UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

BACHARELADO EM ENGENHARIA DE SOFTWARE

CAYK GOMES LIMA BARRETO

SISTEMA DE MONITORAMENTO DE BICICLETAS URBANAS

QUIXADÁ – CEARÁ

2016

CAYK GOMES LIMA BARRETO

SISTEMA DE MONITORAMENTO DE BICICLETAS URBANAS

Monografia apresentada no curso de Engenharia de Software da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Engenharia de Software. Área de concentração: Computação.

Orientador: Prof. Dr. Marcio Espíndola Freire Maia

QUIXADÁ – CEARÁ

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

B261s Barreto, Cayk Gomes Lima.
Sistema de Monitoramento de Bicicletas Urbanas / Cayk Gomes Lima Barreto. – 2016.
54 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2016.
Orientação: Prof. Dr. Marcio Espíndola Freire Maia.

1. Computação ubíqua. 2. Computação móvel. 3. Sistemas embarcados (Computadores). I. Título.
CDD 005.1

CAYK GOMES LIMA BARRETO

SISTEMA DE MONITORAMENTO DE BICICLETAS URBANAS

Monografia apresentada no curso de Engenharia de Software da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Engenharia de Software. Área de concentração: Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Marcio Espíndola Freire Maia (Orientador)
Campus Quixadá
Universidade Federal do Ceará – UFC

Prof.ª Dr.ª Rossana Maria de Castro Andrade
Universidade Federal do Ceará - UFC

Prof. Msc. Bruno Góis Mateus
Campus Quixadá
Universidade Federal do Ceará - UFC

Aos meus pais.

Aos meus amigos e a todos que acreditaram.

AGRADECIMENTOS

Aos meu pais, Sueli e Marcos, que sempre acreditaram em mim e me deram todo o suporte necessário durante a graduação.

Ao meu orientador, Prof. Dr. Marcio Espíndola Freire Maia , pela ideia proposta e confiança.

Aos professores participantes da banca examinadora Prof.ª Dr.ª Rossana Maria de Castro Andrade e Prof. Msc. Bruno Góis Mateus pelo tempo e colaborações dadas ao trabalho.

Agradeço a todos que me ajudaram de alguma forma durante a realização desse trabalho, especialmente ao Luan Lima, que fez contribuições importantes para o desenvolvimento do projeto.

Gostaria de agradecer ao Ricardo Ferreira, meu orientador de bolsa, e aos membros da secretária acadêmica por sempre me atenderem quando precisei.

Agradeço também a todos que estiveram comigo durante esses 4 anos de caminhada.

“Escolha sempre o caminho que pareça o melhor, mesmo que seja o mais difícil; o hábito brevemente o tornará fácil e agradável.”

(Pitágoras)

RESUMO

Dentro da área Internet das Coisas, vários domínios de aplicação podem ser citados. Esse projeto visa trabalhar no domínio de aplicações de mobilidade, especificamente com sistemas de transporte sustentáveis, que tiveram um aumento de atenção nos últimos anos por causa do crescimento do consumo de energia, poluição e trânsito nas cidades. Dentre os meios de trasporte sustentáveis, temos as bicicletas. Esse trabalho tem como objetivo criar uma aplicação que realize o monitoramento de bicicletas urbanas. O projeto é dividido em três módulos. O módulo Cliente é responsável por mostrar as informações obtidas para o usuário, essa tarefa é realizada através de uma aplicação Android. O módulo *WebService* é responsável por persistir os dados utilizando NodeJS e o MongoDB e disponibiliza-los ao módulo Cliente. Já o módulo Placa é responsável por interagir com o ambiente e coletar informações utilizando o Arduíno Uno e o componente SIM808. Para facilitar a implementação e o entendimento dos módulos, foi criado um modelo arquitetural com várias visões diferentes da aplicação. Além disso, no modelo também é mostrado as restrições para a utilização da aplicação. Por fim, é mostrado a aplicação em uso, os problemas enfrentados durante o desenvolvimento, uma discussão sobre os resultados obtidos, possíveis trabalhos futuros e as considerações finais sobre o trabalho.

Palavras-chave: Internet das Coisas. Arduíno. WebService REST. Android. Monitoramento de Bicicletas

ABSTRACT

Within the Internet of Things, we have several different fields that can be used to create applications that exchange information with each other or with end users through the Internet. This project aims to work with Internet of Things, more specifically in the field of mobility to create an application that performs the monitoring of urban bicycles. The project is divided into three modules. The Client module is responsible for displaying the obtained information to the user, performed through an Android application. The WebService module is responsible for persisting the date using NodeJS and MongoDB and making the data available to the Client module. The Board module is responsible for interacting with the environment and collecting information, which is obtained through the Arduino Uno and the SIM808 component. To make the implementation and understanding of the modules easier, an architectural model was created with several different views of the application, in addition, the model also shows the restrictions for the use of the application. Finally, it shows the application in use, the problems faced during the development, a discussion about the results obtained, possible future work and the final considerations about this project.

Keywords: Internet of Things. Arduino. WebService REST. Android. Bicycle Monitoring

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE CÓDIGOS-FONTE

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
BBM	Bicycle-Born Monitor
CAGR	Compounded Annual Growth Rate
CPU	Central Processing Unit
GPRS	General Packet Radio Services
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IoT	Internet of Things
IoT-A	Internet of Things - Architecture
JSON	JavaScript Object Notation
LoCCAM	Loosely Coupled Context Acquisition Middware
REST	Representational State Transfer
RFID	Radio Frequency Identification
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SWB	SensorWebBike
UDDI	Universal Description Discovery and Integration
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus
WSAN	Wireless Sensor and Actor Network

WSDL	Web Service Description Language
WSN	Wireless Sensor Network
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Internet das Coisas (*Internet of Things* - IoT) é um termo amplamente utilizado para se referir à conexão de dispositivos do dia a dia à internet (MIORANDI et al., 2012). Esses dispositivos, que podem trocar informações entre si e executar tarefas, variam de bens de consumo, como ar-condicionado, geladeiras e lâmpadas até máquinas industriais. Estima-se que, atualmente, mais de seis bilhões desses dispositivos estejam conectados à internet e as previsões para o futuro são que esses números continuem aumentando (BASSI et al., 2013).

De acordo com Miorandi et al. (2012), de um ponto de vista conceitual, a Internet das Coisas é construída sobre três pilares relacionados à habilidade de objetos inteligentes: i) ser identificável, em que qualquer coisa se identifica, ii) ser comunicável, em que qualquer coisa se comunica e iii) interagir, em que qualquer coisa interage, seja entre elas ou com usuários finais. Para ocorrer a identificação e comunicação entre os objetos, é necessária a utilização de tecnologias auxiliares.

Dentre essas tecnologias, temos o *Radio Frequency Identification* (RFID), que ajuda na identificação e comunicação embarcada das "coisas" (WANT, 2006). Temos também a *Wireless Sensor Network* (WSN), conjunto de sensores (*smart sensors*, atuadores, tags RFID) que interagem com o ambiente e entre sí para obter informações (WHITMORE; AGARWAL; XU, 2015). Em relação à comunicação e transferência de dados, as principais tecnologias são 3G, 4G, GSM, UMTS, WiFi, infra-vermelho e Bluetooth *Low Energy* (GUBBI et al., 2013). Na computação dos dados, podem ser citadas as placas de prototipagem: Arduíno, Intel Galileo, Raspberry Pi e BeagleBone, juntamente com os *Smart Phones* são as principais tecnologias (HEMALATHA; AFREEN, 2015).

Com a utilização da Internet das Coisas, é possível realizar um número grande de aplicações de diferentes tipos. Pode-se utilizar IoT para aplicações de tipo pessoal ou residenciais, aplicações de nível comercial, como sistemas de controle climático e controle de ambiente, e também em aplicações de mobilidade, como carros e bicicletas.

Especificamente tratando das aplicações de mobilidade, sistemas de transporte sustentáveis ganharam uma atenção significativa nos últimos anos por causa do crescimento do consumo de energia, poluição do ar e sonora (RAZZAQUE; CLARKE, 2015b). Além disso, o uso de bicicletas para transporte é um dos meios de transporte sustentáveis que mais crescem no mundo.

Além dos fatores citados, o uso de bicicletas como meio de transporte é uma forma

de amenizar o trânsito nas cidades, podendo ser útil a utilização de algum meio para controlar e monitorar essas bicicletas.

Assim, esse trabalho propõe a criação de um sistema para o monitoramento de bicicletas urbanas através de sensores. Esses sensores serão responsáveis por coletar a localização das bicicletas e enviá-las para um servidor. O servidor é responsável por persistir e disponibilizar os dados obtidos. Para a visualização das informações será disponibilizada uma aplicação Android.

O trabalho pode ser utilizado tanto para ciclistas individuais que querem saber informações sobre suas viagens com a bicicleta (aplicação), quanto para pessoas ou sistemas que desejam coletar informações (serviço), como por exemplo, as principais ruas que os ciclistas utilizam durante suas viagens.

Além da seção de Introdução, este trabalho está organizado em outras cinco seções. A seção Fundamentação Teórica apresenta os conceitos fundamentais para o desenvolvimento do trabalho: Internet das Coisas, Monitoramento de Bicicletas, *WebServices*, Android e Arduíno. A seção Trabalhos Relacionados apresenta os trabalhos que possuem objetivos semelhantes com os deste trabalho. A seção Proposta explica as etapas de desenvolvimento da aplicação e sua arquitetura. A seção Resultados e Discussão apresenta os resultados obtidos no uso da aplicação e uma breve discussão sobre esses resultados. Por fim, temos a seção Considerações Finais que apresenta as conclusões do projeto e seus trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

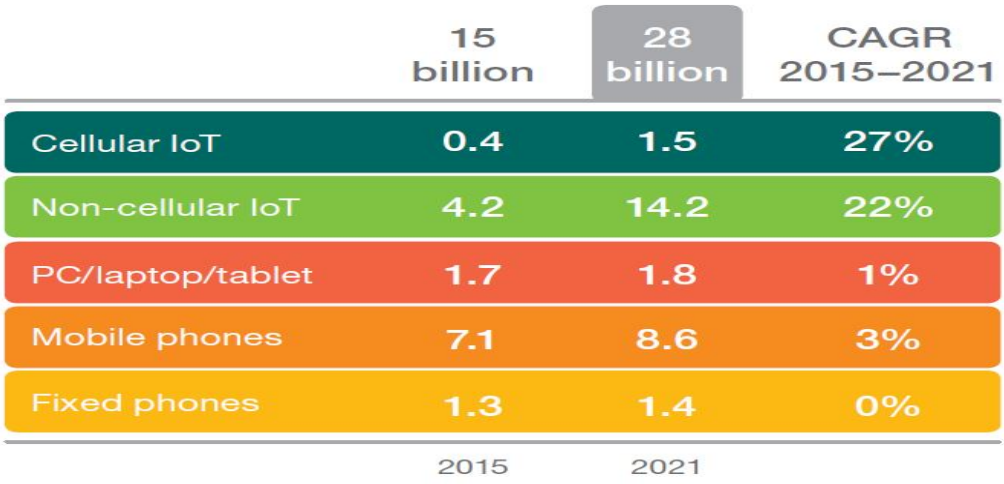
Na fundamentação teórica serão abordados os principais conceitos utilizados para o desenvolvimento do projeto. Na subseção 4.1 será definido a Internet das Coisas (IoT). Na subseção 4.2 será abordado o Monitoramento de bicicletas, mostrando o monitoramento de forma compartilhada e individual . Por fim, subseção 4.3 apresenta os conceitos relacionados ao desenvolvimento da aplicação: *WebService*, Android e Arduíno.

2.1 Internet das Coisas

A Internet das Coisas é um novo paradigma que rapidamente se espalhou no cenário de comunicações *wireless* moderna (ASGHAR; MOHAMMADZADEH; NEGI, 2015). IoT representa o conceito de conectar dispositivos do dia a dia à internet (MIORANDI et al., 2012). Esses dispositivos são chamados de *smart objects* ou "coisas", que podem se comunicar e trocar informações. O termo "coisas" na visão IoT possui um escopo muito amplo e inclui uma variedade diferente de dispositivos físicos. Ela inclui objetos pessoais como *smartphones*, inclui elementos de ambiente como veículos e casas, e por fim inclui coisas que utilizam tags (RFID ou outras) (COETZEE; EKSTEEN, 2011).

Estima-se que, atualmente, mais de seis bilhões desses dispositivos estejam conectados à Internet, com um potencial de mercado estimado em mais de 14 trilhões de dólares (BASSI et al., 2013). As previsões para o futuro são de que esses números continuem aumentando. A Figura 1 mostra as expectativas de crescimento para o ano de 2021 dos *smart objects* em relação a telefones, computadores, *tablets* e *laptops*. Também é mostrado a taxa de crescimento anual composta (*Compounded Annual Growth Rate* – CAGR) para os dispositivos.

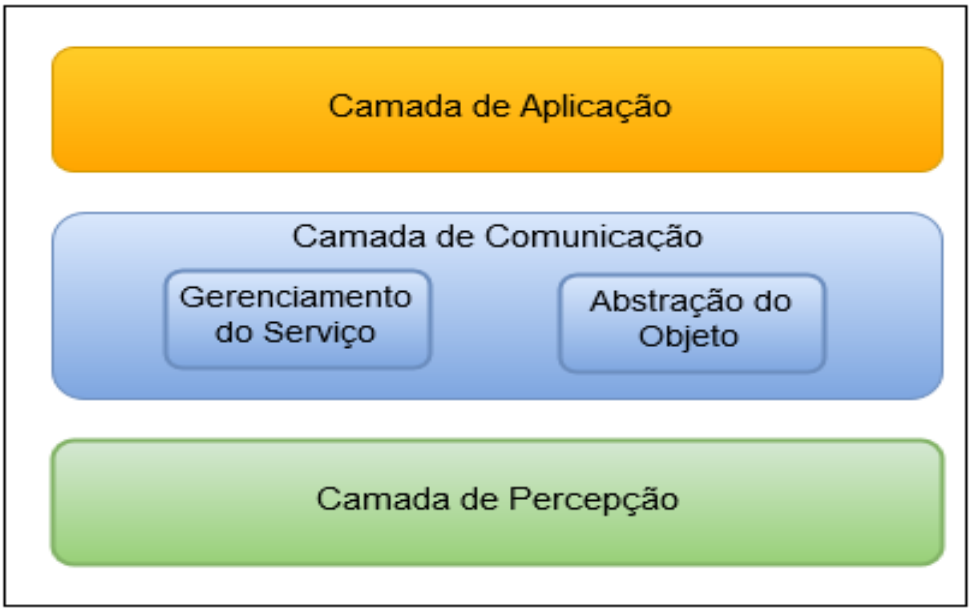
Figura 1 – Previsão para IoT em 2021.



Fonte: Ericsson (2016)

A Internet das Coisas deve ser capaz de interconectar bilhões ou trilhões de objetos heterogêneos através da Internet, portanto é critica a necessidade de uma arquitetura flexível (HEMALATHA; AFREEN, 2015). Ainda não existe um modelo arquitetural definido para IoT, porém temos alguns projetos, como por exemplo, IoT-A (Internet of Things - Architecture) que visam definir um modelo arquitetural de referência, assim garantindo a interconexão dos objetos heterogêneos de forma fácil e coesa (BASSI et al., 2013). Com base nos modelos apresentados em Hemalatha e Afreen, a Figura 2 ilustra uma visão geral IoT.

