smart cities and the future internet: Towards cooperation frameworks for open innovation. In: **The Future Internet: Future Internet Assembly 2011: Achievements and Technological Promises**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 431–446. ISBN 978-3-642-20898-0. Available from Internet: <http://dx.doi.org/10.1007/978-3-642-20898-0_31>.

TIMES, T. T. M. 'smart' maps appear at moscow bus stops. In: . [s.n.], 2013. Available from Internet: <http://www.themoscowtimes.com/news/article/smart-maps-appear-at-moscow-bus-stops/473955.html>.

TOMA, I.; SIMPERL, E.; HENCH, G. A joint roadmap for semantic technologies and the internet of things. In: **Proceedings of the Third STI Roadmapping Workshop**. [S.l.: s.n.], 2009.

TRANSPORT, S. S. C. Smart traffic management. In: . [s.n.], 2016. Available from Internet: <http://www.smartertransport.uk/smart-traffic-management/>.

UCKELMANN, D.; HARRISON, M.; MICHAHELLES, F. Introduction, background and initial visions. In: **Architecting the Internet of Things**. [S.l.]: Springer, 2011.

UN. World's population increasingly urban with more than half living in urban areas. In: . [s.n.], 2014. Available from Internet: <http://www.un.org/en/development/desa/news/population/world-urbanization-prospects-2014.html>.

VERMESAN, O. et al. Internet of things strategic research and innovation agenda. In: **Internet of Things From Research and Innovation to Market Deployment**. [S.l.]: River, 2014.

W3C. All standards and drafts. In: . [s.n.], 2017. Available from Internet: <https://www.w3.org/TR/>.

W3C. Facts about w3c. In: . [s.n.], 2017. Available from Internet: <https://www.w3.org/Consortium/facts.html>.

W3C. Web of things wg. In: . [s.n.], 2017. Available from Internet: <https://labs.w3.org/unitas/?g=95969>.

WG, W. W. **WoT Current Practices**: Unofficial draft 07 july 2017. 2017. Available from Internet: <http://w3c.github.io/wot/current-practices/wot-practices.html>.

WIKIPEDIA. Bicycle sharing system. In: . [s.n.], 2016. Available from Internet: <https://en.wikipedia.org/wiki/Bicycle-sharing_system>.

WIKIPEDIA. Smart cities. In: . [s.n.], 2016. Available from Internet: <https://en.wikipedia.org/wiki/Smart_city\#Amsterdam>.

ZANELLA, A. et al. Internet of things for smart cities. In: **IEEE Internet of Things Journal, Vol. 1, No. 1, February 2014**. [S.l.: s.n.], 2014. p. 22.

Parada de Ônibus Inteligente: Uma Aplicação para Cidades Inteligentes Baseado no Conceito de Web das Coisas

Marcos Kintschner[1], **Alexandre S. Carissimi**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{mkintschner,asc}@inf.ufrgs.br

Abstract. *The Internet of Things aims to interconnect a multitude of heterogeneous devices around us, raising issues related to scalability and compatibility. The Web of Things is suggested as a possible solution aiming to facilitate connectivity and application development through the reuse of Web technologies. When concerns, demands and global trends are added to these new technologies, the idea of Smart City is created, in which issues such as sustainability, quality of life and urban mobility are optimized. Based on this concept, it will be presented a proposal for Bus Stop for Smart Cities, implemented under the approach of Web of Things, aimed at providing real-time informations about the buses that pass by.*

Resumo. *A Internet das Coisas pretende interconectar uma infinidade de dispositivos heterogêneos ao nosso redor, levantando problemas relacionados a escalabilidade e compatibilidade. A Web das coisas é apontada como possível solução, visando facilitar a conectividade e o desenvolvimento de aplicações por meio do reaproveitamento das tecnologias Web. Quando preocupações, demandas e tendências globais são somadas a essas novas tecnologias, é criada a ideia de Cidade Inteligente, em que questões como sustentabilidade, qualidade de vida e mobilidade urbana são otimizadas. Baseado neste conceito será apresentado uma proposta de Parada de Ônibus para Cidades Inteligentes, implementado sob a abordagem de Web das Coisas, cujo objetivo é fornecer diversas informações em tempo real a respeito os ônibus que passam pelo local.*

1. Introdução

A Internet surgiu em laboratórios de pesquisa de universidades americanas no ínicio da década de 1970 e, desde então, foi ampliada por diversas instituições espalhadas pelo mundo com o desenvolvimento de tecnologias, aplicações, protocolos e serviços. Hoje, extrapolando seu propósito inicial, a Internet atinge todos os continentes, interconectando computadores e pessoas por toda a parte. Com a evolução de sistemas embarcados, sensores e tecnologias de comunicação sem fio nos últimos anos, a Internet encontra-se agora a um passo de uma importante transformação.

A chamada Internet das Coisas surge como uma mudança de paradigma onde a conexão à Internet torna-se possível não apenas a qualquer momento ou em qualquer lugar, mas também por meio de qualquer "coisa". Uma imensidão de coisas interconectadas pelo mundo deve movimentar um mercado de US$1,7 trilhão em 2020 de acordo com a

projeção da IDC (*International Data Corporation*) [IDC 2015]. Um número cada vez maior de dispositivos devem ser inseridos na vida cotidiana das pessoas e na indústria, alterando a forma como percebemos e interagimos com objetos ao nosso redor.

