Marcel Popolin de Araújo Cunha

MyBatRecommender: Otimização automatizada do consumo de energia em smartphones Android em nível de software

Sorocaba, SP

Fevereiro de 2016

Marcel Popolin de Araújo Cunha

MyBatRecommender: Otimização automatizada do consumo de energia em smartphones Android em nível de software

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação (PPGCCS) da Universidade Federal de São Carlos como parte dos requisitos exigidos para a obtenção do título de Mestre em Ciência da Computação. Área de concentração: Engenharia de Software e Gestão do Conhecimento.

Universidade Federal de São Carlos – UFSCar

Centro de Ciências em Gestão e Tecnologia – CCGT

Programa de Pós-Graduação em Ciência da Computação – PPGCCS

Orientador: Profa. Dra. Luciana Aparecida Martinez Zaina

Sorocaba, SP

Fevereiro de 2016

Popolin de Araújo Cunha, Marcel

MyBatRecommender: Otimização automatizada do consumo de energia em smartphones Android em nível de software / Marcel Popolin de Araújo Cunha. -- 2016.
139 f. : 30 cm.

Dissertação (mestrado)-Universidade Federal de São Carlos, campus Sorocaba, Sorocaba
Orientador: Luciana Aparecida Martinez Zaina
Banca examinadora: Alexandre Álvaro, Graça Bressan
Bibliografia

1. Gerenciamento de energia em smartphones Android. I. Orientador. II. Universidade Federal de São Carlos. III. Título.

Ficha catalográfica elaborada pelo Programa de Geração Automática da Secretaria Geral de Informática (SIn).
DADOS FORNECIDOS PELO(A) AUTOR(A)

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências em Gestão e Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Marcel Popolin de Araújo Cunha, realizada em 22/02/2016:

Profa. Dra. Luciana Aparecida Martinez Zaina
UFSCar

Profa. Dra. Graça Bressan
USP

Prof. Dr. Alexandre Alvaro
UFSCar

Aos meus pais, exemplo maior de vida.

Agradecimentos

Agradeço,

Aos meus pais Haroldo e Maria Lúcia pelo exemplo de vida a ser seguido e amor incondicional.

A minha família, amigos e namorada por todo apoio e momentos vividos.
A Profa. Dra. Luciana Zaina por todos ensinamentos e paciência durante todos esses anos de estudo.
Ao Venturus Centro de Inovação Tecnológica pelo incentivo e horas cedidas durante essa jornada.
A UFSCar e todos os seus colaboradores pela estrutura e oportunidades oferecidos durante todos esses anos de estudo.

"O Homem pretende ser imortal e para isso defende princípios efêmeros. Um dia, inexoravelmente, descobrirá que para ser imortal deverá defender Princípios Absolutos. Nesse dia, morrerá para a carne, efêmera, e viverá para o Espírito, Eterno. Será Imortal."(Celso Charuri)

Resumo

Os *smartphones* atuais são compostos por uma grande gama de sensores e componentes como GPS, *Bluetooth* e conexão com a Internet através das interfaces de rede Wi-Fi, 3G, entre diversos outros recursos. Junto com a crescente popularização dos *smartphones* ao redor do mundo está também o crescente desenvolvimento e popularização dos aplicativos que fazem uso desses recursos e tendem a diminuir a autonomia dos *smartphones*. Esse fato é considerado como um dos maiores problemas a ser superado no contexto de *smartphones* atualmente. Tendo esse problema em vista, diversas soluções abordam a questão de diferentes maneiras, e podem ser classificados em dois grupos: soluções em nível de software e soluções em nível de hardware. As soluções em nível de software são aquelas que procuram melhorar o consumo de energia dos *smartphones* apenas com alterações nos softwares que compõe o *smartphone*. Por exemplo, melhorando a eficiência da interface de rede Wi-Fi, controlando aplicativos que rodam no *smartphone*, entre outras. Por outro lado, as soluções em nível de hardware procuram melhorar ou ampliar a disponibilidade de energia nos *smartphones* através dos seus componentes físicos, como por exemplo evoluindo a tecnologia das baterias dos mesmos. Esta dissertação apresenta uma abordagem em nível de software para esse problema através de um sistema de gerenciamento dos estados dos sensores e componentes de um *smartphone*, baseado no perfil do usuário, visando a economia de energia. Esse estudo consistiu de três etapas. Na primeira etapa foi feito o levantamento bibliográfico e desenvolvida uma pesquisa de soluções existentes na área, identificando possíveis sistemas e aplicativos com a mesma proposição. Na segunda etapa foi elaborado o mecanismo denominado MyBatRecommender, composto pelas partes servidora e cliente, e implementado para o sistema operacional Android. Por fim, na última etapa, foram aplicadas algumas formas de validação no sistema proposto a fim de verificar a sua eficiência. Os resultados obtidos mostram que o sistema implementado, quando aplicado em cenário de testes controlado, apresenta um resultado que traz uma economia de energia de aproximadamente trinta e dois por cento em relação ao uso sem o mecanismo proposto.

Palavras-chaves: Gerenciamento de energia. Otimização de energia. Aplicativos móveis. Aplicativos sensíveis ao contexto.

Abstract

Nowadays smartphones are composed of a wide range of sensors and resources such as GPS (Global Positioning System), Bluetooth and Internet connection through Wi-Fi, 3G, among others resources. Along with the smartphone's increasing popularity around the world, there is an increasing development and popularity of power-hungry applications: applications that take advantage from these resources and may reduce the smartphones autonomy. This fact is known as one of the biggest to be solved when talking about nowadays smartphones. Considering this, many solutions were proposed and approach this topic in different ways. These solutions can be classified in two major groups: software layer solutions and hardware layer solutions. In one hand, the software layer solutions are the ones that try to reduce the smartphone's energy drain by only changing the software that composes the smartphone. For example, by improving the Wi-Fi interface or managing the running applications of the smartphone. On the other hand, the hardware layer solutions are the ones that try to improve or increase the energy availability of the smartphone changing or improving only the physical components, for example evolving the technology regarding the batteries. This study presents an approach in software layer for this problem: a system for managing the states of the smartphone's sensors and components, based on the user profile, aiming energy savings. This work consisted of three steps. In the first step the literature research was done and also a research of the existing solutions in the same area. In the second step the mechanism, called MyBatRecommender, composed by the server and client sides, was presented and developed for the Android operational system. In the last step some validation tests were applied aiming to verify the system efficiency. The results show that when applied to a controlled scenario, the MyBatRecommender achieves around thirty-two per cent of energy savings.

Key-words: Energy management. Energy optimization. Mobile applications. Context sensitive applications.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface
AutoSync	Automatic Synchronism
CPU	Central Processing Unit
DUT	Device Under Test
DVS	Dynamic Voltage Scheduling
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
LTE	Long Term Evolution
MAC	Media Access Control
PDA	Personal Digital Assistant
PaaS	Platform as a Service
RAM	Random Access Memory
2G	Second Generation
SNS	Social Networking Site
3G	Third Generation
URL	Uniform Resource Locator
WLAN	Wireless Local Area Network

Sumário

1 Introdução

Os *smartphones* são, hoje em dia, aparelhos tecnologicamente evoluídos que, apesar de seu tamanho compacto, possuem uma grande variedade de sensores como o Sistema de Posicionamento Global, GPS (*Global Positioning System*), sensores de luminosidade, acelerômetro entre outros, além de interfaces de redes como Wi-Fi, a Terceira Geração de Telefonia Móvel, 3G (*Third Generation*) e outros componentes como o *Bluetooth* e sua grande capacidade computacional, o que tem possibilitado os mais diversos usos, tornando-os bastante atraentes para o usuário final. Notou-se que na literatura relacionada os termos como sensor(es), componente(s) e interface(s) foram utilizados para se referir ao mesmo elemento do *smartphone*. Por exemplo, Vallina-Rodriguez e Crowcroft (2013) e Koenig, Memon e David (2013) se referem ao GPS como um sensor enquanto Chen et al. (2014) utiliza o termo componente, embora, de forma geral, o termo sensor tenha sido utilizado para indicar elementos do *smartphone* como acelerômetro, giroscópio e o termo componente tenha sido utilizado para Wi-Fi, *Display*, *Bluetooth*. No presente trabalho, toda vez que citados, esses termos serão referidos como componentes, por ser um termo mais abrangente, que remete a um componente de hardware do *smartphone*, englobando os outros termos conhecidos.

Esse cenário possibilitou que uma grande gama de aplicativos para esses *smartphones* fosse desenvolvida, incluindo desde simples gerenciadores de *e-mails* até complexos sistemas de navegação GPS. Muitos desses aplicativos fazem uso intenso de grande parte dos componentes presentes no aparelho, como os aplicativos sensíveis ao contexto do usuário, que são capazes de adaptar a sua forma de operação sem que haja uma ação explícita do usuário, provendo informações obtidas a partir do contexto no qual estão inseridas e que são relevantes para as ações que serão tomadas (ABOWD et al., 1999). Um exemplo desse tipo de aplicativo são os sistemas de navegação, que utilizam o GPS para posicionar o usuário e localizá-lo em um mapa de determinado local.

No entanto, essa abordagem tem seu custo pois ao permitir que os aplicativos tenham acesso livre a esses componentes, esse cenário faz com que o consumo da energia aumente em grande escala, de forma que o *smartphone* tenha uma autonomia de poucas horas por dia. Além disso, outro fator importante é a capacidade das baterias: A maioria dos *smartphones* possuem baterias Li-ion, cuja evolução tecnológica não acompanhou a evolução dos outros componentes do aparelho. Dessa forma, apesar de existirem estudos e esforços a fim de evoluir os recursos destas baterias, em nível de hardware, outras alternativas são necessárias para encontrar formas eficientes de gerenciar e otimizar o seu consumo. Por conta disso, diversos estudos vem se concentrando em desenvolver soluções em nível de software para esse problema.

Os estudos na área de gerenciamento e otimização do consumo de energia existem desde antes da popularização dos *smartphones*, com artigos datados de 2004, quando o mesmo problema ocorria, por exemplo, nos Assistentes Digitais Pessoais, PDAs (*Personal Digital Assistant*) como cita Krintz, Wen e Wolski (2004). Entretanto, os estudos se intensificaram com a popularização dos *smartphones* e das aplicações sensíveis ao contexto, com um aumento significativo a partir de 2011, tanto em quantidade e quanto a forma da abordagem para alcançar a economia de energia. Os estudos tem sido voltados desde a interação do usuário com o aparelho até a integração dos *smartphones* com os serviços da *Cloud Computing*, passando por estudos de otimização do funcionamento de componentes como GPS e Wi-Fi, além de estudos que analisam a forma em que a energia é consumida em um *smartphone* de acordo com seus componentes, visando identificar novos caminhos a serem explorados para atingir a economia de energia. Esses estudos são completos, de forma que conseguem englobar grande parte dos componentes presentes em um celular, medindo desde o consumo do *Display* do aparelho, até componentes utilizados em comunicação de redes, de uma forma detalhada. Recentemente, a empresa Google mostrou seu interesse por essa área de pesquisa ao apresentar em sua conferência anual, Google I/O, nos Estados Unidos da América, uma biblioteca de localização que utiliza os diversos componentes disponíveis no aparelho a fim de apontar a localização do usuário, em uma forma mais econômica em termos de consumo de energia [1].

1.1 Motivação e problema

Há uma grande previsão de crescimento de vendas de *smartphones* para os próximos anos. Apesar da previsão para o ano de 2015 ter diminuído de 11.3% para 10.4%, o número continua bastante expressivo, com uma previsão de 1.9 bilhão de unidades em 2019 (IDC, 2015).

Essa crescente popularidade fez com que as pessoas criassem uma dependência com esses aparelhos no dia a dia de tal forma que a ausência dos mesmos por um dia, ou até mesmo poucas horas de um dia, pode levar a incapacidade de executar tarefas importantes, como por exemplo o acesso ao *e-mail*, função de grande importância para usuários de *smartphones*. Junta-se a isso o fato de a evolução das baterias dos *smartphones* não ter acompanhado a evolução do restante dos componentes, levando-os a um estado comum onde a oferta de energia pela bateria é muito menor do que a demanda dos componentes. Não é raro entre os usuários de *smartphones* acabar em situações nas quais os aparelhos estão com níveis baixos de energia antes mesmo do final do dia, como citado por Robinson (2009). Além disso considera-se o fato que o sistema operacional Android, um dos grandes nomes de sistemas operacionais para *smartphones*, expõe o controle dos componentes de *hardware* para os desenvolvedores, o que permite que sejam desenvolvidos aplicativos

[1] https://developers.google.com/events/io/

ineficientes em termos de consumo de energia, uma vez que podem usar os componentes de forma equivocada (PATHAK et al., 2012).

Nesse cenário, soluções surgem a fim de prover energia extra ou de fornecer meios de gerenciar e otimizar o consumo da energia existente. Para prover energia extra acessórios como carregadores portáteis entre outros são comuns e cumprem bem o seu papel, mas muitas vezes tem um custo elevado e não são práticas, como portar um carregador junto de si ao longo do dia. Por conta disso, as alternativas que visam gerenciar e estender a duração da energia existente são consideradas mais viáveis. Geralmente são soluções em nível de software, como, no caso dos *smartphones*, aplicativos que gerenciam e otimizam o consumo de energia, resultando em uma redução em seu consumo. Além disso, em sua grande maioria as soluções em nível de software são de baixo custo ou gratuitas e cômodas para o usuário final, uma vez que basta instalá-las e utilizá-las para obter essa economia de energia.

1.2 Objetivos

1.2.1 Objetivo geral

O objetivo geral desse projeto é propor um mecanismo que permita realizar um gerenciamento eficiente e uma otimização do consumo de energia em *smartphones* Android, sem que este elemento interfira na experiência do usuário com o *smartphone*, ou seja, a implementação do mecanismo deve executar de forma transparente e automatizada, sem que o usuário tenha que tomar ações para que o mesmo funcione. Além disso, grande parte do processamento computacional do sistema deverá estar localizado em um servidor remoto, para garantir que o *smartphone* não utilize seus recursos computacionais com isso, e se mantenha livre para as outras tarefas do usuário. Outra característica importante do sistema é que deverá se comportar como um sistema de recomendação, de forma que deverá alcançar o gerenciamento e otimização de energia através de ações definidas pela análise de dados coletados relativos a diversos componentes do *smartphone*, que determinam o perfil de uso do usuário. Essas ações são as responsáveis por recomendar o melhor estado para cada componente do *smartphone* em determinados períodos do dia, evitando assim o consumo desnecessário de energia.

O foco do sistema é prevenir que os componentes de um *smartphone* fiquem em determinados estados em períodos em que não são necessários e que consequentemente consumam energia. Por exemplo, um usuário, ao estar em um ambiente que não possui disponibilidade de Wi-Fi não necessita que o componente responsável pelo mesmo permaneça ligado.

Dessa forma entende-se que não faz parte do escopo do projeto as seguintes

propostas:

- Projetar, propor ou analisar soluções ao problema apresentado que envolvam unicamente componentes de hardware;
- Implementar a solução proposta em sistemas operacionais que não o Android;
- Desenvolver funcionalidades que não sejam relacionadas diretamente com a proposta de gerenciamento e otimização do consumo de energia em *smartphones*, como, por exemplo, predição do nível de energia;

A solução proposta é composta de duas etapas: (i) elaboração teórica de um mecanismo; (ii) e sua posterior implementação no formato de um sistema que envolve um aplicativo Android. Além disto outros elementos auxiliares, como a aplicação servidora e componentes auxiliares para processamentos dos dados foram desenvolvidos.

1.2.2 Objetivo específicos e contribuições

A partir do objetivo geral do trabalho, foram definidos as contribuições e objetivos específicos, apresentados a seguir:

- Elaborar uma revisão sistemática que visa identificar o estado da arte em gerenciamento e otimização da energia de *smartphones* em nível de software;
- Identificar os principais aplicativos Android presentes no mercado (Google Play[2]) com funções similares ao presente trabalho;
- Desenvolver um aplicativo para o sistema operacional Android capaz de coletar informações relativas aos principais componentes de um *smartphone*;
- Propor um mecanismo de gerenciamento e otimização da energia em *smartphones*, que será realizado baseado nas recomendações baseadas no perfil de uso do usuário;
- Desenvolver um sistema, baseado no mecanismo proposto, com foco no sistema operacional Android, para economizar energia de *smartphones* Android;
- Validar o sistema desenvolvido de duas formas: (i) através da execução do mesmo em cenários controlados; (ii) e pela execução do mesmo em dois grupos de usuários reais, estudantes da UFSCar - *campus* Sorocaba e colaboradores da empresa Venturus Centro de Inovação Técnológica[3]

[2] https://play.google.com/store
[3] http://www.venturus.org/

1.3 Organização e metodologias

O presente trabalho foi organizado através de técnicas de revisão bibliográfica e estudo de soluções já existentes no mercado. Para isso, duas etapas foram definidas: a primeira etapa foi a responsável pela definição do tema, pesquisa bibliográfica e de soluções existentes que serviram de base para a segunda etapa, que por sua vez foi a responsável pela elaboração e implementação do mecanismo proposto e posterior validação. A Figura 1 apresenta a visão geral dessa metodologia.

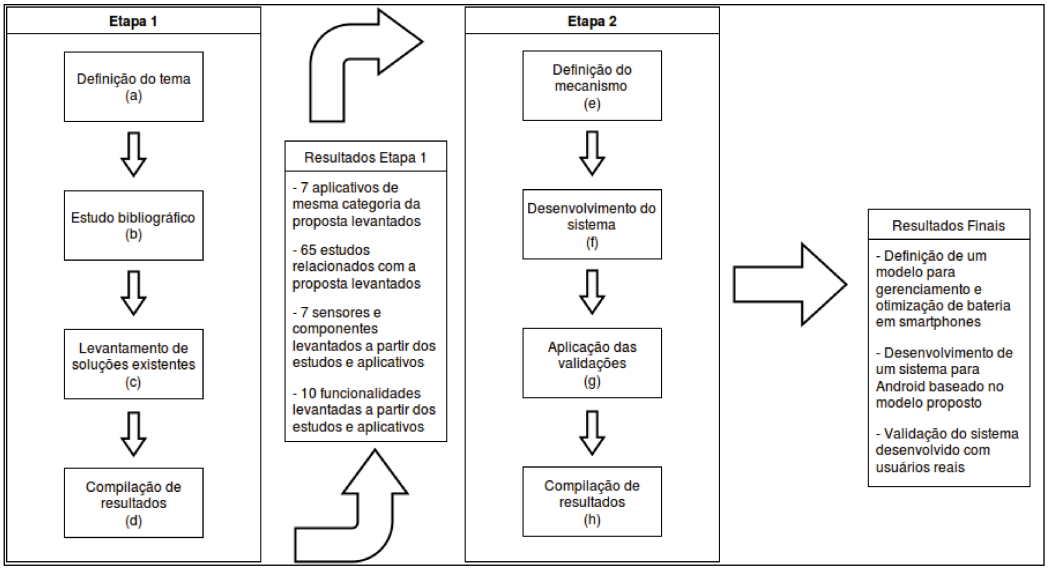


Figura 1 – Visão geral da metodologia utilizada

A Etapa 1 consistiu dos seguintes quatro passos: o primeiro passo, responsável pela definição do tema, gerenciamento e otimização do uso de energia em *smartphones*, que deu base a todos os outros passos do trabalho (a). O segundo passo, responsável pelo estudo bibliográfico, através de uma revisão sistemática, a fim de definir o estado da arte do tema definido (b), que junto com o terceiro passo, responsável pelo levantamento de aplicativos e soluções existentes no mercado (c), criaram possibilidades de identificar as principais funcionalidades de um sistema de gerenciamento e otimização de consumo de energia. Por fim, o quarto passo foi o responsável por compilar os resultados obtidos a fim de utilizá-los da etapa seguinte (d).

A Etapa 2 consistiu dos seguintes quatro passos: o primeiro passo, responsável pela definição do mecanismo proposto como solução ao problema apresentado (e). O segundo passo, responsável pelo desenvolvimento de um sistema que contemplasse o mecanismo proposto (f). O terceiro passo, responsável pela aplicação de estudos de caso a fim de validar a proposta (g) e por fim o quarto passo, responsável pela análise dos dados das validações (h).

1.4 Organização da dissertação

A presente dissertação está dividida em quatro capítulos, além deste primeiro capítulo introdutório, organizados da seguinte forma: o **Capítulo 2** apresenta a revisão sistemática desenvolvida, os principais trabalhos relacionados encontrados e os fundamentos e técnicas nos quais o presente trabalho se baseou. O **Capítulo 3** apresenta a proposta do mecanismo do presente trabalho, assim como as principais partes do sistema desenvolvido que contemplam esse mecanismo. O **Capítulo 4** apresenta a validação da proposta realizada através de estudos de caso e por fim o **Capítulo 5** apresenta as conclusões, contribuições, possíveis melhorias e trabalhos futuros.

2 Fundamentos e Estado da Arte

Para determinar o estado da arte referente ao gerenciamento e otimização de energia de *smartphones* em nível de software, foi realizada uma revisão sistemática da literatura sobre o tema. Com o resultado da revisão foi possível realizar algumas análises sobre os estudos e soluções existentes, que deram suporte para a elaboração do mecanismo proposto por este trabalho.

2.1 Revisão Sistemática

Uma revisão sistemática é uma forma de pesquisa que permite evidenciar propostas, soluções e resultados sobre um determinado tema, através de sua literatura existente. O processo de revisão sistemática se dá através de métodos e passos definidos que formalizam a sua execução, o que permite uma análise organizada dos resultados. Usualmente são seguidos os seguintes três passos: planejamento, execução e sumarização, que serão detalhados na sequência, como propõe Kitchenham (2004). Para auxiliar na execução desse processo da Revisão Sistemática, utilizou-se utilizou-se a ferramenta StArt (*State of the Art through Systematic Reviews*) [1], desenvolvida pelo LaPES-Laboratório de Pesquisa em Engenharia de Software da Universidade Federal de São Carlos.

2.1.1 Planejamento

O planejamento é um passo fundamental para que a revisão seja bem sucedida, pois é onde o protocolo de pesquisa da revisão é definido. No protocolo de pesquisa estão definidos os parâmetros necessários para que a pesquisa possa ser executada, que são: contexto de estudo, questão de pesquisa, palavras-chave, fontes de pesquisa e os critérios de seleção que definem quais estudos farão parte dos resultados da revisão. Essa etapa permite a outros pesquisadores obter resultados similares ao efetuar uma revisão sobre o mesmo tema. A seguir estão descritos os principais pontos que envolvem os parâmetros do protocolo de pesquisa:

1. **Contexto de estudo:** Junto com a crescente popularização dos *smartphones* e seu avanço tecnológico, o número de aplicativos que fazem uso intenso dos componentes disponíveis no *smartphone*, e, consequentemente, diminuem a sua autonomia para apenas algumas horas por dia, também cresceu significantemente. Devido a esse cenário, as pesquisas na área de gerenciamento e otimização de energia se tornaram um tópico de pesquisa mandatório. Baseado nesse contexto, a presente revisão

[1] Disponível em: http://lapes.dc.ufscar.br/tools/start-tool

sistemática objetiva identificar o que está sendo feito e as soluções existentes deste tópico de pesquisa.

2. **Questão de pesquisa:** Baseado no contexto de estudo, elaborou-se a seguinte questão de pesquisa: Como gerenciar e otimizar o uso de energia em *smartphones*, em nível de software?

3. **Fontes de pesquisa:** Para executar a busca de artigos existentes na literatura, 5 fontes de pesquisa foram definidas: IEEEXplore [2], Google Scholar [3], Scopus [4], ACM Digital Library [5] e Science Direct [6].

   A fim de executar a busca nas fontes de pesquisa acima citadas, foi necessário definir palavras-chave para indexar as buscas e fazer com que ela se restrinja a procurar artigos que possuam em seus títulos, conteúdos e resumos as palavras definidas, eliminando, assim, resultados que não sejam relacionados ao tópico de pesquisa desejado. As palavras definidas foram: *energy management, energy efficiency, battery, mobile application, context-aware, context-sensitive*. Para algumas fontes de pesquisa, a saber, ACM e IEEEXplore, como a busca estava retornando um número de resultados não relacionados muito grande, foram definidas mais duas palavras chaves a fim de restringir ainda mais o resultado das buscas, que são *device, smartphone*. Essas palavras foram definidas apenas em Inglês pois considerou-se para o presente trabalho apenas as publicações que continham, no mínimo, o resumo e as palavras-chave escritos em Inglês.

   Definidas as palavras-chave, foi necessário definir a *string* de busca, que explicita a relação entre as palavras-chave e como elas serão abordadas na busca. Foi definido ***((energy management and energy efficiency and battery and mobile application) and (context-aware or context-sensitive))*** para todas as bases de dados exceto ACM e IEEEXplore, que tiveram a *string* de busca composta por uma ou as duas palavras chave adicionais, resultando em ***((energy management and energy efficiency and battery and mobile application) and (context-aware or context-sensitive) and smartphone)*** para ACM e ***((energy management and energy efficiency and battery and mobile application) and (context-aware or context-sensitive) and (smartphone or device))*** para IEEEXplore.

4. **Critérios de seleção:** A fim de selecionar apenas artigos relevantes ao contexto do estudo, foi necessário criar critérios de seleção, que são usados para filtrar os resultados:

[2] http://ieeexplore.ieee.org/search/advsearch.jsp
[3] http://scholar.google.com/
[4] http://www.scopus.com/
[5] http://dl.acm.org/
[6] http://www.sciencedirect.com/

Critérios de inclusão:

- O artigo deve estar escrito em Inglês ou Português; e
- O artigo deve possuir no mínimo o resumo e as palavras-chave escritos em Inglês; e
- O artigo deve endereçar o tema de gerenciamento e otimização do consumo de energia em *smartphones* em nível de software; ou
- O artigo deve discutir as formas de como a energia é consumida pelos componentes em um *smartphone*; ou
- O artigo deve discutir temas relacionados ao consumo de energia em *smartphones*.