Figura 2 – Visão Geral IoT.



Fonte: Adaptada de Hemalatha e Afreen (2015)

A camada de Aplicação é responsável por prover serviços de alta qualidade para atender as necessidades dos usuários. Ela abrange inúmeros nichos de mercados, tais como *smart home*, *smart building*, transporte, automação industrial e *smart healthcare* (HEMALATHA; AFREEN, 2015).

A camada de Comunicação é responsável por fazer a transmissão dos dados entre o usuário final e o *smart object*. Nela, a subcamada de Gerenciamento do Serviço pareia um serviço com o seu requisitor baseando-se em endereços e nomes. Ela também abstrai os tipos de hardware, assim os programadores podem trabalhar com os objetos sem se preocupar com uma plataforma de hardware especifica. Por fim, a subcamada também processa os dados recebidos, toma decisões e entrega o serviço requerido (HEMALATHA; AFREEN, 2015). Já a subcamada Abstração do Objeto é responsável por transferir os dados produzidos na camada de Percepção para a subcamada de Gerenciamento.

A camada de Percepção representa os sensores físicos que coletam e processam os dados. Essa camada inclui sensores e atuadores que realizam diferentes funções como medir temperatura, medir batimentos cardíacos, pressão etc. A camada de Percepção digitaliza e transfere os dados para a subcamada de Abstração do Objeto (HEMALATHA; AFREEN, 2015).

Um exemplo de uma abordagem que funciona de modo parecido com a visão apresentada é o LoCCAM (*Loosely Coupled Context Acquisition Middware*), que provê a aquisição auto-adaptativa de informações do contexto (MAIA et al., 2013).

LoCCAM funciona como a camada de Comunicação, em que ele recebe as informações do contexto (Camada de Percepção) e notifica as aplicações (Camada de Aplicação) sobre a existência do conteúdo. Após a notificação, as aplicações podem requisitar e receber as informações coletadas.

Para que um serviço possa funcionar de acordo com a Figura 2, e assim garantir a identificação e comunicação entre os objetos, é necessário a utilização de tecnologias auxiliares, as principais são RFID e WSN/WSAN.

O *Radio Frequency Identification* (RFID), geralmente chamado de tag RFID, é um microchip modelado para transmissão *wireless* de dados. Uma tag RFID transmite dados pelo ar em resposta a uma pergunta de um *reader* RFID (JUELS, 2006). Um *reader* RFID usualmente é composto por um rádio transmissor, um receptor de rádio, uma unidade de controle e uma unidade de memória. A principal função do *reader* é viabilizar que as tags RFID e o servidor possam trocar mensagens (HEMALATHA; AFREEN, 2015). Tags RFID podem ser dividias

em duas classes: ativas e passivas (WANT, 2006). Tags ativas possuem bateria e um radio transmissor, o que dá a elas o poder de emitir sinais para se comunicar com os *readers*, porém elas são mais caras que as tags passivas, essas não possuem bateria e utilizam o poder dos sinais transmitidos pelos *readers* para se comunicar com eles.

As tags passivas atualmente são utilizadas em vários cartões de bancos e em pedágios de rodoviária, já as tags ativas são bastante utilizadas em containers portuários para o monitoramento de cargas (GUBBI et al., 2013). As tags também podem ser utilizadas para a identificação de sensores.

Sensores são dispositivos que monitoram as características do ambiente ou do objeto. Quando múltiplos sensores são utilizados em conjunto e interagem, eles formam uma *Wireless Sensor Networks* (WSN) (WHITMORE; AGARWAL; XU, 2015) ou *Wireless Sensor and Actor Network* (WSAN) (NAIR et al., 2015).

Gubbi et al. (2013) definem que para construir uma rede de monitoramento WSN é necessário incluir: a) Hardware WSN - geralmente um nó contendo as interfaces de sensores, unidades de processamento e fonte de energia; b) Pilha de Comunicação WSN - os nós precisam estar em uma rede para poder se comunicar entre si e com o mundo externo através da internet; c) *Middleware* WSN para prover acesso aos recursos disponibilizados pelos sensores; d) Agregação de Dados Segura - para aumentar o ciclo de vida da rede e também para garantir que a coleta de dados dos sensores é confiável.

Enquanto os sensores realizam o monitoramento dos objetos e do ambiente, os atuadores realizam ações para afetar o ambiente ou o objeto de alguma forma. Atuadores podem afetar o ambiente emitindo sons, luz etc. A combinação de sensores e atuadores possibilita que os objetos possam simultaneamente estar ciente do ambiente e interagir com pessoas, ambos objetivos da IoT (WHITMORE; AGARWAL; XU, 2015).

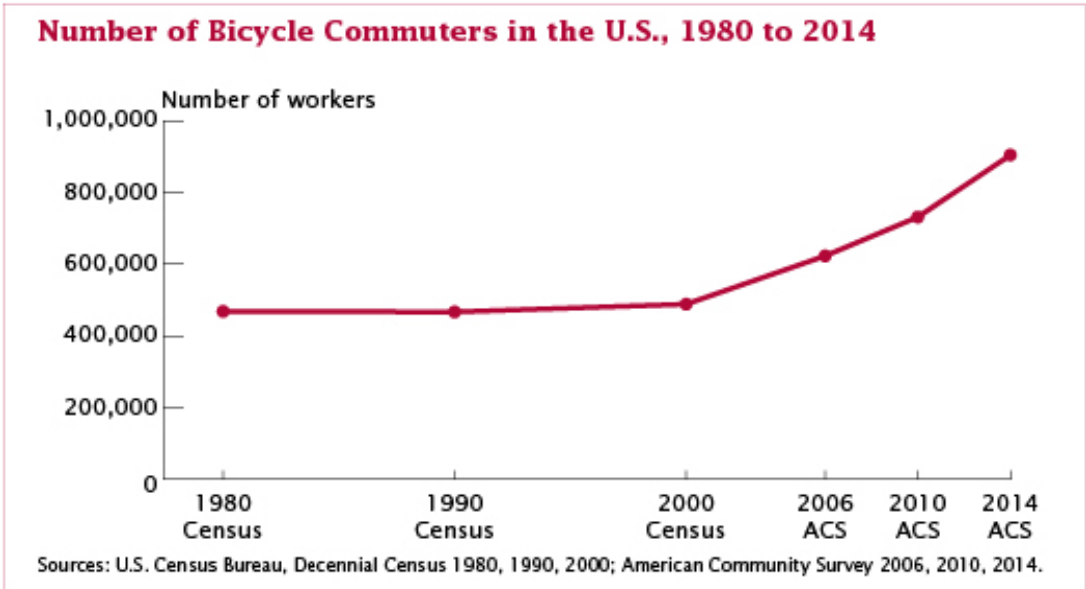
O potencial de criação de aplicações com IoT é imenso, existem vários nichos diferentes onde a utilização de Internet das Coisas é bem sucedida. Este trabalho se encaixa tanto no nicho de uso pessoal, quanto no de *smart cities*, pois ele visa monitorar bicicletas em ambientes urbanos.

Internet das Coisas é o conceito principal por trás deste trabalho. Serão utilizados sensores para realizar o monitoramento das bicicletas e também a coleta de dados, os resultados obtidos serão enviados para o servidor. A comunicação entre o servidor e a placa que contém os sensores será realizada através de conexões GPRS.

2.2 Monitoramento de Bicicletas

Sistemas de transporte sustentáveis tiveram um aumento significativo de atenção nos últimos anos por causa do crescimento do consumo de energia, barulho e poluição do ar (RAZZAQUE; CLARKE, 2015b). Dentre esses meios de transportes, temos a bicicleta, que além dos motivos citados anteriormente, também teve um crescimento no seu uso devido a problemas ambientais, sociais, de saúde e sustentabilidade (RAZZAQUE; CLARKE, 2015a). A Figura 3 mostra o crescimento no uso de bicicletas nos Estados Unidos entre 1980 e 2014.

Figura 3 – Crescimento do uso de bicicletas como meio de transporte nos EUA



Fonte: Bureau (2016)

Bicicletas e seu ecossistema em geral não tinham testemunhado quaisquer mudanças significativas até recentemente, com a integração de sensores (por exemplo, GPS) (PIRAMUTHU; ZHOU, 2016). Com a integração de sensores/atuadores e outros dispositivos com a bicicleta, uma nova janela de oportunidades se abre e operações que antes eram muito complicadas ou impossíveis, agora se tornam realidade.

De forma simplificada, podemos dividir o uso de bicicletas em duas formas: individual e coletiva. No uso individual a bicicleta é utilizada somente pelo dono, já na coletiva a utilização pode ser feita por várias pessoas. Um exemplo de uso coletivo são os sistemas de compartilhamento de bicicletas, em que os usuários pegam uma bicicleta em uma estação e podem devolver em qualquer outra (MIDGLEY, 2011), sem ter os custos e responsabilidades de um dono (SHAHEEN; GUZMAN; ZHANG, 2010). O objetivo de monitoramento da bicicleta

pode mudar, dependendo da forma, individual ou coletiva.

No monitoramento individual de uma bicicleta, o usuário pode estar interessado em saber os status da bicicleta em relação às ações que ele realizou. No monitoramento coletivo (sistemas de compartilhamento de bicicletas) o foco é na análise coletiva dos dados obtidos em relação às bicicletas, por exemplo, saber os principais pontos de início de viagem, e assim saber quais são as estações que necessitam de mais bicicletas, localização, para evitar que as bicicletas sejam furtadas etc.

Este trabalho visa realizar o monitoramento de bicicletas urbanas de forma individual, em que as bicicletas terão uma placa embarcada que conterá um módulo GPS e GPRS. A placa será responsável por monitorar as ações realizadas na bicicleta, como localização, e enviar essas informações para o servidor.

2.3 Aplicação

Nessa seção serão abordados os aspectos relacionados ao desenvolvimento da aplicação. Na subseção 4.3.1 será falado sobre *Webservices*, já a subseção 4.3.2 abordará aspectos relacionados ao *Android*, por fim, a subseção 4.3.3 irá descrever a placa de prototipação e seus sensores.

2.3.1 WebService

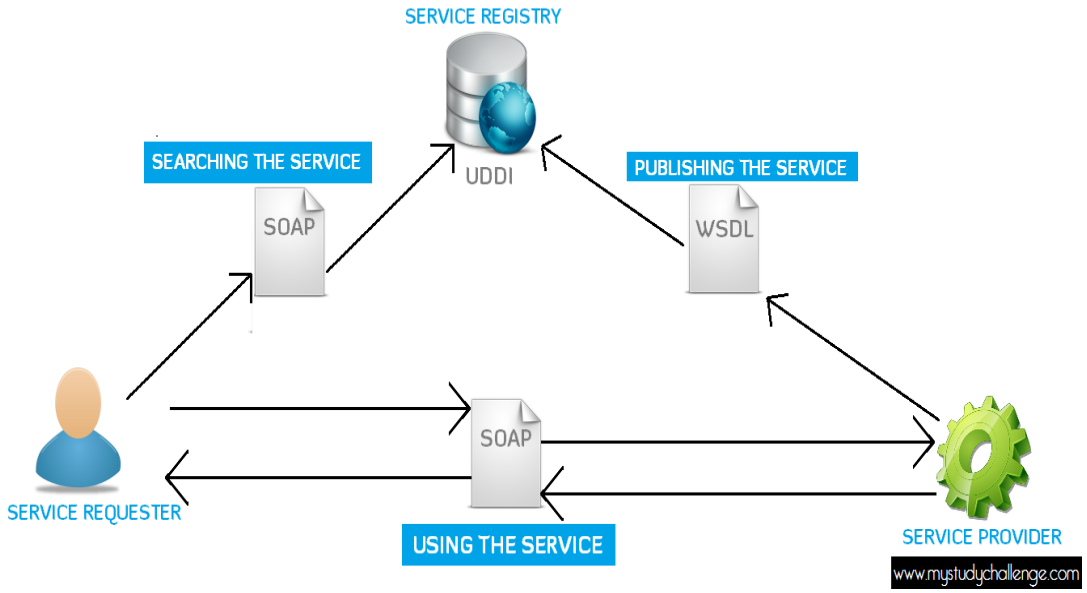
WebServices são definidos como unidades modulares de lógica de aplicação que proveem funcionalidades de negócio através de conexão com a Internet (SRIVASTAVA; KOEHLER, 2003). Conceitualmente, um serviço é um componente de software disponibilizado através de um *endpoint* acessível pela rede. Serviços provedores e consumidores utilizam mensagens para trocar informações dos tipos *request* e *response* na forma de arquivos autocontidos que fazem abstrações sobre as capacidades tecnológicas do recebedor (ORACLE, 2013).

WebServices podem ser implementados de várias formas. Oracle (2013) as dividiu em duas: *Big webservices* e *webservices RESTful*.

Big webservices integram aplicações web usando XML (*eXtensible Markup Language*), SOAP (*Simple Object Access Protocol*), WSDL (*Web Service Description Language*) e UDDI (*Universal Description Discovery and Integration*) através de comunicações

HTTP (*HyperText Transfer Protocol*). XML é usado para marcar os dados, SOAP é utilizado para a transferência dos dados, WSDL é usado para descrever os serviços disponíveis e UDDI é utilizado para escutar quais serviços estão disponíveis (GORTON, 2006). Uma das grandes vantagens de *Big webservices* é que, por terem interfaces bem definidas, a comunicação entre elementos se torna mais fácil, apesar de deixar o envio das mensagens mais custoso. A Figura 4 mostra o funcionamento de um *Big WebService*.

Figura 4 – Big WebServices.

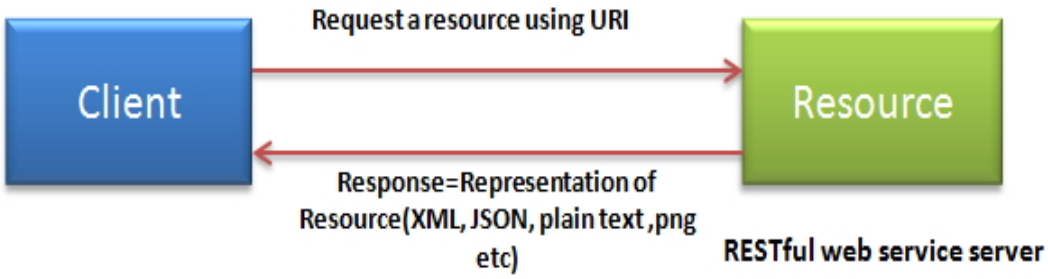


Fonte: My Study Challenge (2012)

O requisitante do serviço procura por um serviço na UDDI através de mensagens SOAP, o provedor do serviço publica o serviço na UDDI através da WSDL, em que são especificadas as interfaces do serviço. Por fim, temos a comunicação entre o requisitante e o provedor, onde ocorre a troca de dados através de mensagens SOAP.

Webservices RESTful oferecem serviços de forma mais leve do que os *Big webservices*, pois eles não precisam utilizar XML para o corpo das mensagens e nem WSDL para definir as interfaces, assim facilitando a interação com o protocolo HTTP. Esses *webservices* são baseados na arquitetura REST (*Representational State Transfer*) que especifica restrições que são aplicadas aos *webservices* para alcançar propriedades desejadas, como desempenho, escalabilidade e modificabilidade e assim garantindo um bom funcionamento do serviço (ORACLE, 2013). A Figura 5 mostra o funcionamento de um *WebService RESTful*.