Diversas soluções para a viabilização da Internet das Coisas se encontram disponíveis atualmente. No entanto, muitas dessas soluções se apresentam de forma heterogênea. Como consequência, muitos dispositivos tendem a ser incompatíveis entre si, tornando complicado o acesso e a identificação uniforme na Internet das Coisas. À luz desse problema, a Web das Coisas emerge implantando tecnologias Web extensamente utilizadas juntamente com a aplicação do estilo REST às coisas [Guinard et al. 2011]. Identificadas unicamente por URIs (*Uniform Resource Identifier*), essas podem ser acessadas por um navegador ou mesmo interagir com outras coisas integradas à Web, utilizando para isto uma interface uniforme padrão.

As mudanças trazidas pela Internet e suas novas tecnologias, juntamente com novas tendências globais, tem feito cidades ao redor mundo debater questões relacionadas à sustentabilidade, utilização de recursos, poluição e qualidade de vida da população em geral. Esses assuntos, quando associados às novas tecnologias de comunicação e informação tem dado origem ao termo *Cidades Inteligentes*, amplamente discutidos atualmente. Gestão de recursos, mobilidade, saúde, segurança e áreas públicas inteligentes são alguns dos tópicos de maior interesse nestas cidades.

Neste trabalho, será proposto um sistema baseado no conceito de Web das Coisas para uma *Parada de Ônibus Inteligente*. O sistema contará com dispositivos acoplados a monitores projetados para ficar em áreas públicas (paradas de ônibus) de uma cidade inteligente. Esses monitores fornecerão na tela informações em tempo real sobre os ônibus que passam por ali. As informações mostradas podem incluir o tempo de chegada dos ônibus (à partir de dados de GPS enviados por esses), além de avisos ou notícias transmitidos a partir de uma central.

2. Internet das Coisas

A Internet das Coisas é um paradigma de comunicação relativamente recente que antecipa um mundo de coisas interconectadas ao nosso redor através da Internet e por meio de tecnologias diversas [Zanella et al. 2014]. Apesar de ser um pensamento geral quando se fala em Internet das Coisas (dado o seu próprio nome), não há uma definição única para ela, e as diversas definições existentes na literatura podem acabar tornando o seu real significado confuso devido às diferenças que tendem a surgir de autor para autor. Tais divergências se devem à forma como as partes interessadas escolhem abordar o tema, em geral guiadas por interesses particulares, ou motivadas pela área de conhecimento, ou tecnologias que dominam, dependendo também, por vezes, do contexto da época em que o assunto foi tratado.

O termo "Internet das Coisas" foi usado pela primeira vez em 1999 pelo britânico Kevin Ashton que, então, trabalhava na *Auto-ID Labs*, uma rede de laboratórios de pesquisa acadêmica focada no desenvolvimento de redes RFID (*Radio Frequency IDentification*). O nome foi utilizado para fazer uma ligação entre dois termos de interesse crescente na época: RFID e Internet ''[Ashton 2009].

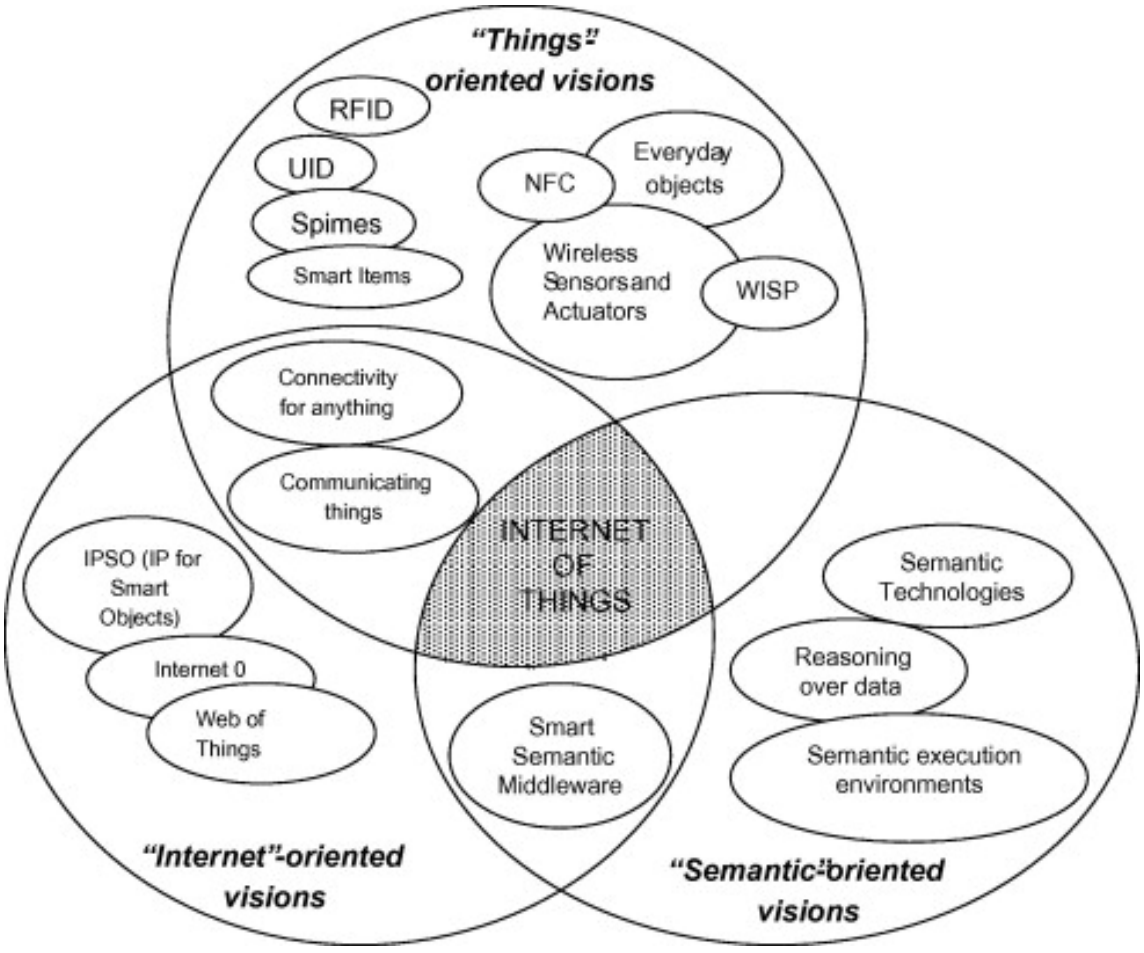
Desde então, o termo tem sido amplamente usado por pesquisadores para descrever a interação do mundo real com o mundo virtual através de tecnologias de identificação,