Critérios de exclusão:

- O artigo não discute sobre gerenciamento e otimização do consumo de energia em *smartphone*; ou
- O artigo não discute sobre nenhum tema relacionado ao consumo de energia em *smartphones*;
- O artigo discute sobre o tema proposto, mas apenas em nível de hardware;
- O artigo estava incompleto, ou sem informação o suficiente para ser analisado.

2.1.2 Execução

O pesquisador executa a pesquisa buscando e armazenando trabalhos encontrados a partir dos parâmetros definidos na fase de planejamento. A execução da presente revisão sistemática se deu entre Março/2013 e Janeiro/2014, sendo atualizada em Março/2015 e resultou em 1162 artigos encontrados, distribuídos nas fontes de pesquisa como mostra a Figura 2.

A partir dos artigos coletados, os critérios de seleção foram aplicados, considerando o título, resumo e conteúdo dos artigos, o que resultou em uma coleção de 65 artigos que foram considerados relevantes para o presente estudo, e que serão detalhados nas próximas seções.

2.1.3 Sumarização

Por fim, na etapa de sumarização, o pesquisador sumariza os resultados obtidos ao longo da execução da revisão a fim de facilitar novas análises e obter resultados sobre o tópico de pesquisa em questão. Dessa forma, no presente trabalho, algumas análises foram feitas com a intenção de identificar o potencial do tópico de pesquisa do presente estudo.

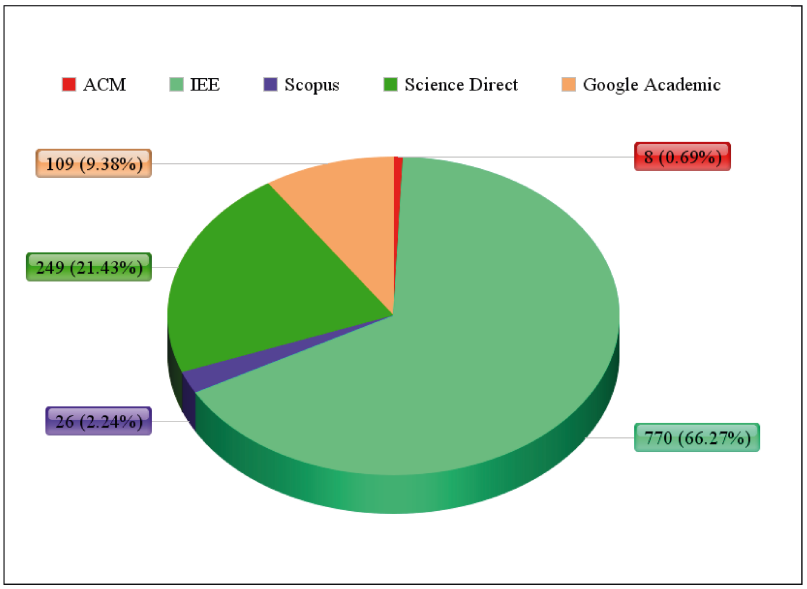


Figura 2 – Distribuição dos artigos nas fontes de pesquisa

Primeiramente, foi feita uma análise temporal da data de publicação dos artigos considerados nesse trabalho, como mostra a Figura 3. A partir da análise, pode-se perceber que o assunto vem sendo estudado desde o início da Era *smartphone*, em 2007. No entanto, o número de artigos publicados começou a crescer significantemente a partir de 2011, época próxima a quando houve uma popularização dos *smartphones* (POCKETNOW, 2016). Desde então o número de publicações vem crescendo, de forma que apenas os anos de 2013 e 2014 possuem cerca de 55% do total de artigos publicados. Nota-se também que, até o começo do ano de 2015, data da atualização da presente revisão sistemática, já havia dois artigos publicados. Dessa forma, os dados provenientes da análise temporal permitem concluir que esse é um relevante tópico de pesquisa, e que continuará a se expandir, uma vez que os *smartphones* estão cada vez mais populares e presentes no nosso dia a dia.

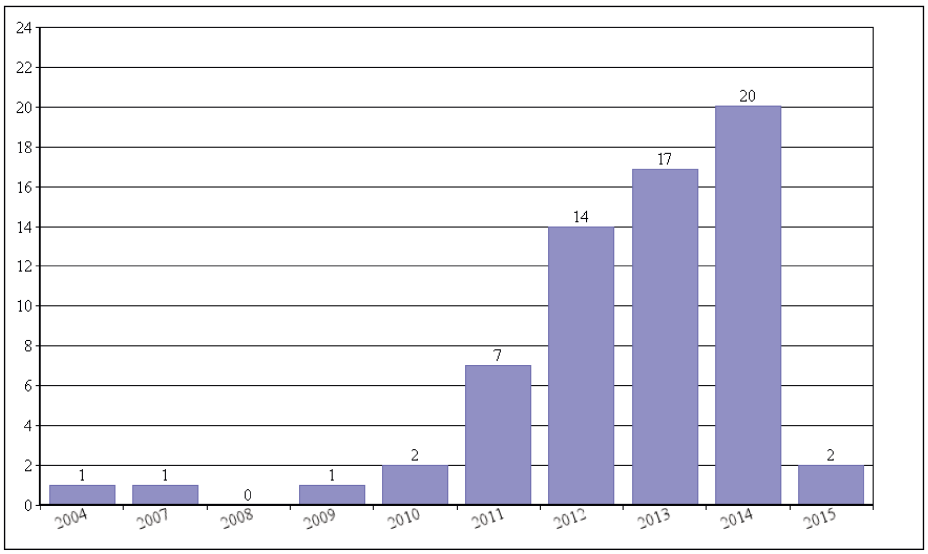


Figura 3 – Distribuição dos artigos pelos anos de publicação

A segunda análise dos artigos se concentrou no conteúdo dos mesmos. A ideia era identificar os principais métodos e soluções encontradas, podendo assim avaliar quais métodos já se encontram em um estágio avançado de pesquisa e quais não. Considerando isso, foram criadas categorias que agrupam os artigos de acordo com a forma que abordam o tópico de pesquisa, que são as seguintes:

- Categoria I, "Consumo de energia no *smartphone*": trata de artigos que focam em como a energia é consumida dentro de um *smartphone*, pelos seus componentes, aplicativos e outros fatores. Esses artigos apresentam soluções que em muitos casos se propõe a medir esse consumo utilizando ou não ferramentas auxiliares, alcançando uma boa precisão nas medições. Além disso, comparações são feitas entre modelos diferentes de dispositivos, sistemas operacionais diferentes, componentes diferentes e também entre os distintos estados de cada componente. Por exemplo, diferenciam o consumo dos estados *idle*, ligado, desligado e escaneando do componente rede Wi-Fi. Por fim, alguns desses artigos também propões modelos de consumo de energia, que tem como objetivo predizer o consumo de energia de um *smartphone* baseado nesses conhecimentos.

- Categoria II, "Gerenciamento de componentes do *smartphone* e aplicativos sensíveis ao contexto": contém artigos que propõem soluções para o consumo de energia através de algoritmos que procuram gerenciar os estados dos componentes do *smartphone* para economizar energia. Para isso, esses algoritmos coletam informações contextuais do usuário, como por exemplo sua localização, e através do processamento destas informações definem qual é o estado mais adequado para determinado componente. Por exemplo, ligar ou desligar o GPS quando o usuário estiver dentro ou fora de um prédio.

- Categoria III, "Consumo de energia em *smartphones*: a visão dos usuários e desenvolvedores": envolve artigos que buscam mostrar como os usuários de *smartphone* e os desenvolvedores de aplicativos compreendem a questão do consumo de energia nos *smartphones*.

- Categoria IV, "Otimização do funcionamento dos componentes de *smartphones*": possui artigos que tentam alcançar a redução do consumo de energia melhorando o funcionamento ou aproveitando-se de detalhes do comportamento de componentes específicos do *smartphone*. Uma das formas para tal é utilizar componentes alternativos, que consumam menos energia, para obter resultados semelhantes, ao invés dos principais, que consomem mais energia. Por exemplo, utilizar *Bluetooth* e/ou Wi-Fi para conseguir a localização de um usuário ao invés de utilizar o GPS.

- Categoria V, "Revisões sistemáticas e resumos sobre o tópico de pesquisa": considera os trabalhos que possuem uma visão geral sobre o que já existe sobre o tópico de pesquisa do presente trabalho.
- Categoria VI, "Integração *Cloud* e *Mobile*": contém artigos que exploram uma solução mais recente, que surgiu junto com a popularização dos serviços em *Cloud*, que consiste na transferência de parte do processamento do aplicativo para servidores em nuvem, evitando assim um consumo de energia maior nos *smartphones* causado pelo alto processamento computacional.

As Tabelas 1, 2, 3, 4, 5 e 6 mostram o agrupamento de parte dos artigos selecionados nas categorias criadas e contém uma breve descrição de cada trabalho, referenciados pela coluna **Resumo** enquanto a coluna **Id** se refere ao número daquele trabalho, criado pela Tabela 41, e a coluna **Citações** mostra o número de citações de cada artigo, obtido pela informação fornecida pelo Google Scholar[7].

Tabela 1 – Trabalhos relacionados na Categoria I

Id	Resumo	Citações
[1]	Mostra o consumo do *Bluetooth*, Wi-Fi e 3G nos aparelhos Nokia E71 e Motorola Milestone, considerando diferentes estados desses componentes como *standby*, escaneando e transferindo dados	1
[2]	Apresentam o *SEMO*, sistema Android que monitora o consumo de energia por aplicativos e classifica-os de acordo com a sua taxa de consumo, dados que podem ser úteis tanto para desenvolvedores quanto para usuários	31
[3]	O artigo analisa duas plataformas de *smartphone* para investigar como a energia é consumida pelos aplicativos que estão rodando em segundo plano e pelo tipo da conexão com a Internet. A partir dessas análises o artigo também propõe algumas otimizações que poderiam ser feitas para economizar no consumo de energia	8
[4]	Considerando a importância de se realizar testes de consumo de energia em aplicativos, o artigo propõe um processo de como realizar esses testes em um formato que desenvolvedores possam seguir para avaliar os seus aplicativos desenvolvidos	18

[7] https://scholar.google.com

[5]	Mostra um gerador de perfil de consumo de energia para aplicativos Android, mostrando como ela é consumida pelos componentes de hardware do *smartphone*. Os testes executados mostram que esse sistema consegue estimar o consumo de energia de um aplicativo com uma margem de erro de 10% e que o custo operacional, em termos de consumo de energia, desse sistema é de 3.8%	3
[6]	Propõe um modelo com uma implementação de uma ferramenta baseada nesse modelo, chamada *GreenDroid*, capaz de automaticamente analisar um aplicativo e a utilização que o mesmo faz dos componentes do *smartphone* e identificar problemas que podem aumentar o consumo de energia	30
[7]	Utilizando-se a ferramenta *GreenDroid*, mostra testes feitos com 173 aplicativos Android reais, identificando e discutindo os problemas com eles encontrados	18
[8]	Propõe um método capaz de predizer o consumo de energia de aplicativos rodando no *smartphone* e para isso não utilizam nenhum aparelho externo, e sim apenas informações de consumo de energia coletados do próprio aparelho	13
[9]	Apresenta um modelo, chamado *eLens*, para estimar o consumo de energia de aplicativos Android, sem a necessidade de ferramentas auxiliares externas, além de permitir que a análise seja feita para todo o aplicativo, métodos ou linhas de código. Pelos testes realizados com 6 aplicativos do Google Play, mostra que consegue predizer o consumo com uma taxa de erro de aproximadamente 10%	94
[10]	Mostra uma análise do consumo de energia de alguns aplicativos utilizando-se a ferramenta *Power Tutor*, que exibe o consumo de alguns componentes por aplicativo	3
[11]	Apresenta um método de modelagem do consumo de energia através de padrões de consumo coletados e armazenados no celular, sem utilizar ferramentas externas e sem a necessidade de conhecer-se o hardware do *smartphone*. Além disso apresenta a implementação da ferramenta *PowerDoctor* que implementa o modelo proposto	3

[12]	Apresenta uma ferramenta desenvolvida, chamada *CharM*, para o sistema operacional Android, que coleta dados relativos a utilização da energia do *smartphone* em diferentes estados operacionais do mesmo e apresenta uma análise dos dados coletados para que possa ser possível identificar possíveis otimizações	13
[13]	Mostra uma investigação sobre o consumo de energia por diversos componentes como acelerômetro, giroscópio entre outros do *smartphone*, usando ferramentas externas e APIs internas do Android. Por fim, mostra uma comparação do resultado dessas duas técnicas	12
[14]	Propõe um método com auxílio de ferramentas externas para estimar o consumo de energia em *smartphones* Android pelos seus principais componentes, como GPS e Wi-Fi em diferentes estados. Por fim, mostram o porque alguns desses componentes foram considerados grandes consumidores	2
[15]	Apresenta uma análise do consumo de energia dos componentes Wi-Fi, GPS e as redes móveis 3G e 4G usando aplicativos Android. Além disso fazem a comparação dessas medições entre quatro gerações da mesma série de *smartphone*, o Samsung Galaxy, evidenciando a melhora de consumo entre as gerações	4
[16]	Considerando a importância da coleta de informações relevantes ao consumo de energia para poder elaborar otimizações, propõe uma ferramenta designada a coletar informações relacionadas a energia a partir do contexto do usuário, chamada *UserScope*, capaz de operar com um custo operacional baixo, de 0.8% em termos de processamento da Unidade Central de Processamento, CPU (*Central Processing Unit*)	7
[17]	Mostra um estudo detalhado sobre os diversos estados e propriedades de consumo do Wi-Fi	4
[52]	Mede o consumo de energia dos diferentes estados de diversos componentes como Wi-Fi, *Bluetooth*, GPS entre outros, utilizando como *device* para teste um Nokia N95, sistema operacional Symbian OS 9.2	186

[53]	Mostra o consumo do componente conhecido como Evolução de Longo Termo, LTE (*Long Term Evolution*), através de medições relativas ao volume de dados trafegado e variáveis da bateria realizadas durante a utilização de uma rede LTE comercial	6
[54]	Descreve o *PowerBooter*, uma técnica automatizada de construção de modelos de consumo de energia através do gerenciamento do consumo de energia dos diferentes estados dos componentes de um *smartphone*, não necessitando de nenhuma ferramenta externa para tal. Além disso, descreve o *PowerTutor*, uma ferramenta que utiliza-se do modelo criado pelo *PowerBooter* para criar as análises de estimativa do consumo de energia	686
[55]	Explora as otimizações da Memória de Acesso Aleatório, RAM (*Random Access Memory*), existentes para computadores e verifica se elas são válidas para o ambiente dos *smartphones*	34
[58]	Investiga em qual nível a dissipação da energia causada pela execução de aplicativos pode ser analisada através de ferramentas via software. Também apresenta uma técnica que pode estimar, para um aplicativo qualquer, o seu consumo de energia	49

Tabela 2 – Trabalhos relacionados na Categoria II

Id	Resumo	Citações
[29]	Mostra um sistema probabilístico que controla os estados do componente GPS, considerando a forma de locomoção e localização do usuário, entre outras variáveis, obtidas através da análise de outros dados, coletados por sensores como acelerômetro, giroscópio entre outros	7
[18]	Mostra um sistema sensível ao contexto do usuário que adapta o funcionamento de outros aplicativos rodando no *smartphone* através de mudanças na taxa de coleta de informação, de acordo com as informações contextuais do usuário. Testes mostram que a técnica pode melhorar o consumo de energia em até cinco vezes	11

[19]	Considerando a necessidade de fluxo de informação constante entre uma aplicação sensível ao contexto do usuário e um servidor remoto, propõe um *middleware*, *gLCB*, para *smartphones* Android que é capaz de coletar informações contextuais do usuário, enviando-as a um servidor remoto de uma forma eficiente em termos de consumo de energia	5
[20]	Propõe um *middleware*, *NetMaster*, que, através de informações contextuais do usuário, controlam as atividades e componentes de comunicação de rede do *smartphone* a fim de economizar energia. Resultados mostram uma melhora no consumo de energia, das atividades relacionadas a comunicação de rede, de 77.8%	1
[21]	Mostra um sistema de gerenciamento de energia, *PhoneJoule*, que muda entre o Serviço de Rádio de Pacote Geral, GPRS (*General Packet Radio Services*) e Wi-Fi utilizando-se de informações contextuais. Além disso provê formas para o usuário encerrar aplicações que podem estar consumindo energia de forma indesejada e apresenta perfis de controle dos componentes que podem ser ativados	2
[22]	Propõe um método que controla o estado da conexão com a Internet quando o *smartphone* está com o *Display* desligado	3
[23]	Propõe estratégias para determinar onde colocar *wakelocks*[8], que por sua vez determinam a utilização do CPU, por exemplo, de acordo com a utilização das aplicações do *smartphone*. Testes mostram que essa técnica é capaz de obter economia de energia de cerca de 32%	6
[32]	Apresenta técnicas que exploram dados contextuais de localidade e tempo para predizer a configuração dos componentes Wi-Fi e GPS de forma a economizar energia. Utiliza-se diversos algoritmos de aprendizagem de máquina para tal	3
[56]	Apresenta um aplicativo desenvolvido para o sistema operacional Android que permite controlar o estado de alguns componentes como Wi-Fi, GPS e *Bluetooth* de acordo com o estado (ligado ou desligado) do *Display*	13
[57]	Apresenta uma técnica que otimiza o consumo de energia dos componentes CPU e *Display* ao mesmo tempo que mantém um nível aceitável de desempenho e utilização	13

[8] http://developer.android.com/intl/pt-br/reference/android/os/PowerManager.WakeLock.html

Tabela 3 – Trabalhos relacionados na Categoria III

Número	Resumo	Citações
[44]	Através de informações coletadas de uso do *smartphone* e respostas de questionários, tentam entender qual é o comportamento e as expectativas do usuário em relação ao seu consumo de energia no *smartphone*, mostrando que, por exemplo, os usuários desejam saber mais como a energia é consumida nos mesmos	24
[24]	Utiliza informações coletadas do Stack Overflow[9] provenientes das perguntas e respostas de usuários relativas ao consumo de energia em *smartphones*, visando entender como os desenvolvedores de aplicativos se preocupam e entendem sobre o assunto de energia em *smartphones*	31

Tabela 4 – Trabalhos relacionados na Categoria IV

Id	Resumo	Citações
[25]	Apresenta um sistema de posicionamento que usa o acelerômetro ao invés de GPS para reduzir o consumo de energia, mostrando resultados de até 27% de economia de energia	24
[26]	Apresenta um método que, baseado no nível de energia do *smartphone*, muda a forma de funcionamento do Serviço de Redes Sociais, SNS (*Social Networking Site*), evitando que conexões com a Internet sejam feitas, por exemplo, quando não houver energia o suficiente para tal	6
[27]	Propõe um algoritmo de roteamento para a Rede Local Sem Fios, WLANs (*Wireless Local Area Network*) ad-hoc, que, considerando o nível de energia, adapta o funcionamento e a forma com que o roteamento é feito, a fim de economizar energia	3
[28]	Apresenta um método que tenta reduzir o consumo de energia através de um gerenciador de tarefas em execução no *smartphone* baseado na técnica de Dimensionamento Dinâmico de Tensão, DVS (*Dynamic Voltage Scaling*)	9
[30]	Mostra um método que reduz o consumo do *Display* mudando a sua taxa de atualização, sem comprometer a experiência do usuário	5

[9] http://stackoverflow.com/

[31]	Propõe um sistema colaborativo onde usuários com alto nível de energia podem ajudar a carregar dados de usuários com pouca energia, reduzindo as chances desses últimos ficarem sem energia	11
[33]	Detalha o consumo de energia do componente Wi-Fi e propõe um sistema que utiliza-se de dois esquemas para melhorá-lo, chamado *BattMan*. Além disso, desenvolvem o *BreezChirp*, ferramenta responsável pela medição da largura de banda do Wi-Fi, necessário para os testes do primeiro sistema	1
[34]	Mostra uma técnica que expande a implementação do componente Wi-Fi para que ele possa identificar novos blocos de informação de uma forma melhor em termos de consumo de energia. Essa técnica é implementável através de extensões para o *driver* do componente Wi-Fi que podem ser instalados através de uma atualização do *firmware* do componente	5
[35]	Mostra um *framework* que consegue identificar localidades de interesse de forma econômica em termos de energia, oferecendo assim suporte para a execução automática de alguns aplicativos através da localização do usuário	3
[36]	Melhoram a implementação de um sistema existente, o *EnTracked*, que determina de forma econômica, em termos de energia, quando e quais componentes responsáveis pelas informações de localização buscar as informações necessárias	9
[37]	Apresenta uma arquitetura, *MADNet*, para descobrir qual é o melhor ponto de acesso Wi-Fi, em termos de consumo de energia, para executar tarefas remotamente, considerando operadoras de celular, redes Wi-Fi e usuários finais	36
[50]	Apresentam um sistema, *ASP*, que pausa o funcionamento de aplicativos em execução no *smartphone*, mas mantém um controle para que eles possam ser reiniciados quando necessário, por exemplo, quando uma nova mensagem é recebida	2

Tabela 5 – Trabalhos relacionados na Categoria V

Id	Resumo	Citações
[38]	Mostra uma visão geral de quatro principais maneiras de como economizar energia em *smartphones*, mostrando algumas ideias que usam informações contextuais do usuário	32

[39]	Uma revisão sistemática que estuda diversos artigos na área de gerenciamento e otimização de energia em *smartphones*	95
[40], [51]	Revisões sistemáticas que estudam artigos que tratam sobre computação em nuvem para dispositivos móveis	12, 0
[41]	Mostra uma visão geral de sistemas para dispositivos móveis que são adaptáveis ao contexto e como isso pode ser usado para incluir informações sobre o nível de energia	17
[42]	Considerando o problema das baterias em *smartphones* propõe e mostra um resumo sobre as duas principais áreas relacionadas ao problema, que são formas de buscar mais energia e como gerenciar a energia já existente	2
[43]	Mostra um estudo sobre o problema de otimização de energia em *smartphones*, comparando cinco soluções principais	0

Tabela 6 – Trabalhos relacionados na Categoria VI

Id	Resumo	Citações
[45]	Mede o consumo relativo à operação de processar jogos de *smartphones* na *Cloud*	4
[46]	Apresenta um método que executa parte dos aplicativos na *Cloud*, através de análise feita que determina se o método deve rodar local ou remotamente	4
[47]	Apresenta um trabalho que mede a eficiência das operações de executar tarefas na *Cloud*	4
[48]	Apresenta um trabalho que tenta descobrir através de comparações de métricas e resultados, qual é o melhor método para executar tarefas remotamente	1
[49]	Mostra uma proposta que melhora a capacidade dos *smartphones*, tanto em termos de eficiência de consumo de energia quanto em processamento, usando computação em nuvem para executar tarefas complexas	0

Além disso as Tabelas 7, 8 e 9 mapeiam como os componentes são endereçados por cada artigo das Categorias I, II e IV. Essas Categorias foram consideradas pois elas se relacionam diretamente com o estudo de componentes de diferentes formas, como por exemplo o gerenciamento dos estados desses componentes, e por conta disso, relacionam-se diretamente com a proposta do presente trabalho. Nessas tabelas foram considerados todos os componentes analisados pelos estudos dessas categorias. Caso algum dos componentes

listados, presentes na horizontal da tabela, seja alvo de algum dos estudos, presentes na vertical da tabela, um X é utilizado para indicar que aquele artigo estuda aquele componente.

Tabela 7 – Componentes endereçados pela Categoria I

Id	CPU	*Display*	GPS	BT	Wi-Fi	2G	3G	LTE	Voz	Áudio	Apps
[52]	x	x		x	x	x	x		x		
[1]				x	x		x				
[2]											x
[53]								x			
[54]	x	x	x		x		x			x	
[3]	x				x		x				
[4]											x
[55]											
[5]	x						x	x			
[6]											x
[7]											x
[8]											x
[9]	x		x		x						x
[10]											x
[11]	x	x			x						
[12]			x	x	x						
[13]	x						x				
[14]	x	x	x	x	x	x			x	x	
[15]			x		x		x	x			
[16]	x	x	x		x		x		x		x
[17]					x						
[58]											x

Tabela 8 – Componentes endereçados pela Categoria II

Id	CPU	Display	GPS	BT	Wi-Fi	2G	3G	LTE	Voz	Áudio	Apps
[29]			x								
[18]											x
[56]	x		x	x	x	x					
[19]			x	x	x		x		x	x	
[20]					x		x				
[21]					x	x	x				
[22]							x				
[23]	x	x									
[32]			x		x						
[57]	x	x									

Por fim, a Figura 4 mostra a distribuição dos estudos das Categorias I, II e IV pelos componentes que os mesmos abordam em seu conteúdo ao longo dos anos de publicação. Para essa comparação, considerou-se apenas os componentes que são abordados pelo

Tabela 9 – Componentes endereçados pela Categoria IV

Id	CPU	Display	GPS	BT	Wi-Fi	2G	3G	LTE	Voz	Áudio	Apps
[25]			x								
[26]					x		x				
[27]					x						
[28]	x										x
[29]	x										
[30]		x									
[31]								x			
[33]					x						
[34]					x						
[35]			x								x
[36]			x								
[37]					x						
[50]	x										x

mecanismo proposto por este estudo, como explicado na Seção 3, ao invés de todos considerados por todos estudos. A proposta desta análise é identificar a importância que certos componentes têm para o tópico de estudo e presente trabalho, através do número de artigos publicados ao longo dos anos que tratem sobre os mesmos em seus conteúdos.