Figura 5 – WebServices RESTful.



Fonte: Java tutorial for beginners (2013)

O cliente faz uma requisição ao serviço que retorna com uma resposta, o formato da resposta será feito pelo desenvolvedor do *WebService*, existem vários tipos, os principais são JSON e XML.

Na realização deste trabalho, para persistir e disponibilizar os dados foi decidido utilizar um *Webservice*, para o qual a placa enviará os dados coletados pelos sensores e a aplicação móvel irá consumir os dados disponibilizados pelo serviço. A abordagem utilizada neste trabalho será a *RESTful*, pois ela é mais leve e simples de se utilizar do que a *Big webservices*.

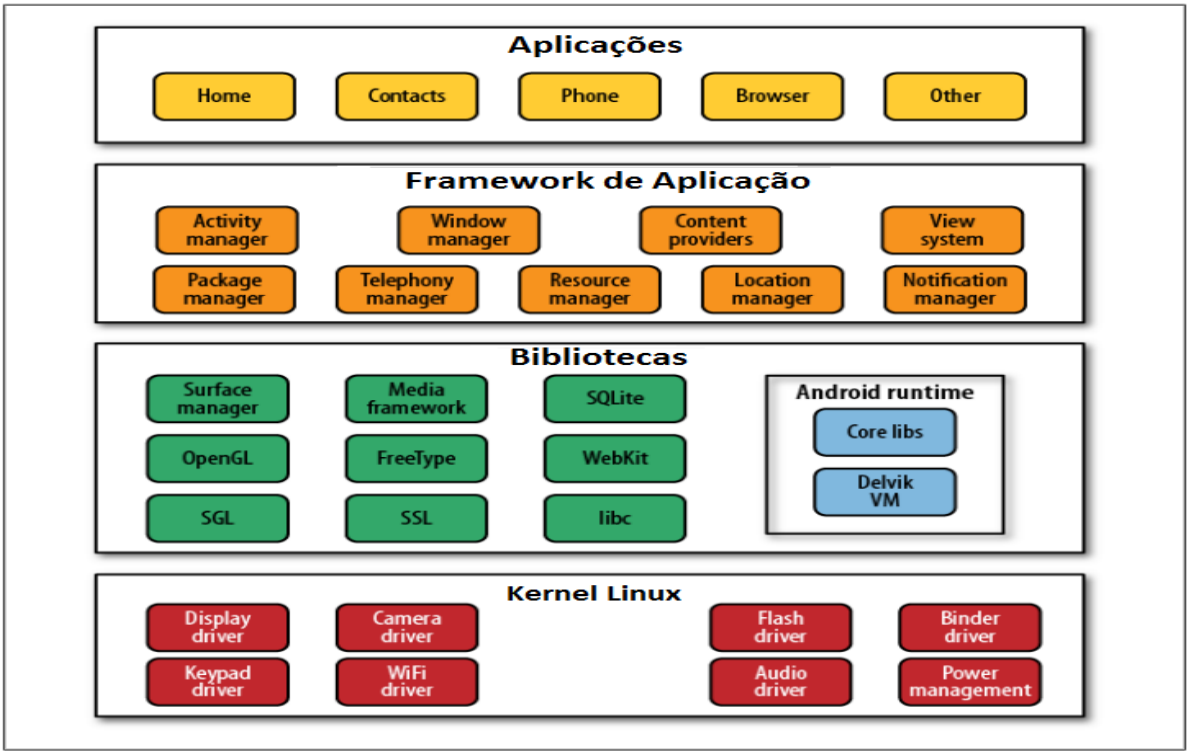
2.3.2 Android

Android é um sistema operacional para dispositivos móveis como *smart phones*, *tablets*, e notebooks. Android é desenvolvido atualmente pela Google e é baseado no Linux Kernel (SHANKER; LAL, 2011).

O Android foi criado por uma empresa chamada Android Inc., fundada por Andy Rubin, adquirida em 2005 pela Google. Entre a aquisição e o anúncio para o mundo em novembro de 2007, pouca ou quase nenhuma informação sobre o Android foi passada pela Google (YAGHMOUR, 2013). A primeira versão oficial do Android foi lançada em setembro de 2008, atualmente estamos na versão 6, conhecida como *Marshmallow*. O Android está entre as maiores plataformas mobile do mundo, estando presente em mais de 190 países (ANDROID DEVELOPERS, 2016).

A plataforma Android possui uma vasta quantidade de funcionalidades e características que podem ser utilizadas para desenvolvimento. Yaghmour (2013) cita algumas: framework de aplicação, maquina virtual Dalvik, SQLite, tecnologias de conexão e sensores. A Figura 6 mostra a arquitetura do Android.

Figura 6 – Arquitetura Android.



Fonte: Paulo (2014)

A camada mais baixa, Kernel Linux, fornece uma abstração entre o dispositivo de hardware e as camadas superiores da pilha de software do Android, além de ficar responsável pelo gerenciamento dos *drivers*. Bibliotecas possui as bibliotecas utilizadas para desenvolvimento, como por exemplo, biblioteca SQL utilizada para banco de dados. Ainda na camada Bibliotecas, temos a subcamada Android *Runtime* que é responsável por executar as aplicações no dispositivo android através de uma maquina virtual Dalvik. A camada Framework de Aplicação possui todos os recursos que o Android disponibiliza para a criação de uma aplicação. No topo da pilha, temos a camada Aplicações onde ficam as aplicações, tanto as nativas como navegador e contatos, quanto às criadas por outros.

Para desenvolver para Android o recomendado é utilizar a plataforma de desenvolvimento Android *Studio* que é a IDE oficial.

Na implementação deste trabalho será necessário dispor de uma forma de visualizar os dados, portanto foi decidida a criação de uma aplicação móvel onde será disponibilizado os dados para os usuários. Dentre as plataformas disponíveis, foi escolhida a plataforma Android, pois possui uma grande base de usuários e por ser *open source*.

2.3.3 Arduíno

Arduíno é uma plataforma de prototipagem eletrônica *open source* baseada em software e hardware flexíveis e fáceis de usar. Placas Arduíno são capazes de ler entradas (sensores) e transformar em saída (atuadores) (ARDUINO, 2016).

A placa Arduíno é uma pequena placa microcontroladora, ou seja, um circuito de pequeno porte, que contém um computador inteiro dentro de um pequeno chip (BANZI; SHILOH, 2015). A Figura 7 apresenta a estrutura de uma placa Arduino Uno.

Figura 7 – Estrutura de uma placa Arduino Uno.

Fonte: Silva, Silva e Lima (2015)

Nela é possível visualizar a CPU ATMEL composta por um microcontrolador ATmega328P, 6 entradas analógicas, 14 entradas e saídas digitais, conversor serial para USB, fonte de alimentação externa e os pinos de energia com 3,3 V, 5 V e Terra (GND) (SILVA; SILVA; LIMA, 2015).

Sensores e atuadores são componentes eletrônicos que permitem que um equipamento eletrônico interaja com o mundo. Quando as informações geradas pelos sensores são lidas, o microcontrolador tem o poder de decisão para executar uma ação ou não, as ações são executadas por meio dos atuadores.

Na realização deste trabalho, será necessário a utilização de uma placa para realizar o monitoramento das bicicletas. Foi então escolhido a placa Arduíno, pela sua facilidade de usar.

3 TRABALHOS RELACIONADOS

Torres et al. (2015) propõe uma funcionalidade de cálculo de rotas para a aplicação BeCity. O algoritmo utilizado para o cálculo combina as informações providas pelo o usuário e as informações das rotas da cidade para calcular a melhor rota, levando em consideração a distância da viagem, disponibilidade do caminho e segurança do caminho. A aplicação BeCity foi criada com o intuito de melhorar a vida dos ciclistas nas cidades. Torres et al. (2015) e este trabalho compartilham um objetivo em comum, que é saber as rotas percorridas por uma bicicleta, a grande diferença é que Torres et al. (2015) foca em pegar as rotas por usuário, já este trabalho foca em pegar as rotas realizadas pelas bicicletas. Uma desvantagem de Torres et al. (2015) é que para coletar dados é necessário utilizar a bicicleta com um smartphone com a aplicação BeCity instalada, já nesse trabalho basta utilizar a placa.

Em Vagnoli et al. (2014), é proposto um framework chamado SensorWebBike (SWB). O framework é divido em duas partes, a primeira é uma aplicação web responsável pela visualização dos dados, a segunda é o sensor, responsável pela coleta dos dados. Vagnoli et al. (2014) utilizam uma placa "AirQuino" que fica responsável por monitorar o ambiente (umidade, barulho, temperatura), qualidade do ar (CO, CO2, NO2), qualidade das ruas e os índices de bem-estar nos ambientes urbanos. Os autores embarcam a placa nas bicicletas, como propomos, mas os tipos de dados coletados pelos sensores que eles utilizam são voltados para medir a qualidade de vida, enquanto focamos em utilizar os sensores para coletar informações sobre as bicicletas. Outra diferença é em relação a visualização dos dados, neste trabalho à visualização será realizada através de um dispositivo móvel, já em Vagnoli et al. (2014) é realizada através de uma aplicação web.

Em Liu et al. (2015) é desenvolvido um dispositivo que mede a emissão de gás dos veículos motorizados e também a qualidade do ar nas estradas. O dispositivo (*Bicycle-Born Monitor* - BBM) consiste em um processador *single-chip*, um receptor microGPS, um detector de partículas, um sensor de gás de escape, um cartão microSD e um módulo Bluetooth. Esse dispositivo é embarcado em bicicletas do sistema de bicicletas públicas. Liu et al. (2015) e este trabalho possuem o objetivo comum de embarcar sensores em bicicletas. As diferenças são em relação aos objetivos finais: enquanto Liu et al. (2015) tem como objetivo monitorar a qualidade do ar das cidades através das bicicletas, este trabalho tem como objetivo o monitoramento das bicicletas. Uma desvantagem de Liu et al. (2015) é que somente bicicletas públicas são utilizadas, já nesse trabalho a placa pode ser embarcada tanto em bicicletas publicas, quanto privadas.

O Quadro 1 mostra as características do trabalha proposto e dos trabalhos relacionados.

Quadro 1 – Características

Características	**Torres et al. (2015)**	**Vagnoli et al. (2014)**	**Liu et al. (2015)**	**Trabalho Proposto**
Código aberto	Não	Não	Não	Sim
Visualização dos dados	Móvel	Web	Não informado	Móvel
Sensores	Celular	AirQuino	Dispositivo BBM	Arduíno/SIM808
Monitoramento	Bicicletas	Ambiente	Ambiente	Bicicletas

Em relação ao código aberto, nenhum dos trabalhos relacionados disponibiliza o código dos seus trabalhos. A visualização dos dados nesse trabalho e em Torres et al. (2015) é realizada através de dispositivos móveis, Vagnoli et al. (2014) faz a visualização através de uma aplicação web, já em Liu et al. (2015) não é informado como é feita a visualização dos dados.

Torres et al. (2015) utiliza o celular como sensor para pegar as informações da bicicleta, Vagnoli et al. (2014) utiliza uma placa AirQuino, que contém sensores de umidade, barulho, temperatura e outros, para monitorar o ambiente. Liu et al. (2015) utiliza o dispositivo BBM para realizar o monitoramento. Esse trabalho utiliza o Arduíno com o módulo SIM808 para a coleta e envio de dados.

O objetivo de monitoramento nos trabalhos de Vagnoli et al. (2014) e Liu et al. (2015) é o ambiente, em que eles utilizam as bicicletas para monitorar o ambiente no qual elas estão inseridas, já em Torres et al. (2015) e nesse trabalho o objetivo do monitoramento são as bicicletas.

4 PROPOSTA

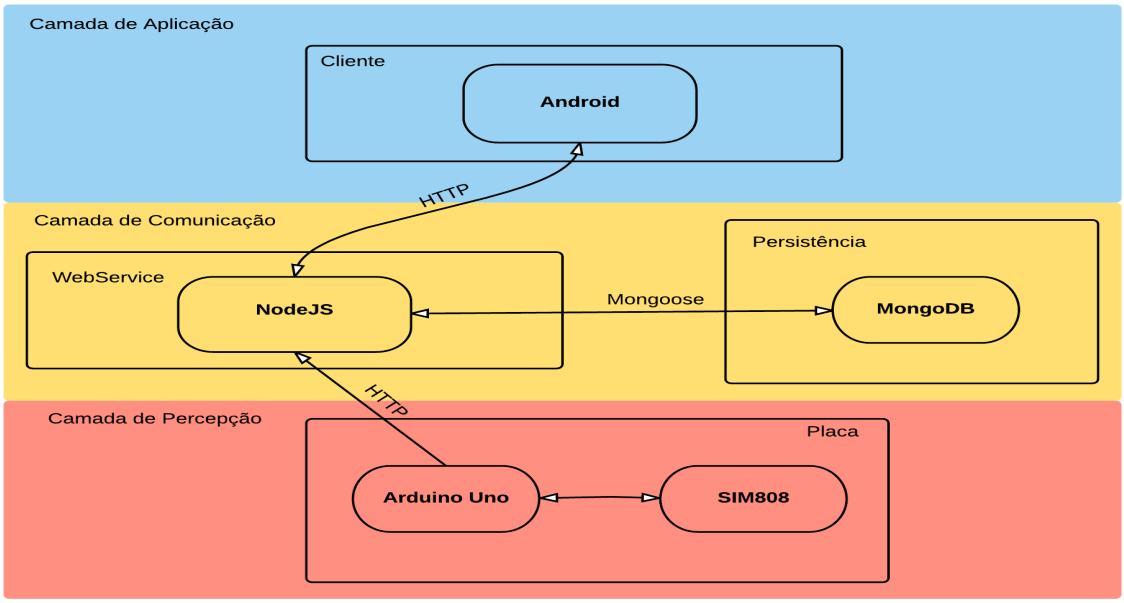
Esse trabalho propõe uma aplicação para o monitoramento de bicicletas urbanas através de sensores. Esses sensores coletam as localizações das bicicletas e envia para um servidor, que persisti e disponibiliza os dados obtidos. Um dos objetivos secundários desse trabalho é disponibilizar o servidor como um serviço, em que qualquer um pode fazer requisições diretas ao servidor, para a obtenção de dados. Para a visualização das informações será disponibilizada uma aplicação Android.

Na seção 4.1 iremos abordar os módulos do projeto, a seção 4.2 irá descrever a arquitetura da aplicação, mostrando algumas de suas visões, na seção 4.3 será mostrado a API para a utilização do serviço, por fim, temos a seção 4.4 que contém informações sobre a placa.

4.1 Módulos do Projeto

O projeto é dividido em três módulos: Cliente [1], *WebService* [2] e a Placa [3]. Nessa seção iremos abordar esses módulos, suas tecnologias e como ocorre a comunicação entre eles. A Figura 8 ilustra a representação arquitetural do sistema, baseada na arquitetura de referência, representada de forma simplificada na Figura 2.

Figura 8 – Módulos do Sistema



Fonte: Elaborada pelo autor.