sistemas de localização em tempo real, sensores e atuadores [Uckelmann et al. 2011].

Algums centros de pesquisa têm se empenhado em tratar de maneira mais geral o conceito de Internet das Coisas. O IERC (*European Research Cluster for Internet of Things*) define a Internet das Coisas como: "um paradigma que considera que coisas possam se conectar a qualquer momento, em qualquer lugar, a qualquer coisa, utilizando, idealmente, qualquer rede ou serviço, e que sejam capazes de interagir e colaborar uns com os outros afim de criar novos serviços e aplicações." [Vermesan et al. 2014]. Já a ITU-T (*International Telecommunication Union - Telecommunication Standardization Sector*) apresenta o conceito de Internet das Coisas como: "Uma infraestrutura global que fornece serviços avançados baseados na interconexão de objetos físicos ou virtuais que possuam a capacidade de serem identificados e integrados em redes de comunicação."

Dada a forma decentralizada como o tema Internet das Coisas foi inicialmente tratado, esta pode ser compreendida a partir de três diferentes perspectivas: orientada a coisas, orientada a Internet e orientada a semântica, onde a Internet das Coisas parece resultar da composição das tecnologias abordadas nas três visões, conforme mostrado na Figura 1 [Atzori et al. 2010].

Figura 1. Diferentes visões do Internet das Coisas.



Fonte: [Atzori et al. 2010].

Na visão orientada a coisas, o foco é no desenvolvimento de dispositivos, sensores, atuadores, tecnologias de identificação e rastreamento. A ideia de Internet das Coisas foi criada sob essa perspectiva em uma época que os proprietários estavam interessados em incluir seus dispositivos à Internet. Entre as tecnologias de maior destaque estão as de identificação e comunicação sem fio, como RFID, NFC (*Near Field Communications*) e o WSAN (*Wireless Sensor and Actuator Networks*).

Na visão orientada a Internet, o objetivo é desenvolver soluções para as diversas "coisas" se interconectarem à Internet. A IPSO (*IP for Smart Objects*) Alliance defende o uso do protocolo IP como meio de conectar os *Smart Objects* ao redor do mundo [Dunkels and Vasseur 2008]. Além desta, foi proposta a Internet 0, uma protocolo de roteamento da camada física cujo objetivo é apresentar uma complexidade inferior a do IP.

Finalmente, na visão orientada a semântica é defendido que tecnologias semânticas possam ser uma solução para as consequências que a quantidade enorme de coisas conectadas na Internet trará quanto a representação, armazenamento, interconexão, busca e organização de informação [Toma et al. 2009].

Ainda na perspectiva orientada a Internet, surge a Web das Coisas, uma importante abordagem da Internet das Coisas cujas características serão descritas na próxima seção.

3. Web das Coisas

A Web das Coisas é uma visão que surgiu para tentar resolver problemas relacionados principalmente a interconexão devida à heterogeneidade de dispositivos e protocolos associados à Internet das Coisas. Sua principal vantagem está no fato de que um desenvolvedor que segue esta abordagem não tem de se preocupar com protocolos específicos ou padrões de rede. A Web das Coisas lida essencialmente com a camada de aplicação do modelo OSI (em contraste com a Internet das Coisas que tem potencial para lidar com todas elas). A Web das Coisa efetivamente quebra o padrão "um protocolo, um dispositivo, uma aplicação", frequente na atual Internet das Coisas, ao integrar as "coisas" na Web, explorando padrões e tecnologias já bem estabelecidos como HTTP e JSON. Ao transformar "coisas" em recursos web, a conexão entre dispositivos heterogêneos se torna muito mais simples e o desenvolvedor pode focar na lógica da aplicações em vez de ter se de preocupar com questões que fogem de seu escopo ou área de conhecimento. [Guinard and Trifa 2016b].

Uma das maneiras sugeridas para se implementar a Web das Coisas é através da integração das coisas em uma arquitetura de Web Service padronizado [Guinard et al. 2011]. No entanto, Web Services que implementam padrões como SOAP (*Simple Object Access Protocol*) ou WSDL (*Web Services Description Language*) podem ser custosos demais em termos de processamento e consumo de memória. Considerando que muitos dispositivos que formam a Internet das Coisas possam conter sistemas de hardware e software muito simples, essa opção pode não ser a mais adequada.

Uma alternativa recente para a solução baseada em Web Services tem sido o uso de servidores Web embarcados (EWS) que podem ser executados em dispositivos com apenas alguns kB de memória RAM ou EEPROM [Duquennoy et al. 2009]. Nesse contexto de Web das Coisas, o sistema de um cliente web geralmente é mais poderoso que

o de um servidor embarcado. Ao contrário de um servidor web comum, porém, ele não precisa lidar com milhares de conexões e requisições. Existem conjuntos de técnicas bem conhecidas, como AJAX (Asynchronous Javascript and XML) e Comet, que possibilitam que o cliente realize o processamento que, de outra forma, seria feito pelo servidor.