Pela análise da Figura 4, é possível notar que o número de componentes sendo estudados está crescendo de forma geral ao longo dos anos. Isso se deve ao fato de que o tópico de estudo está em evidência. Mas pode-se notar um aumento significativo principalmente nos componentes Wi-Fi, Redes Móveis, *Display* e GPS, visto que eles são um dos maiores consumidores de energia de um *smartphone*, o que explica o porque o foco nesses componentes.

2.2 Resultados obtidos

A revisão sistemática ajudou a agrupar e formalizar resultados do tópico de pesquisa através de diversos estudos sobre o tema. Pelas análises feitas a partir desses resultados, nota-se que o problema em torno de gerenciamento e otimização do consumo de energia em *smartphones* está se tornando um tópico de pesquisa com crescente interesse, visto que os *smartphones* são aparelhos cada vez mais populares e que ainda existe uma diferença entre o que as baterias podem oferecer em quantidade de energia e o que os usuários estão requisitando, o que torna esse problema ainda mais grave. Além disso, através dos artigos selecionados pela revisão sistemática, pode-se perceber que existe um tema, abordado pela Categoria II, que explora as soluções que gerenciam os estados dos componentes de um *smartphone* através de ações baseadas em informações contextuais do usuário. Essas soluções são interessantes pois, ao considerar informações do usuário, conseguem adequar-se ao funcionamento do *smartphone*, aumentando a sua eficiência e levando a

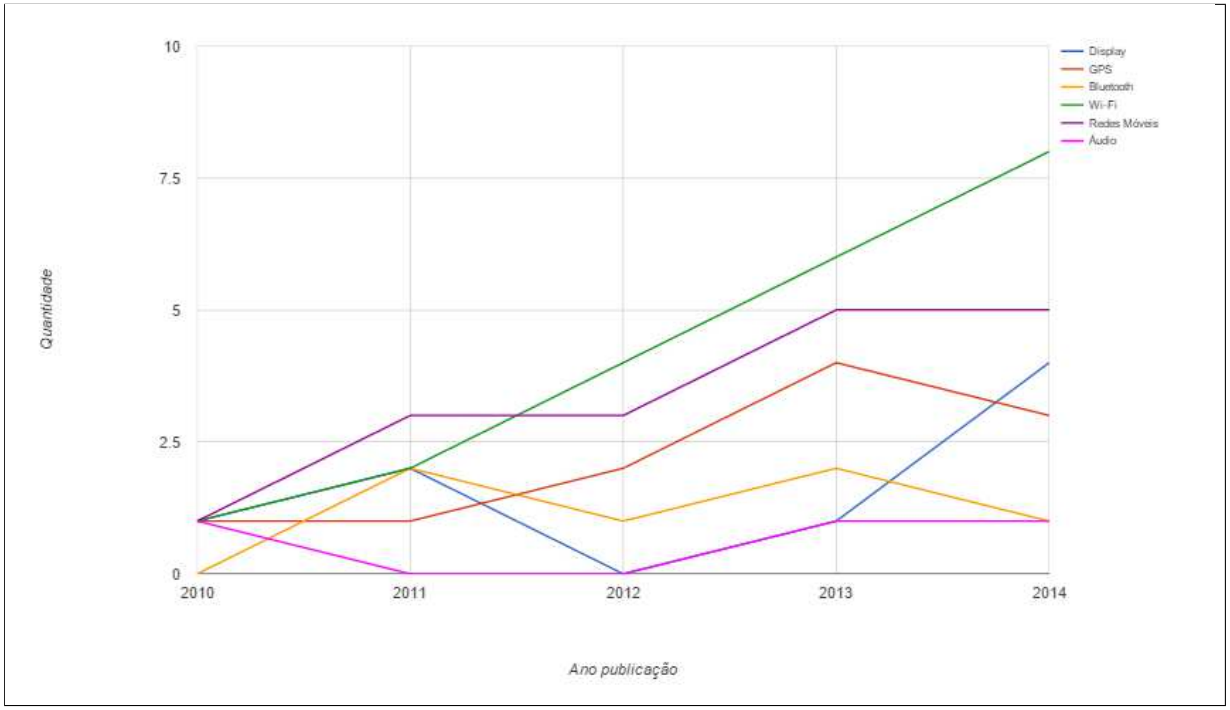


Figura 4 – Distribuição dos artigos das Categorias I, II e IV

resultados mais satisfatórios.

2.2.1 Trabalhos relacionados selecionados

A partir da revisão sistemática, buscou-se detalhar alguns dos artigos que poderiam trazer contribuições diretas para o contexto de soluções em nível de software, pois este é o objetivo desta dissertação. Para isto os mesmos foram agrupados nas subseções a seguir, seguindo a classificação das categorias apresentada nas Tabelas 1, 2, 3, 4, 5 e 6

2.2.1.1 Consumo de energia no *smartphone*

Os trabalhos que estudam como a energia é consumida nos *smartphones* são relevantes pois mostram o consumo de energia de cada componente do celular, em seus diferentes estados. Dessa forma pode-se analisar quais são os maiores consumidores e quando esse consumo ocorre, informações que podem ser usadas para auxiliar na elaboração de soluções posteriores.

É assim que Perrucci, Fitzek e Widmer (2011) relata como diversos componentes consomem a energia em um *smartphone* Nokia N95, sistema operacional Symbian[10]. O estudo apresenta um comparativo desses componentes e seus estados. Por exemplo, para o componente *Bluetooth*, é mostrado o consumo quando o mesmo está desligado, ligado, conectado, procurando outros aparelhos, enviando dados e recebendo dados com os valores de consumo de 12, 15, 67, 223, 425, 432 mW, respectivamente. Pode-se notar pela leitura

[10] http://www.nokia.com/us-en/support/product/nokia-n95-8gb/

desses valores que existem diferenças de consumo entre os estados do componente. Os estados ligado e desligado, por exemplo, diferem em 3 mW. Essa mesma comparação é feita para os outros componentes do *smartphone*, de forma que fica fácil analisar os maiores consumidores. Apesar de não utilizarem o sistema operacional foco do presente estudo, Android, essa visão geral mantém sua importância uma vez que os aparelhos utilizados para teste se assemelham com os *smartphones* atuais em sua composição de componentes, possuindo Wi-Fi, *Bluetooth*, redes móveis entre outros.

O estudo de Abreu e Villapol (2012) também mede o consumo de componentes dos *smartphones*, mas é voltado para as comunicações de rede. Usando como dispositivos para testes um *smartphone* Nokia E71, sistema operacional Symbian[11] e um Motorola Milestone, sistema operacional Android[12], o artigo apresenta o consumo dos componentes Wi-Fi, *Bluetooth* e redes móveis, em específico 3G, e a partir de casos experimentais abrangendo os estados dos componentes, assim como o estudo anterior, faz comparações a fim de descobrir quais componentes são melhores em determinadas situações. Além disso mostra comparações entre os dois dispositivos utilizados para testes, a fim de avaliar como os sistemas operacionais se portam nos cenários propostos.

Além dos estudos que medem o consumo da energia pelos componentes de um celular, existem também aqueles que reportam a medição do consumo de energia pelos aplicativos instalados nos *smartphones*. É dessa forma que Ding et al. (2011) desenvolve um sistema para dispositivos que possuam o sistema operacional Android, que monitora o consumo dos aplicativos e mostra ao usuário uma classificação com os aplicativos que mais consomem energia. Isso é feito através do sistema proposto, denominado *SEMO*, que através de seus módulos coleta informações sobre a bateria e os aplicativos em execução no *smartphone* a cada minuto e com essas informações pode, posteriormente, através de análises, criar uma lista com os aplicativos e o seu consumo relativo de energia. Essa informação, quando exibida para os usuários, pode servir para que os mesmos saibam quais aplicativos podem ser encerrados a fim de economizar energia.

2.2.1.2 Gerenciamento de componentes do *smartphone* e aplicativos sensíveis ao contexto

Essa é a categoria de estudos que tenta adaptar a solução ao perfil de uso do usuário de forma a deixá-la dinâmica e, dessa forma, atingir um grande número de usuários.

O conceito de sensível ao contexto, aqui, diz respeito ao fato de soluções que coletam dados do usuário como localização, nível de bateria, estados dos componentes e aplicações sendo usadas entre diversos outros dados a fim de processá-los, estudá-los e, consequentemente, prover uma solução. Alguma dessas soluções envolvem o gerenciamento dos estados dos componentes dos *smartphones*.

[11] http://www.nokia.com/us-en/support/product/nokia-e71/
[12] http://www.motorola.com.br/search?q=Motorola%20milestone

É assim que Yi e Cho (2012) apresenta os dados coletados de diversos componentes e utiliza redes Baysianas[13] a fim de desenvolver um sistema probabilístico que infere se o usuário está dentro ou fora de algum prédio ou construção e baseado nessa inferência controla o estado do GPS do *smartphone*. Testes mostram que a acurácia desse sistema é de 77% em dias de semana e 68% em finais de semana, devido a diferença de hábitos do usuário entre esses dias.

Da mesma forma, Herrmann, Zappi e Rosing (2012) desenvolveram um sistema que, de acordo com o contexto do usuário, encerra ou inicia os aplicativos em execução no *smartphone*. Para isso, cada aplicativo registra-se no sistema, informando qual é o contexto que tem interesse. Para computar os resultados, eles testaram o sistema proposto com uma ferramenta utilizada para medir variáveis ambientais, junto com um *smartphone* Android, e mostram que o sistema proposto conseguiu estender a autonomia em cinco vezes.

Zahid, Ali e Nassr (2011) desenvolvem um sistema similar com os anteriores dessa categoria, uma vez que também gerencia o estado dos componentes dos *smartphones*, no caso Wi-Fi, GPS e *Bluetooth*, mas ele não considera informações contextuais do usuário, e sim regras pré-estabelecidas pelo mesmo. Por exemplo, quando a energia chegar a determinado nível, o sistema desliga alguns componentes, deixando apenas os fundamentais como rede de telefonia móvel.

Donohoo, Ohlsen e Pasricha (2011) propõe um método que reduz gradativamente o brilho do *Display* e também propõe uma técnica que reduz o processamento da CPU, considerando o intervalo entre as ações do usuário, de forma a manter a experiência do usuário em um nível aceitável, e diminuir o consumo da energia.

Apesar de estar classificado na categoria de soluções que utilizam o gerenciamento dos componentes do *smartphone*, Donohoo et al. (2014) apresenta uma contribuição importante relacionada ao consumo dos componentes 3G, Wi-Fi e GPS ao mostrar que os mesmos, quando ligados, mesmo em seu estado *idle*, possuem um consumo de energia de mais de 25%, como mostra a Figura 5, adaptada do trabalho desses autores. É a partir desses dados que eles propõem uma técnica de gerenciamento do estado de alguns componentes, desligando os componentes em tempo de execução, para economizar energia. Para tal utilizam diversas técnicas de aprendizagem de máquina e dados contextuais do usuário.

2.2.1.3 Consumo de energia em *smartphones*: a visão dos usuários e desenvolvedores

Existem estudos que exploram a interação do usuário com o *smartphone* através de diversos métodos como questionários, tentando entender a forma que os mesmos enxergam o consumo de energia nos *smartphones*.

[13] http://research.microsoft.com/apps/pubs/?id=69588

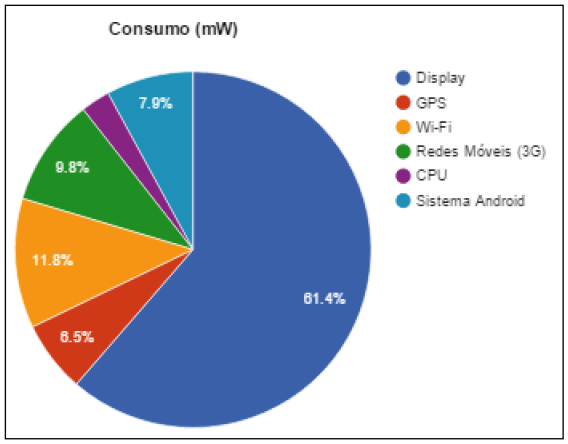


Figura 5 – Consumo dos componentes em seu estado *idle* de um *smartphone* Google Nexus One (adaptado de Donohoo et al. (2014))

É dessa forma que, através de informações coletadas de uso do *smartphone* e respostas de questionários, Heikkinen et al. (2012) tenta entender qual é o comportamento e as expectativas do usuário em relação ao seu consumo de energia no *smartphone*, mostrando que, por exemplo, os usuários desejam saber mais como a energia é consumida nos mesmos pois estão interessados em prolongar a sua duração, e desejam para isso ter o controle do consumo da mesma.

2.2.1.4 Otimização do funcionamento dos componentes de *smartphones*

As soluções que tentam otimizar a utilização de componentes específicos como GPS e Wi-Fi nos *smartphones* são discutidas por alguns autores. Em geral, mas não exclusivamente, esses estudos utilizam outras informações provenientes de diferentes componentes disponíveis no dispositivo, e que consomem menos energia, a fim de complementar ou substituir os componentes que consomem mais energia.

É dessa forma que Ananthanarayanan e Stoica (2009) descrevem o *Blue-Fi*: um sistema que prediz a disponibilidade de redes *wireless* utilizando-se dos sinais e padrões de comunicação do componente *Bluetooth* e da rede de celular. Dessa forma, evita que o componente Wi-Fi fique em estados como o de buscando redes, o que incorre, consequentemente, em uma redução no consumo de energia.

Considerando o gasto energético de manter o GPS ligado para posicionamento do usuário, Oshin, Poslad e Ma (2012) descrevem um sistema que foi desenvolvido para o sistema operacional Android visando diminuir o consumo de energia desse componente. Foi desenvolvido uma arquitetura que detecta o estado da mobilidade do usuário através do acelerômetro e baseado nesses estados gerencia o estado o GPS, ligando-o ou desligando-o, sem comprometer a acurácia do posicionamento.

Além disso, Hyeon et al. (2012) apresenta um método que, baseado no nível de energia do *smartphone*, caso o mesmo seja menor que um valor estabelecido, muda o modo

de operação do *smartphone* para o modo de economia de energia. Nesse modo, o tempo dos intervalos entre as mensagens enviadas ao Serviço de Redes Sociais, SNS (*Social Network Service*) é alterado de acordo com uma função que considera entre alguns fatores, o nível de energia. Dessa forma evita que conexões com a Internet sejam feitas, por exemplo, quando não houver energia suficiente para tal.

2.2.1.5 Revisões sistemáticas e resumos sobre o tópico de pesquisa

Os trabalhos presentes nessa categoria apresentam uma visão geral das soluções existentes para o problema de gerenciamento e otimização do consumo de energia em *smartphones*.

Alguns desses trabalhos apresentam uma visão que abrange diversas soluções existentes ao problema, como em o presente trabalho. Vallina-Rodriguez e Crowcroft (2013) também o faz, apresentando um estudo das soluções encontradas entre 1999 e 2011 em seis diferentes níveis.

Também existem os trabalhos que se focam em dar uma visão geral dos trabalhos existentes para uma metodologia específica, como Jiao et al. (2013), que cobre apenas as soluções que envolvem técnicas relacionadas com *Cloud*.

2.2.1.6 Integração *Cloud* e *Mobile*

Estudos também exploram tecnologias mais recentes como a *Cloud Computing*[14], através de diferentes técnicas, como o *offloading* que consiste em reduzir o processamento computacional do *smartphone* delegando algumas funções para serem executadas nos servidores remotos, reduzindo consequentemente o consumo de energia no *smartphone*.

Cuervo et al. (2010) apresenta uma arquitetura que determina em tempo de execução métodos que podem ser executados remotamente, e tenta dessa foram alcançar economia da energia do celular. Para a tomada dessa decisão esse sistema analisa dados e limitações da conexão atual do *smartphone*. Por fim, o modelo proposto é testado com alguns aplicativos, mostrando um bom resultado em termos de economia de energia.

Como citado na Seção 1.2.1 o objetivo do presente trabalho é propor um mecanismo que irá a partir dos dados coletados da utilização do *smartphone* e seus componentes pelo usuário, ser capaz de gerenciar o estado desses componentes, deixando-os sempre no estado apropriado, economizando energia. Por conta dessa característica o presente trabalho pode ser considerando como parte da Categoria II. Além disso, o mecanismo proposto possui uma característica importante da Categoria VI, uma vez que utiliza a tecnologia da *Cloud Computing* para poder executar métodos que exigem um alto processamento computacional, evitando que o mecanismo proposto tenha um consumo alto de energia.

[14] http://www.webopedia.com/TERM/C/cloud_computing.htm

A ideia do controle e gerenciamento dos estados dos componentes proposta pelo presente trabalho se assemelha ao proposto por Zahid, Ali e Nassr (2011). Mas, ao invés de utilizar regras preestabelecidas, como por exemplo quando o *display* for desligado, para tomar as ações de controle, o mecanismo proposto pelo presente estudo toma essas ações com base em análises feitas de informações da utilização do *smartphone* pelo usuário, que definem um perfil para o usuário. Essa abordagem pode levar a melhores resultados em termos de eficiência, uma vez que considera as peculiaridades de cada usuário.

2.2.2 Fundamentação Teórica

2.2.2.1 Sistema de recomendação

Com a imensa disponibilidade de informações nos diversos sistemas de hoje em dia, os usuários desses sistemas, muitas vezes, se encontram em dúvida sobre o que fazer com as mesmas, ou qual opção escolher, por exemplo. Isso acontece pois estes usuários podem não ter o conhecimento específico necessário para identificar o que esses dados representam. É nesse ponto que os sistemas de recomendação têm papel fundamental. Eles auxiliam no processo conhecido como processo de indicação, que através de entradas relativas ao usuário são capazes de fornecer dados e opções que sejam significativas ao usuário, facilitando na tomada de decisões (CAZELLA, 2010).

Os sistemas de recomendação também tem um papel importante na questão da personalização do sistema, que é o ato de adequar um produto ou serviço para atender as necessidades de um indivíduo. Como cita Cazella (2010), uma das funções necessárias para esse tipo de recomendação é monitorar toda a interação e informações do usuário enquanto o mesmo navega no sistema ou utiliza o produto. Dessa forma é possível sempre fazer recomendações atualizadas, o que pode aumentar a eficiência do sistema em questão.

O mesmo cenário ocorre para *smartphones*. São produtos que oferecem diversas funcionalidades e aplicativos, mas que funcionam de forma obscura aos usuários. Um exemplo desse relacionamento é interação entre aplicativos e consumo de energia: como estudado por Heikkinen et al. (2012), os usuários desejam ter um maior entendimento de como os aplicativos consomem a energia, para que possam ter maior controle sobre os mesmos. Considerando isso, vários aplicativos se propõem a identificar e mostrar para o usuário quais são os aplicativos e quanto de energia estão consumindo, muitas vezes oferecendo meios para encerrá-los, funcionando como um sistema de recomendação.

Existem diferentes maneiras de recomendação para personalizar o sistema, dentre as quais pode-se destacar a recomendação por análise de sequências de ações. Esse tipo de recomendação toma como base a sequência de ações que o usuário toma no sistema, e considerando a variável tempo conseguem fazer as inferências necessárias (CAZELLA, 2010).

2.2.2.2 Perfil do usuário

Para que a recomendação seja possível, é necessário conhecer o usuário que receberá a recomendação, e uma das formas de se fazer é através do Perfil do Usuário, que são formas de representar o interesse de um usuário com relação a vários assuntos em um dado momento, e pode ser visto como uma base de dados que armazena esses interesses e pode ser dinamicamente mantida (CAZELLA, 2010).

Para se construir o perfil do usuário é necessário conhecer qual o tipo de informação que é de fato relevante para representar esse usuário, considerando o funcionamento do sistema ou produto em questão. Considerado esses dados, deve-se então pensar nas estratégias de coleta de dados.

As formas de coleta de dados, segundo Cazella (2010), podem ser duas, que são os mecanismos explícitos e implícitos. Na sua forma explícita, a coleta de dados se dá através da ação espontânea e ativa do usuário, indicando os dados referentes a ele que são úteis para o sistema em questão. Isso pode se dar através de diversos meios como questionários, formulários entre outros. A coleta implícita, por sua vez, infere esses dados úteis ao sistema através da análise das ações do usuário sem que o mesmo tenha que explicitamente indicá-las. Essa coleta se dá, geralmente, através de ferramentas que são capazes coletar e armazenar as ações do usuário para posterior análise.

2.2.2.3 *Mobile*

Alguns conceitos de uma área maior, *mobile*, foram considerados como base para a fundamentação do mecanismo proposto, uma vez que o mesmo lida com *smartphones*, que são agentes importantes dessa área.

Ao se considerar o ambiente dos *smartphones*, um dos principais nomes de sistemas operacionais para os mesmos é o Android, sistema operacional baseado em Unix que opera em grande parte dos *smartphones* atuais. Desenvolvido pela *Open Handset Alliance*, uma aliança entre várias empresas das quais uma dela é o Google, possui diversas características que fazem dele o mais utilizado atualmente, como o fato de o código do sistema operacional ser disponibilizado pelo Google sob licença de código aberto, o que ajuda a baixar o custo de criação de um produto com o sistema (BRODKIN, 2012). Além disso esse fato ajuda a criar um grande grupo de programadores de aplicativos para Android, uma vez que não existe quase nenhum custo adicional para se programar para essa plataforma, o que torna o desenvolvimento atraente, de forma que existem mais de um milhão de aplicativos disponíveis em sua loja oficial de aplicativos, o Google Play (WARREN, 2013).

Esse fato tem seu lado positivo pois torna o ecossistema Android mais atraente que seus competidores como o Windows Phone[15], uma vez que é possível encontrar uma

[15] https://www.microsoft.com/pt-br/windows/phones

grande gama de aplicativos disponíveis, e em grande parte gratuitos. Mas também tem seu lado negativo, uma vez que deixa à disposição dos programadores funções importantes como o controle dos estados dos componentes. Isso se dá, principalmente, pela forma que o Android organiza a sua política de gerenciamento de energia. O Android é organizado em diversas camadas que possuem diferentes serviços e funcionalidades, de acordo com necessidades específicas. A primeira camada é baseada em uma customização do *kernel* do Linux, permitindo a comunicação entre os componentes de hardware e o software através dos *drivers*. Acima dessa camada, existe uma outra camada com uma série de bibliotecas desenvolvidas nas linguagens de programação C e C++ para a execução das requisições necessárias do *smartphone* e comunicação com a camada dos *drivers*, além de uma camada de bibliotecas Java que são a interface de comunicação com a última camada, que é a camada dos aplicativos Android (CORRAL et al., 2013). É essa organização que permite que os aplicativos Android da primeira camada consigam ter acesso aos recursos de hardware, da última camada, como mostra a Figura 6.

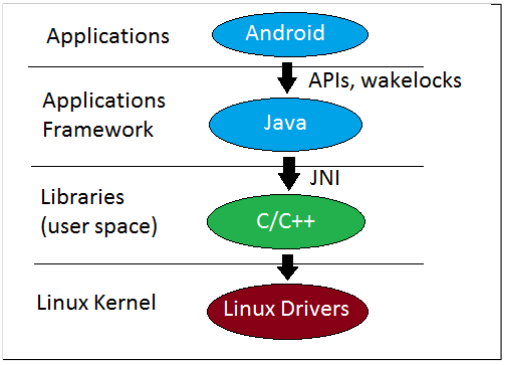


Figura 6 – Representação do *framework* de gerenciamento de energia Android (baseado em Instruments (2016))

Considerando esse esquema, a política de gerenciamento de energia do Android é distribuída entre essas camadas a fim de garantir que exista uma comunicação entre os aplicativos Android e a camada de comunicação com o hardware. Dessa forma, os aplicativos Android, quando necessitam de algum recurso computacional, precisam requere-los através dos chamados *wake-locks*[16], de forma a garanti-los durante a execução do aplicativo. Isso é feito através da classe PowerManager[17] do Android. Ou seja, caso algum programador queira utilizar algum recurso computacional ele consegue obtê-lo, no momento desejado, através de métodos e funções disponíveis publicamente na plataforma Android. Dessa mesma forma, fica a encargo do desenvolvedor desligá-lo no momento em que o componente não seja mais necessário. A não execução dessa segunda parte, no entanto, pode levar a cenários onde diversos componentes estão em estados que não são necessários, o que, consequentemente, pode elevar o consumo desnecessário de energia.

[16] http://developer.android.com/intl/pt-br/reference/android/os/PowerManager.WakeLock.html
[17] http://developer.android.com/intl/pt-br/reference/android/os/PowerManager.html

Junto a isso, deve-se considerar que a oferta de energia nos *smartphones* é limitada. Os *smartphones*, em sua maioria, utilizam baterias de Li-Ion pois elas possuem uma capacidade de armazenamento de energia muito boa para uma bateria de tamanho compacto, item essencial em um *smartphone* (SHEARER, 2008b). No entanto, apesar de existirem diversas pesquisas na área, essa capacidade é limitada e sua evolução tecnológica é menor do que os outros componentes do *smartphone* (SHEARER, 2008a), de forma que hoje consegue-se aumentar a oferta de energia principalmente aumentando o tamanho da bateria, o que vai no sentido contrário da evolução do tamanho da maioria dos *smartphones*.

2.3 Considerações Finais

O presente capítulo apresentou uma revisão da literatura existente sobre o tópico de pesquisa de gerenciamento e otimização de energia de *smartphones* em nível de software, bem como os principais trabalhos relacionados. Além disso, conceitos utilizados na elaboração do restante do trabalho, como sistemas de recomendação, perfil de usuário e algumas características do sistema operacional Android também foram apresentados.