[1] <https://github.com/Cayk/smb-app>
[2] <https://github.com/Cayk/smb-server>
[3] <https://github.com/Cayk/smb-arduino>

O cliente foi implementado na plataforma Android, e se comunica com o servidor através do protocolo HTTP, os dados enviados nessas requisições são representados na notação JSON. No servidor foi utilizado o NodeJS, plataforma de desenvolvimento baseada em JavaScript, para a criação das rotas de acesso aos dados e o MongoDB, banco de dados orientado a documentos, para a persistência desses dados. A comunicação entre o NodeJS e MongoDB é realizada através do Mongoose. Por fim, temos a placa, em que os dados são enviados para o servidor através de requisições HTTP, a comunicação entre o Arduíno Uno e o SIM808 é realizada via hardware.

As subseções a seguir abordam de maneira mais detalhada cada um dos módulos do projeto em relação às suas tecnologias.

4.1.1 Cliente

O módulo cliente é responsável pela visualização e manipulação dos dados. Ele se comunica com o servidor através de requisições HTTP.

Para esse projeto, foi definido que o cliente seria uma aplicação móvel. Dentre as plataformas de desenvolvimento móvel mais conhecidas, Android, iOS e Windows Phone, para a implementação deste trabalho foi escolhido a plataforma Android. A escolha do Android leva em consideração vários fatores, como por exemplo, possuir uma maior presença de mercado, onde o Android representa 79% do mercado de dispositivos móveis. A Figura 9 traz um comparativo entre Android, iOS e Windows Phone.

Figura 9 – Comparação entre plataformas de desenvolvimento móvel

Questão	Android	iOS	Windows Phone
Linguagem de desenvolvimento	Java	Objective-C	.Net C#
Disponibilidade de debug	Excelente	Muito Boa	Excelente
Velocidade de deploy	Relativamente rápida	Rápida	Relativamente rápida
Suporte e comunidade de desenvolvimento	Muito grande	Muito grande	Média
Presença de mercado	Muito grande	Grande	Limitada
Disponibilidade de IDEs	Excelente – suportado por todas as principais IDEs	Suporte muito bom através da Apple Xcode e Jetbrains Appcode	Muito bom, porém limitada ao Microsoft Visual Studio
Custo das ferramentas de desenvolvimento	Grátis	Necessita de computador com iOS para desenvolvimento	Grátis para emular

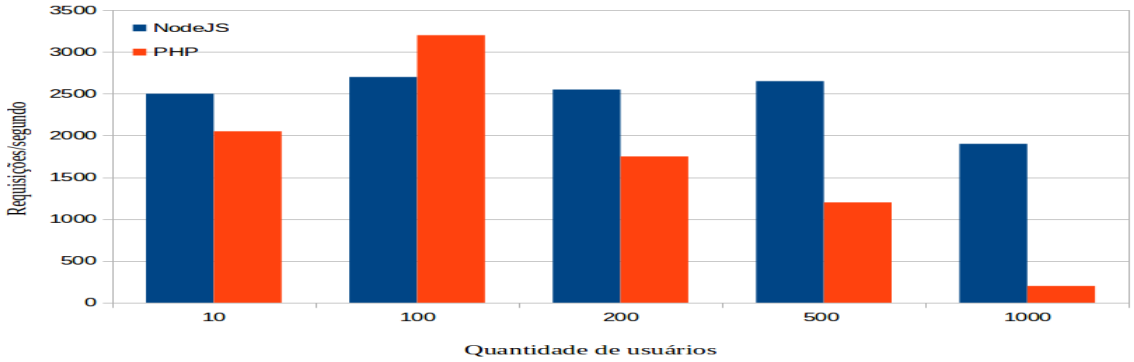
Fonte: Adaptada de Grønli et al. (2014).

4.1.2 WebService

O módulo *WebService* é responsável por salvar e disponibilizar os dados obidos da Placa. Ele se comunica com a Placa através de requisições HTTP para a coleta de dados, também realiza conexões HTTP com o cliente Android para a manipulação desses dados. Outro papel do *WebService* é ser um serviço, em que sistemas de terceiros podem se comunicar com o servidor sem necessariamente utilizar o cliente criado nesse trabalho.

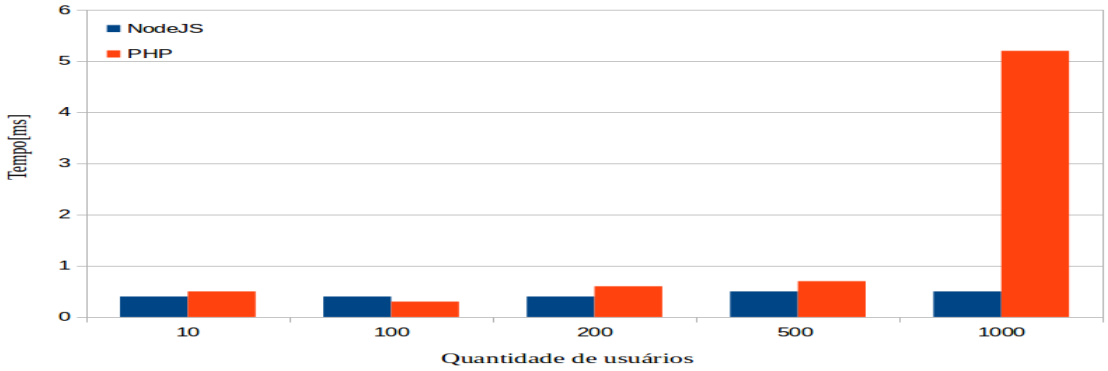
Na hora de escolher a linguagem de programação que seria utilizada para a criação do servidor, houve um dúvida entre utilizar Php ou JavaScript (mais precisamente o NodeJs), pois as duas linguagens trabalham nativamente com o formato de dados JSON, que foi utilizado para a troca de dados entre a aplicação móvel e o servidor. Para a tomada de decisão, foi considerado o estudo realizado por Nobre (2016) sobre o desempenho das duas linguagens. A Figura 10 mostra a quantidade de requisições atendidas por segundo, já a Figura 11 mostra o tempo gasto no atendimento da requisição.

Figura 10 – Desempenho: Requisições por segundo



Fonte: Adaptado de Nobre (2016).

Figura 11 – Desempenho: Tempo de cada requisição em milissegundos



Fonte: Adaptado de Nobre (2016).

Com base nas informações obtidas nas figuras 10 e 11, percebemos que o NodeJS tem um desempenho melhor que o PHP, sendo assim escolhida como a linguagem de programação utilizada para a construção do servidor.

Em relação ao banco de dados, foi escolhido o MongoDB, que é um banco não relacional, orientado a documentos e que trabalha nativamente com objetos JSON. Além disso, oferece esquemas dinâmicos, em que mudanças nas estruturas do banco podem ser realizadas sem maiores dificuldades. Tendo em mente as características do MongoDB, concluímos que ele possui alta compatibilidade com aplicações NodejS.

O *WebService* está hospedado na Amazon. Essa escolha foi realizada por dois fatores específicos: i) Conhecimento prévio sobre como montar um servidor Amazon e ii) Os incentivos dados à alunos de universidades, em que alguns serviços disponibilizados não precisam ser pagos.

4.1.3 Placa

O módulo Placa é responsável pela coleta dos dados(localização) no ambiente. Ele possui dois componentes, Arduíno que realiza o envio dos dados para o servidor e o módulo SIM800 que coleta as localizações.

A placa escolhida foi a Arduíno Uno, pelo fato de ser uma placa bastante conhecida e com isso consegue-se um maior suporte da comunidade, pois muitos dos exemplos e bibliotecas encontradas na internet são voltadas ou compatíveis com o Arduíno Uno. O módulo SIM808 foi escolhido pois ele possui tanto o sensor de GPS para pegar a localização, quanto a antena GPRS para realizar a conexão com o servior.

4.2 Arquitetura da Aplicação

Esta seção apresenta a arquitetura proposta para a aplicação Android de monitoramento de bicicletas urbanas. O objetivo é mostrar as decisões arquiteturais que foram tomadas em relação ao sistema. As próximas subseções irão mostrar as visões arquiteturais que abordadas no projeto e as restrições da arquitetura.

4.2.1 Representação arquitetural

A aplicação foi desenvolvida com base no padrão arquitetural Camadas. Nesse padrão, funcionalidades parecidas são agrupadas em uma mesma camada, em que cada camada possui um papel específico e responsabilidades dentro da aplicação (SILVA; PAULA, 2001). Por exemplo, a camada de Apresentação é responsável por lidar com as interações/requisições dos usuários com a aplicação, já a camada Lógica é responsável por executar as regras de negocio associadas com as requisições. Uma das principais vantagens em utilizar esse padrão é que cada camada da arquitetura forma uma abstração de como o trabalho está sendo realizado. Voltando ao exemplo mostrado anteriormente, a camada de Apresentação não precisa se preocupar em como os dados do usuário são gerados, ela simplesmente precisa pegar e mostrar aquela informação na tela.

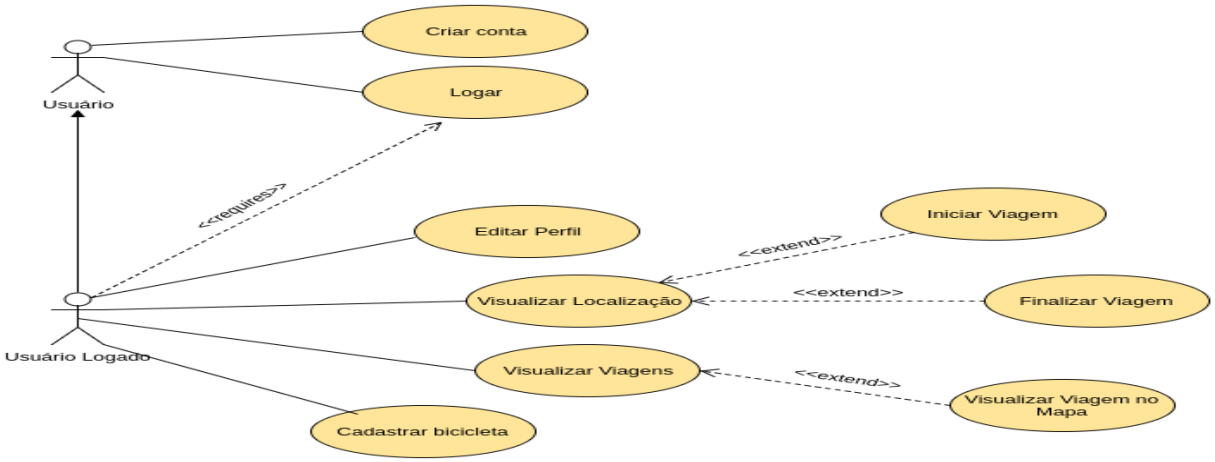
4.2.2 Restrições da Aplicação

Para a execução da aplicação, é necessário que o dispositivo do usuário possua um sistema operacional Android com versão 16 (*Jelly Bean*) ou maior; também é necessário que o dispositivo tenha conexão com a internet. Caso nenhum dos requisitos sejam cumpridos a aplicação não irá funcionar.

4.2.3 Visão de Casos de Uso

Nessa seção serão mostrados os casos de uso da aplicação. A Figura 12 mostra o usuário e as funcionalidades que podem ser realizadas na aplicação.

Figura 12 – Casos de uso da aplicação



Fonte: Elaborada pelo autor.

Para utilizar os recursos da aplicação, o usuário precisa estar logado no sistema. Para que as funcionalidades de iniciar e finalizar uma viagem funcionem é necessário visualizar a localização da bicicleta.

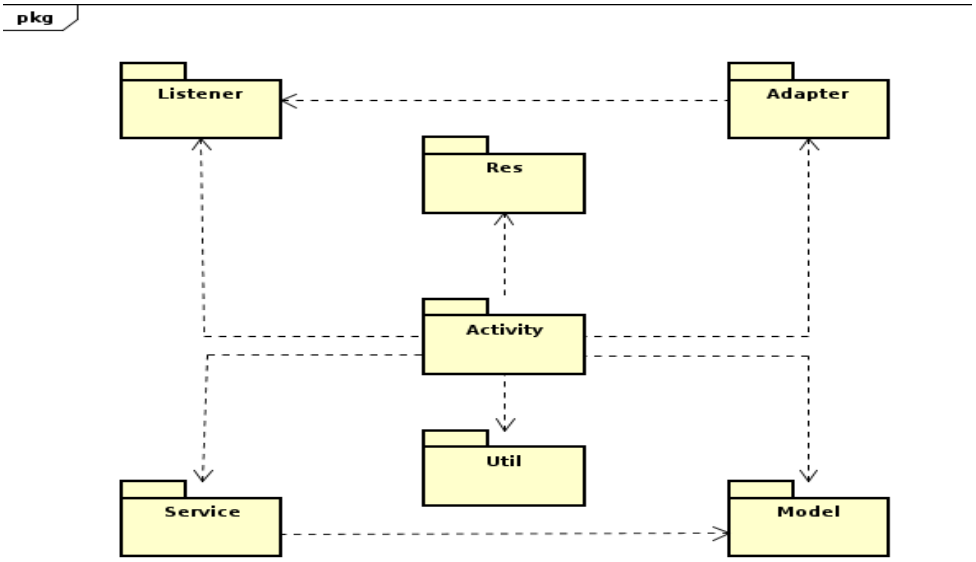
4.2.4 Visão Lógica

Está seção apresenta as visões lógicas de pacotes e classes da aplicação, para que se possa visualizar como funciona a comunicação entre esses elementos.

4.2.4.1 Visão de Pacotes

A Figura 13 mostra a estrutura de pacotes da aplicação e como eles se comunicam.

Figura 13 – Visão Lógica de Pacotes



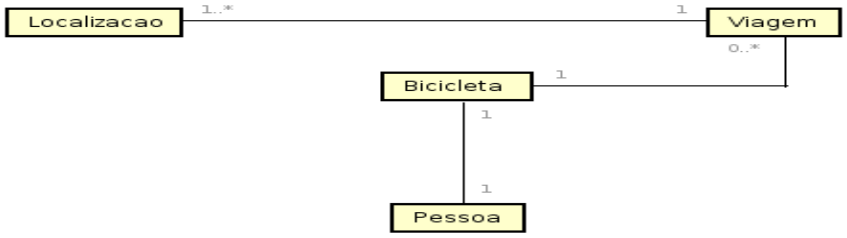
Fonte: Elaborada pelo autor.

O pacote **Listener** contém a interface necessária para que seja possível realizar ações ao clicar em um item de uma *RecyclerView*. O pacote **Adapter** possui o adaptador responsável por montar as listas do tipo *RecyclerView*. No pacote **Res** temos vários sub pacotes de estilo que são necessários para gerar a visualização do sistema. O pacote **Activity** é responsável por realizar a comunicação entre a aplicação e o servidor. No pacote **Service** temos um serviço que funciona em *background* e fica recebendo as localizações da bicicleta. No pacote **Util** temos uma classe que realiza a validação dos dados, por exemplo, se um campo está vázio ou não. Por fim, temos o pacote **Model** que possui as classes do sistema.

4.2.4.2 Visão de Classes

A Figura 14 mostra as classes de modelo da aplicação e seus relacionamentos.