Juntamente aos servidores embarcados, é desejávem aplicar o estilo REST para assegurar diversas características importantes para a Web das Coisas conforme será mostrado a seguir.

3.1. Arquitetura REST

REST (*REpresentation State Transfer*) é um estilo arquitetural da Web que visa melhorar propriedades de sistema como desempenho, escabilidade e segurança [Fielding 2000a]. Essa arquitetura destina-se a guiar a criação de serviços para que sejam fracamente acoplados e de fácil reusabilidade ao impor um conjunto de restrições sobre elementos que a compõe. Um sistema que aplica as restrições exigidas pelo estilo arquitetural REST pode ser chamado de *RESTful*.

Um dos aspectos mais importantes do REST é a definição de uma interface uniforme entre os componentes da arquitetura. Baseado na identificação e manipulação de recursos por meio de URIs e através da aplicação de métodos HTTP (GET, PUT, POST, DELETE, etc.), ela garante o desacoplamento entre a semântica da aplicação e os serviços fornecidos por esta, além de uma melhoria na visibilidade das interações entre os componentes.

Com isso, o acesso aos mais diversos dispositivos pertencentes a Web das Coisas é simplificado. Pessoas poderiam facilmente, através de um navegador Web, por exemplo, procurar, acessar, interagir e compartilhar informações sobre os dispositivos, da mesma forma que hoje é feito com qualquer recurso Web identificável.

Além da exigência de uma interface uniforme, duas restrições são de grande importância para tornar um sistema RESTful: a separação explícita das responsabilidades entre cliente e servidor e a condição de que o servidor não guarde qualquer informação sobre o estado atual de seus clientes - isso é, uma requisição do cliente deve conter toda a informação necessária para que o servidor possa tomar a decisão. A primeira restrição é necessária para que os componentes do sistema possam evoluir de forma independente. Já a segunda, melhora conceitos de visibilidade, confiabilidade e escalabilidade [Fielding 2000b].

Um servidor também precisa declarar (implícita ou explicamente), na resposta de uma requisição, se os dados requisitados devem, ou não, ser mantidos em cache. Essa requisição tem por objetivo melhorar a eficiência, escabilidade e o desempenho percebido pelo usuário ao ter o tempo médio das respostas reduzido.

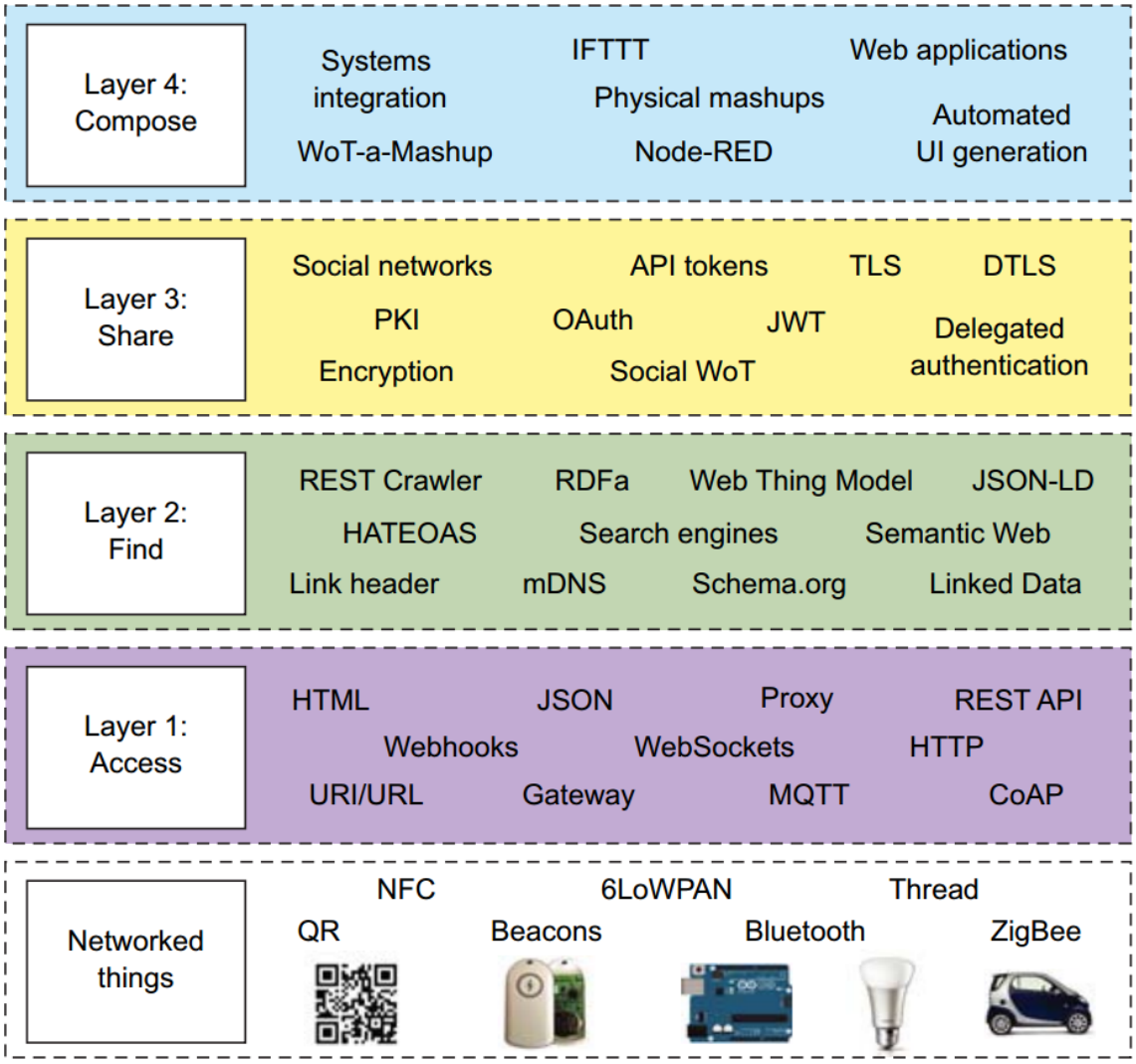
Por fim, a última restrição do REST define a implementação de um sistema em camadas, de forma que cada componente (cliente ou servidor) não tenha conhecimento de outras camadas, além daquelas que estão interagindo. Dessa forma, é possível um cenário onde hajam servidores intermediários ao longo do caminho sem que o cliente tenha como distingui-los do servidor final. Através do balanceamento de carga e da utilização de caches compartilhadas, esses servidores intermediários podem melhorar a escabilidade e o desempenho do sistema. Além disso, podem aplicar políticas de segurança ao agirem