3 MyBatRecommender: mecanismo proposto

3.1 Considerações iniciais

O consumo de energia em *smartphones* se dá de acordo com o perfil de uso de seus usuários, que usam funcionalidades e aplicativos em diferentes intensidades ao longo do dia. Esse consumo se dá através dos componentes que os *smarthpones* dispõem e que são ativados para prover recursos a fim de tornar as funcionalidades possíveis o que, consequentemente, eleva o consumo de energia. Dessa forma, uma maneira de se economizar energia é encontrar um meio de gerenciar esses componentes para que eles sejam ativados quando necessário e desativados quando não forem mais necessários, evitando assim o consumo indesejado de energia.

Considerando isso, o presente trabalho propõe um mecanismo denominado MyBatRecommender cujo objetivo é gerenciar e otimizar o uso da energia em *smartphones* com o propósito de economizá-la. No mecanismo proposto, o gerenciamento da energia se dá através do controle dos estados dos componentes do *smartphone*, feito por meio de um sistema de recomendação, que define ações para o perfil de uso do usuário; o perfil é criado utilizando-se variáveis coletadas continuamente ao longo dos dias de uso do usuário, baseando-se nos conceitos apresentados na subseção 2.2.2. Junto a isso, a otimização da energia se dá pela aplicação desses controles em momentos oportunos, ou seja, momentos em que a energia do *smartphone* poderia estar sendo consumida indevidamente.

Um exemplo, seria um componente que estava em seu estado ligado, mas não era necessário. Como mostram alguns trabalhos apresentados na subseção 2.2.1, como por exemplo Perrucci, Fitzek e Widmer (2011), o consumo de energia dos componentes varia de acordo com algumas variáveis, entre elas os seus estados. Por isso, manter o componente no estado necessário para aquele momento evita que a energia seja desperdiçada.

A partir da definição do mecanismo, foi desenvolvido um sistema para validação da proposta deste trabalho, composto de dois aplicativos Android e uma aplicação servidora. Embora a escolha tenha sido para o sistema operacional Android, o mecanismo proposto pode ser migrado para outros sistemas operacionais de *smartphone*, como iOS[1] e Windows Phone, considerando as peculiaridades de cada um.

Para dar maiores subsídios à proposta foi realizado um estudo de aplicativos com funções semelhantes. Este estudo será apresentado antes da proposta, na subseção 3.2. A

[1] http://www.apple.com/br/ios/

Seção 3.3 apresenta a visão geral do mecanismo, bem como a descrição detalhada de cada parte; a Seção 3.4 apresenta a visão da implementação do sistema; e por fim a Seção 3.6 discorre sobre as considerações finais referentes a este capítulo.

3.2 Estudo de aplicativos semelhantes

Além do estudo da literatura do tópico de pesquisa, apresentada na Seção 2, o presente trabalho realizou uma busca por aplicativos existentes que fossem similares à proposta para fundamentar a solução apresentada. Esta pesquisa se deu, principalmente, em torno das funcionalidades dos aplicativos Android com mesma função existentes no Google Play[2] e foi importante já que este projeto possui contribuições tecnológicas, sendo necessário observar tais soluções. A próxima subseção expõe esta pesquisa, explorando as características positivas e negativas comuns aos aplicativos estudados.

3.2.1 Estudo exploratório

Com o objetivo de identificar semelhanças entre os aplicativos existentes com foco em gerenciamento e otimização da energia no Google Play, foi feito um estudo exploratório considerando as funcionalidades dos aplicativos (i) e quais componentes eles gerenciavam (ii). Para a busca considerou-se o veículo oficial mais comum de publicação de aplicativos Android, o Google Play, e através dele fez-se a pesquisa, utilizando a palavra *battery* como filtro. Selecionou-se os seis aplicativos gratuitos mais populares, baseando-se no número de downloads, que possuíam características funcionais semelhantes ao que se pretendia propor como mecanismo deste trabalho. Ou seja, considerou-se válidos aqueles aplicativos que de alguma forma apresentavam meios para gerenciar os estados dos componentes do *smartphone*. Foram selecionados: Battery Doctor [3], DU Battery Saver [4], Avast Battery Saver[5], JuiceDefender[6], GO Battery Saver[7], MPower[8] e Battery Optimizer & Cleaner[9]. Dessa lista, o aplicativo MPower foi escolhido manualmente pois o mesmo havia sido estudado anteriormente, durante o processo de revisão sistemática, concluindo-se que ele possuía características semelhantes ao mecanismo proposto pelo presente estudo. O MPower é descrito em Ferroni et al. (2013).

[2] https://play.google.com/store
[3] https://play.google.com/store/apps/details?id=com.ijinshan.kbatterydoctor_en
[4] https://play.google.com/store/apps/details?id=com.dianxinos.dxbs
[5] https://play.google.com/store/apps/details?id=com.avast.android.batterysaver
[6] https://play.google.com/store/apps/details?id=com.latedroid.juicedefender
[7] https://play.google.com/store/apps/details?id=com.gau.go.launcherex.gowidget.gopowermaster
[8] https://play.google.com/store/apps/details?id=org.morphone.mpower
[9] https://play.google.com/store/apps/details?id=com.mcafee.batteryoptimizer

3.2.1.1 Análise das funcionalidades

As funcionalidades apresentadas pelos aplicativos escolhidos foram bastantes semelhantes, de forma que foram listadas as dez funcionalidades principais, apresentadas na lista abaixo:

(a) Consumo: *display* indicando a porcentagem atual da energia e seus principais atributos como temperatura, voltagem e tecnologia;

(b) Previsão de bateria: indica uma previsão do tempo para o final da energia;

(c) *Status bar*: barra rápida para gerenciamento manual dos principais componentes do *smartphone* como: Wi-Fi, *Bluetooth* e brilho do *Display*;

(d) Histórico: mostra o histórico de consumo de energia ao longo dos dias e das horas;

(e) Controle de carga: mostra os estágios de carga de bateria, o que significa cada um e, baseado no nível de energia do *smartphone* indica o que fazer, como por exemplo, carregar completamente entre outros;

(f) Controle de aplicativos: mostra os aplicativos que estão rodando no *smarpthone* atualmente e estão consumindo energia. Além disso oferece opções para encerrar o processo desses aplicativos;

(g) Controle de componentes por perfil: permite controlar o estado dos componentes através da ativação de perfis preestabelecidos;

(h) Criação de perfis: permite ao usuário criar um perfil de controle dos estados dos componentes manualmente, além dos preestabelecidos;

(i) Ativação de perfis por regras: permite que os perfis entrem em vigência através de regras pré estabelecidas, por exemplo: nível de energia abaixo de um certo valor ou a partir de um certo horário;

(j) Automatização de perfis: permite a criação e ativação dos perfis de forma automatizada, baseado no perfil de uso do usuário do *smartphone*;

Além disso, a Tabela 10 apresenta um comparativo resumido sobre os aplicativos e as funcionalidades da lista apresentada que os mesmos possuem. A partir da Tabela 10 observa-se que a grande maioria, 85% dos aplicativos, possui uma forma de gerenciamento dos estados dos componentes através de perfis pré estabelecidos como forma de economizar energia, indicado por (g). Entretanto, nenhum deles possui alguma forma de automatização da criação e ativação do perfil, representado por (j). Ou seja, em todos os aplicativos o usuário tem que manualmente selecionar um perfil para ativar um modo de economia de

Tabela 10 – Análise das funcionalidades presentes em cada aplicativo

	a	b	c	d	e	f	g	h	i	j
Battery Doctor	x	x	x	x	x	x	x	x	x	
DU Battery Saver	x	x		x	x	x	x	x	x	
Avast Battery Saver	x	x				x	x		x	
JuiceDefender							x	x		
GO Battery Saver	x			x	x	x	x	x	x	
MPower	x	x	x				x			x*
Battery Optimizer & Cleaner	x	x				x				

* parcial, diferença relatada no texto

energia relacionado, ou através de regras preestabelecidas. Apenas o aplicativo MPower se aproxima desta funcionalidade, e, a partir de dados de uso do *smartphone* coletados, gerar propostas de perfis para controle dos componentes automaticamente, mas mesmo assim não faz a ativação desses perfis automaticamente, delegando essa tarefa ao usuário.

Outra funcionalidade, presente em 57% dos aplicativos, representada por (i), diz respeito a forma de ativação dos perfis de economia de energia. Se é ativada, por exemplo, através de alguma regra preestabelecida, como um horário específico ou o quando a energia do *smartphone* atinge um certo nível. Essa característica é importante para aumentar a eficiência do aplicativo pois retira a responsabilidade do usuário de ter que ativar esses perfis manualmente, o que pode minimizar as falhas, como no caso de o usuário esquecer de ativá-los.

Além disso, as funcionalidades representadas por (a) e (b) também estão presentes em grande parte dos aplicativos estudados, mas como elas não tem relação direta com a economia de energia, e sim com informações sobre a bateria, não foram consideradas importantes para serem exploradas com mais detalhes no estudo.

3.2.1.2 Análise dos componentes

A segunda análise teve como função levantar os principais componentes considerados pela funcionalidade (g) da subseção anterior. Essa funcionalidade foi considerada importante para o estudo, assim como as funcionalidades (h), (i) e (j), pois elas se relacionam diretamente a proposta do mecanismo, uma vez que tratam do gerenciamento dos componentes do *smartphone*. Para isso, os mesmos aplicativos considerados anteriormente foram analisados. A Tabela 11 apresenta todos os componentes que foram identificados durante a análise dos aplicativos, indicando quando o componente em questão é gerenciado pelo aplicativo. Os seguintes componentes foram encontrados:

(a) *Display*: considera aplicativos que controlam o brilho, *timeout* e orientação do *display*;

(b) Wi-Fi: considera aplicativos que controlam o estado do Wi-Fi;

Tabela 11 – Análise dos componentes gerenciados por cada aplicativo

	a	b	c	d	e	f	g
Battery Doctor	x	x	x	x		x	
DU Battery Saver	x	x	x	x		x	
Avast Battery Saver	x	x	x	x		x	
JuiceDefender	x	x			x		x
GO Battery Saver	x	x	x	x		x	
MPower	x	x	x		x		x
Battery Optimizer & Cleaner							

(c) *Bluetooth*: considera aplicativos que controlam o estado do *Bluetooth*;

(d) AutoSync: considera aplicativos que controlam o Sincronismo Automático, AutoSync (*Automatic Synchronism*), que permite que aplicativos se conectem de tempos em tempo na Internet para atualizar o conteúdo;

(e) Redes móveis: considera aplicativos que controlam o estado da conexão com a Internet através de redes móveis, como 3G e a LTE;

(f) Áudio: considera aplicativos que controlam os volumes de áudio do *smartphone*, como volume de toque e de *media* e também a vibração dos *smartphones*;

(g) GPS: considera aplicativos que controlam o estado do GPS;

Analisando-se a Tabela 11, nota-se que apenas os componentes representados por (e) e (g) foram os menos considerados dos aplicativos. Contudo, no presente trabalho, esses componentes foram considerados importantes e incluídos como componentes a serem gerenciados, pois a Figura 4, da Seção 2, demonstrou um crescimento para esses componentes, evidenciando a sua importância no assunto do consumo de energia em *smartphones*.

3.3 MyBatRecommender: mecanismo proposto

3.3.1 Visão Geral

O mecanismo proposto por esse trabalho tem como principal funcionalidade buscar economizar a energia de *smartphones* Android. Como visto nas seções anteriores, existem diversas formas de atingir esse resultado, e entre elas está o gerenciamento dos estados dos componentes do *smartphone*, que é a forma abordada pelo presente estudo. Mas, mesmo dentro dessa categoria de gerenciamento de componentes existem diversas formas de modelar e implementar a solução. A estrutura exposta pela Figura 7 e explicada a seguir apresenta o MyBatRecommender, mecanismo de gerenciamento de componentes de um *smartphone* Android proposto pelo presente trabalho.

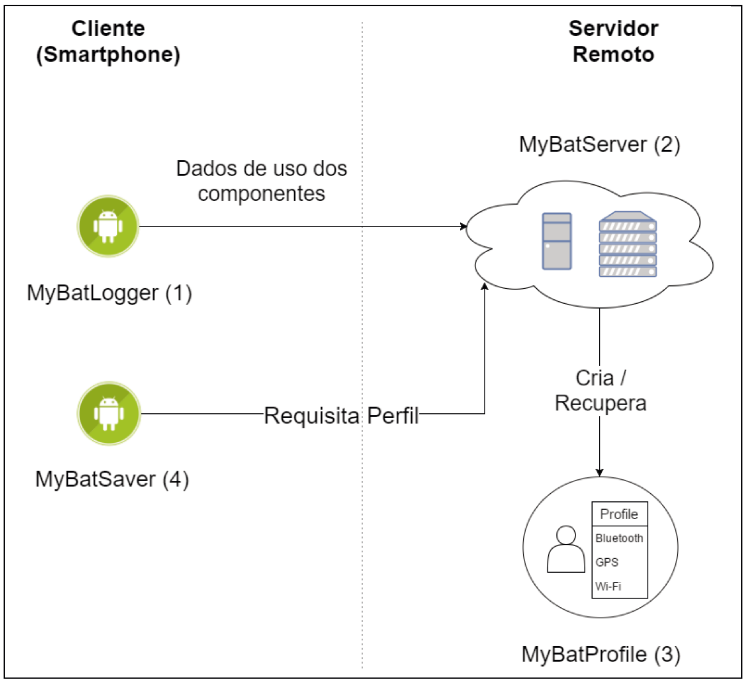


Figura 7 – MyBatRecommender: mecanismo proposto

O mecanismo é composto por quatro elementos, sendo o MyBatLogger (1), aplicativo Android responsável por coletar dados relativos aos componentes do *smartphone* e enviá-los a um servidor remoto. Uma vez no servidor remoto, o MyBatServer (2), os dados são armazenados e processados através de algoritmos a fim de gerar um perfil de uso do *smartphone* para cada usuário, o MyBatProfile (3), que representa a forma como o usuário utiliza seu *smartphone* ao longo dos dias, relacionando os componentes usados em cada dia e horário. Por fim, o elemento MyBatSaver (4), tem como função requisitar o MyBatProfile através de interação com o MyBatServer e aplicar ações definidas, controlando os estados dos componentes do *smartphone* automaticamente de acordo com esse perfil.

A estrutura do mecanismo é cíclica, ou seja, o MyBatLogger está sempre coletando e enviando os dados para o servidor remoto, que por sua vez está sempre atualizando o perfil de um determinado usuário para que o MyBatRecommender possa aplicar ações sempre atualizadas e cada vez mais eficientes para o usuário.

O mecanismo descrito na Figura 7 é composto de quatro partes principais, cada uma com uma funcionalidade específica, a saber, MyBatLogger, MyBatServer, MyBatProfile e MyBatSaver. A arquitetura e funcionamento dessas três partes serão explicadas com mais detalhes nas subseções a seguir.

3.3.2 MyBatLogger

Buscando entender como os componentes se comportam nos *smartphones* de usuários reais, o MyBatLogger foi especificado para ser a parte responsável pela coleta dos

dados relativos aos componentes dos usuários e posterior envio ao servidor remoto.

A funcionalidade de coleta dos dados foi denominada "gerar *log*": em intervalos de tempo de aproximadamente um minuto o aplicativo executa uma ação que consiste em coletar os dados relativos a todos os componentes do *smartphone*. Junto a esses dados também são coletados o *id* do usuário, único para cada *smartphone* e o *timestamp*, valor em milissegundos que representa o horário em que a ação foi executada. O atributo *id* do usuário é um valor que identifica o *smartphone* do usuário e não é um valor nominal, de forma que os usuários não podem ser identificados a partir dele. Todos esses dados coletados, juntos, compõe um *log*, que determina as configurações dos componentes abrangidos pelo presente estudo em um determinado momento.

A Figura 8 apresenta o ciclo de vida do MyBatLogger. Cada *log* gerado é armazenado no banco de dados local do *smartphone* e assim, quando houver um número de *logs* representativo, eles são enviados ao servidor remoto através da Internet. Além disso, qualquer alteração de estado de algum componente do *smartphone* faz com que um *log* seja gerado apenas para aquele componente em questão.

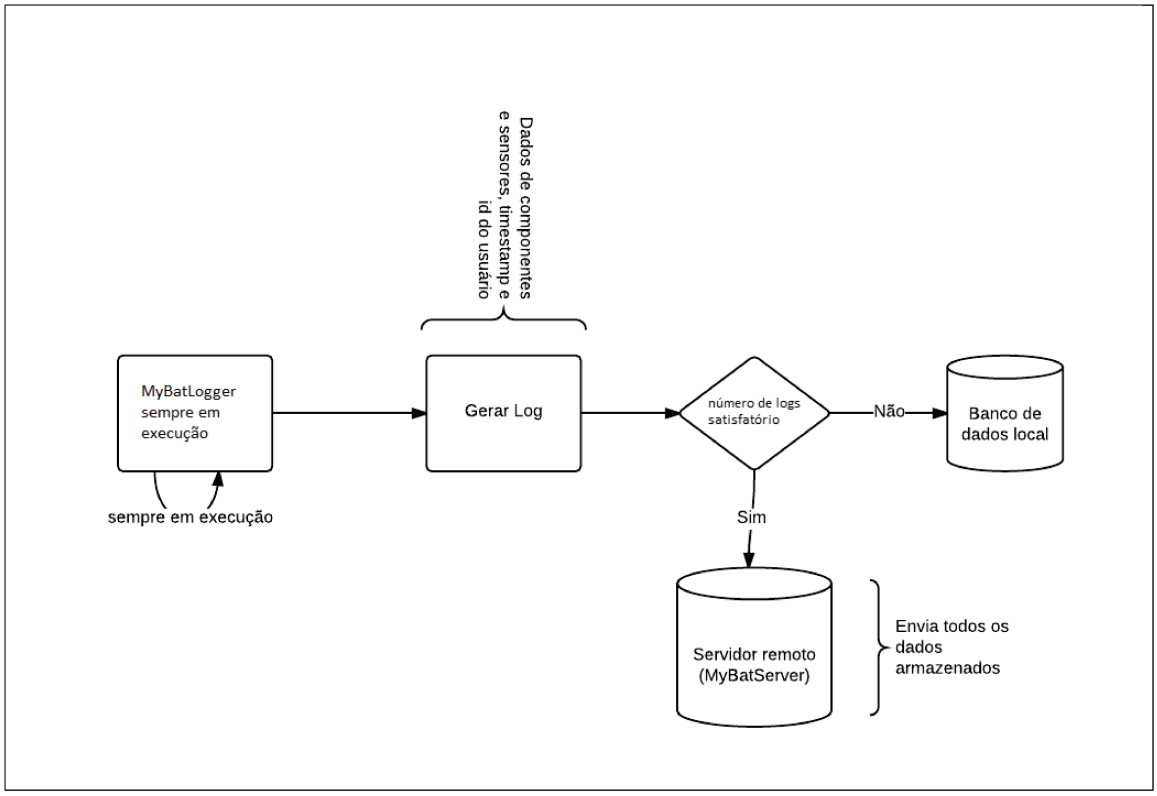


Figura 8 – Visão do funcionamento do MyBatLogger

3.3.3 MyBatServer

O MyBatServer é a parte servidora do sistema, responsável por receber as requisições do lado cliente, MyBatSaver e MyBatLogger, e processá-las de acordo com as suas

ações relacionadas. Esse elemento possui um papel importante pois retira a função de analisar os dados coletados e gerar o MyBatProfile do *smartphone*, tarefa que exige grande processamento computacional, o que elevaria o consumo de energia nos *smartphone* pelo MyBatRecommender.

Além das requisições existentes, o MyBatServer também possui um banco de dados, que considera os atributos coletados pelo MyBatLogger: cada componente possui uma tabela específica representando-o, com todos os campos necessários para poder armazenar os seus dados relativos. As ações que são tratadas pelo servidor podem ser resumidas em três principais, explicadas a seguir.

3.3.3.1 Inserir *Logs*

Essa ação é utilizada apenas pelo elemento MyBatLogger e é a responsável por inserir os dados coletados pelo MyBatLogger no banco de dados do servidor remoto, para que posteriormente possam ser utilizados a fim de gerar o MyBatProfile do usuário. Possui um retorno indicando se a operação foi bem sucedida ou não, para que o requisitante possa tomar ações necessárias.

3.3.3.2 Criar o MyBatProfile

Essa ação é utilizada apenas pelo elemento MyBatSaver sempre antes de requisitar o MyBatProfile. Essa é a requisição responsável por executar os métodos que são responsáveis por gerar o MyBatProfile, baseando-se nos dados coletados e armazenados.

Existem diversas formas de criar um perfil de uso para o usuário considerando diferentes dados. O MyBatRecommender não descreve um forma fixa a ser utilizada, deixando a critério da implementação.

3.3.3.3 Requisitar o MyBatProfile do usuário

Essa ação é utilizada apenas pelo mecanismo MyBatSaver que a requisita toda vez que é necessário, com ou sem intervenção do usuário. É através dela que se consegue o MyBatProfile, contendo todas as informações relativas aos estados dos componentes.

3.3.4 MyBatSaver

Para colocar em prática as ações definidas pelo MyBatProfile de cada usuário, o MyBatSaver é o responsável por aplicar os controles que irão de fato gerenciar os estados dos componentes do *smartphone*. Basicamente esse elemento, a partir da intervenção ou não do usuário, é executado e, através de interação com o MyBatServer, obtém o MyBatProfile que corresponde ao usuário que está interagindo e seu *smartphone*, utilizando-se para isso o *id* do usuário, único para cada *smartphone*.

Uma vez em posse do MyBatProfile, o MyBatSaver se programa para executar novamente em horários preestabelecidos, com base nas configurações definidas para aquele usuário, para poder aplicar as ações de gerenciamento dos estados dos componentes do *smartphone*, também definidas pelas configurações. Dessa forma, através das configurações, é possível identificar o estado em que cada componente deve estar, para cada horário de um dia específico.

Além disso, essa configuração é salva localmente, no banco de dados do *smartphone*, pois para executar a requisição de perfil é necessário que o *smartphone* esteja conectado à Internet, e caso essa condição não seja verdadeira, o aplicativo utilizará o último perfil que já tenha sido previamente baixado com sucesso.

A Figura 9 mostra o ciclo de vida do MyBatSaver, acima descrito.

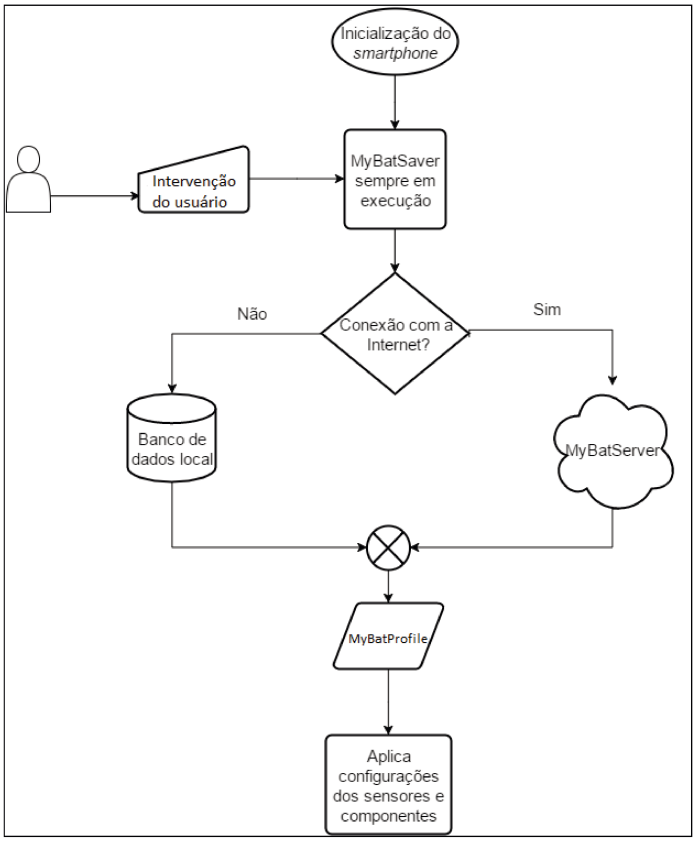


Figura 9 – Visão do funcionamento do MyBatSaver

3.4 MyBatRecommender: implementação

A implementação da proposta do mecanismo MyBatRecommender se deu através do desenvolvimento de dois aplicativos Android e uma aplicação servidora, representados pelos elementos MyBatLogger, MyBatServer e MyBatSaver. A escolha pelo sistema operacional Android se deu pelo fato do mesmo ser um sistema operacional de código aberto e bem documentado, tornando o desenvolvimento dos aplicativos mais fácil e de baixo custo.

Além disso, o Android oferece o controle de diversos componentes de hardware através das suas APIs, item necessário para a implementação da abordagem proposta, uma vez que a mesma se baseia no gerenciamento dos estados dos componentes.

Para os dois aplicativos Android utilizou-se a IDE (*Integrated Development Environment*) oficial de desenvolvimento Android, o Android Studio[10]. Para o desenvolvimento do aplicativo servidor, optou-se por utilizar o serviço PaaS (*Platform as a Service*) em *Cloud* oferecido por OpenShift[11] uma vez que o mesmo é gratuito para utilização dentro de determinados parâmetros como volume de dados trafegados e outros, dentro dos quais a implementação da abordagem proposta se encaixa. Os detalhes das outras tecnologias utilizadas na implementação da aplicação servidora estão detalhadas na subseção 3.4.2. A seguir será detalhada a implementação de cada elemento do mecanismo. A descrição da implementação do MyBatProfile será realizada juntamente com a descrição do MyBatServer, uma vez que estão atrelados.

3.4.1 MyBatLogger

Como o sistema operacional alvo do estudo é o Android, o MyBatLogger foi implementado na forma de um aplicativo Android. Para isso, um Serviço[12] que roda o tempo inteiro em *background* foi desenvolvido, ou seja, um aplicativo Android sem interface gráfica, que está executando desde o momento em que o *smartphone* é ligado e sem que o usuário perceba.