Figura 14 – Visão de classes do modelo



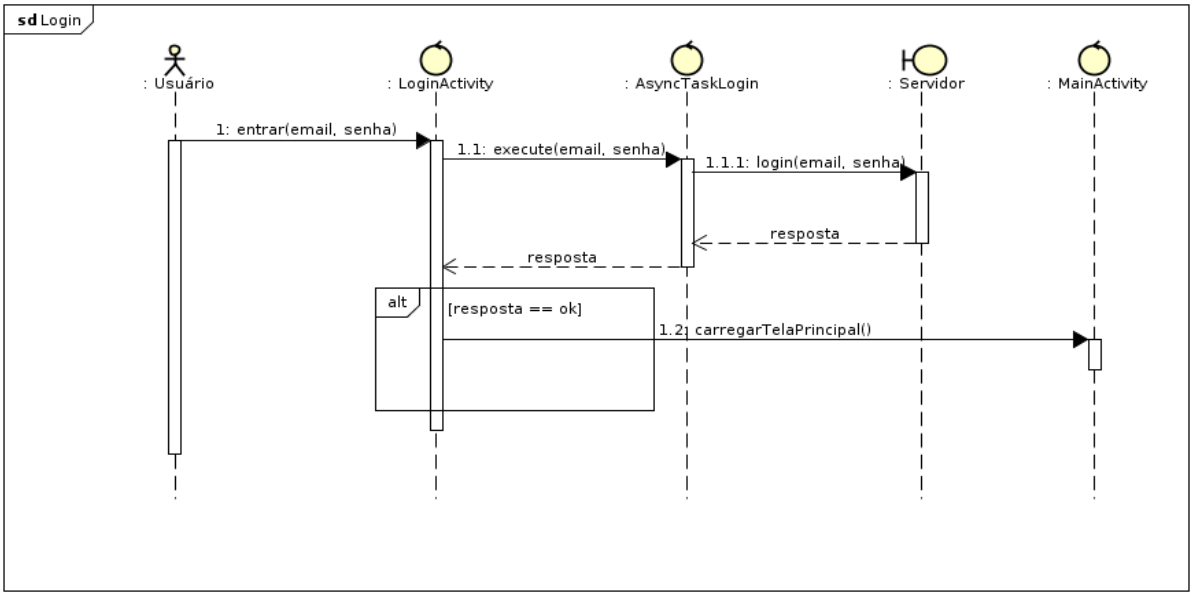
Fonte: Elaborada pelo autor.

São 4 classes: Localizacao Viagem, Bicicleta e Pessoa. Em que uma Pessoa tem uma Bicicleta, que pode ter nenhuma ou várias viagens, e uma Viagem pode ter uma ou mais Localizações.

4.2.5 Visão de Processos

Esta seção descreve a decomposição do sistema em processos, sendo evidenciado os principais processos do ponto de vista arquitetural. A Figura 15 mostra o processo de Login na aplicação.

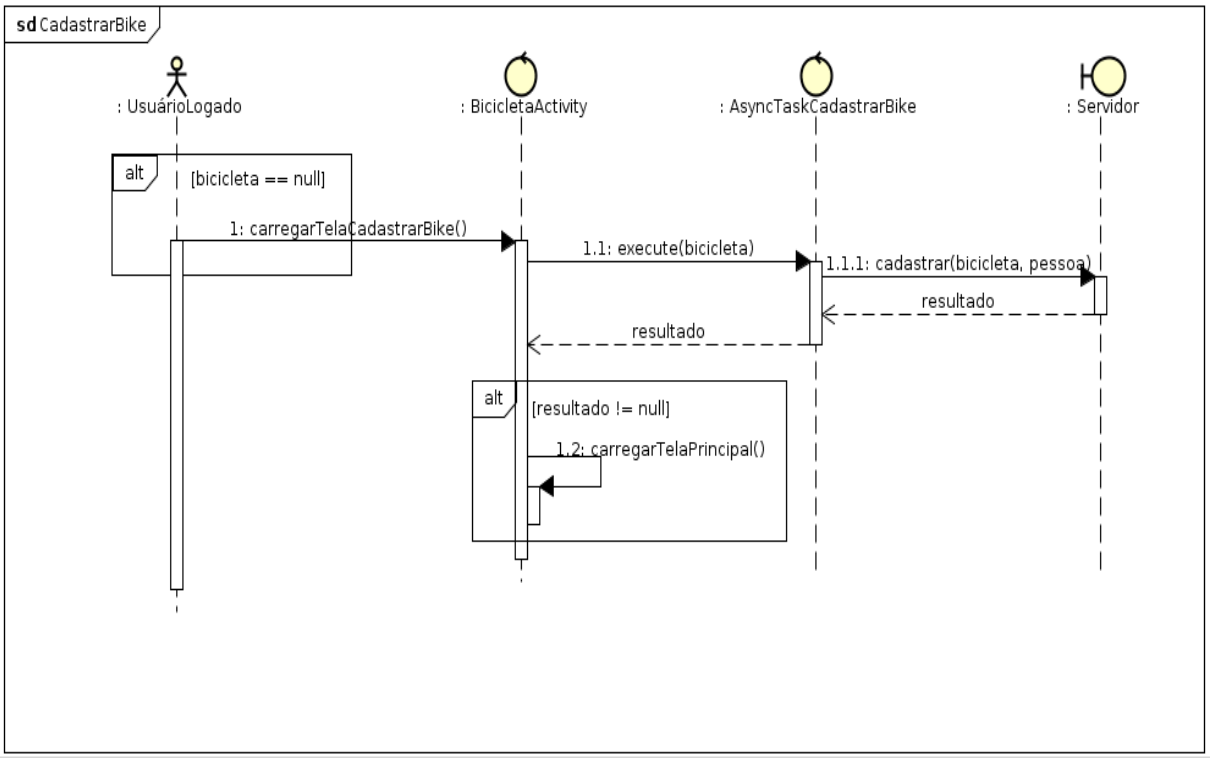
Figura 15 – Processo de Login



Fonte: Elaborada pelo autor.

A *activity* de login passa os dados informados pelo usuário para a sua classe privada AsyncTaskLogin, que manda uma requisição para o servidor com os dados; se tudo estiver correto, o método para carregar a tela principal da aplicação é chamado. A próxima figura mostra o processo de cadastro de uma bicicleta na aplicação.

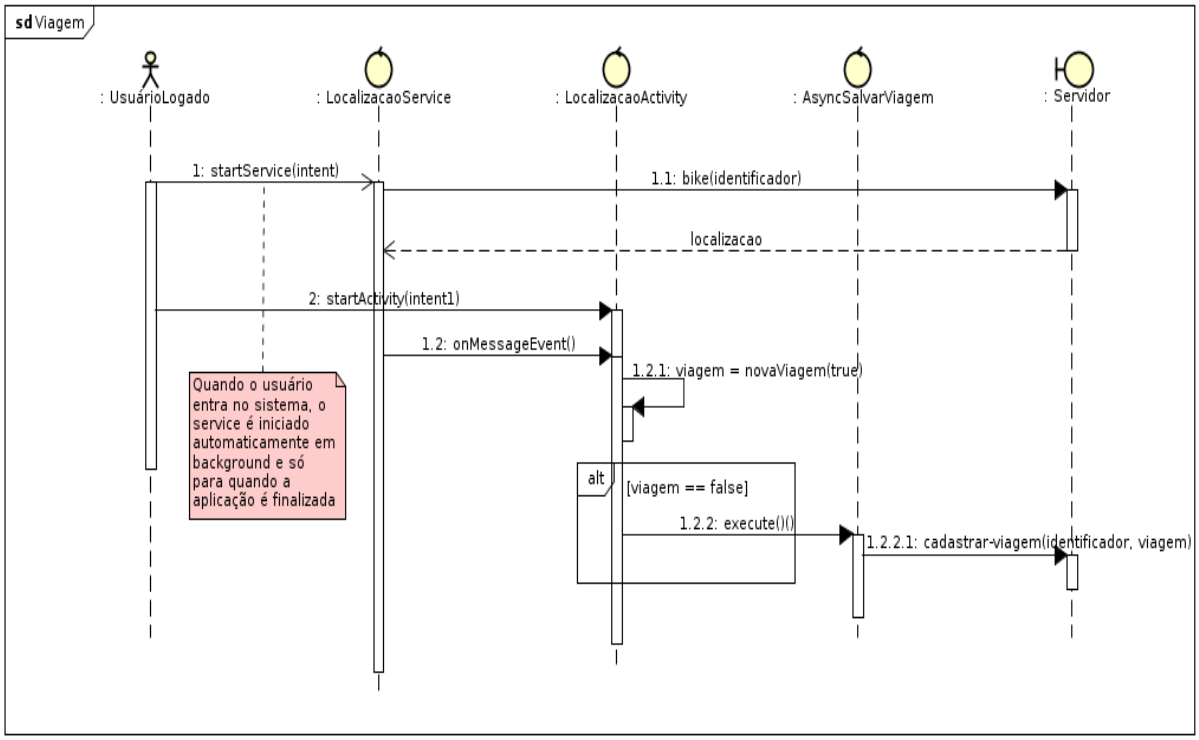
Figura 16 – Processo de cadastro de uma bicicleta



Fonte: Elaborada pelo autor.

Quando o usuário entra pela primeira vez no sistema, ele não possui qualquer bicicleta cadastrada; logo ao invés de ir para a tela principal, o usuário será redirecionado para a tela de cadastrar bicicleta. Na tela de cadastrado, ele irá informar o identificador da bicicleta, que será passado para a AsyncTaskCadastrarBike, que irá mandar uma requisição para o servidor cadastrar aquele identificador. Após o cadastro da bicicleta, o usuário será redirecionado para a tela principal da aplicação. A Figura 17 mostra o processo para a criar e finalizar uma viagem.

Figura 17 – Processo de criação e finalização de uma viagem



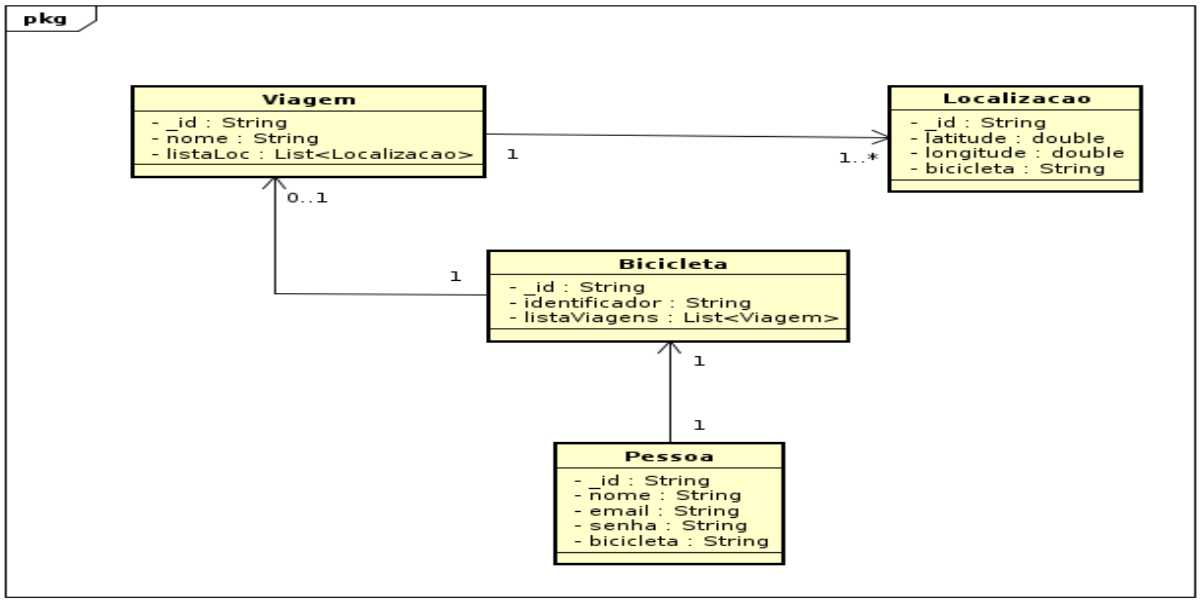
Fonte: Elaborada pelo autor.

Assim que o usuário entra no sistema, um serviço é iniciado em *background*. Esse serviço é responsável por ficar requisitando a localização da bicicleta ao servidor, e ele só é finalizado quando o ciclo de vida da aplicação termina. Para iniciar uma viagem, o usuário precisa estar na tela de localização. Essa tela que está vinculada com a LocalizacaoActivity que, quando iniciada, fica recebendo localizações do *service* através do método onMessageEvent(). Ao finalizar uma viagem, a classe privada AsyncTaskSalvarViagem é chamada, ela envia uma requisição para o servidor salvar uma nova viagem, passando como parâmetro o identificador da bicicleta e uma lista de localizações.

Visão de Implementação

Está seção descreve a estrutura geral do modelo de implementação. A Figura 18 mostra todas as entidades de modelo da aplicação, com seus respectivos atributos.

Figura 18 – Entidades do modelo



Fonte: Elaborada pelo autor.

4.2.6 Visão de Dados

As coleções de documentos JSON que irão ser utilizadas no sistema são: Pessoas, Bicicletas e Localizacoes. A Figura 19 mostra os esquemas de dados que são utilizados para definir os atributos mínimos que um documento deve ter, tendo em vista que, por utilizarmos o MongoDB, os documentos de cada coleção podem ter mais atributos, se necessário.

Figura 19 – Modelo de Dados

```
var schema = mongoose.Schema;

var PessoaSchema = schema({
  nome:{type:String, require:true},
  senha:{type:String, require:true},
  email:{type:String, require:true},
  bicicleta:{type:String, require:false}
}, {versionKey: false});

var LocalizacaoSchema = schema({
  latitude:{type:Number, require:true},
  longitude:{type:Number, require:true},
  bicicleta:{type:String, require:true}
}, {versionKey: false});

var BicicletaSchema = schema({
    identificador:{type:String, require:true},
    listaViagens:[{
      nome:{type:String, require:true},
      listaLoc:[{
        latitude:{type:Number, require:true},
        longitude:{type:Number, require:true},
        time:{type:Number, require:false}
      }]
    }]
}, {versionKey: false});
```

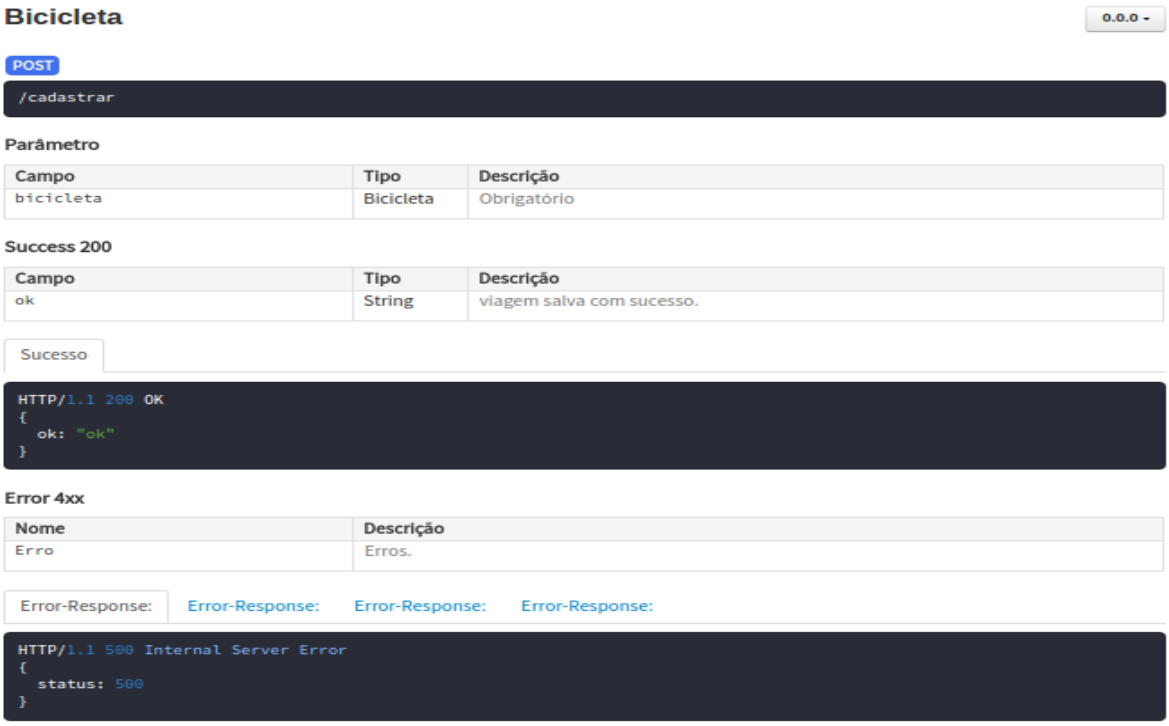
Fonte: Elaborada pelo autor.