como *firewalls*, por exemplo.

3.2. Arquitetura em Camadas

Uma arquitetura em camadas para Web das Coisas foi criada por [Guinard and Trifa 2016a]. A ideia é que cada camada possa resolver um conjunto de problemas específicos para servir à camada superior. As quatro camadas são mostradas na Figura 2. São elas: (1) Acessar (*Access*), (2) Encontrar (*Find*), (3) Compartilhar (*Share*) e (4) Compor (*Compose*).

Figura 2. Arquitetura em camadas da Web das Coisas.



Fonte [Guinard and Trifa 2016a].

A camada 1 (Acessar) se preocupa em conectar as coisas à Web utilizando para isso uma API Web. Essa camada transforma as "coisas" em recursos programáveis da Web das Coisas e expõe seus serviços através de uma interface padrão (REST).

A camada 2 (Encontrar) é responsável por deixar as coisas acessíveis na Web, tornando-as visíveis de forma que alguém (ou algo) seja capaz de encontrá-las através de

um motor de busca, por exemplo. Mas, esta camada não se limita a isso. A camada 2 implementa protocolos que aplicações devam seguir para que possam discernir e identificar os serviços fornecidos por outras coisas.

A camada 3 (Compartilhar) é focada em lidar com o compartilhamento de forma segura e eficiente das coisas que já são acessíveis e que já podem ser encontradas na Web.

A camada 4 (Compor) tem o objetivo facilitar a criação de aplicações que envolvam coisas e serviços Web virtuais. Ferramentas que operam nessa camada incluem: Web Toolkits, *dashboards* programáveis e ferramentas de *mashup* físico como Node-RED. Esta camada também lida com a visualização e o significado da grande quantidade de dados que podem ser coletados.

4. Cidades Inteligentes

De acordo com o Departamento de Assuntos Sócio-Econômicos da ONU [UN 2014], 54% da população mundial vivia em cidades no ano de 2014, e a perspectiva é que essa porcentagem tenda a aumentar ainda mais nas próximas décadas. Associados a este fenômeno urbano, são cada vez maiores os problemas relacionados a gestão de resíduos, escassez de recursos, poluição do ar, problemas de saúde, congestionamentos de trânsito, bem como problemas de natureza social e organizacional [Chourabi et al. 2012]. Cresce, portanto, o interesse em tornar mais eficientes os diversos setores de uma cidade, assim como tornar locais públicos mais inteligentes e mais agradáveis para a população em geral.

Dessas motivações surge o conceito de "Cidades Inteligentes", termo que vem sendo amplamente debatido nos últimos anos. Diversas cidades já se auto-proclamam "inteligentes", não havendo, no entanto, um consenso que defina formalmente o momento em que uma cidade se torna uma. De um modo geral, uma "cidade inteligente" surge quando tecnologias de informação e comunicação são aplicadas a centros urbanos visando fornecer uma infraestrutura que garanta formas de se alcançar o desenvolvimento social, econômico e urbano de maneira sustentável, melhoria na qualidade de vida da população e maior eficiência na utilização de recursos. [Gea et al. 2013].

De acordo com [Giffinger et al. 2007], uma Cidade Inteligente é uma cidade construída através esforço de cidadãos conscientes e que se destaca positivamente em seis características: Economia Inteligente, População Inteligente, Administração Inteligente, Mobilidade Inteligente, Ambiente Inteligente e Modo de Vida Inteligente. A Figura 3 mostra as seis características e os fatores associados que as descrevem.

Como pré-requisito para uma cidade tornar-se inteligente é necessário que ela usufrua de uma infraestrutura de redes sem fio para permitir a alta demanda de conexões dos cidadãos e de organizações locais. A infraestrutura e ambientes públicos da cidade devem ser equipadas com dispositivos inteligentes, possibilitando o gerenciamento de dados em tempo real, além da emissão de avisos e processamento de informações [Schaffers et al. 2011].

Figura 3. Fatores e características de uma Cidade Inteligente.