O MyBatLogger coleta e armazena informações de diversos componentes de um *smartphone* Android e isso é feito através das APIs que o Android expõe. Essas informações coletadas e armazenadas constituem um *log*, que representa o estado dos componentes para um determinado momento. Esses *logs* coletados são armazenados localmente e enviados ao MyBatServer quando houver um número grande que possa ser enviado pela Internet de forma que o tempo de conexão não se exceda. Isso é feito para que o MyBatLogger produza poucas requisições com o intuito de consumir menos energia. Esse envio ocorre apenas quando o *smartphone* estiver conectado à uma rede Wi-Fi, para que não haja gastos extras ao usuário através do consumo do pacote de dados.

Para o presente estudo, considerando a análise feita na subseção 3.2.1.2, os seguintes componentes foram considerados:

1. Aplicativos: são coletados o nome dos processos dos aplicativos que estão em execução no *smartphone*;

[10] http://developer.android.com/intl/pt-br/sdk/index.html
[11] https://www.openshift.com/
[12] http://developer.android.com/intl/pt-br/guide/components/services.html

2. *Bluetooth*: são coletados o estado do componente, ligado ou desligado, além do número de conexões atuais que esse componente tem e os estados atual e anterior de conexão do componente;

3. *Display*: são coletados o estado, ligado ou desligado e orientação, *portrait* ou *landscape*, representados por um inteiro. Além disso a altura, largura, brilho do *Display* e taxa de atualização;

4. Wi-Fi: são coletados o estado, ligado ou desligado, e o estado da conexão atual. Além disso, a quantidade de *bytes* transmitidos e recebidos desde a última inicialização do *smartphone* além da velocidade do link da conexão Wi-Fi, força de sinal e Endereço de Controle de Acesso, o endereço MAC (*Media Access Control*) da interface de rede do *smartphone*;

5. Bateria: são coletados o tipo da tecnologia da bateria, por exemplo, Li-ion, e se a bateria está ligada ou não ao carregador. Além disso, a temperatura e voltagem da bateria, seus níveis máximo e atual da carga de bateria e a vida útil da mesma;

6. Volumes: são coletados os níveis do volume de música, toque e sistema, se alguma música está tocando ou não e se o alto falante está ligado ou não. Além disso, qual é o modo atual do áudio;

7. Redes móveis: são coletados o estado atual, ligado ou desligado, o tipo de conexão, estado da ligação, força do sinal e atividade da conexão. Além disso, a quantidade de *bytes* transmitidos e recebidos desde a última inicialização do *smartphone* também são coletados;

8. GPS: é coletado o estado, ligado ou desligado, do componente;

A Tabela 12 apresenta resumidamente os principais componentes e variáveis coletadas pelo aplicativo MyBatLogger, bem como a classe do sistema operacional Android responsável por prover esses dados de cada componente.

Tabela 12 – Visão geral dos componentes e variáveis analisados por MyBatLogger

Componente	**Ações**	**Classe Android**
Wi-Fi	Estado do componente (ligado/desligado), Estado da conexão (conectando, recebendo dados, etc), Bytes transmitidos/recebidos desde a última inicialização, Velocidade da conexão, Força do sinal e Endereço MAC	WifiManager

Bluetooth	Estado do componente (ligado/desligado), Número de conexões, Estado da conexão (conectando, recebendo dados, etc), Estado da conexão anterior (conectando, recebendo dados, etc)	BluetoothManager[13]
Display	Estado do componente (ligado/desligado), Orientação (portrait, landscape), Altura, Largura, Taxa de atualização e Nível de brilho	PowerManager[14]
Bateria	Tecnologia, Plugada na tomada (sim ou não), Temperatura, Voltagem, Nível máximo, Nível atual e Vida útil	BatteryManager[15]
GPS	Estado do componente (ligado/desligado)	LocationManager[16]
Redes Móveis	Estado do componente (ligado/desligado), Tipo de conexão, Estado da ligação, Força do sinal e Atividade (enviando, recebendo dados, etc)	TelephonyManager[17]
Áudios	Volume de música do sistema, Volume de ligação do sistema, Volume do sistema, Música tocando (sim ou não), Speaker (ligado ou desligado) e Modo do áudio (ligação, normal, etc)	AudioManager[18]
Aplicativos	Nome dos processos dos aplicativos rodando	ActivityManager[19]

Além desses dados relativos aos componentes, também são coletados o *id* do usuário, obtido pela função *getDeviceId()* da classe TelephonyManager e o *timestamp*, que representa o momento em que aquele *log* foi gerado, em millisegundos desde 1 de Janeiro de 1970, obtida através da função *currentTimeMillis()* da classe System [20].

[13] http://developer.android.com/intl/pt-br/reference/android/bluetooth/BluetoothManager.html
[14] http://developer.android.com/intl/pt-br/reference/android/os/PowerManager.html
[15] http://developer.android.com/intl/pt-br/reference/android/os/BatteryManager.html
[16] http://developer.android.com/intl/pt-br/reference/android/location/LocationManager.html
[17] http://developer.android.com/intl/pt-br/reference/android/telephony/TelephonyManager.html
[18] http://developer.android.com/intl/pt-br/reference/android/media/AudioManager.html
[19] http://developer.android.com/intl/pt-br/reference/android/app/ActivityManager.html
[20] http://developer.android.com/intl/pt-br/reference/java/lang/System.html

Para executar essa coleta, o MyBatLogger dispõe de dois mecanismos de coleta principais, um intervalar e outro através de BroadcastReceiver[21]. O mecanismo intervalar é o responsável por executar a coleta dos dados de todos os componentes a cada aproximadamente um minuto. Para tal, utilizou-se o método *setRepeating()* da classe AlarmManager[22], fazendo com que o serviço em execução gerasse o *log* naquele momento programado.

O segundo mecanismo foi estruturado de forma a utilizar um recurso fornecido pelo sistema operacional Android, BroadcastReceiver, que permite que os aplicativos Android respondam e tratem a eventos provenientes de outros aplicativos Android instalados no mesmo *smartphone* e também de eventos do próprio sistema operacional. Para isso, é necessário que o aplicativo defina qual é o evento que está interessado, através das ações desses eventos, como explicado em Developers (2015b). Dessa forma, o MyBatLogger é capaz de responder a eventos provenientes do sistema operacional relativos a mudança de estado dos diversos componentes do *smartphone* em que está instalado. A Tabela 13 apresenta os componentes que possuem essa forma de coleta e as ações necessárias para que possam receber os eventos. Observa-se que os componentes definidos como Aplicativos, Volumes, Redes Móveis e GPS não são contemplados pelo mecanismo *Broadcast*. Isso se dá pois o sistema operacional Android não disponibiliza acesso aos estados dos mesmos através do mecanismo de BroadcastReceiver. Dessa forma, esses componentes tem seus *logs* apenas através do primeiro mecanismo, o intervalar.

Tabela 13 – Ações necessárias para receber eventos relativos aos componentes

Componente	**Ações**
Wi-Fi	android.net.wifi.WIFI_STATE_CHANGED, android.net.wifi.WIFI_STATE_CHANGED e android.net.wifi.supplicant.CONNECTION_CHANGE
Bluetooth	android.bluetooth.adapter.action.CONNECTION_STATE_CHANGED e android.bluetooth.adapter.action.STATE_CHANGED
Display	android.intent.action.SCREEN_OFF e android.intent.action.SCREEN_ON
Bateria	android.intent.action.BATTERY_CHANGED

3.4.2 MyBatServer

O elemento MyBatServer foi desenvolvido utilizando-se a solução OpenShift [23] que é uma plataforma de hospedagem de aplicações na nuvem. As tecnologias utilizadas foram: Java para o desenvolvimento do aplicativo servidor, Tomcat 7 [24] para servidor

[21] http://developer.android.com/intl/pt-br/reference/android/content/BroadcastReceiver.html
[22] http://developer.android.com/intl/pt-br/reference/android/app/AlarmManager.html
[23] https://www.openshift.com/
[24] http://tomcat.apache.org/

web, MySQL 5.5 [25] para o desenvolvimento do banco de dados e phpMyAdmin [26] para o gerenciamento do banco de dados. As próximas subseções detalham o funcionamento e implementação das requisições que o MyBatServer trata.

3.4.2.1 Inserir *Logs*

Essa ação está acessível através do método do Protocolo de Transferência de Hipertexto, HTTP (*Hypertext Transfer Protocol*) POST em um Localizador Padrão de Recursos, URL (*Uniform Resource Locator*). Ao utilizar essa requisição, o módulo MyBatLogger envia todos os dados coletados dos componentes na forma de um JSONArray, que é um *array* de Notação de Objetos JavaScript, o JSON[27] (*JavaScript Object Notation*). Cada objeto JSON nesse *array* representa um *log* específico de um componente coletado que pode ser identificado através do seu atributo inteiro *type*, organizado da seguinte forma:

1. Aplicativos: *logs* coletados dos aplicativos rodando no *smartphone*;
2. Volumes: *logs* coletados com informações dos níveis de volume do sistema;
3. Bateria: *logs* coletados com informações de bateria;
4. *Bluetooth*: *logs* coletados com informações do *Bluetooth*;
5. Error: *logs* coletados de erros de execução do aplicativo MyBatLogger;
6. GPS: *logs* coletados com informações do GPS;
7. *Display*: *logs* coletados com informações do *display*;
8. Redes Móveis: *logs* coletados com informações das redes móveis;
9. Wi-Fi: *logs* coletados com informações do Wi-Fi;

Além do atributo *type*, cada objeto JSON contém também os atributos relacionados ao componente que representam, o *id* do usuário e o *timestamp* daquele *log*, como explicado na subseção 3.4.1. Dessa forma, ao receber a requisição de inserir *logs*, o MyBatServer itera sobre todos os elementos do *array* recebido e salva-os no banco de dados para que possam ser utilizados posteriormente. Caso algum erro ocorra durante a inserção de algum dos *logs*, o servidor desfaz as operações de inserção dos *logs* anteriores daquele mesmo *array* recebido e retorna o erro ocorrido. Caso contrário, o número de instâncias inseridas com sucesso é retornado.

[25] https://www.mysql.com/
[26] https://www.phpmyadmin.net/
[27] http://www.json.org/

3.4.2.2 Criar o MyBatProfile

Essa ação está acessível em uma URL, através do método HTTP GET. Essa URL recebe como parâmetro o *userOp* que representa a ação a ser tomada, onde valor "1" representa a ação de criar o MyBatProfike do usuário e o parâmetro *userId* que representa o *id* do usuário, onde o valor deve ser o obtido como explicado na subseção 3.4.1. Não há retorno para essa requisição, mas é através dela que o MyBatProfile é criado e armazenado no banco de dados em uma tabela específica, para retorná-lo quando houver a requisição desse perfil, apresentado na subseção 3.4.2.3.

Como citado na subseção 3.3.3.2, o mecanismo MyBatRecommender deixa a encargo da implementação como será feita a criação do MyBatProfile. Considerando isso, o presente trabalho criou uma hipótese baseando-se na existência de uma rotina diária de uso dos *smartphones*. Ou seja, considerou-se que para determinados grupos de pessoas a sua rotina durante os dias da semana e durante as semanas possa se repetir com uma dada frequência. Por exemplo, dado um grupo de estudantes de uma faculdade, os mesmos poderiam ter uma rotina diária que envolveria se transportar de suas casas para a faculdade, assistir aulas e se transportar da faculdade para a casa ao término das aulas. O mesmo seria válido para outros grupos, como por exemplo trabalhadores de uma certa empresa, que poderiam ter uma rotina diária que envolveria o transporte de suas casas ao local de trabalho, trabalhar e o transporte do local de trabalho de volta para suas casas.

Assim, através da rotina de uma pessoa pode-se extrair uma rotina de uso dos *smartphones*. Isso se dá pois o uso dos *smartphones* está atrelado a diversas variáveis, entre elas o ambiente em que o usuário está inserido, que dita a disponibilidade de elementos como redes Wi-Fi e também o estado do usuário, que interfere, por exemplo, na orientação do *smartphone*, que muda de acordo com a rotina do usuário. Para o exemplo do grupo de estudantes acima, pode-se considerar que enquanto o usuário está se transportando não há disponibilidade de redes Wi-Fi, logo o mesmo tem acesso à Internet apenas através das redes móveis. De forma análoga, quando ele está em casa ou durante as aulas, as redes Wi-Fi estão disponíveis, logo as redes móveis não são mais utilizadas.

Dessa forma, o presente trabalho propõe a criação do MyBatProfile de forma que o mesmo possuirá as configurações dos estados dos componentes do *smartphone* para cada período do dia e para todos os dias de uma semana. Para isso o mecanismo fará a coleta dos dados relativos aos componentes do *smartphone* durante uma semana para que os mesmos possam ser analisados e o MyBatProfile gerado para que então, a partir da segunda semana de utilização, o MyBatRecommender possa aplicar as recomendações baseadas no perfil de uso do usuário.

Para isso, fez-se necessário definir por quantos períodos um dia seria composto. A definição dos períodos pode ser feita de duas maneiras principais, onde a primeira

considera-os estáticos, através de um número definido de períodos para um dia. A segunda forma considera-os dinâmicos, de forma que se adaptam em quantidade e horários ao cotidiano do usuário. A implementação atual utilizou a criação de períodos estáticos, utilizando-se para tal do número de ambientes diferentes visitados por um usuário nos períodos diurno, vespertino e noturno de um dia, uma vez que, como discutido acima, cada ambiente interfere diretamente na forma em que o *smartphone* é utilizado.

Para conseguir ter uma suposição do número de períodos de cada dia, foi aplicado um questionário com seis pessoas, apresentado pelo Anexo B, e através do seu resultado chegou-se a uma média de 5,4 períodos por dia. Como o número de períodos tem que ser um número inteiro, considerou-se o valor arredondado para cima, 6, definidos da seguinte maneira: o período 1 compreende o horário das 00:00 horas até as 03:59 horas. O período 2 compreende o horário das 04:00 horas até as 07:59 horas. O período 3 compreende o horário das 08:00 horas até as 11:59 horas. O período 4 compreende o horário das 12:00 horas até as 15:59 horas. O período 5 compreende o horário das 16:00 horas até as 19:59 horas e por fim o período 6 compreende o horário das 20:00 horas até as 23:59 horas. Além disso, fez-se necessário definir a configuração dos estados dos componentes para cada período.

Para definir e criar o MyBatProfile, o módulo MyBatServer executa alguns algoritmos em sequência, baseado nos dados coletados e já armazenados no banco de dados. A seguir estão apresentados os algoritmos executados em sua ordem de execução e a função de cada um no processo de geração do MyBatProfile.

Inicialmente o método ***separateByDays*** é executado: considerando que o MyBatProfile irá possuir a configuração de cada componente para os períodos de um dia, durante os dias de uma semana, é necessário que os *logs* coletados sejam analisados em grupo, onde cada grupo possui os *logs* relativos a um período de um dia da semana. Para isso, o primeiro passo da geração do MyBatProfile executa o método *separateByDays()*. Esse método tem a função de agrupar todos os *logs* existentes no banco de dados do servidor em grupos de *logs* que tenham sido coletados no mesmo dia. Ou seja, considera-se o valor do atributo *timestamp* que representa a data em que o *log* foi coletado, e através de comparações utilizando-se os atributos da classe Calendar[28] define-se se dois *logs* foram coletados no mesmo dia ou não. Dois *logs* são considerados do mesmo dia se os anos, representado por Calendar.YEAR, forem iguais e também o dia do ano, representado por Calendar.DAY_OF_YEAR, forem iguais, como mostra a Figura 10.

O resultado da execução do método *separateByDays()* é uma lista de objetos do tipo **Dia**, que contém uma lista de objetos **Log** que representa todos os *logs* coletados daquele dia. O objeto **Dia** possui ainda uma lista de objetos do tipo **Periodo**, que será preenchida após a execução do método seguinte, *separateByPeriods()*, explicado na sequência. O

[28] http://developer.android.com/intl/pt-br/reference/java/util/Calendar.html

esquema de classes e seus relacionamentos da aplicação servidora estão representados na Figura 11.

```java
/**
 * Check if two items are in the same day
 *
 * @param item1 the first {@link BaseItem}
 * @param item2 the second {@link BaseItem}
 * @return true if they are in the same day, false otherwise
 */
public static boolean isSameDay(BaseItem item1, BaseItem item2) {
    Date date1 = new Date(item1.getTimestamp());
    Date date2 = new Date(item2.getTimestamp());
    return isSameDay(date1, date2);
}

/**
 * Check if two dates are in the same day
 *
 * @param date1 the first date
 * @param date2 the second date
 * @return boolean indicating if they are in the same day
 */
public static boolean isSameDay(Date date1, Date date2) {
    Calendar cal1 = Calendar.getInstance();
    Calendar cal2 = Calendar.getInstance();
    cal1.setTime(date1);
    cal2.setTime(date2);
    boolean sameDay = cal1.get(Calendar.YEAR) == cal2.get(Calendar.YEAR)
            && cal1.get(Calendar.DAY_OF_YEAR) == cal2
                    .get(Calendar.DAY_OF_YEAR);
    return sameDay;
}
```

Figura 10 – Trecho do código da função que verifica se dois *logs* foram coletados no mesmo dia

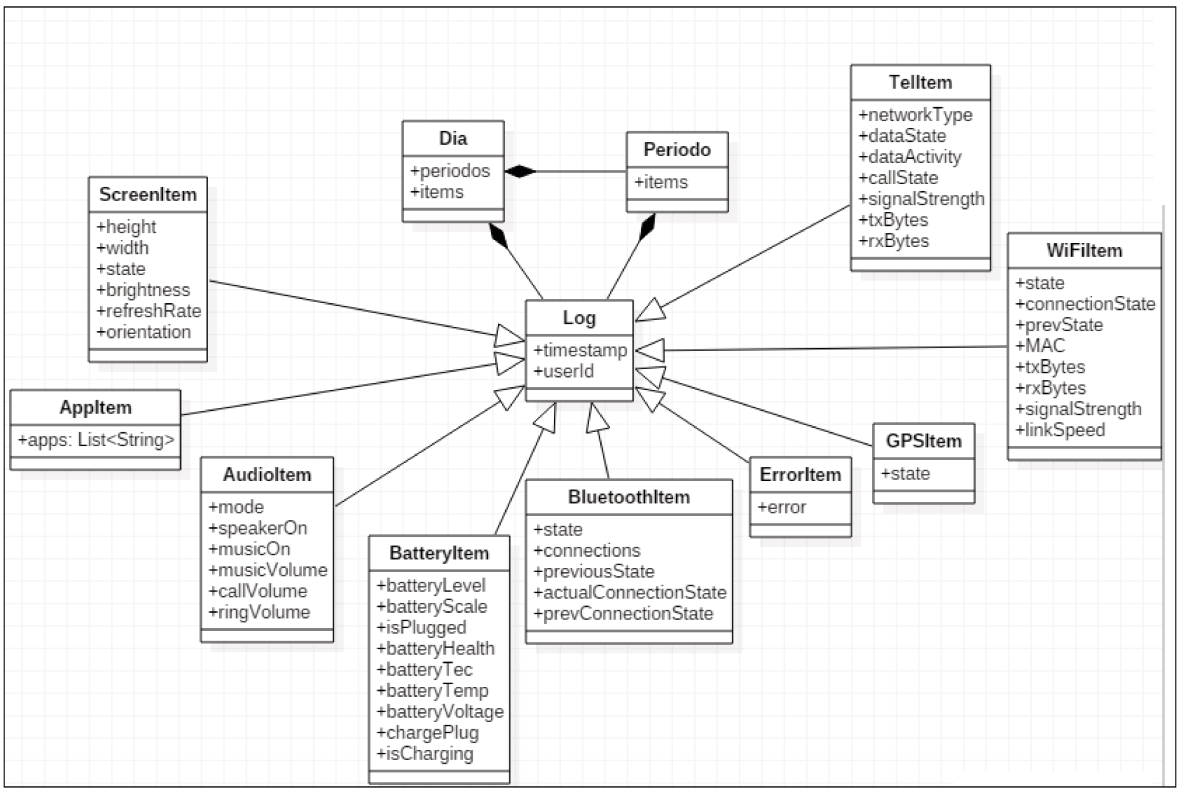


Figura 11 – UML das classes do MyBatLogger

Na sequência executa-se o método ***separateByPeriods***: da mesma forma que o método *separateByDays()*, o método *separateByPeriods()* segue a mesma lógica e tem

como objetivo agrupar os *logs* de um objeto **Dia** nos períodos definidos, descritos no começo dessa subseção. Ou seja, da mesma forma considera-se o atributo *timestamp* e através de comparações utilizando-se o atributo Calendar.HOUR_OF_DAY da classe Calendar define-se a qual período um *log* pertence, como mostra a Figura 12. O resultado da execução do método *separateByPeriods()* é preencher a lista de objetos do tipo **Periodo** que um objeto do tipo **Dia** possui. Um objeto do tipo **Periodo** contém uma lista de objetos do tipo **Log** que representa todos os *logs* coletados daquele período do dia.

```
/**
 * Separate this day in periods and fulfill the array o periods.
 */
public void separateByPeriods() {
    // Build the periods
    if (mItems != null) {
        for (int i = 0; i < mItems.size(); i++) {
            Object specificItem = mItems.get(i);
            long timestamp = 0;
            if (specificItem instanceof Log) {
                timestamp = ((Log) specificItem).getTimestamp();
            }

            Calendar calendar = Calendar.getInstance();
            calendar.setTimeInMillis(timestamp);
            int hours = calendar.get(Calendar.HOUR_OF_DAY);

            if (hours >= 0 && hours < 4) {
                mPeriods.get(0).add(specificItem);
            } else if (hours >= 4 && hours < 8) {
                mPeriods.get(1).add(specificItem);
            } else if (hours >= 8 && hours < 12) {
                mPeriods.get(2).add(specificItem);
            } else if (hours >= 12 && hours < 16) {
                mPeriods.get(3).add(specificItem);
            } else if (hours >= 16 && hours < 20) {
                mPeriods.get(4).add(specificItem);
            } else if (hours >= 20 && hours < 24) {
                mPeriods.get(5).add(specificItem);
            }
        }
    }
```

Figura 12 – Trecho do código da função que agrupa os *logs* de um dia por períodos

Feito isso, o método ***createUserProfile*** também é executado e é o responsável por efetivamente criar as configurações de cada período: após os *logs* terem sido agrupados em dias e períodos eles estão prontos para serem analisados a fim de gerar o MyBatProfile. É esse o papel da função *createUserProfile()*, que percorre todos os objetos **Periodo** criados, que estão dentro dos objetos **Dia** criados e executa a função ***analyzeType()***. Essa última função analisa todos os objetos do tipo **Log** de cada objeto **Periodo** da seguinte maneira: para cada objeto analisa-se primeiramente a qual componente aquele *log* pertence através do atributo *type*, como explanado na subseção 3.4.2.1. A partir daí, verificações específicas são feitas para cada tipo de objeto, o que define o estado de cada componente para aquele determinado período. Algumas dessas verificações utilizaram uma condição que define se um item é considerado **usado ou não usado**, condição essa explicada na Tabela 14. A lista a seguir mostra a intenção e a análise feita para os *logs* de cada tipo de componente:

- Aplicativos: encontrar quais são os aplicativos mais usados pelo usuário em um período. Para isso conta-se a quantidade de vezes que cada aplicativo apareceu nos *logs* coletados e ordena-se a lista com os nomes dos aplicativos considerando

a quantidade de vezes que apareceram nos *logs*. Feito isso, obtém-se os nomes dos cinco aplicativos que mais apareceram nos *logs*;

- Volume: definir o valor do nível dos volumes dos diferentes tipos de áudio para aquele período. Para isso faz-se uma média simples com os valores coletados de todos os *logs* de cada tipo de áudio considerado. Feito isso, obtém-se um vetor de três valores inteiros, representando os níveis dos volumes para cada tipo de áudio;

- Bateria: manter um controle do nível de bateria do *smartphone* durante a execução do aplicativo e também identificar os períodos em que o *smartphone* é carregado. Para o nível de energia faz-se uma média simples dos valores coletados por todos os *logs* daquele período. Para a identificação de quando o *smartphone* estava sendo carregado, considera-se apenas os *logs* que foram considerados **não utilizados**. Caso essa quantidade for maior que a metade de todos os *logs* de bateria daquele período, considera-se que o *smartphone* estava sendo carregado naquele período;

- *Bluetooth*: definir qual é o estado do mesmo para o período. Para isso analisou-se a quantidade de *logs* desse tipo que foram considerados **usados**. Caso essa quantidade seja maior que a metade de todos os *logs* de *Bluetoot* daquele período, considera-se que o estado desse componente era ligado naquele período;

- *Error*: coletar erros de execução do aplicativo MyBatLogger. Esses dados são utilizados apenas para controle da execução do aplicativo e não influenciam na criação do MyBatProfile do usuário;

- GPS: definir qual é o estado do mesmo para o período. Para isso analisou-se a quantidade de *logs* desse tipo que foram considerados **usados**. Caso essa quantidade seja maior que a metade de todos os *logs* de GPS daquele período, considera-se que o estado desse componente era ligado naquele período;

- *Display*: definir qual é o nível do brilho do *display* e taxa de atualização do *display* para o período. Para isso fez-se uma média simples de todos os *logs* que foram considerados **usados** daquele período, obtendo-se os dois valores desejados;

- Redes Móveis: definir qual é o estado do mesmo para o período. Para isso analisou-se a quantidade de *logs* desse tipo que foram considerados **usados**. Caso essa quantidade seja maior que a metade de todos os *logs* de Redes Móveis daquele período, considera-se que o estado desse componente era ligado naquele período;

- Wi-Fi: a intenção em analisar os *logs* do tipo Wi-Fi é definir qual é o estado do mesmo para o período. Para isso analisou-se a quantidade de *logs* desse tipo que foram considerados **usados**. Caso essa quantidade seja maior que a metade de todos os *logs* de Wi-Fi daquele período, considera-se que o estado desse componente era ligado naquele período;

Tabela 14 – Condições para considerar componentes usados

Componente	**Condições para considerar-se usado**
Bateria	O estado da bateria deve ser diferente do valor definido por BATTERY_STATUS_CHARGING, da classe BatteryManager, indicando que o *smartphone* não estava sendo carregado naquele momento
Bluetooth	O estado do *Bluetooth* deve ser igual ao valor definido por STATE_ON e o estado da conexão atual deve ser igual ao valor definido por STATE_CONNECTED ou STATE_CONNECTING definidos pela classe BluetoothAdapter, indicando que o componente estava ligado e conectado ou conectando a algum elemento
GPS	O estado do GPS deve ser igual ao valor GPS_EVENT_STARTED ou GPS_EVENT_SATELLITE_STATUS definido pela classe GpsStatus indicando que o GPS estava ligado e executando
Display	O estado do *display* deve ser ligado, representado pelo Intent.ACTION_SCREEN_ON da classe Intent, indicando que o *display* estava ligado
Redes Móveis	O estado da conexão deve ser igual ao valor DATA_CONNECTING ou DATA_CONNECTED e o estado da transferência de dados deve ser DATA_ACTIVITY_IN, DATA_ACTIVITY_OUT ou DATA_ACTIVITY_INOUT, definidos pela classe PhoneStateListener, indicando que o componente estava ligado e executando
Wi-Fi	O estado da conexão deve ser igual ao valor WIFI_STATE_ENABLED definido pela classe WifiManager e o estado da conexão deve ser igual a CONNECTED ou CONNECTING definido pela classe NetworkInfo.DetailedState, indicando que o componente estava ligado e executando

Por fim, baseado no resultado da análise de cada componente, o método ***insertProfile*** insere o perfil gerado em uma tabela específica no banco de dados do servidor para que as próximas chamadas para requisitar o MyBatProfile do usuário retornem o perfil atualizado e de forma rápida.