4.3 Serviço

Um dos objetivos do trabalho é criar um serviço, em que qualquer pessoa/sistema teria acesso aos dados obtidos, sem ter a necessidade de usar a aplicação Android desenvolvida nesse trabalho.

Para ter acesso aos dados, o requisitante precisa utilizar as rotas de acesso que o serviço disponibiliza. Existem três tipos de rotas: Pessoa, Bicicleta e Localizacao. A primeira rota é destinada somente ao sistema que está sendo desenvolvido nesse trabalho, pois essa rota lida com as informações dos usuários da aplicação. As rotas Bicicleta e Localizacao são as que estão disponíveis para qualquer pessoa utilizar. Nessas rotas encontram-se informações sobre a localização das bicicletas, suas viagens etc. A Figura 20 mostra o método para cadastrar uma bicicleta no servidor.

Figura 20 – API do Serviço



Fonte: Elaborada pelo autor.

Os demais métodos disponibilizados pelo serviço estão descritos no Apêndice A - API SMB Service.

4.4 Código Arduíno

Nesta sub-seção será explicado o código utilizado no Arduíno para a obtenção da localização e seu envio para o servidor. O Código-fonte 1 mostra os métodos setup e loop do arduíno.

Código-fonte 1 – Métodos Setup e Loop

```
1  void setup(){
2    Serial.begin(9600);
3    if (gsm.begin(2400)) {
4      Serial.println(F("\nstatus=READY"));
5      //Garantir que esteja ligado
6      gsm.forceON();
7      started=true;
8    }else
9      Serial.println(F("\nstatus=IDLE"));
10 }
11
12 void loop(){
13   //Pegar localizacao
14   gps_enviar();
15   delay(10000);
16   //Enviar os dados para o servidor
17   enviarMensagem();
18   delay(10000);
19 }
```

No *setup*, é realizada a configuração inicial do dispositivo. O método forceOn() é invocado para garantir que o GSM não fique em modo de recarga. No *loop*, ocorre a chamada aos métodos gps_enviar() e enviarMensagem(); essas chamadas ficam ocorrendo até que a placa seja desligada. O Código-fonte 2 mostra o método gps_enviar() responsável por pegar a localização da bicicleta.

Código-fonte 2 – Método para pegar localização

```
void gps_enviar(){
  if(started){
    if (gps.attachGPS()){
      Serial.println(F("status=GPSREADY"));
    }
    else
      Serial.println(F("status=ERROR"));

    delay(20000); //Time for fixing

    stat=gps.getStat();
    if(stat==1){
      Serial.println(F("NOT FIXED"));
    else if(stat==0)
      Serial.println(F("GPS OFF"));
    else if(stat==2)
      Serial.println(F("2D FIXED"));
    else if(stat==3)
      Serial.println(F("3D FIXED"));

    delay(5000);
    gps.getPar(lon,lat,alt,time,vel);
  }
}
```

Primeiro é verificado se o Arduíno está iniciado, depois é verificado se o recurso está disponível. Após essas etapas de verificação, um *delay* é dado para que o módulo SIM808 consiga pegar a localização. Após o *delay*, o método getStat() é executado e retorna com o estado do módulo. Se o retorno for igual a 1, significa que o SIM808 não foi acessado; retorno igual 0 significa que o módulo foi acessado, porém não conseguiu pegar a localização; retorno igual a 2 ou 3 representa que o módulo está funcionando e que ele conseguiu pegar a localização. Por fim,

o método getPar é chamado para pegar os dados obtidos.

O Código-fonte 3 mostra de forma simplificada o método enviarMensagem(), responsável por enviar os dados obtidos para o servidor.

Código-fonte 3 – Método para enviar os dados para o servidor

```
1  void enviarMensagem(){
2    if(started){
3      if (inet.attachGPRS("timbrasil.br", "tim", "tim"))
4        Serial.println(F("status=ATTACHED"));
5      else
6        Serial.println(F("status=ERROR"));
7
8      delay(1000);
9      //Read IP address.
10     gsm.SimpleWriteln("AT+CIFSR");
11
12     delay(5000);
13     inet.httpGET("35.161.237.63", 3000, enviar, msg, 50);
14   }
15 }
```

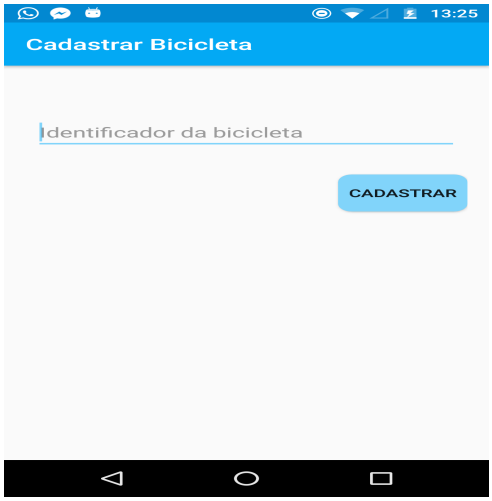
Nesse método também é verificado se o recurso está disponível e se o Arduíno foi iniciado, após essas etapas, uma requisição é feita ao servidor através do método httpGET(), nesse método é passado o IP do servidor, a porta e a mensagem com os dados.

5 RESULTADOS E DISCUSSÃO

Nesta seção serão mostrados os resultados do uso da aplicação, os problemas encontrados durante a execução e as considerações sobre as informações obtidas.

A seguir, serão mostradas imagens com a aplicação Android em execução. A Figura 21 mostra a tela de cadastrar bicicleta.

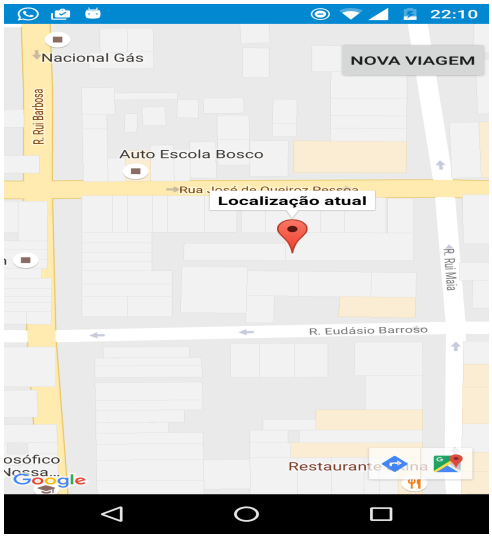
Figura 21 – Tela Cadastrar Bicicleta



Fonte: Elaborada pelo autor.

Ao entrar na aplicação pela primeira vez, é solicitado que o usuário cadastre uma bicicleta no sistema, para que, assim, o servidor possa receber as localizações. A Figura 22 mostra a tela de localização da bicicleta.

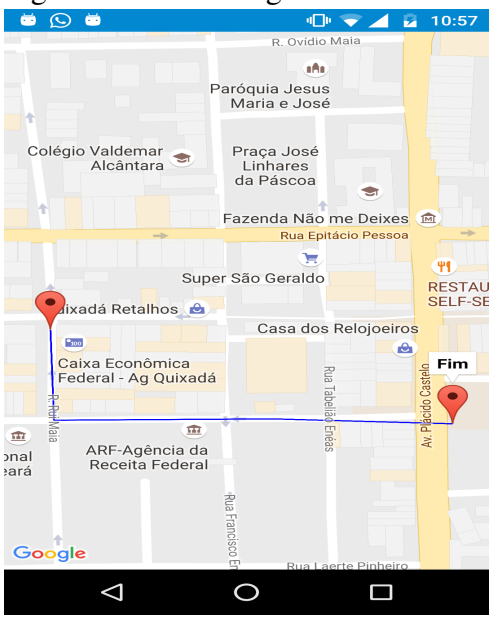
Figura 22 – Tela Localização



Fonte: Elaborada pelo autor.

Nessa tela, é mostrada a localização atual da bicicleta, para iniciar ou finalizar uma viagem, basta clicar no botão. Ao longo do uso da aplicação, foi identificado um pequeno problema: ao iniciar uma viagem, se o usuário sair da tela de localização a viagem é encerrada. Até o momento, nenhuma solução para esse problema foi encontrada, porém, para o uso da aplicação, esse problema não causa muitas perdas porque, se o usuário continuar na tela de localização do inicio ao fim da viagem, nenhum erro ocorrerá. A Figura 23 mostra a tela de uma viagem.

Figura 23 – Tela Viagem



Fonte: Elaborada pelo autor.

Nessa tela, podemos ver todos os locais por onde o usuário passou durante uma viagem. O caminho do Início ao Fim é formado por todos os pontos de localização que a placa pegou e enviou para o servidor.

5.1 Lições Aprendidas

Durante a implementação do Arduíno, alguns problemas foram encontrados. Inicialmente, foi prevista a utilização do MQTT, protocolo que realiza a troca de mensagens entre aplicações com base no padrão arquitetural *publish/subscriber* para a comunicação entre a placa e o servidor. Porém, como o MQTT é baseado no TCP/IP (VICENZI, 2015), seu uso em conexões de baixa qualidade como o GSM se torna muito complexo, portanto, foi utilizada a biblioteca inetGSM para a comunicação com o servidor.

Outro problema encontrado é em relação ao SIM808, módulo responsável por pegar a localização da bicicleta: se a placa estiver em local fechado as chances de não conseguir pegar uma localização válida são grandes.

O sinal de celular também tem grande influência no funcionamento da aplicação, visto que é utilizado um chip no módulo para que a placa consiga se conectar a internet. Assim, é importante sempre andar por áreas com um sinal bom, pois, do contrário, a placa não conseguirá enviar os dados para o servidor.

5.2 Considerações sobre os Resultados

Os problemas encontrados durante a execução do trabalho tiveram um grande impacto no funcionamento e desempenho da aplicação. O fato do SIM808 só funcionar corretamente em locais abertos e com um bom sinal telefônico, limita o potencial de uso da aplicação, pois ela restringe os locais onde a aplicação pode funcionar, geralmente áreas urbanas.

O desempenho da aplicação com a utilização do Arduíno para a coleta dos dados não foi satisfatório, pois a placa demora muito tempo para poder realizar todo o processo de obtenção da localização e envio para o servidor.

Esse processo leva geralmente entre 35-50 segundos para ocorrer, o que é inviável. Somente para que a placa consiga pegar uma localização válida são necessários 20 segundos de espera; então, mesmo que o código seja otimizado, pelo menos 20 segundos serão necessários para que a operação seja realizada.

Conclui-se que, para conseguir pegar informações sobre as bicicletas de forma mais eficiente, seria necessário a utilizações de outras tecnologias mais sofisticadas.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Esse trabalho desenvolveu um sistema de monitoramento de bicicletas urbanas, além disso, foi disponibilizado um serviço, em que os dados gerados podem ser acessados por qualquer pessoa ou sistema.

Ao longo do trabalho foram abordados os principais conceitos por trás do projeto. Também foi mostrado todas as tecnologias envolvidas no processo de implementação do sistema, no qual foi buscado o uso de tecnologias conhecidas, atuais e que trouxessem contribuições positivas para o projeto.

Para melhor entendimento da aplicação, foi criado uma Arquitetura do projeto, onde são mostradas varias visões arquiteturais, entre elas, visão lógica, de processos e de caso de uso. Além de melhorar o entendimento, o modelo arquitetural do sistema ajuda na manutenção futura do código e no intendimento do código por terceiros.

Após a implementação, foi apresentado a aplicação em funcionamento, mostrando assim, que é possível criar um projeto com as tecnologias escolhidas que realize o monitoramento de bicicletas. Por fim, podemos concluir que os resultados apresentados são satisfatórios.

Ao final de todas as etapas de desenvolvimento do trabalho, foram observadas algumas possibilidades de trabalho futuros. A seguir temos uma lista com possíveis trabalhos.

1. Otimizar o código do Arduíno para que a coleta dos dados e comunicação com o servidor seja feita de forma mais rápida.
2. Utilizar outros sensores e comparar com os utilizados nesse trabalho, para saber qual o melhor tipo de tecnologia para o desenvolvimento de trabalhos parecidos.
3. Aumentar o número de funcionalidades oferecidas pela aplicação Android, por exemplo, histórico de viagens, principais rotas utilizadas e tempo de uma viagem.
4. Permitir que os usuários possam interagir com redes sociais através da aplicação.
5. Realizar uma avaliação de usabilidade da aplicação, para saber o nível da aplicação em relação à interação com os usuário.

REFERÊNCIAS

ANDROID DEVELOPERS. **Android, the world's most popular mobile platform**. [S.l.], 2016. Disponível em: <https://developer.android.com/about/index.html>. Acesso em: 16 maio 2016.

ARDUINO. **What is Arduino ?** [S.l.], 2016. Disponível em: <https://www.arduino.cc/en/Guide/Introduction>. Acesso em: 27 jun. 2016.

ASGHAR, M. H.; MOHAMMADZADEH, N.; NEGI, A. Principle application and vision in internet of things (iot). In: IEEE. **Computing, Communication & Automation (ICCCA), 2015 International Conference on**. Noida, 2015. p. 427–431.

BANZI, M.; SHILOH, M. **Primeiros Passos com o Arduino–2ª Edição**. São Paulo: Novatec Editora, 2015.

BASSI, A.; BAUER, M.; FIEDLER, M.; KRAMP, T.; KRANENBURG, R. V.; LANGE, S.; MEISSNER, S. Enabling things to talk. **Designing IoT Solutions With the IoT Architectural Reference Model**, Springer, São Paulo, p. 163–211, 2013.

BUREAU, U. S. C. **A Look at the Nearly 1 Million Who Ride Their Bikes to Work in the U.S.** [S.l.], 2016. Disponível em: <http://blogs.census.gov/2016/05/19/a-look-at-the-nearly-1-million-who-ride-their-bikes-to-work-in-the-u-s/>. Acesso em: 20 nov. 2016.