Economia Inteligente (Competitividade)	População Inteligente (Capital Humano e Social)	Administração Inteligente (Participação)
Espírito inovador	Nível de qualificação	Participação na tomada de decisões
Empreendedorismo	Afinidade com aprendizagem ao longo da vida	Serviços públicos
Imagem econômica e marcas registradas	Pluralidade ética e social	Serviços sociais
Produtividade	Flexibilidade	Administração transparente
Flexibilidade do mercado de trabalho	Criatividade	
Inserção internacional	Cosmopolitanismo	Perspectivas e estratétias políticas
Capacidade de transformar	Participação na vida pública	
Mobilidade Inteligente (Transporte e TCI)	**Ambiente Inteligente (Recursos Naturais)**	**Modo de Vida Inteligente (Qualidade de Vida)**
Acessibilidade local	Atratividade das condições naturais	Centros culturais
Acessibilidade (inter)nacional	Poluição	Condições de saúde
		Segurança individual
Disponibilidade de infraestrutura de TCI	Proteção ambiental	Qualidade de moradia
		Instalações educacionais
Sistemas de transporte sustentáveis, inovadores e seguros	Gerenciamento sustentável de recursos	Atrações turísticas
		Coesão social

Fonte: adaptado de [Giffinger et al. 2007].

4.1. Mobilidade Inteligente

A Mobilidade Inteligente é uma característica de uma Cidade Inteligente que visa melhorar aspectos como segurança, sustentabilidade e eficiência no transporte urbano, assim como tornar mais fácil e rápido o deslocamento de pessoas ao longo da cidade [City 2015a]. Aplicações para Mobilidade Inteligente são de grande interesse devido aos impactos que tem sobre a cidade. Com o objetivo de contextualizar, serão apresentados brevemente quatro exemplos de aplicações.

Uso compartilhado de bicicletas. Após efetuado cadastro em um aplicativo, usuários podem utilizar bicicletas, por um determinado período de tempo, que estão estacionadas em diversas áreas espalhadas pela cidade. Exemplos de cidades que implementam: Milão, Paris, Nova Iorque, Barcelona, Porto Alegre [Wikipedia 2016a] [BikePoa 2016].

Controle de tráfego inteligente. Inclui sistema de sensores e controle de semáforos com o objetivo de regular o fluxo de trânsito de acordo com a demanda [Transport 2016]. Em Barcelona, os semáforos ficam verdes quando há veículos de emergência em rota [City 2015b]. Exemplos de cidades que implementam: Barcelona, Amsterdã [City 2016].

Paradas de ônibus inteligentes. Podem fornecer diversas informações em tempo real para os passageiros, tais como horários, tipos de ônibus (com acessebilidade, com suporte para biciletas, etc.), itinerários, e informações gerais. Exemplos de cidades que implementam: Seul [Government 2010], Moscou [Times 2013].

Estacionamentos inteligentes. São aplicações que podem indicar o melhor local para o motorista estacionar. Baseiam-se no número de vagas disponíveis em estacionamentos e nos locais aonde o usuário pretende se deslocar. Exemplos de cidades que implementam: Amsterdã [Wikipedia 2016b].

5. Proposta

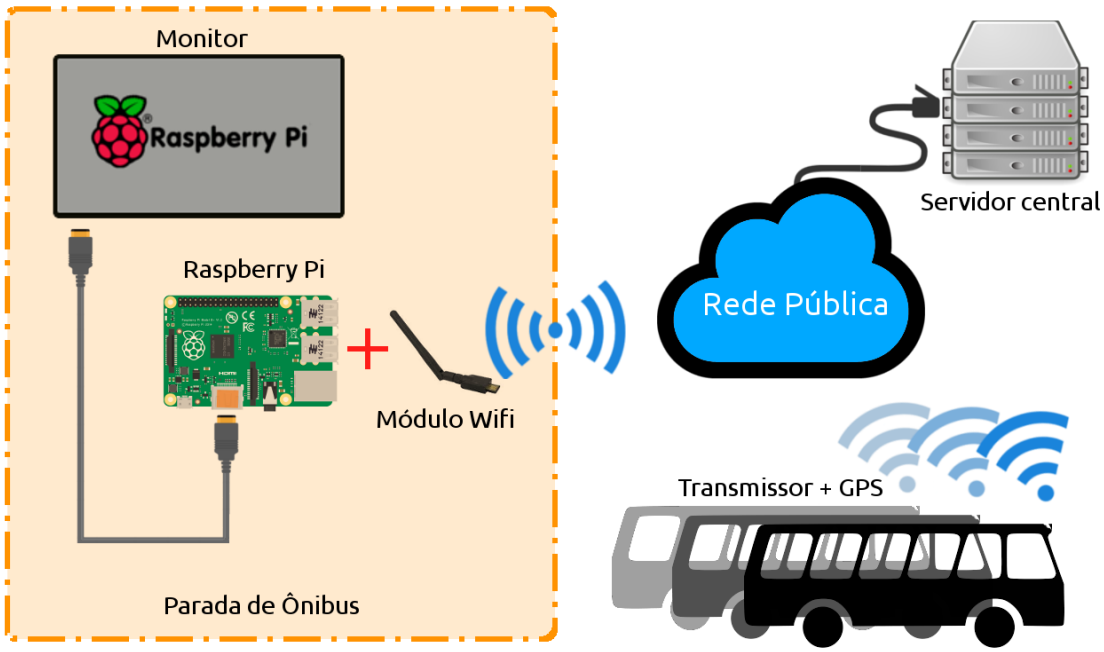
A proposta para este trabalho de graduação é realizar um estudo sobre os conceitos que envolvem e interrelacionam a Internet das Coisas, Web das Coisas e Cidades Inteligentes. Com base nos estudos feitos, será desenvolvido um protótipo focado em Cidades Inteligentes para uma *Parada de Ônibus Inteligente*. O objetivo é aplicar os conceitos de Internet das Coisas (sob a abordagem de Web das Coisas) em locais públicos de uma cidade, neste caso, paradas de ônibus.