Ao final da execução sequencial desses métodos desenvolvidos, o usuário possuirá o elemento MyBatProfile criado e armazenado nos bancos de dados do servidor para que possa ser retornado sempre que requisitado pelos outros elementos.

3.4.2.3 Requisitar MyBatProfile

Essa ação está acessível em uma URL através do método HTTP GET, que recebe como parâmetros o *userOp*, que representa a ação a ser tomada, onde o valor "0" representa a ação de requisitar o MyBatProfile do usuário e o parâmetro *userId* que representa o *id* do usuário, onde o valor deve ser o obtido como apresentado na subseção 3.4.1.

O retorno dessa requisição é um objeto JSON, como mostra a Figura 13. Essa figura mostra apenas um exemplo de retorno, onde os valores dos campos não representam

valores válidos. A representação de cada campo desse objeto é dada pelos campos a seguir:

(a) *id*: representa o *id* do usuário, o mesmo usado como parâmetro na requisição;

(b) *days*: é uma lista de objetos JSON, onde cada objeto representa as configurações do usuário para determinado dia;

(c) *daysCount*: campo inteiro que indica quantos objetos existem na lista *days*;

(d) *periods*: é uma lista de objetos JSON, contida por cada objeto da lista *days*, onde cada objeto representa as configurações do MyBatProfile para um determinado período de um determinado dia;

(e) *periodsCount*: campo inteiro contido por cada objeto da lista *days* que indica quantos objetos existem na lista *periods*;

(f) *dayName*: campo inteiro contido por cada objeto da lista *days* que indica o dia relativo da semana que aquele objeto representa, baseado na classe Calendar[29];

(g) *configuration*: é um objeto JSON, contido por cada objeto da lista *periods*, que possui diversos atributos que representam as características dos componentes relativas ao MyBatProfile do usuário para um dado período de um determinado dia;

(h) *start*: é um campo *long* contido por cada objeto da lista *periods* que indica o horário de início da configuração;

(i) *end*: é um campo *long* contido por cada objeto da lista *periods* que indica o horário de término dessa configuração;

(j) *ring*: indica o volume de toque;

(k) *call*: indica o volume da ligação;

(l) *sound*: indica o volume de som do sistema;

(m) *mobileUsed*: indica a quantidade de eventos de redes móveis considerados utilizados, como explicado na subseção 3.4.2.2;

(n) *gpsUsed*: indica a quantidade de eventos de GPS considerados utilizados, como explicado na subseção 3.4.2.2;

(o) *wifiUsed*: indica a quantidade de eventos de Wi-Fi considerados utilizados, como explicado na subseção 3.4.2.2;

(p) *btUsed*: indica a quantidade de eventos de *Bluetooth* considerados utilizados, como explicado na subseção 3.4.2.2;

[29] http://developer.android.com/intl/pt-br/reference/java/util/Calendar.html

(q) *charging*: indica o estado, carregando ou não, da bateria;

(r) *bright*: indica o brilho da *display* da configuração;

(s) *bt*: indica o estado, ligado ou desligado, do *Bluetooth*;

(t) *wifi*: indica o estado, ligado ou desligado, do Wi-Fi;

(u) *gps*: indica o estado, ligado ou desligado, do GPS;

(v) *mobile*: indica o estado, ligado ou desligado, das Redes Móveis;

(w) *apps*: indica os cinco aplicativos mais utilizados;

(x) *battery*: indica o nível de carga do celular;

(y) *synctime*: indica o tempo de AutoSync;

(z) *refresh*: indica o tempo de atualização do *display*;

3.4.3 MyBatSaver

Assim como o MyBatLogger, tendo em vista que o sistema operacional alvo do estudo é o Android, o MyBatSaver foi implementado na forma de um aplicativo Android. Este aplicativo é o responsável por aplicar as configurações de gerenciamento dos estados dos componentes com base no MyBatProfile. Esses controles são feitos todos baseados nas APIs que o sistema operacional Android dispõe, disponíveis no site oficial para desenvolvedores Android (DEVELOPERS, 2015a), sem que seja necessário fazer alterações no nível do próprio sistema operacional, tornando o sistema disponível para usuários comuns de *smartphones* Android.

Para isso, o MyBatSaver foi implementado da seguinte forma: um Serviço Android, sem interface gráfica, ou seja, invisível para o usuário, inicia-se automaticamente no momento em que o *smartphone* é ligado, sem que seja necessário que o usuário execute-o manualmente. Dessa forma, toda primeira vez que o MyBatSaver é inicializado, uma busca é realizada no servidor para que seja baixado ou atualizado o MyBatProfile. Essa busca é feita através do *id* do usuário, que é conseguido da mesma forma que o aplicativo MyBatLogger faz, uma vez que esse método sempre retorna um *id* único para cada *smartphone*, usando as requisições explicadas na subseção 3.4.2.

Além disso, o MyBatSaver, quando executado manualmente através do menu de aplicativos padrão do Android, possui um interface gráfica que permite que o usuário possa, a qualquer instante, forçar a atualização do MyBatProfile. A Figura 14 apresenta a tela do aplicativo que contém essa opção.

```
{
  "id":"004402147264133",
  "days":[
    {
      "periods":[
        {
          "start":0,
          "configuration":{
            "ring":0,
            "call":0,
            "mobileUsed":0,
            "gpsUsed":0,
            "charging":false,
            "brigth":0,
            "bt":true,
            "wifi":true,
            "sound":0,
            "wifiUsed":0,
            "apps":"nullnullnullnullnull",
            "battery":0,
            "gps":true,
            "synctime":40,
            "autosync":true,
            "btUsed":0,
            "refresh":0,
            "mobile":true
          },
          "end":0
        },
        { },
        { },
        { },
        { },
        { }
      ],
      "periodsCount":6,
      "dayName":0
    },
    { }
  ],
  "daysCount":2
}
```


Figura 13 – Atributos dos objetos JSON representando o MyBatProfile do usuário retornado pelo MyBatServer

Para executar a ação de requisitar o MyBatProfile é necessário que o *smartphone* esteja conectado à Internet. Caso essa condição não seja verdadeira, o aplicativo tentará usar algum perfil que já tenha sido previamente baixado com sucesso, e que consequentemente estará salvo no banco de dados local do aplicativo.

Uma vez em posse do MyBatProfile no formato JSON, como mostrado na Figura 13, o aplicativo MyBatSaver decodifica-os e cria objetos representativos para uma melhor organização do processo, como representados na Figura 15. A partir desse ponto, algoritmos são executados para encontrar a configuração atual e dar início ao processo de gerenciamento dos estados dos componentes.

Considerando o objeto **Perfil** criado a partir da decodificação da resposta do MyBatServer, procura-se na sua lista de objetos **Dia** o dia referente ao dia da semana atual através do atributo *dayName*. Como esse atributo é um inteiro, que representa o dia da semana com base na classe Calendar, utiliza-se o valor Calendar.DAY_OF_WEEK

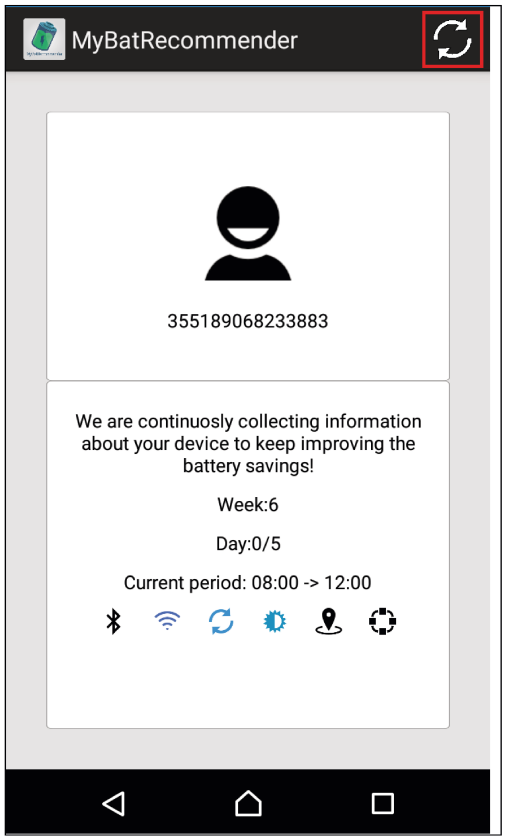


Figura 14 – MyBatSaver: requisição manual de perfil

dessa mesma classe para identificar o dia da semana atual. Encontrado esse objeto, procura-se na sua lista de objetos **Período** o período atual do dia, através dos seus atributos *start* e *end*. Para isso, esse **Período** tem que ter o horário de início, representado por *start*, menor ou igual o horário atual do *smartphone* e o horário de término, representado por *end*, maior que o horário atual do *smartphone*. Por fim, com o objeto **Período** encontrado, extrai-se diretamente o objeto **Configuração** que representa a configuração dos estados dos componentes que deve ser aplicada no *smartphone* durante o período compreendido entre o seu horário de início e término.

Com o objeto **Configuração** definido, o aplicativo programa dois alarmes através das APIs do AlarmManager[30] do Android, que provê formas de programar a execução do aplicativo para datas e horários futuros. Assim, o primeiro alarme é programado adicionando-se um minuto ao horário atual do *smartphone* para programar o início da configuração. Isso representa que, um minuto após o aplicativo baixar o MyBatProfile do usuário do servidor, ele colocará em prática as ações definidas pela configuração atual. O processo de colocar em prática as ações definidas por uma configuração será explicado a seguir. O segundo alarme é programado para o horário de término do período a qual aquela configuração pertence. Isso representa que, chegado esse horário, o aplicativo repetirá os passos descritos anteriormente para encontrar a próxima configuração e colocar em prática

30 http://developer.android.com/intl/pt-br/reference/android/app/AlarmManager.html

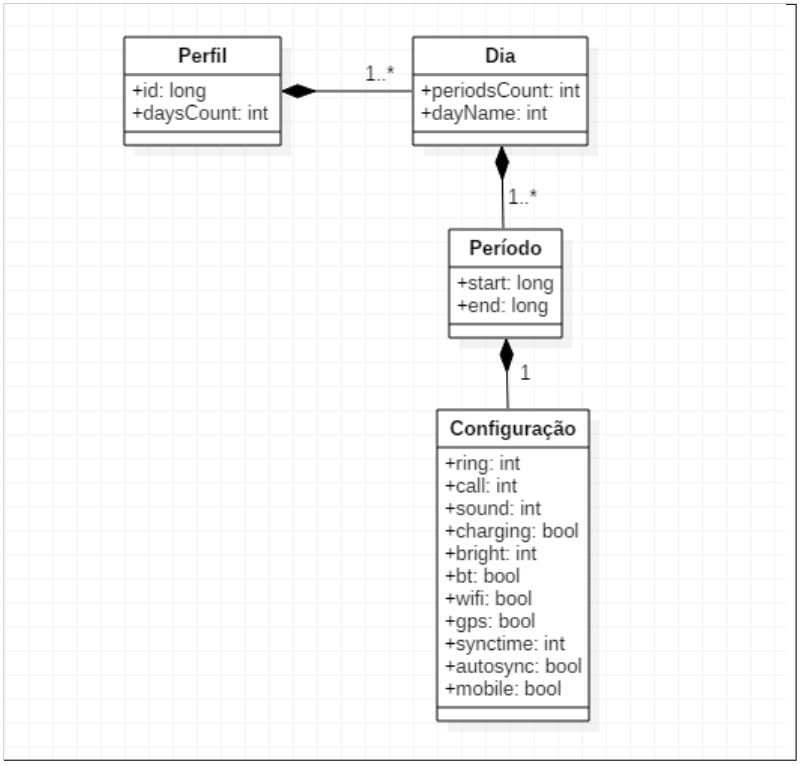


Figura 15 – UML das classes do MyBatSaver

as ações da configuração relacionada ao período seguinte. Esse processo está representado na Figura 16.

Para colocar em prática as ações de uma configuração, o aplicativo baseia-se nos atributos dessa configuração, definidos pelo objeto **Configuração**, como mostra a Figura 15, acionando controles de gerenciamento dos estados dos componentes através das APIs do Android. A lista a seguir relaciona os atributos de um objeto Configuração com suas APIs:

1. *ring, call, sound*: Através da classe AudioManager, utiliza-se o método *setStreamVolume()*, que permite alterar o volume de diferentes tipos de áudio do sistema;
2. *bright*: Através da classe WindowManager, é possível alterar o parâmetro *screenBrightness*, que muda o brilho do *display*;
3. bt: Através da classe BluetoothAdapter, é possível habilitar ou desabilitar o *Bluetooth*, pelos dos métodos *enable()* e *disable()*;
4. wifi: Através da classe WifiManager[31] é possível habilitar ou desabilitar o Wi-Fi, pelo método *setWifiEnabled()*;
5. gps: Através da classe Settings.Secure é possível habilitar ou desabilitar o GPS pelo método *putString()*;

[31] http://developer.android.com/intl/pt-br/reference/android/net/Wi-Fi/Wi-FiManager.html

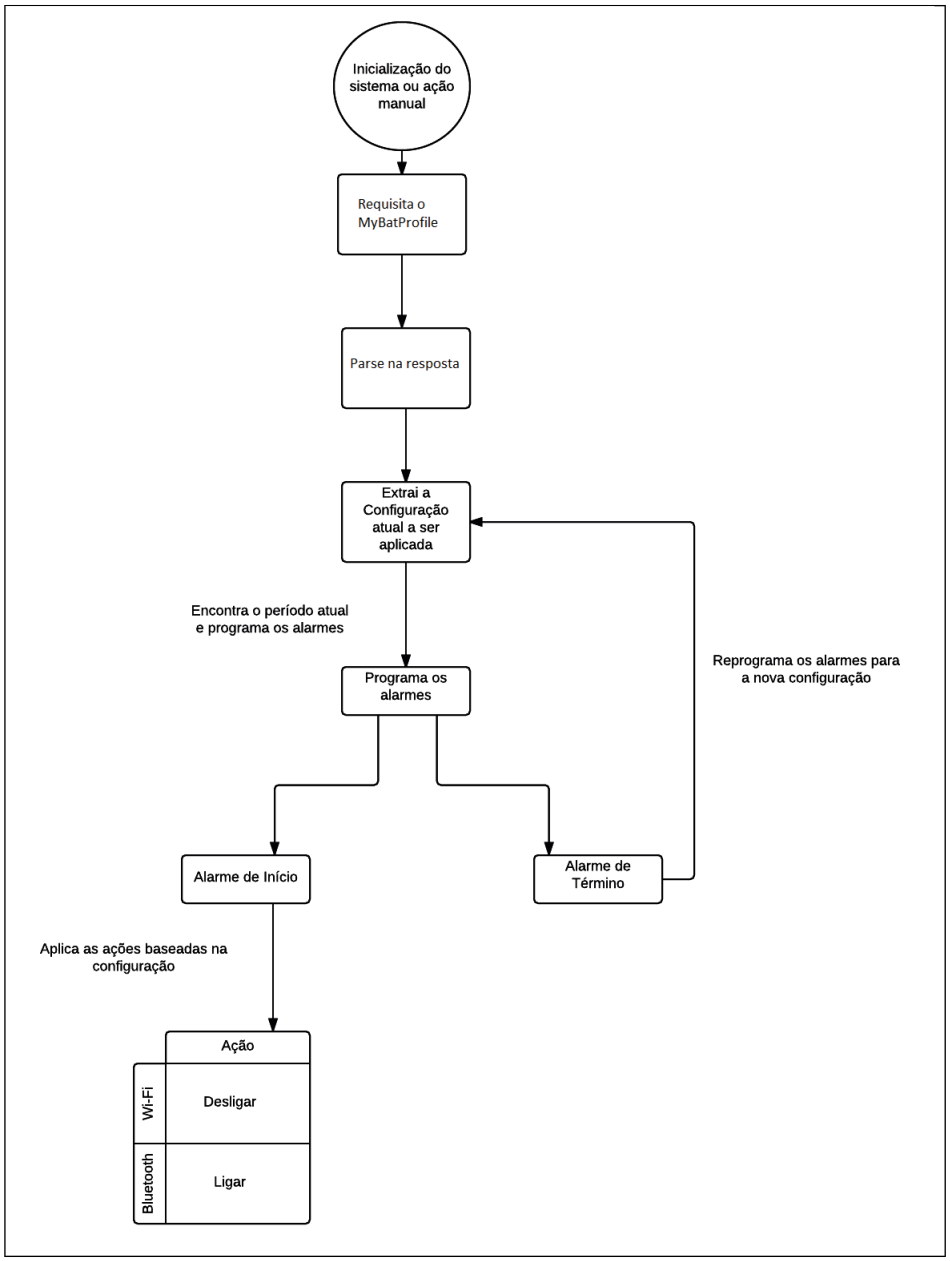


Figura 16 – Visão geral do fluxograma do aplicativo MyBatSaver

6. *mobile*: Através da classe ConnectivityManager é possível habilitar ou desabilitar as redes móveis pelo método *setMobileDataEnabled()*;

7. *autosync*: Através da classe ContentResolver é possível habilitar ou desabilitar o AutoSync pelo método *setMasterSyncAutomatically()*;

   Apesar de o MyBatProfile incluir em seus atributos um indicador do AutoSync, esse valor não foi considerado para o controle das configurações pelo MyBatSaver, uma vez que o MyBatLogger não coletou informações relativas ao mesmo, sendo retornado um valor padrão pelo MyBatServer.

   Para o usuário do aplicativo, existe uma representação visual de cada configuração vigente. A Figura 17 mostra as telas do aplicativo MyBatSaver que indicam as

configurações vigentes.

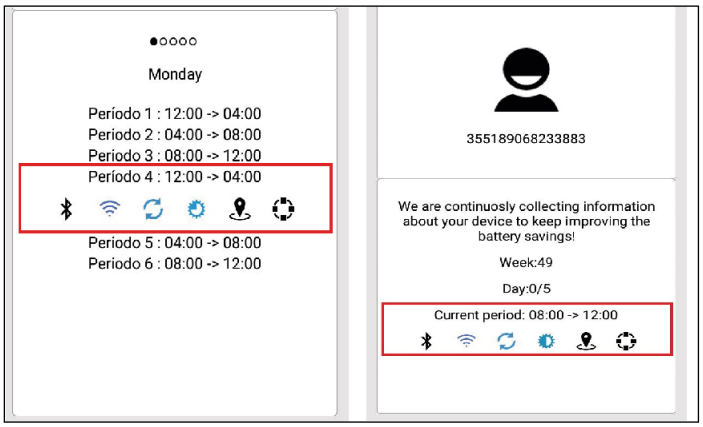


Figura 17 – MyBatSaver: visão da configuração atual

Notou-se durante a implementação, através de testes funcionais do aplicativo MyBatSaver, que existiam operações que não eram permitidas em algumas versões de Android. Por exemplo, a operação de desligar/ligar o GPS através das APIs não era permitida em algumas versões de Android, a saber, as posteriores ao Android Lollipop[32]. Isso ocorre pois existe um bloqueio no nível do sistema operacional que não pode ser transposto pelo aplicativo. Nesses casos uma notificação é exibida ao usuário de forma que ele possa tomar a ação necessária manualmente, como mostra a Figura 18.

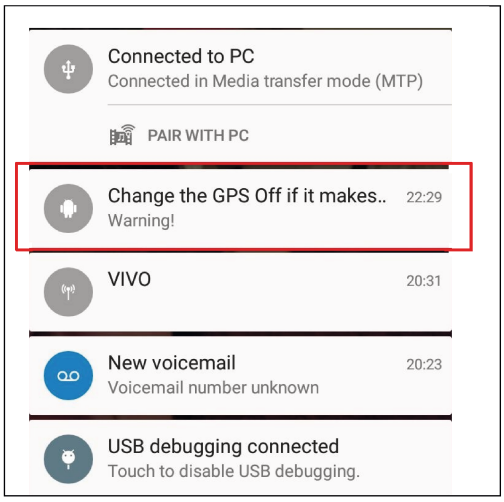


Figura 18 – MyBatSaver: visão das notificações

3.5 Comparação com aplicativos similares

Com o propósito de comparar o mecanismo proposto com os aplicativos escolhidos na pesquisa executada nesse capítulo, as Tabelas 10 e 11 foram replicadas, incluindo

[32] https://www.android.com/intl/pt-BR_br/versions/lollipop-5-0/

o mecanismo proposto, MyBatRecommender. Os mesmos índices que representam as características para a Tabela 10 e os componentes para a Tabela 11 foram mantidos.

Pela análise da Tabela 15 pode-se notar que o mecanismo é focado no controle dos estados dos componentes através geração de perfil automatizada, uma vez que não apresenta outras características como previsão de duração da energia. Com relação as funcionalidades de geração de perfil, o MyBatRecommender apenas não permite que os usuários criem manualmente os perfis, uma vez que a característica do mecanismo é gerá-los automaticamente a partir da análise dos dados de utilização dos *smartphone*.

Tabela 15 – Análise das características presentes em cada aplicativo incluindo o MyBatRecommender

	a	b	c	d	e	f	g	h	i	j
Battery Doctor	x	x	x	x	x	x	x	x	x	
DU Battery Saver	x	x		x	x	x	x	x	x	
Avast Battery Saver	x	x				x	x		x	
JuiceDefender							x	x		
GO Battery Saver	x			x	x	x	x	x	x	
MPower	x	x	x				x			x
Battery Optimizer & Cleaner	x	x				x				
MyBatRecommender							x		x	x

Pode-se observar pela análise da Tabela 16 que o mecanismo MyBatRecommender abrange todos os componentes levantados na pesquisa, excetuando-se apenas o AutoSync.

Tabela 16 – Análise dos componentes gerenciados por cada aplicativo incluindo o MyBatRecommender

	a	b	c	d	e	f	g
Battery Doctor	x	x	x	x		x	
DU Battery Saver	x	x	x	x		x	
Avast Battery Saver	x	x	x	x		x	
JuiceDefender	x	x			x		x
GO Battery Saver	x	x	x	x		x	
MPower	x	x	x		x		x
Battery Optimizer & Cleaner							
MyBatRecommender	x	x	x		x	x	x

3.6 Considerações Finais

Foi apresentado nesse capítulo o mecanismo denominado MyBatRecommender, que tem como objetivo gerenciar e otimizar o uso da energia com o objetivo de

economizá-la nos *smartphones*. Essa economia é alcançada através do gerenciamento dos estados dos componentes do *smartphone*, baseado em um perfil de utilização do usuário, evitando que os mesmos permaneçam em estados consumindo energia quando não necessário. Além da apresentação geral do mecanismo, também foram apresentadas as partes que o compõe, que são: o MyBatLogger, responsável por coletar as informações relativas aos componentes do *smartphone* e enviá-las a um servidor remoto, o MyBatServer, que além de armazenar essas informações é o responsável por rodar algoritmos a fim de extrair um perfil de uso para o usuário, denominado MyBatProfile, e por fim o MyBatSaver, que aplica ações para gerenciar os estados dos componentes baseado no perfil de uso do usuário. O capítulo também detalhou a implementação do mecanismo proposto para a plataforma Android.

4
Validação da proposta

4.1 Considerações Iniciais

Novos métodos, modelos e ferramentas devem ser sempre apresentados em conjunto com a sua experimentação e validação. Considerando isso, existem diferentes métodos para conduzir os experimentos na área de Engenharia de Software, com o propósito de medir e analisar o efeito da adoção do modelo proposto (TRAVASSOS; GUROV; AMARAL, 2002).