COETZEE, L.; EKSTEEN, J. The internet of things-promise for the future? an introduction. In: IEEE. **IST-Africa Conference Proceedings, 2011**. Gaborone, 2011. p. 1–9.

ERICSSON. **Ericsson Mobility Report: On The Pulse of The Network Society**. [S.l.], 2016. Disponível em: <https://www.ericsson.com/res/docs/2016/ericsson-mobility-report-2016.pdf>. Acesso em: 19 jun. 2016.

GORTON, I. **Essential software architecture**. São Paulo: Springer Science & Business Media, 2006.

GRØNLI, T.-M.; HANSEN, J.; GHINEA, G.; YOUNAS, M. Mobile application platform heterogeneity: Android vs windows phone vs ios vs firefox os. In: IEEE. **2014 IEEE 28th International Conference on Advanced Information Networking and Applications**. [S.l.], 2014. p. 635–641.

GUBBI, J.; BUYYA, R.; MARUSIC, S.; PALANISWAMI, M. Internet of things (iot): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, Elsevier, Amsterdã, v. 29, n. 7, p. 1645–1660, 2013.

HEMALATHA, D.; AFREEN, B. E. Development in rfid (radio frequency identification) technology in internet of things (iot). **International Journal of Advanced Research in Computer Engineering & Technology**, v. 4, n. 11, 2015.

JAVA TUTORIAL FOR BEGINNERS. **RESTful web service tutorial**. [S.l.], 2013. Disponível em: <http://www.java2blog.com/2013/04/restful-web-service-tutorial.html>. Acesso em: 11 maio 2016.

JUELS, A. Rfid security and privacy: A research survey. **IEEE journal on selected areas in communications**, IEEE, v. 24, n. 2, p. 381–394, 2006.

LIU, X.; LI, B.; JIANG, A.; QI, S.; XIANG, C.; XU, N. A bicycle-borne sensor for monitoring air pollution near roadways. In: IEEE. **Consumer Electronics-Taiwan (ICCE-TW), 2015 IEEE International Conference on**. Taipei, 2015. p. 166–167.

MAIA, M. E.; FONTELES, A.; NETO, B.; GADELHA, R.; VIANA, W.; ANDRADE, R. Loccam-loosely coupled context acquisition middleware. In: ACM. **Proceedings of the 28th Annual ACM Symposium on Applied Computing**. Coimbra, 2013. p. 534–541.

MIDGLEY, P. Bicycle-sharing schemes: enhancing sustainable mobility in urban areas. **United Nations, Department of Economic and Social Affairs**, Nova York, p. 1–12, 2011.

MIORANDI, D.; SICARI, S.; PELLEGRINI, F. D.; CHLAMTAC, I. Internet of things: Vision, applications and research challenges. **Ad Hoc Networks**, Elsevier, Amsterdã, v. 10, n. 7, p. 1497–1516, 2012.

MY STUDY CHALLENGE. **All you need to know about web services**. [S.l.], 2012. Disponível em: <http://mysc.altervista.org/all-you-need-to-know-about-web-services/>. Acesso em: 11 maio 2016.

NAIR, K.; KULKARNI, J.; WARDE, M.; DAVE, Z.; RAWALGAONKAR, V.; GORE, G.; JOSHI, J. Optimizing power consumption in iot based wireless sensor networks using bluetooth low energy. In: IEEE. **Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on**. Noida, 2015. p. 589–593.

NOBRE, P. S. d. O. **VOIPBOX – Uma Solucão de Gerenciamento VOIP para Universidade Federal do Ceará Campus Quixadá**. Ceará: Universidade Federal do Ceará (UFC), 2016.

ORACLE. **The Java EE 6 Tutorial**. [S.l.], 2013. Disponível em: <http://docs.oracle.com/javaee/6/tutorial/doc/giqsx.html>. Acesso em: 09 maio 2016.

PAULO, J. V. d. **Desenvolvimento de um aplicativo Android e de uma interface bluetooth para um dinamômetro biomédico**. São Paulo: Universidade Estadual Paulista (UNESP), 2014.

PIRAMUTHU, O. B.; ZHOU, W. Bicycle sharing, social media, and environmental sustainability. In: IEEE. **2016 49th Hawaii International Conference on System Sciences (HICSS)**. Koloa, 2016. p. 2078–2083.

RAZZAQUE, M.; CLARKE, S. A security-aware safety management framework for iot-integrated bikes. In: IEEE. **Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on**. Milan, 2015. p. 92–97.

RAZZAQUE, M.; CLARKE, S. Smart management of next generation bike sharing systems using internet of things. In: IEEE. **Smart Cities Conference (ISC2), 2015 IEEE First International**. Guadalajara, 2015. p. 1–8.

SHAHEEN, S.; GUZMAN, S.; ZHANG, H. Bikesharing in europe, the americas, and asia: past, present, and future. **Transportation Research Record: Journal of the Transportation Research Board**, Transportation Research Board of the National Academies, Washington, n. 2143, p. 159–167, 2010.

SHANKER, A.; LAL, S. Android porting concepts. In: IEEE. **Electronics Computer Technology (ICECT), 2011 3rd International Conference on**. Kanyakumari, 2011. v. 5, p. 129–133.

SILVA, J. T.; SILVA, J. T.; LIMA, G. F. Controle e monitoramento de nível utilizando plataforma open source arduino. **Revista INNOVER**, Maranhão, v. 1, n. 4, p. 85–92, 2015.

SILVA, L. F. da; PAULA, V. C. C. de. Um meta-modelo para especificação de arquiteturas de software em camadas. 2001.

SRIVASTAVA, B.; KOEHLER, J. Web service composition-current solutions and open problems. In: **ICAPS 2003 workshop on Planning for Web Services**. [S.l.: s.n.], 2003. v. 35, p. 28–35.

TORRES, S.; LALANNE, F.; CANTO, G. del; MORALES, F.; BUSTOS-JIMÉNEZ, J.; REYES, P. Becity: sensing and sensibility on urban cycling for smarter cities. In: IEEE. **2015 34th International Conference of the Chilean Computer Science Society (SCCC)**. Santiago, 2015. p. 1–4.

VAGNOLI, C.; MARTELLI, F.; FILIPPIS, T. D.; LONARDO, S. D.; GIOLI, B.; GUALTIERI, G.; MATESE, A.; ROCCHI, L.; TOSCANO, P.; ZALDEI, A. The sensorwebbike for air quality monitoring in a smart city. In: IET. **Future Intelligent Cities, IET Conference on**. Londres, 2014. p. 1–4.

VICENZI, A. **BUSTRACKER: Sistema de rastreamento para transporte coletivo**. Blumenau: Universidade Regional de Blumenau, 2015.

WANT, R. An introduction to rfid technology. **Pervasive Computing, IEEE**, IEEE, Nova York, v. 5, n. 1, p. 25–33, 2006.

WHITMORE, A.; AGARWAL, A.; XU, L. D. The internet of things—a survey of topics and trends. **Information Systems Frontiers**, Springer, v. 17, n. 2, p. 261–274, 2015.

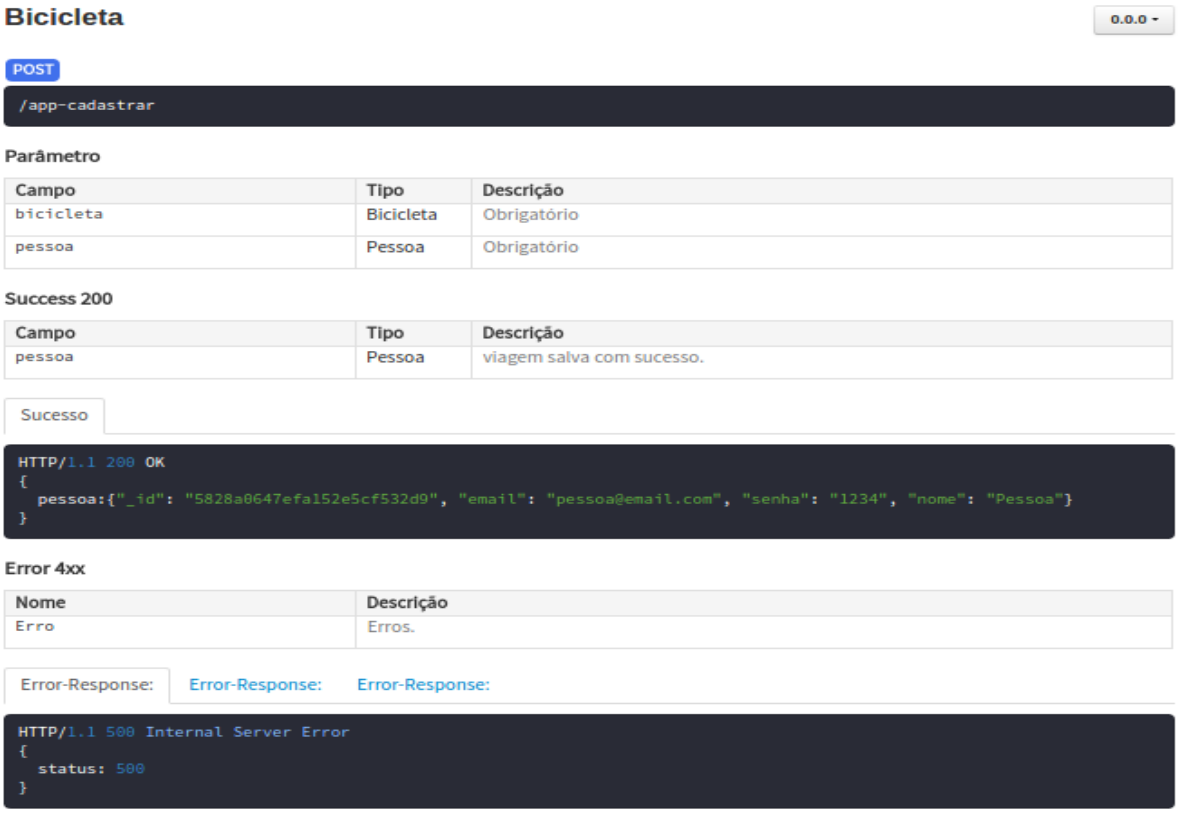
YAGHMOUR, K. **Embedded Android: Porting, Extending, and Customizing**. [S.l.]: "O'Reilly Media, Inc.", 2013.

APÊNDICE A – API SMB Service

Neste apêndice será mostrado os métodos oferecidos pelas rotas Bicicleta e Localizacao para acesso aos dados. Como dito na seção 4.3, a rota Pessoa não entra no escopo do serviço, tendo em vista que ela lida com os dados dos usuários da aplicação Android.

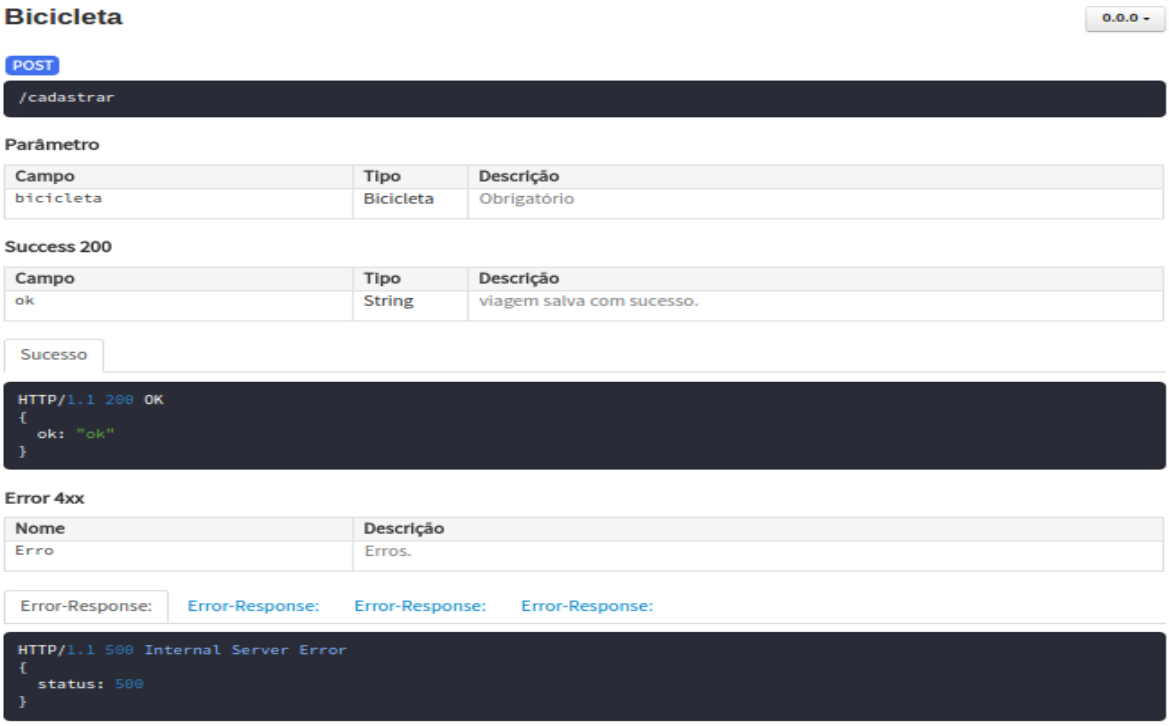
Rota Bicicleta

Figura 24 – Cadastrar Bicicleta via Aplicação



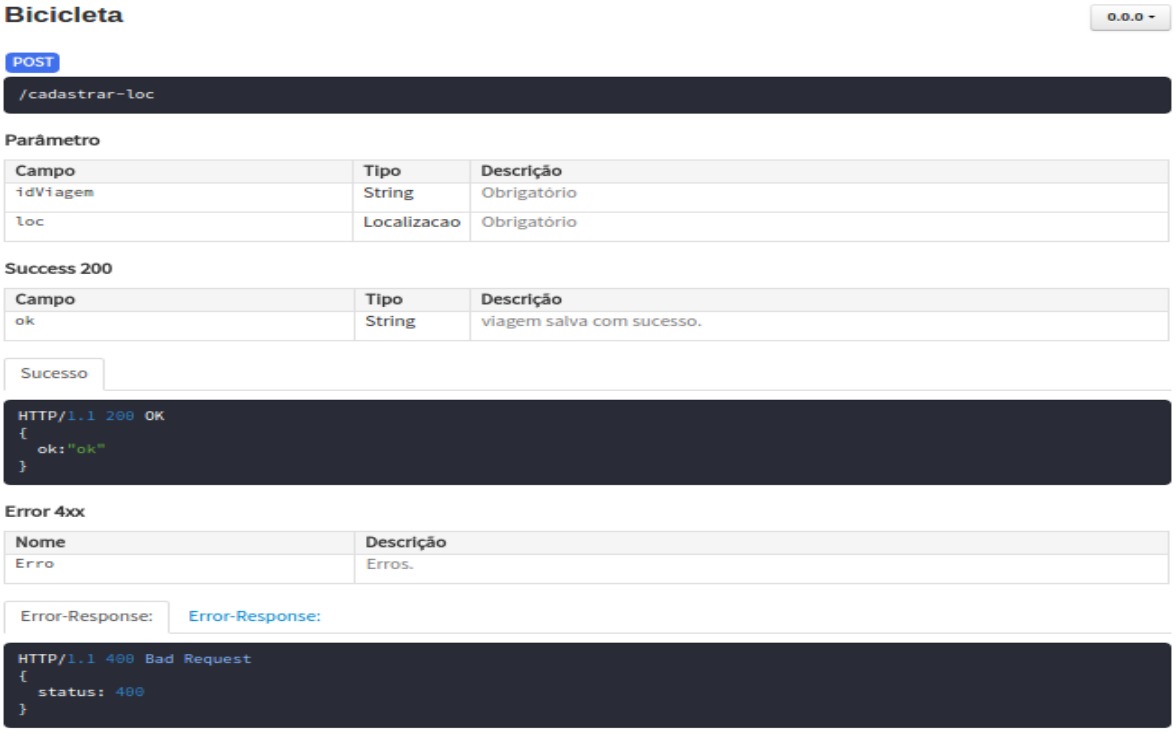
Fonte: Elaborada pelo autor.