A ideia geral deste projeto é tornar *Inteligentes* as paradas de ônibus de uma cidade, equipando-as com dispositivos que possam informar os dados dos ônibus que circulam por ela. Tais dados podem estar relacionados com localização (GPS), identificação (ID, nome), tipo (“cadeirantes”, “bicicletas”) ou estado (“danificado”, “atrasado”, etc.). O propósito final é conectar o dispositivo a um monitor (ou display LCD) que será fixado às paradas, onde as informações sobre os ônibus serão mostradas para as pessoas que estiverem por perto. Além de dados inerentes aos ônibus, outras informações como notícias e avisos poderiam ser transmitidos para todas as paradas (ou para uma específica) a partir de um servidor central.

Seguindo os conceitos de Web das Coisas, cada dispositivo de uma parada teria seu próprio servidor HTTP (REST) em execução. Para que isto seja possível, é necessário que os seus componentes possam ter conexão sem fio com a Internet. Além disso, é preciso que o protótipo tenha um equipamento processamento básico de vídeo para que seja possível gerar a interface gráfica. É desejável também que esses dispositivos sejam baratos, visto que serão espalhados por muitos locais públicos da cidade. Placas da família Raspberry Pi parecem ser uma boa escolha, já que satisfazem todos esses requisitos (por meio de módulos WIFI, etc.) e ainda podem ser encontradas por um valor relativamente baixo no mercado (como o Raspberry Pi Zero, anunciado por US$5). A Figura 4 mostra a organização geral do sistema utilizando Raspberry Pi. No entanto, dispositivos Android e placas Arduino poderão ser utilizadas em seu lugar.

É requisito necessário que cada ônibus esteja equipado com equipamentos de GPS e algum transmissor sem fio (WIFI, 2G, etc.), pois cada ônibus irá transmitir o sinal com

Figura 4. Esquemático das partes que compõe o sistema.



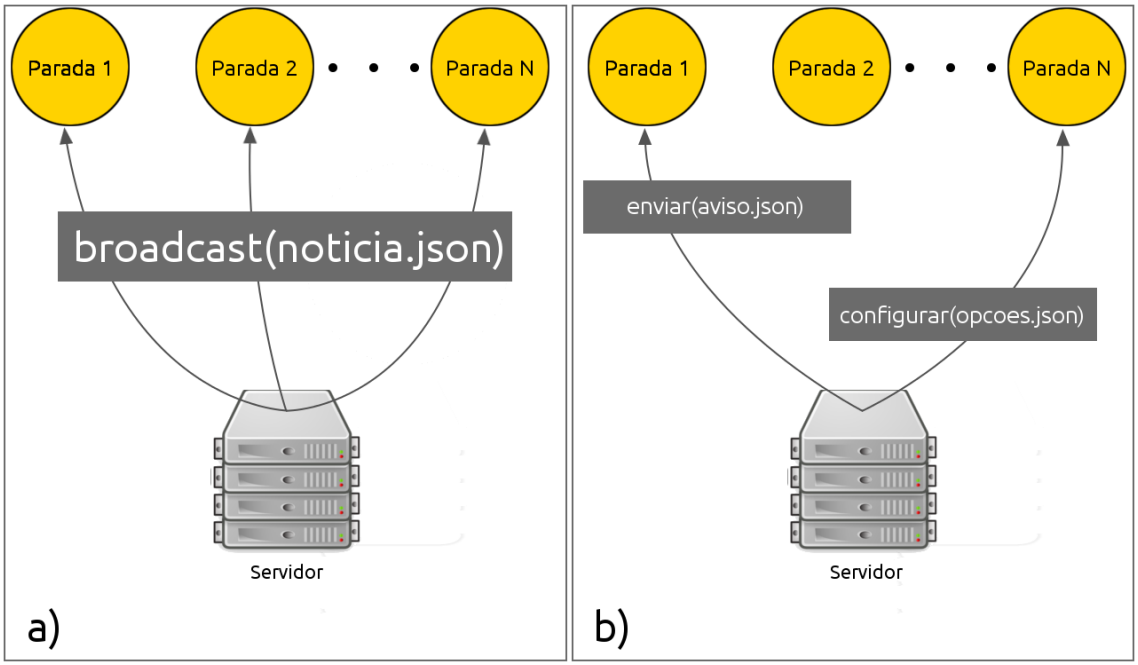
Fonte: autor.

seus dados para todas as paradas que estiver associado. As paradas, então, poderão gerar dados a respeito da distância ou posição atual dos ônibus, ou o tempo que levarão para alcançá-la, por exemplo, exibindo o resultado ao público.

O servidor central terá acesso ao banco de dados com informações sobre as paradas da cidade (nome, localização, quais ônibus passam por elas, etc.). Portanto, além de responsável pela propagação de avisos e informações, também serão pela configuração das paradas e ônibus.

Na Figura 5 são apresentados dois casos de uso do servidor central. No caso a), o servidor transmite mensagens com notícias para todas as paradas de ônibus que passarão, então, a exibi-las. No caso b), o servidor manda mensagens específicas para determinadas paradas. Os tipos de mensagem podem ser, por exemplo, de aviso ou configuração. Uma mensagem de aviso teria o objetivo de informar uma ou mais paradas sobre assuntos que são relevantes apenas para aquela área. Um exemplo seria o envio de um anúncio aos passageiros de que será alterado o itinerário de um dos ônibus que por ali passam. Já as mensagens de configuração podem ser utilizadas para configurar uma parada pela primeira vez, ou para atualizar seus atributos, como a lista de ônibus que passam por ela.

Figura 5. Exemplos de casos de usos do servidor central.



Fonte: autor.