Considerando a importância da validação para novos métodos propostos, este capítulo tem como objetivo apresentar e validar as hipóteses criadas sobre o mecanismo MyBatRecommender. Para isso foram realizadas validações sobre diversas perspectivas, que estão de acordo com os objetivos propostos, que são: verificação do *overhead* da implementação do mecanismo proposto, dois experimentos com cenário controlado e um experimento com usuários reais, realizados para análise da eficiência do mecanismo proposto.

Para todos os estudos de casos que serão apresentados na sequência, dois *smartphones* com mesma configuração foram utilizados. A configuração básica, bem como o site com as especificações técnicas mais detalhadas desses *smartphones* estão apresentadas na Tabela 17.

Tabela 17 – Detalhamento dos *smartphones* utilizados nas validações

Nome comercial	Sony Xperia Z2
Número do Modelo	D6503
Número da *Build*	23.1.A.0.690
Versão Android	4.4 (KitKat)
Especificações Técnicas	http://www.sonymobile.com/br/products/phones/xperia-z2/specifications/

Considera-se que a configuração padrão desses aparelhos é a que se obtém após o aparelho ter sido ligado a sua primeira vez. As configurações variáveis, como novos aplicativos instalados, caso existam, dependem do experimento e serão apresentadas nas seções relativas a cada experimento.

Para organização do restante das seções desse capítulo os dois *smartphones* utilizados serão chamados de DUT 1 e DUT 2, onde DUT é a sigla para *Device Under Test*, indicando que é o aparelho que está sendo usado para testes.

4.2 Experimento 1: Verificação do *overhead* do aplicativo

O mecanismo proposto, quando implementado para o sistema operacional Android, exige processamento computacional o que, consequentemente, incorre em um consumo de energia. Esse consumo oriundo da execução do mecanismo proposto é um dos *overheads* do sistema, que também se aplica a outras variáveis como tempo, por exemplo. Para a presente validação, como a variável de interesse é a energia consumida, o termo *overhead* será utilizado apenas para indicar a sobrecarga de energia consumida.

Mesmo que esse consumo possa ser pequeno, uma vez que o mecanismo foi projetado para ter o seu processamento mais pesado em um servidor remoto, é necessário estimá-lo para que ele possa ser considerado nas análises das validações de eficiência propostas. Além disso, o objetivo desse experimento também contempla analisar se as formas de executar a medição do *overhead*, com ou sem ferramentas externas, possuem um resultado equivalente.

4.2.1 Planejamento

O presente experimento elaborou duas formas para calcular o *overhead*, que utilizam ou não uma ferramenta externa própria para medição de corrente elétrica dos *smartphones*.

4.2.2 Execução

A execução do presente experimento se deu pelo planejamento e execução de outros dois cálculos do *overhead*, sendo um através de ferramentas externas, apresentado em 4.2.2.1 e o outro através de medições de software, apresentado em 4.2.2.2.

4.2.2.1 Cálculo do *overhead* com ferramentas externas

Uma das formas de se calcular o consumo de um aplicativo é por meio de ferramentas externas. Este cálculo utilizou a ferramenta *Keithley Source Measure Unit/Charge Simulator*[1], cedida pelo Venturus Centro de Inovação Tecnológica. Esse aparelho é capaz de medir a corrente elétrica do dispositivo durante um período de tempo e através de softwares auxiliares armazená-la em arquivos para posterior análise no computador.

[1] http://www.tek.com/sites/tek.com/files/media/media/resources/2308.pdf

4.2.2.1.1 Pré-condições

Para o presente cálculo, elaborou-se as seguintes configurações, utilizadas durante a condução do mesmo:

- Configuração 1 (C1): o DUT 1, **sem** o MyBatRecommender instalado, ligado junto ao equipamento que coleta os dados, durante uma hora, em modo avião, com a tela apagada e apenas com o Wi-Fi ligado, para que não houvesse influência de outros fatores além do necessário para o funcionamento do aplicativo. O Wi-Fi foi mantido ligado pois o mecanismo necessita de conexão com a Internet via Wi-Fi para poder enviar os dados coletados ao servidor remoto, e essa tarefa exige processamento computacional, por isso tinha que ser considerada;
- Configuração 2 (C2): o DUT 1, **com** o MyBatRecommender instalado, ligado junto ao equipamento que coleta os dados, durante uma hora, em modo avião, com a tela apagada e apenas com o Wi-Fi ligado, para que não houvesse influência de outros fatores além do necessário para o funcionamento do aplicativo. O Wi-Fi foi mantido ligado pois o mecanismo necessita de conexão com a Internet via Wi-Fi para poder enviar os dados coletados ao servidor remoto, e essa tarefa exige processamento computacional, por isso tinha que ser considerada;

4.2.2.1.2 Condução

O cálculo se deu pela execução das configurações C1 e C2 em sequência. Inicialmente conectou-se o DUT 1, com a configuração C1, na máquina de coleta de dados durante uma hora. Após essa hora, salvou-se os dados coletados e reiniciou-se a máquina para que os dados da primeira coleta não afetassem a segunda. Logo após conectou-se o DUT 1, com a configuração C2, na máquina de coleta de dados por mais uma hora. Após essa hora, salvou-se os dados coletados e finalizou-se a execução da coleta.

4.2.2.1.3 Análise

Uma amostra dos dados coletados pela *Keithley Source Measure Unit* é apresentada na Tabela 18, ilustrando o formato armazenado pela máquina. A totalidade dos dados não será exibida no presente trabalho devido a quantidade de amostras.

Além disso, a Tabela 19 mostra a legenda de cada coluna da Tabela 18.

Considerou-se para a análise dos dados as amostras coletadas desde o início da execução de cada uma das configurações até uma hora após o início de cada uma delas, excluindo-se as amostras que excedessem esse limite. A ferramenta *Keithley Source Measure Unit* tem uma taxa de amostragem de aproximadamente dois

Tabela 18 – Exemplo de dados coletados pela *Keithley Source Measure Unit*

Time (ms)	**Time**	**Offset (ms)**	**Current (mA)**
1452099728914	2016-00-06 15:02:08	531967	7
1452099730789	2016-00-06 15:02:10	533842	7
1452099732664	2016-00-06 15:02:12	535717	7
1452099734540	2016-00-06 15:02:14	537593	7
1452099736415	2016-00-06 15:02:16	539468	7

Tabela 19 – Legenda da Tabela 18

Legenda	**Significado**
Time (ms)	Representa o horário em que o dado foi coletado, em milissegundos, baseado na *Unix Timestamp*
Time	Representa o horário em que o dado foi coletado no formato ano-mês-dia hora:minuto:segundo
Offset (ms)	Representa o acumulado de tempo, em milissegundos, desde o primeiro dado coletado
Current (mA)	Representa a corrente elétrica do dispositivo no momento da coleta do dado, em miliamperes

segundos, o que permitiu que as amostras pudessem ser delimitadas dentro do limite estipulado com uma boa precisão.

Como ambas as coletas de dados tiveram a mesma duração, com a mesma taxa de amostragem e consequentemente o mesmo número de amostras coletadas, os dados coletados das duas configurações C1 e C2 foram colocados em um mesmo gráfico para comparação. A Figura 19 mostra essa comparação.

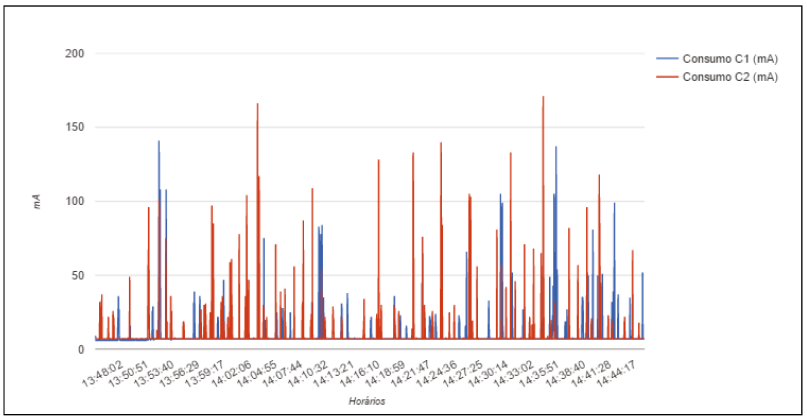


Figura 19 – Análise do consumo de energia de ambas configurações

Pela análise da Figura 19 percebe-se que a corrente elétrica no dispositivo com a configuração C2, com o mecanismo proposto, se apresenta de forma mais intensa e com valores mais altos, indicando um maior consumo de energia durante o período da execução do experimento. Esse comportamento é mensurado e apresentado na sequência do capítulo.

Após a delimitação dos dados, foi calculada a média simples (MED) da corrente elétrica amostrada do dispositivo, de ambas as configurações, para comparação. Para esse cálculo, somou-se todos os valores da corrente elétrica amostrados de cada configuração e dividiu-se pelo número de amostras de cada configuração:

$$MED = \frac{\sum Current(corrente\ elétrica)}{Número\ de\ amostras} \tag{4.1}$$

Com esses valores calculados, e considerando a amperagem do DUT 1, de 3200 mAh, miliampere-hora, que identifica a transferência de carga elétrica por meio de uma corrente estável de um ampere ao longo de uma hora, calculou-se a representatividade da média da corrente elétrica (REP) de cada uma das configurações perante a amperagem do DUT 1:

$$REP = \frac{MED\ *\ 100}{3200} \tag{4.2}$$

Com isso obteve-se o valor relativo ao percentual do consumo de energia por hora das duas configurações C1 e C2, e, consequentemente a relação entre esses consumos, que indica que a configuração C2 consumiu 0,0285% de energia por hora a mais do que a C1. Esses dados estão apresentados na Tabela 20.

Tabela 20 – Resultados das análises dos dados coletados

C	**Horário Início**	**Horário Fim**	**MED (mA)**	**REP**
1	13:45:15	14:45:15	9.212389381	0.2878%
2	15:02:08	16:02:09	10.12324493	0.3163%
			Diferença	0,0285%

Considerando que durante a execução do estudo a diferença apresentada entre as configurações foi a presença ou ausência do MyBatRecommender, conclui-se que existem indícios para se afirmar que o mecanismo MyBatRecommender possui um consumo de 0,0285% por hora. Este valor será utilizado nos próximos experimentos do presente capítulo.

4.2.2.2 Cálculo do *overhead* sem ferramentas externas

Além do método de cálculo do *overhead* com ferramentas externas, pode-se fazer o mesmo cálculo através de medições manuais do próprio *smartphone* ou de softwares instalados no mesmo. O presente experimento utiliza-se de medições do próprio sistema operacional Android para estimar o *overhead* do MyBatRecommender.

4.2.2.2.1 Pré-condições

Para o presente cálculo, considerou-se que as seguintes condições já existiam para a condução do mesmo:

- Utilizou-se o cenário descrito no Anexo D, elaborado para os estudos de caso com cenário controlado;
- O MyBatProfile, gerado pela execução do MyBatRecommender após uma semana de coleta de dados, já existia;
- O DUT 1 rodou o sistema MyBatRecommender e o DUT 2 rodou com a configuração padrão do *smartphone*, sem nenhum aplicativo instalado no mesmo;
- A forma de se registrar o nível da energia de ambos os DUTs durante a execução foi através de capturas de tela da tela de análise de consumo de energia do *smartphone*, como mostra a Figura 20;
- A duração da coleta foi de cinco dias e não houve período de recarga do *smartphone*;

Por questões de espaço não serão mostrados todas as capturas da tela obtidas, e sim a Tabela 21 com os dados coletados.

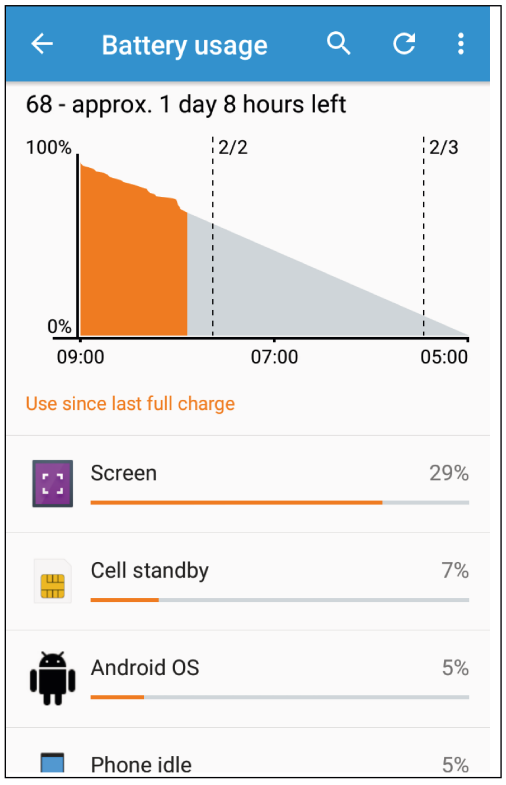


Figura 20 – Representação das capturas da tela para verificação do nível de energia

4.2.2.2.2 Condução

A execução desse experimento se deu pelo acompanhamento do cenário, aplicando as configurações definidas pelo mesmo para cada período do dia. Além disso, toda vez que o DUT 1 recebia uma recomendação e alterava o estado de algum componente do *smartphone*, a mesma alteração era replicada manualmente no DUT 2. A intenção dessa execução é garantir que ambos DUTs estavam sempre com a mesma configuração para que na análise qualquer diferença no nível de energia pudesse ser atribuída para a diferença entre eles, no caso, apenas o aplicativo instalado do modelo proposto, MyBatRecommender. Também, a cada alteração de configuração, era feito uma captura de tela para salvar as informações necessárias daquele momento.

4.2.2.2.3 Análise

A Tabela 21 apresenta o nível de energia de ambos DUTs durante a execução do presente experimento.

Tabela 21 – Nível de energia de ambos DUTs durante execução do cálculo de *overhead*

Horário	**Energia DUT 1 (%)**	**Energia DUT 2 (%)**	**Diferença (DUT 2 - DUT 1) (%)**
07/09 - 00:00	100	100	0
07/09 - 04:00	97	98	1
07/09 - 08:00	96	96	0
07/09 - 12:00	93	94	1
07/09 - 16:00	90	91	1
07/09 - 20:00	88	89	1
08/09 - 00:00	88	88	0
08/09 - 04:00	85	86	1
08/09 - 12:00	75	83	8
08/09 - 20:00	72	79	7
08/09 - 23:30	71	79	8
09/09 - 06:00	71	77	6
09/09 - 17:00	65	71	6
09/09 - 23:00	64	70	6
10/09 - 21:30	54	63	9
11/09 - 19:00	44	54	10
12/09 - 00:00	43	52	9
		MEDD	4.35

A média das diferenças (MEDD), para o cálculo do *overhead*, foi considerado como sendo a média dos valores do nível de energia do DUT 2 subtraído do valores do nível de energia do DUT 1 para os mesmos horários:

$$MEDD = \frac{\sum (Energia\ DUT\ 2\ -\ Energia\ DUT\ 1)}{Número\ de\ horas} \tag{4.3}$$

Pela análise da média das diferenças concluiu-se que o DUT 2 consumiu, em média, 4,35% a menos de energia do que o DUT 1, que estava rodando com o mecanismo proposto.

Considerando que o cenário e as configurações para ambos os DUTs foram as mesmas durante toda a execução, conclui-se que existem indícios para se afirmar que o mecanismo MyBatRecommender possui em geral um *overhead* de 4,35% para o período apresentado, de 120 horas, o que resulta em um valor de *overhead* de 0,03627% por hora.

4.2.3 Análise

A análise executada possui uma questão de pesquisa e hipóteses onde apenas uma hipótese é aceita, de acordo com as análises realizadas dos dados coletados.

Para a análise do presente experimento, considerou-se a seguinte questão de pesquisa, elaborada antes do experimento, e respondida a partir do experimento: *há diferença entre a medição do overhead entre os cálculos com e sem ferramentas externas?*. Para a formulação das hipóteses, foi considerada a seguinte métrica:

- μ_T: o *overhead* do mecanismo em porcentagem de energia consumida por hora;

A hipótese nula e suas correspondentes alternativas são:

- Hipótese Nula (H_0Exp_0): Não há diferença significativa (maior ou igual a 0,01%) entre a medição de μ_T com e sem ferramentas externas;
- Hipótese Alternativa (H_1Exp_0): A medição de μ_T com ferramentas externas possui um resultado significantemente (maior ou igual a 0,01%) menor do que a medição sem ferramentas externas;
- Hipótese Alternativa (H_2Exp_0): A medição de μ_T sem ferramentas externas possui um resultado significantemente (maior ou igual a 0,01%) menor do que a medição com ferramentas externas;

Comparando-se os valores calculados nas subseções 4.2.2.1 e 4.2.2.2, chega-se a um valor bastante próximo ao apresentado pelo cálculo sem ferramentas externas,

diferindo de uma porcentagem de 0,00777%, aceitando a hipótese nula rejeitando as alternativas.

Dessa forma, conclui-se que existem indícios para afirmar que ambas as formas, com e sem auxílio de ferramentas externas, podem ser utilizadas de forma equivalente para o cálculo do *overhead* do aplicativo.

4.3 Experimento 2: Estudo de caso com cenário controlado

O experimento com cenário controlado teve como objetivo analisar o consumo de energia dos dois DUTs, de forma que o DUT 1 estivesse rodando o mecanismo proposto, MyBatRecommender, e o outro, DUT 2, com a configuração padrão do aparelho. No entanto, para essa validação, utilizou-se o aplicativo MyBatLogger para a coleta das informações relativas ao nível de energia, de forma que o DUT 2 também possuísse, além da configuração padrão, o aplicativo MyBatLogger instalado.

4.3.1 Planejamento

Para o presente experimento foi necessário elaborar um cenário para ser seguido, onde estariam presentes todas as ações do usuário com seu *smarpthone* durante os dias da semana. As ações são descritivas, indicando o que o usuário estaria fazendo em um dado período do dia e para cada ação existe uma configuração atrelada, indicando quais os componentes do *smarpthone* estavam sendo utilizados naquele momento. O cenário elaborado para esta validação está descrito no Anexo D.

Considerando a estrutura do modelo proposto, MyBatRecommender, a qual necessita de uma semana de coleta de dados para aprendizagem e geração do MyBatProfile, elaborou-se o seguinte calendário:

- Semana 1: Ambos DUT 1 e DUT 2 rodaram durante essa semana apenas com o módulo MyBatLogger instalado. A intenção nesse caso era, para o DUT 1, coletar dados para a geração do MyBatProfile, além de registrar o nível da energia do mesmo. Para o DUT 2, a intenção era apenas de registar o nível da energia durante essa semana de uso;
- Semana 2: O DUT 1 passou a rodar o sistema MyBatRecommender completo, uma vez que seus dados referentes a primeira semana já tinham sido coletados e seu MyBatProfile gerado. Ou seja, para o DUT 1, a segunda semana serviu para a aplicação das recomendações de gerenciamento dos estados dos componentes do celular e para registar o nível da energia do mesmo. O DUT 2 continuou apenas com o MyBatLogger instalado, com a intenção de apenas de registar o nível da energia durante essa semana de uso;

Todos os dados logados pelo MyBatLogger ficam armazenados no MyBatServer, de forma que é possível acessá-los posteriormente para fazer as análises.

4.3.2 Execução

A execução desse experimento se deu pelo acompanhamento e execução do cenário proposto em ambos dos DUTs. Para isso, no início de cada semana, ambos DUTs estavam com todos os seus componentes no estado desligado. Após isso, para o DUT 1, a cada início de um novo período, o MyBatRecommender aplicava automaticamente as configurações relativas aquele período. Para o DUT 2 essas configurações eram aplicadas manualmente, replicando o estado necessário para aquele período.

Caso houvesse, para o DUT 2, durante a aplicação de configurações, algum componente que permanecesse em um estado que não fosse mais necessário, o mesmo era mantido nesse estado para poder simular os casos em que a energia é consumida desnecessariamente. Esse mesmo cenário, teoricamente, não deveria acontecer, ou acontecer em proporções menores para o DUT 1 pois ao rodar o MyBatRecommender esses estados são automaticamente gerenciados.

4.3.3 Análise

A análise executada possui uma questão de pesquisa e hipóteses onde apenas uma hipótese é aceita, de acordo com as análises realizadas dos dados coletados.

Para a análise do presente experimento, considerou-se a seguinte questão de pesquisa, elaborada antes do experimento, e respondida a partir do experimento: *há diferença entre o consumo de energia entre os DUTs?*. Foram elaboradas três hipóteses para o experimento. Para a formulação das hipóteses, foram consideradas as seguintes métricas:

- τ: tempo total, em horas, em que o *smartphone* ficou consumindo energia, não considerando-se o tempo que o mesmo estava carregando
- ϕ: porcentagem da energia total consumida;
- μ_{T}: a média de porcentagem de energia consumida por hora;
- ε: representa a eficiência de um mecanismo em termos de economia de energia, em porcentagem de nível de energia economizada;

A hipótese nula e suas correspondentes alternativas são:

- Hipótese Nula ($\mathrm{H_0Exp_2}$): Não há diferença significativa (maior ou igual a 5%) de consumo de energia dos dois DUTs, com respeito à eficiência (ε) do mecanismo;

- Hipótese Alternativa (H_1Exp_2): O DUT 1, com o sistema MyBatRecommender instalado, possui um consumo de energia significantemente (maior ou igual a 5%) menor do que o DUT 2;
- Hipótese Alternativa (H_2Exp_2): O DUT 2, sem o sistema proposto instalado, possui um consumo de energia significantemente (maior ou igual a 5%) menor do que o DUT 1;

A variável dependente selecionada para o estudo foi *eficiência do mecanismo*. As variáveis independentes foram os aparelhos utilizados e a sua configuração, o cenário proposto e os aplicativos de economia de energia. Uma vez que o objetivo era investigar as consequências do do MyBatRecommender na economia de energia, a variável denominada "aplicativos de economia de energia" foi estabelecida como fator cujo efeito deve ser observado na variável dependente. As demais variáveis independentes foram mantidas constantes.

Considerando as métricas definidas, pode-se dizer que o DUT 1 é mais eficiente em termos de economia de energia (ε) do que o DUT 2 caso $\mu_{T1} < \mu_{T2}$, para todos os períodos considerados.

A Tabela 22 apresenta todos os campos e seus significados que são utilizados na Tabela 23.

Tabela 22 – Legenda da Tabela 23

Legenda	**Significado**
Início	Representa o horário de início daquele período de descarga de energia
Fim	Representa o horário de fim daquele período de descarga de energia
τ	Representa quantas horas aquele período de descarga de energia tem
ϕ	Representa o percentual de energia consumido durante aquele período
$\mu_{\mathbf{T}}$	É a taxa média de consumo de energia por hora do período
Média	É média dos valores calculados de μ_T de todos os períodos de um DUT

A Tabela 23 apresenta os dados coletados de ambos os DUTs na primeira semana de execução do presente experimento. Cada linha representa um período onde a energia estava sendo descarregada, uma vez que não foram considerados os períodos em que a mesma estava sendo carregada, pois a intenção é calcular a taxa com que a energia é consumida.

Tabela 23 – Dados coletados da Semana 1 de ambos DUTs

		Início	**Fim**	τ	ϕ	μ_T
DUT 1	**Período 1**	18/08 00:00:03	20/08 18:58:37	66	94	1.4242
	Período 2	21/08 06:08:11	22/08 09:17:37	27	26	0.9629
	Período 3	22/08 10:18:07	24/08 16:37:06	54	45	0.8333
					Média	1.0734
DUT 2	**Período 1**	18/08 00:00:20	20/08 18:53:07	66	83	1.2575
	Período 2	21/08 04:31:07	22/08 07:52:29	27	29	1.0740
	Período 3	22/08 10:24:02	24/08 19:09:01	56	33	0.5892
					Média	0,9735

Da mesma forma, a Tabela 24 apresenta os dados coletados de ambos os DUTs na segunda semana de execução do presente experimento. As mesmas considerações e legendas apresentadas anteriormente são válidas.

Tabela 24 – Dados coletados da Semana 2 de ambos DUTs

		Início	**Fim**	τ	ϕ	μ_T
DUT 1	**Período 1**	25/08 20:08:04	28/08 08:14:50	60	35	0.5833
					Média	0.5833
DUT 2	**Período 1**	25/08 17:32:51	28/08 06:50:36	61	55	0.9016
					Média	0.9016

Pela análise da Tabela 23 observa-se que a média de descarga de energia dos períodos de ambos os DUTs é muito próxima para a primeira semana, com uma diferença de aproximadamente 0,1% de energia por hora. Esse resultado é considerado esperado uma vez que ambos os DUTs estavam, para a primeira semana, exatamente com a mesma configuração, de forma que o consumo dos mesmos deveria ser bastante aproximado.

Ao analisar a Tabela 24 vemos que o mesmo cenário não se repete, de forma que o DUT 1, com o mecanismo proposto instalado, tem uma média de descarga de energia relativamente menor do que o DUT 2, com uma diferença de 0,3183% por hora. Esse valor representa a diferença entre os consumos considerando o DUT 2 com a presença do elemento MyBatLogger, uma vez que o mesmo foi utilizado para registrar o nível da energia durante o experimento. Dessa forma, a diferença real de consumo se dá pela subtração do valor calculado do *overhead*, apresentado na subseção 4.2, de 0,3183%, o que resulta em um valor de 0,2898% por hora.

Deseja-se verificar com algum grau de significância se é possível rejeitar a hipótese nula em favor das alternativas com o conjunto de dados obtidos. Para isso foi necessário verificar se as amostras possuíam distribuição normal ou não, para então definir o método a ser aplicado para comparar as amostras. Utilizou-se então o

método Ryan-Joiner[2] através da ferramenta Minitab[3] para as duas amostras de dados coletadas. Calculou-se que ambas as amostras possuem o p-valor < 0.01, concluindo-se que não possuem distribuição normal, como mostra a Figura 21.