Figura 25 – Cadastrar Bicicleta



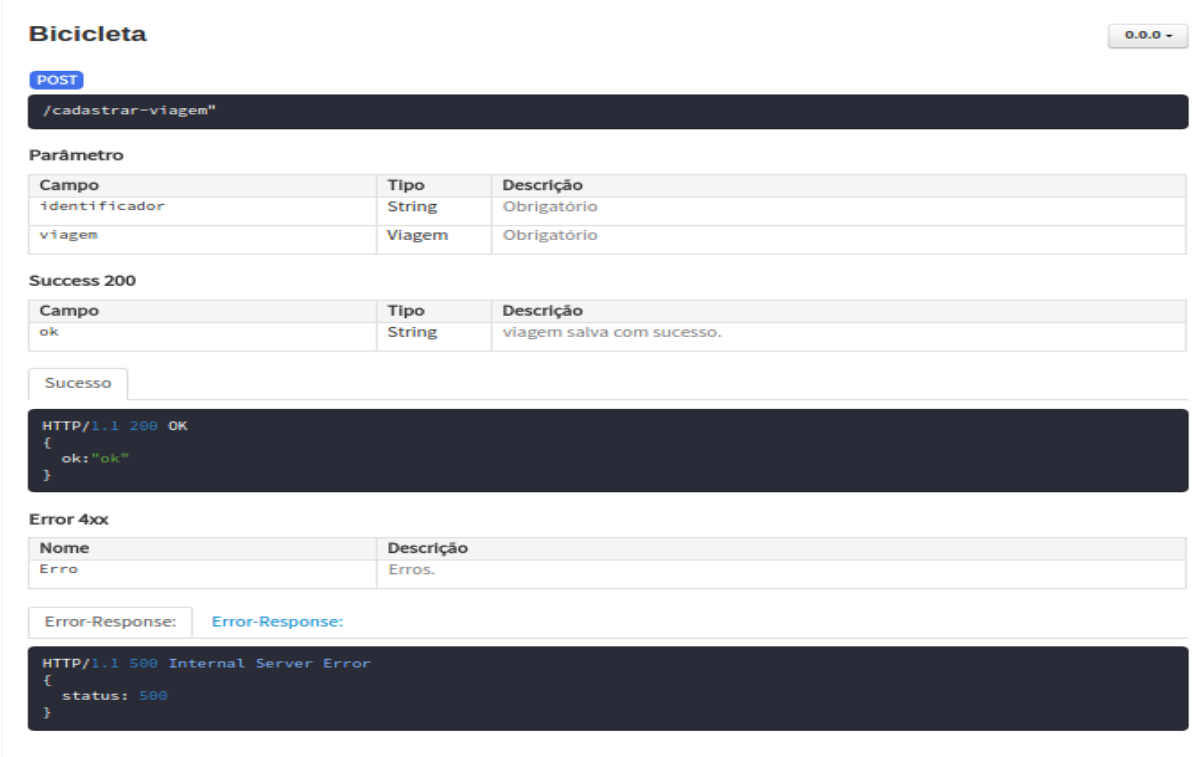
Fonte: Elaborada pelo autor.

Figura 26 – Cadastrar Localização em uma viagem



Fonte: Elaborada pelo autor.

Figura 27 – Cadastrar Viagem



Bicicleta 0.0.0

POST

`/cadastrar-viagem"`

Parâmetro

Campo	Tipo	Descrição
identificador	String	Obrigatório
viagem	Viagem	Obrigatório

Success 200

Campo	Tipo	Descrição
ok	String	viagem salva com sucesso.

Sucesso

```
HTTP/1.1 200 OK
{
  ok:"ok"
}
```

Error 4xx

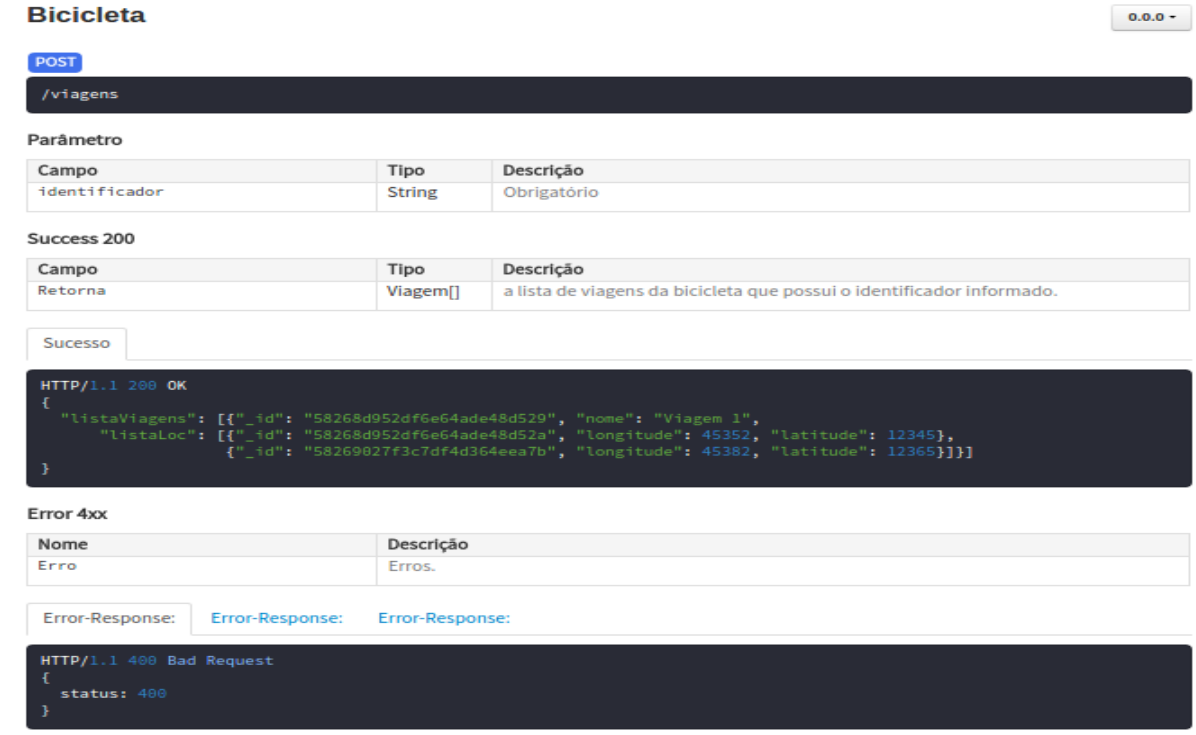
Nome	Descrição
Erro	Erros.

Error-Response: Error-Response:

```
HTTP/1.1 500 Internal Server Error
{
  status: 500
}
```

Fonte: Elaborada pelo autor.

Figura 28 – Viagens de uma bicicleta



Bicicleta 0.0.0

POST

`/viagens`

Parâmetro

Campo	Tipo	Descrição
identificador	String	Obrigatório

Success 200

Campo	Tipo	Descrição
Retorna	Viagem[]	a lista de viagens da bicicleta que possui o identificador informado.

Sucesso

```
HTTP/1.1 200 OK
{
  "listaViagens": [{"_id": "58268d952df6e64ade48d529", "nome": "Viagem 1",
      "listaLoc": [{"_id": "58268d952df6e64ade48d52a", "longitude": 45352, "latitude": 12345},
                   {"_id": "58269027f3c7df4d364eea7b", "longitude": 45382, "latitude": 12365}]}]
}
```

Error 4xx

Nome	Descrição
Erro	Erros.

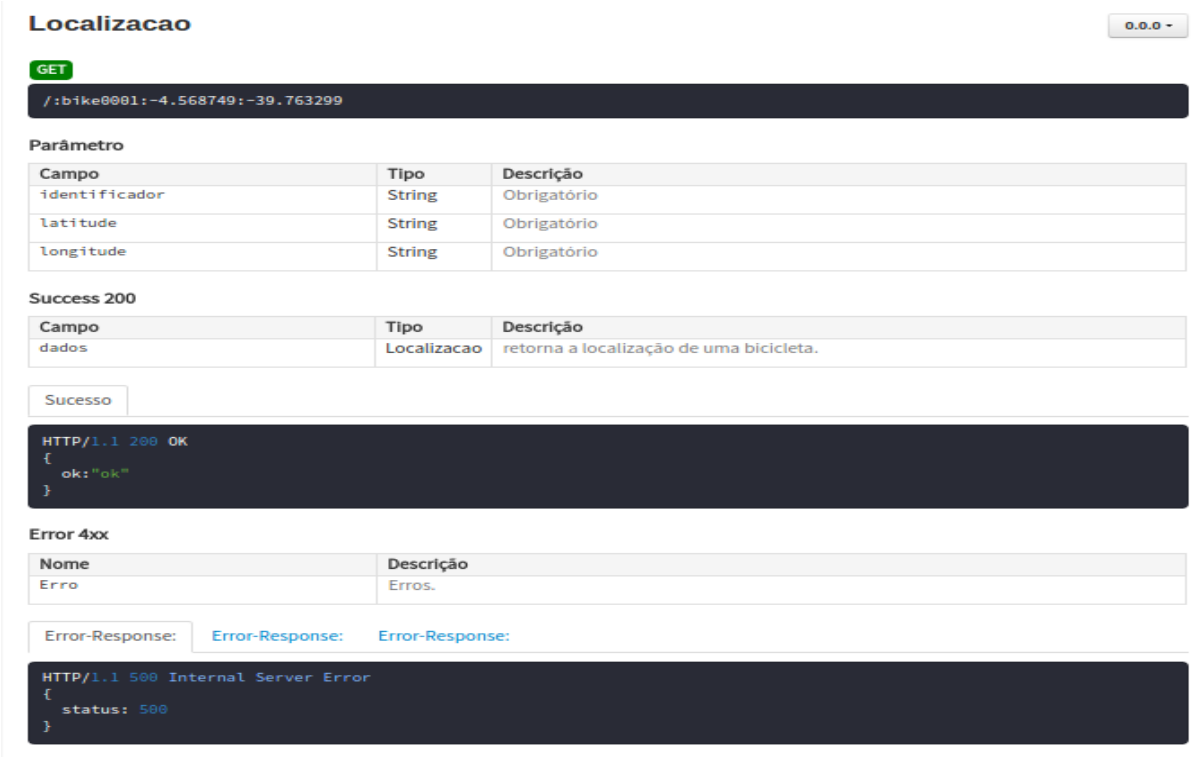
Error-Response: Error-Response: Error-Response:

```
HTTP/1.1 400 Bad Request
{
  status: 400
}
```

Fonte: Elaborada pelo autor.

Rota Localizacao

Figura 29 – Cadastrar localização da bicicleta



Localizacao

0.0.0

GET

/:bike0001:-4.568749:-39.763299

Parâmetro

Campo	Tipo	Descrição
identificador	String	Obrigatório
latitude	String	Obrigatório
longitude	String	Obrigatório

Success 200

Campo	Tipo	Descrição
dados	Localizacao	retorna a localização de uma bicicleta.

Sucesso

```
HTTP/1.1 200 OK
{
  ok:"ok"
}
```

Error 4xx

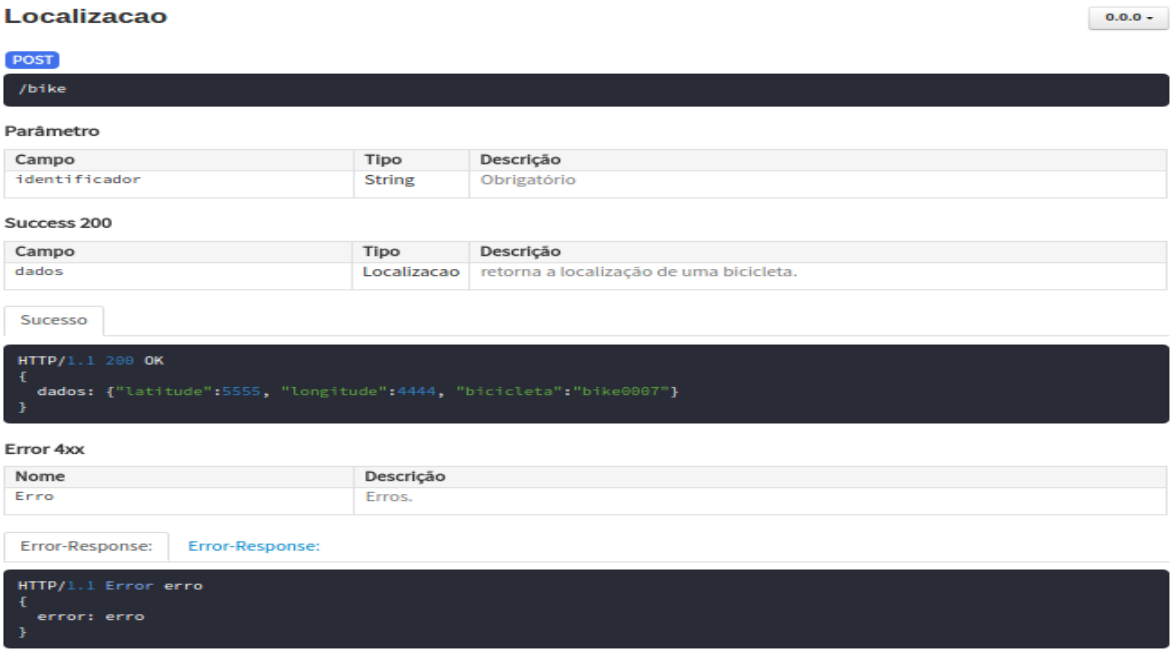
Nome	Descrição
Erro	Erros.

Error-Response: Error-Response: Error-Response:

```
HTTP/1.1 500 Internal Server Error
{
  status: 500
}
```

Fonte: Elaborada pelo autor.

Figura 30 – Buscar localização da bicicleta



Localizacao

0.0.0

POST

/bike

Parâmetro

Campo	Tipo	Descrição
identificador	String	Obrigatório

Success 200

Campo	Tipo	Descrição
dados	Localizacao	retorna a localização de uma bicicleta.

Sucesso

```
HTTP/1.1 200 OK
{
  dados: {"latitude":5555, "longitude":4444, "bicicleta":"bike0007"}
}
```

Error 4xx

Nome	Descrição
Erro	Erros.

Error-Response: Error-Response:

```
HTTP/1.1 Error erro
{
  error: erro
}
```

Fonte: Elaborada pelo autor.