6. Cronograma

Tabela 1. Cronograma de atividades

	Jul	Ago	Set	Out	Nov	Dez
Aquisição de equipamentos	X					
Desenvolvimento da aplicação	X	X	X			
Escrita do TCC		X	X	X		
Testes e ajustes				X	X	
Revisão do TCC					X	
Entrega e apresentação						X

Referências

Ashton, K. (2009). That 'internet of things' thing. In *RFID Journal*. http://www.rfidjournal.com/articles/view?4986.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. In *Computer Networks*.

BikePoa (2016). Sobre o bike poa.

Chourabi, H., Nam, T., Walker, S., Gil-Garcia, J. R., Mellouli, S., Nahon, K., Pardo, T. A., and Scholl, H. J. (2012). Understanding smart cities: An integrative framework. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 2289–2297.

City, A. A. S. (2016). Smart traffic management. https://amsterdamsmartcity.com/projects/smart-traffic-management.

City, B. S. (2015a). Smart mobility. In *Smart City Areas*. http://smartcity.bcn.cat/en/new-bus-network.html.

City, B. S. (2015b). Smart traffic lights. In *Smart City Areas*. http://smartcity.bcn.cat/en/smart-traffic-lights.html.

Dunkels, A. and Vasseur, J. (2008). Ip for smart objects, internet protocol for smart objects (ipso) alliance, white paper #1. http://dunkels.com/adam/dunkels08ipso.pdf.

Duquennoy, S., Grimaud, G., and Vandewalle, J. J. (2009). The web of things: Interconnecting devices with high usability and performance. In *Embedded Software and Systems, 2009. ICESS '09. International Conference on*, pages 323–330.

Fielding, R. T. (2000a). Network-based application architectures. In *Architectural Styles and the Design of Network-based Software Architectures (Ph.D.)*. University of California, http://www.ics.uci.edu/ fielding/pubs/dissertation/net_app_arch.htm.

Fielding, R. T. (2000b). Representational state transfer (rest). In *Architectural Styles and the Design of Network-based Software Architectures (Ph.D.)*. University of California, http://www.ics.uci.edu/ fielding/pubs/dissertation/rest_arch_style.htm.

Gea, T., Paradells, J., Lamarca, M., and Roldán, D. (2013). Smart cities as an application of internet of things: Experiences and lessons learnt in barcelona. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013 Seventh International Conference on*, pages 552–557.

Giffinger, R., Fertner, C., Kramar, H., Kalasek, R., Pichler-Milanovic, N., and Meijers, E. (2007). Smart cities: Ranking of european medium-sized cities. http://www.smart-cities.eu/download/smart_cities_final_report.pdf.

Government, S. M. (2010). Smart bus stops in seoul. http://english.seoul.go.kr/smart-bus-stops-in-seoul/.

Guinard, D. and Trifa, V. (2016a). Using the web to build the iot. Manning Publications Co.

Guinard, D., Trifa, V., Mattern, F., and Wilde, E. (2011). From the internet of things to the web of things: Resource-oriented architecture and best practices. In *Architecting the Internet of Things*. Springer.

Guinard, D. D. and Trifa, V. M. (2016b). Building the web of things. Manning.

IDC (2015). Explosive internet of things spending to reach $1.7 trillion in 2020. http://www.idc.com/getdoc.jsp?containerId=prUS25658015.

Schaffers, H., Komninos, N., Pallot, M., Trousse, B., Nilsson, M., Oliveira, Alvaro", e. J., Galis, A., Gavras, A., Zahariadis, T., Lambert, D., Cleary, F., Daras, P., Krco, S., Müller, H., Li, M.-S., Schaffers, H., Lotz, V., Alvarez, F., Stiller, B., Karnouskos, S., Avessta, S., and Nilsson, M. (2011). Smart cities and the future internet: Towards cooperation frameworks for open innovation. In *The Future Internet: Future Internet Assembly 2011: Achievements and Technological Promises*, pages 431–446. Springer Berlin Heidelberg, Berlin, Heidelberg.

Times, T. T. M. (2013). 'smart' maps appear at moscow bus stops. http://www.themoscowtimes.com/news/article/smart-maps-appear-at-moscow-bus-stops/473955.html.

Toma, I., Simperl, E., and Hench, G. (2009). A joint roadmap for semantic technologies and the internet of things. In *Proceedings of the Third STI Roadmapping Workshop*.

Transport, S. S. C. (2016). Smart traffic management. http://www.smartertransport.uk/smart-traffic-management/.

Uckelmann, D., Harrison, M., and Michahelles, F. (2011). Introduction, background and initial visions. In *Architecting the Internet of Things*. Springer.

UN (2014). World's population increasingly urban with more than half living in urban areas. http://www.un.org/en/development/desa/news/population/world-urbanization-prospects-2014.html.

Vermesan, O., Friess, P., Guillemin, P., Sundmaeker, H., Eisenhauer, M., Moessner, K., Arndt, M., Spirito, M., Medagliani, P., Giaffreda, R., Gusmeroli, S., Ladid, L., Serrano, M., Hauswirth, M., and Baldini, G. (2014). Internet of things strategic research and innovation agenda. In *Internet of Things From Research and Innovation to Market Deployment*. River.

Wikipedia (2016a). Bicycle sharing system. https://en.wikipedia.org/wiki/Bicycle-sharing_system.

Wikipedia (2016b). Smart cities. https://en.wikipedia.org/wiki/Smart_city#Amsterdam.

Zanella, A., Bui, N., Castellani, A., Vangelista, L., and Zorzi, M. (2014). Internet of things for smart cities. In *IEEE Internet of Things Journal, Vol. 1, No. 1, February 2014*, page 22.