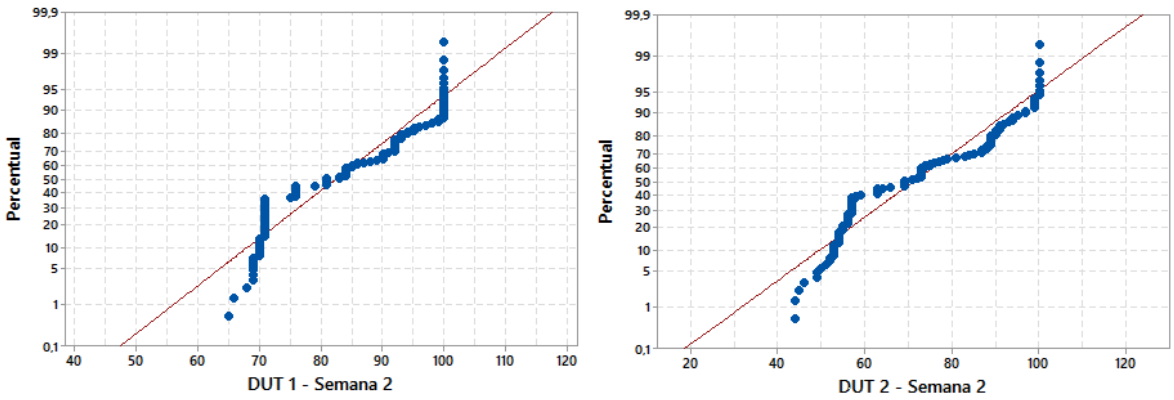


Figura 21 – Teste de normalidade das amostras coletadas de ambos os DUTs na Semana 2

Dessa forma, utilizou-se o método Teste-U de Mann-Whitney através da ferramenta Minitab [4] considerando um grau de significância α=0,05. Através dos cálculos realizados pela ferramenta, chegou-se que o p-valor=0,0 (ajustado pela ferramenta para empates) é menor que o grau de significância estabelecido (0.05), rejeitando a hipótese nula de forma que possa se confirmar com 95% de confiança que a diferença entre as amostras é causada pela presença do MyBatRecommender.

Assim, considerando-se o valor do consumo do DUT 2 durante a segunda semana do experimento como o consumo esperado de um *smartphone*, uma vez que ele não possuía nenhum aplicativo de economia de energia instalado, e comparando-se esse valor com a diferença de consumo entre os DUTs nessa mesma semana, conclui-se que houve uma economia de energia de aproximadamente 32%.

Assim, observando os resultados apresentados, verifica-se que, como $\mu_{T1} < \mu_{T2}$, resultando em uma eficiência (ε) de 32% em termos de economia de energia, aceita-se a hipótese H_1Exp_2 rejeitando-se as outras hipóteses. Dessa forma conclui-se que existem indícios para se afirmar que o DUT 1 utilizando o mecanismo MyBatRecommender possui, em geral, um consumo menor de energia do que o DUT 2, não utilizando nenhum mecanismo.

[2] http://www.statsref.com/HTML/index.html?ryan-joiner.html
[3] http://www.minitab.com/pt-br/
[4] https://statistics.laerd.com/minitab-tutorials/mann-whitney-u-test-using-minitab.php

4.4 Experimento 3: Estudo de caso com cenário controlado: comparação entre MyBatRecommender e BatteryDoctor

O segundo estudo com cenário controlado teve como objetivo comparar a eficiência do modelo proposto, MyBatRecommender, com o aplicativo que segundo as análises da subseção 3.3.2 mais se aproxima das características do modelo proposto, o BatteryDoctor.

4.4.1 Planejamento

Considerando-se o cenário descrito no Anexo D, elaborado para os estudos de caso com cenário controlado, e considerando-se que o MyBatProfile, gerado pela execução do MyBatRecommender após uma semana de coleta de dados, já existe, elaborou-se o seguinte calendário: o DUT 1 rodou o sistema MyBatRecommender completo, uma vez que seus dados referentes a primeira semana já tinham sido coletados e o MyBatProfile gerado. O DUT 2 rodou com o aplicativo BatteryDoctor instalado, utilizando o perfil de gerenciamento de estados dos componentes chamado *General*, como mostra a Figura 22.

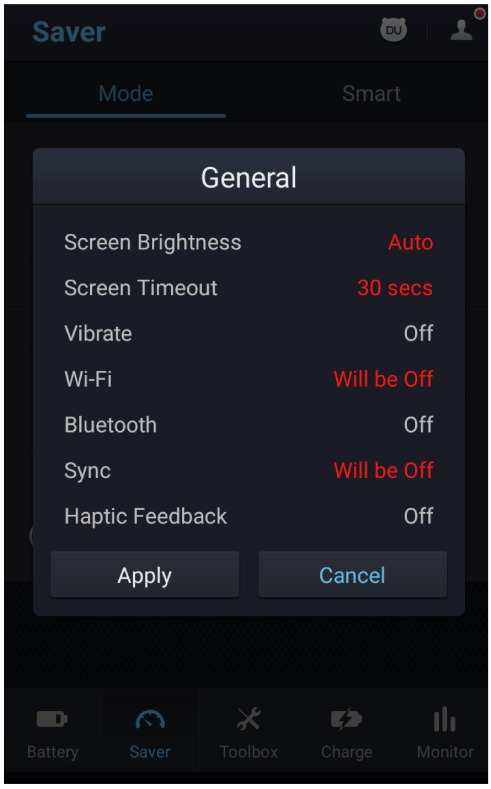


Figura 22 – Representação da tela do perfil General do aplicativo Battery Doctor

4.4.2 Execução

A execução desse experimento se deu pelo acompanhamento e execução do cenário proposto em ambos os DUTs, da mesma forma que na subseção 4.3.2. A única alteração é que a forma de coleta dos dados difere. Como o DUT 2 não tem o MyBatLogger instalado, foi necessário coletar os dados manualmente via capturas da tela, como explicado na subseção 4.2.2.2.2. Para isso, a cada aproximadamente quatro horas, período de troca de configuração do MyBatRecommender, foi feito uma captura da tela de análise de consumo de energia do *smartphone*, como mostra a Figura 20. A partir dessa captura da tela é possível identificar o nível de energia do *smartphone*, necessário para a análise.

4.4.3 Análise

A análise executada possui uma questão de pesquisa e hipóteses onde apenas uma hipótese é aceita, de acordo com as análises dos dados coletados.

Para a análise do presente experimento, considerou-se a seguinte questão de pesquisa, elaborada antes do experimento, e respondida a partir do experimento: *há diferença entre o consumo de energia entre os DUTs?*. Foram elaboradas três hipóteses para o experimento. Para a formulação das hipóteses, foram consideradas as seguintes métricas:

- ϕ: nível da energia de um DUT em um determinado horário;
- δ: diferença entre o nível de energia entre o DUT 2 e DUT 1 em um determinado horário;
- μ_B: média dos valores do nível de energia do DUT 2 subtraído dos valores do nível de energia do DUT 1 para os mesmos horários, de todos os horários coletados;
- ε: representa a eficiência de um mecanismo em termos de economia de energia, em porcentagem de nível de energia economizada;

A hipótese nula e suas correspondentes alternativas são:

- Hipótese Nula (H_0Exp_3): Não há diferença significativa (maior ou igual a 5%) de consumo de energia dos dois DUTs, com respeito à eficiência (ε) do mecanismo;
- Hipótese Alternativa (H_1Exp_3): O DUT 1, com o sistema MyBatRecommender instalado, possui um consumo de energia significantemente (maior ou igual a 5%) menor do que o DUT 2, com o BatteryDoctor instalado;
- Hipótese Alternativa (H_2Exp_3): O DUT 2, com o BatteryDoctor instalado, possui um consumo de energia significantemente (maior ou igual a 5%) menor do que o DUT 1, com o MyBatRecommender instalado;

A variável dependente selecionada para o estudo foi *eficiência do mecanismo*. As variáveis independentes foram os aparelhos utilizados e sua configuração, o cenário proposto e os aplicativos de economia de energia. Uma vez que o objetivo era investigar as consequências do uso dos aplicativos na economia de energia, a variável denominada "aplicativos de economia de energia" foi estabelecida como fator cujo efeito deve ser observado na variável dependente. As demais variáveis independentes foram mantidas constantes.

Considerando as métricas definidas, pode-se dizer que o DUT 1 é mais eficiente em termos de economia de energia (ε) do que o DUT 2 caso $\delta < 0$.

A Tabela 25 apresenta todos os campos e seus significados que são utilizados na Tabela 26.

Tabela 25 – Legenda da Tabela 26

Legenda	**Significado**
Horário	Representa o horário de coleta daquele dado
ϕ_1	Representa o nível de energia do DUT 1 naquele horário
ϕ_2	Representa o nível de energia do DUT 2 naquele horário
δ	Representa a diferença do nível de energia entre o DUT 2 e DUT 1
Média das Diferenças	Representa a média dos valores da diferença do nível de energia dos DUTs de todos os horários coletados

A Tabela 26, por sua vez, mostra o nível de energia de ambos DUTs durante a execução do presente experimento, além dos horários de coleta dos dados.

Tabela 26 – Nível de energia de ambos DUTs durante execução do estudo de caso comparativo

Horário	ϕ_1	ϕ_2	δ**(DUT 2 - DUT 1)**
29/09 - 00:47	100	100	0
30/09 - 07:00	97	99	2
30/09 - 11:00	95	97	2
30/09 - 12:30	94	96	2
30/09 - 14:20	94	96	2
30/09 - 17:20	93	95	2
01/10 - 08:50	85	88	3
01/10 - 14:20	84	87	3
01/10- 17:00	80	85	5
01/10 - 20:20	80	85	5
01/10 - 21:30	78	84	6

02/10 - 07:20	76	82	6
02/10 - 09:40	74	81	7
02/10 - 11:00	73	79	6
02/10 - 16:20	71	77	6
02/10 - 22:00	69	75	6
03/10 - 00:20	68	74	6
03/10 - 17:00	68	61	7
		Média das Diferenças	4.22

Pela análise da média das diferenças, nota-se que o DUT 2, com o BatteryDoctor instalado teve um consumo de energia em média 4.22% menor que o DUT 1, com o MyBatRecommender instalado, mostrando ser mais eficiente. Apesar dessa diferença, verifica-se que ela não foi suficientemente grande para satisfazer alguma das hipóteses alternativas, aceitando a hipótese nula, H_0Exp_3 e rejeitando as hipóteses alternativas. Dessa forma conclui-se que existem indícios para se afirmar que o DUT 1 e o DUT 2 possuem aproximadamente o mesmo consumo de energia, de forma que os mecanismos propostos são equivalentes em termos de consumo de energia.

Apesar dessa diferença, o MyBatRecommender possui algumas características que o BatteryDoctor não possui, como a automatização da criação e ativação do perfil, fazendo com que o usuário não necessite tomar nenhuma ação além de instalar o aplicativo, fatores que podem ser considerados como contrapartida ao analisar o seu *overhead*.

4.5 Experimento 4: Estudo de caso: usuários reais

O experimento com usuários reais teve como objetivo analisar o comportamento do mecanismo proposto em cenários não controlados, como o dia a dia dos usuários. Para tal, cinco voluntários foram escolhidos para que utilizassem o mecanismo proposto durante um período de tempo. O primeiro grupo foi composto de alunos da Universidade Federal de São Carlos - *campus* Sorocaba e o segundo grupo foi formado por colaboradores do instituto Venturus Centro de Inovação Tecnológica.

4.5.1 Planejamento

Como não havia um cenário estabelecido, elaborou-se o seguinte calendário:

- Semana 1: a primeira semana foi utilizada para a coleta de dados. Assim os voluntários receberam o aplicativo MyBatLogger para que pudessem instalar em seus *smartphones*. Para melhor organização do resto do capítulo essa semana será denominada S1;
- Semana 2: após sete dias do início da primeira semana, os voluntários receberam o aplicativo correspondente ao módulo MyBatSaver para que pudessem instalar em seus *smartphones* e dessa forma ficar com a solução completa em execução. A segunda semana foi utilizada para o gerenciamento dos estados dos componentes baseado no perfil criado para cara voluntário através dos dados coletados na Semana 1. Para melhor organização do resto do capítulo essa semana será denominada S2;

Os voluntários receberam as instruções descritas acima através de *e-mail* enviado contendo um anexo com conteúdo do Anexo C. Além disso, dias antes do início de cada semana os voluntários recebiam, também por *e-mail*, o arquivo com extensão *apk*, que é o compilado do aplicativo pronto para ser instalado no *smartphone*, referente aquela semana. Como o aplicativo, para a sua funcionalidade principal, não requer interação com o usuário através de interfaces gráficas, não foi necessário treiná-los para tal.

Além disso, os voluntários receberam um Termo de Consentimento Livre e Esclarecido como mostra o Anexo E.

4.5.2 Execução

A execução desse experimento se deu pela aplicação do mecanismo proposto, seguindo o calendário estipulado. Os mesmos deveriam utilizar seus *smartphones* como sempre usam, sem nenhuma alteração. Como em ambas as semanas os voluntários tinham sempre, no mínimo, o MyBatLogger instalado, não foi necessário nenhuma outra forma de coletar os dados relativos ao nível de energia para análise, uma vez que esse módulo já faz essa coleta e os envia para o servidor remoto.

4.5.3 Análise

A análise executada possui uma questão de pesquisa e hipóteses onde apenas uma hipótese é aceita, de acordo com as análises realizadas dos dados coletados.

Para a análise do presente experimento, considerou-se a seguinte questão de pesquisa, elaborada antes do experimento, e respondida a partir do experimento: *há diferença entre o consumo de energia entre as semanas de uso, com e sem o mecanismo de recomendação proposta?*. Foram elaboradas três hipóteses para o experimento. Para a formulação das hipóteses, foram consideradas as seguintes métricas:

- τ: tempo total, em horas, em que o *smartphone* ficou consumindo energia, não considerando-se o tempo que o mesmo estava carregando;
- ϕ: porcentagem da energia total consumida;
- μ_{T}: a média de porcentagem de energia consumida por hora;
- ε: representa a eficiência de um mecanismo em termos de economia de energia, em porcentagem de nível de energia economizada;

A hipótese nula e suas correspondentes alternativas são:

- Hipótese Nula ($\mathrm{H_0Exp_4}$): Não há diferença significativa (maior ou igual a 5%) de consumo de energia entre a S1 e S2, com respeito à eficiência (ε) do mecanismo;

- Hipótese Alternativa (H_1Exp_4): A S1, sem o mecanismo proposto instalado, possui um consumo de energia significantemente (maior ou igual a 5%) menor que a S2, com o mecanismo proposto instalado;
- Hipótese Alternativa (H_2Exp_4): A S2, com o mecanismo proposto instalado, possui um consumo de energia significantemente (maior ou igual a 5%) menor que a S1, sem o mecanismo proposto instalado;

Considerando as métricas definidas, pode-ser dizer que a S1 foi mais eficiente (ε) do que a S2 caso $\mu_{T1} < \mu_{T2}$.

As análises dos dados foram feitas primeiramente considerando os resultados coletados para cada voluntário de forma individual. Para isso, foram comparados os resultados para verificar qual das semanas foi a mais eficiente em termos de economia de energia. Depois, uma média com o resultado de todos os usuários foi feita para analisar o desempenho do mecanismo proposto de maneira geral.

Os dados que serão apresentados nas tabelas a seguir são dados compilados dos dados brutos coletados, representando cada período de descarga de energia, uma vez que os dados brutos são em grande quantidade. Para a compilação e análise dos dados, considerou-se apenas os períodos durante a execução em que o *smartphone* estava sendo descarregado, ou seja, sua energia estava sendo continuamente consumida, desconsiderando os períodos em que o mesmo estava sendo carregado por alguma fonte de energia.

As legendas apresentadas pela Tabela 27 são válidas para as Tabelas de 28 até 37 apresentadas a seguir.

Tabela 27 – Legenda das Tabelas de 28 até 37

Legenda	**Significado**
τ	Representa quantas horas aquele período de descarga de energia tem
BI	Representa o percentual de energia no horário de início do período
BF	Representa o percentual de energia no horário de fim do período
$\mu_{\mathbf{T}}$	É a taxa média de consumo de energia por hora do período

As Tabelas de 28 até 32 mostram os dados coletados da S1 de uso para todos os voluntários, da mesma forma que as Tabelas de 33 até 37 mostram os dados coletados da S2 de uso de todos os voluntários. Cada linha dessas tabelas representa um período de descarga de energia ocorrido durante a execução do experimento.

Por fim, a Tabela 38 apresenta o resultado agrupado de todos os voluntários nas duas semanas de execução, e pela sua análise vemos que apenas os voluntários 1 e 2 tiveram

Tabela 28 – Voluntário 1

τ	**BI**	**BF**	$\mu_{\mathbf{T}}$
5	72	41	6.2
9	87	40	5.2
10	100	49	5.1
7	50	6	6.28
8	100	17	10.3
1	37	20	17
2	94	78	8
7	100	57	6.14
9	100	51	5.4
3	100	60	13.3
4	63	17	11.5
2	87	41	23
4	100	73	6.75
9	100	1	11
14	100	44	4
		Média	9.26

Tabela 29 – Voluntário 2

τ	**BI**	**BF**	$\mu_{\mathbf{T}}$
10	59	31	2.8
13	100	79	1.61
28	86	34	1.85
13	69	42	2.076
19	59	29	1.57
8	57	27	3.75
1	29	22	7
34	100	29	2.08
		Média	2.84

Tabela 30 – Voluntário 3

τ	**BI**	**BF**	$\mu_{\mathbf{T}}$
6	91	76	2.5
12	100	42	4.83
8	100	3	12.12
7	100	51	7
9	94	5	9.88
4	100	80	5
12	100	17	6.91
1	19	7	12
11	100	14	7.81
3	100	62	12.66
6	100	26	12.33
2	90	49	20.5
3	84	55	9.66
2	100	55	22.5
7	100	19	11.57
		Média	10.48

Tabela 31 – Voluntário 4

τ	**BI**	**BF**	$\mu_{\mathbf{T}}$
0.5	58	51	14
2	100	89	5.5
2	90	72	9
5	100	72	5.6
1	100	96	4
1	100	96	4
5	100	61	7.8
6	100	43	9.5
9	100	41	6.55
6	81	26	9.16
7	100	50	7.14
3	100	69	10.33
		Média	7.71

Tabela 32 – Voluntário 5

τ	**BI**	**BF**	$\mu_{\mathbf{T}}$
17	100	34	3.88
13	100	40	4.61
16	100	3	6.0625
12	100	39	5.08
12	100	79	1.75
15	100	27	4.86
		Média	4.37

uma leve melhora no consumo de energia na S2, enquanto os outros voluntários tiveram um consumo maior na S2. Isso resulta em uma média de 0,61% a mais de consumo de energia por hora na S2. Apesar dessa diferença, verifica-se que ela não foi suficiente para satisfazer nenhuma das hipóteses alternativas, aceitando-se a hipótese nula, H_0Exp_4 e rejeitando-se as alternativas. Dessa forma conclui-se que existem indícios para se afirmar que as semanas S1 e S2 possuem aproximadamente o mesmo consumo de energia, de forma que o mecanismo proposto, MyBatRecommender, não apresentou economia de energia no experimento com usuários reais.

Tabela 33 – Voluntário 1

τ	**BI**	**BF**	$\mu_{\mathbf{T}}$
15	100	13	5.8
2	51	33	9
4	100	31	17.25
1	34	17	17
9	82	25	6.33
8	100	50	6.25
3	54	21	11
2	71	55	8
7	100	34	9.42
8	66	5	7.62
7	100	39	8.71
4	65	17	12
4	39	8	7.75
15	100	20	5.33
1	100	95	5
		Média	9.09

Tabela 34 – Voluntário 2

τ	**BI**	**BF**	$\mu_{\mathbf{T}}$
20	72	31	2.05
2	51	44	3.5
31	67	26	1.32
30	87	7	2.66
41	91	44	1.14
38	100	65	0.92
		Média	1.93

Tabela 35 – Voluntário 3

τ	**BI**	**BF**	$\mu_{\mathbf{T}}$
1	100	94	6
4	100	11	22.25
2	87	68	9.5
2	100	60	20
6	100	42	9.66
8	100	34	8.25
2	74	44	15
13	100	11	6.84
11	100	24	6.90
7	53	22	4.42
15	100	24	5.06
9	100	22	8.66
4	46	20	6.5
14	100	21	5.64
0.5	100	87	26
		Média	10.71

Tabela 36 – Voluntário 4

τ	**BI**	**BF**	$\mu_{\mathbf{T}}$
7	69	60	1.28
1	71	54	17
4	100	55	11.25
3	100	82	6
2	86	54	16
13	99	44	4.23
2	49	31	9
2	64	46	9
2	51	31	10
1	100	92	8
3	97	71	8.66
6	97	50	7.83
		Média	9.02

Tabela 37 – Voluntário 5

τ	**BI**	**BF**	$\mu_{\mathbf{T}}$
16	100	10	5.62
12	100	16	7
11	100	9	8.27
14	100	24	5.42
5	100	26	14.8
1	44	40	4
3	62	25	12.33
4	36	28	2
10	100	5	9.5
3	71	68	1
		Média	6.99

Esse resultado mostra que o mecanismo, quando aplicado ao cenário de usuários reais, pode ter sido influenciado por fatores externos que não são observados quando o mesmo foi experimentado com um cenário controlado. Dentre esses fatores está a forma que o MyBatProfile é gerado, que considera a existência de seis períodos estáticos de igual duração. Para verificar a influência desse último parâmetro na eficiência do mecanismo, fez-se a seguinte análise: considerando os dados de um voluntário, criou-se o MyBatProfile para cada semana do experimento, analisando apenas os dados referentes a cada semana. O resultado dessa execução foram dois

Tabela 38 – Comparação do consumo entre S1 e S2

Voluntário	**S 1**	**S 2**	**Resultado**
1	9.26	9.09	0.17
2	2.84	1.93	0,91
3	10.48	10.71	-0,23
4	7.71	9.02	-1,31
5	4.37	6.99	-2,62
		Média	-0,61

MyBatProfile criados, o primeiro relativo aos dados coletados da S1 e o segundo relativo aos dados coletados da S2. Feito isso, comparou-se a configuração criada de todos os períodos, para um mesmo dia da semana, de ambos MyBatProfile criados. Essa comparação foi realizada através da análise dos valores dessas configurações, como mostra a Tabela 39, que mostra os valores das configurações de cada componente para todos os períodos de um dia do MyBatProfile gerado, de ambas as semanas. Nessa tabela os valores são apresentados da seguinte forma, para a entrada referente ao estado de um componente de um período: **Estado componente na S1 / Estado componente na S2**, sendo que para os componentes Wi-Fi, GPS, *Bluetooth* e Redes móveis o valor do estado era *ON* ou *OFF*, indicando que o componente estava ligado ou desligado, respectivamente. Para o componente *Display* o estado mostra o nível do brilho desse componente naquele momento. Essa análise indica que as configurações possuem uma semelhança em relação aos estados dos componentes de 50%,

Tabela 39 – Comparação do MyBatProfile criado da S1 e S2 para seis períodos

Período	**Wi-Fi**	**GPS**	***Bluetooth***	***Display***	**Redes Móveis**
1	ON / OFF	ON / OFF	OFF / OFF	0 / 0	OFF / OFF
2	ON / OFF	OFF / ON	OFF / OFF	36 / 7	OFF / OFF
3	ON / OFF	ON / ON	OFF / ON	28 / 8	ON / OFF
4	ON / ON	ON / ON	ON / OFF	0 / 0	ON / OFF
5	ON / ON	ON / ON	ON / OFF	0 / 0	ON / OFF
6	ON / ON	ON / ON	ON / OFF	0 / 0	ON / OFF

Essa mesma análise foi realizada alterando-se o número de períodos estáticos de seis para doze. Ou seja, gerou-se o MyBatProfile para ambas as semanas S1 e S2 com doze períodos estáticos cada dia. Feito isso, comparou-se as configurações geradas de todos os períodos de um dia, o mesmo dia da primeira análise, para ambos perfis criados. Essa análise mostrou que a semelhança entre as configurações passou a ser de 64,4% entre os perfis gerados, como mostra a Tabela 40, que usa a mesma formatação dos dados apresentada pela Tabela 39.

Essas análises indicam que um dos parâmetros envolvidos na criação do MyBatProfile, o número de períodos, possui influência na eficiência do mecanismo. Dessa forma,

Tabela 40 – Comparação do MyBatProfile criado da S1 e S2 para doze períodos

Período	**Wi-Fi**	**GPS**	***Bluetooth***	***Display***	**Redes Móveis**
1	ON / OFF	ON / OFF	OFF / OFF	0 / 0	OFF / OFF
2	ON / OFF	OFF / OFF	OFF / OFF	0 / 0	ON / OFF
3	ON / OFF	OFF / ON	OFF / OFF	9 / 4	OFF / OFF
4	ON / ON	OFF / ON	OFF / OFF	54 / 10	ON / ON
5	ON / ON	ON / ON	OFF / OFF	21 / 9	ON / OFF
6	ON / ON	ON / ON	OFF / OFF	49 / 5	ON / ON
7	ON / ON	ON / ON	ON / OFF	0 / 0	ON / OFF
8	ON / ON	ON / ON	ON / OFF	0 / 0	ON / OFF
9	ON / ON	ON / ON	ON / OFF	0 / 0	ON / ON
10	ON / ON	ON / ON	ON / OFF	0 / 0	ON / OFF
11	ON / ON	ON / ON	ON / OFF	0 / 0	ON / OFF
12	ON / ON	ON / ON	ON / OFF	0 / 0	ON / OFF

a configuração atual desse parâmetro, que considera a existência de seis períodos estáticos com mesma duração durante um dia, pode ser considerado como uma das causas para a baixa eficiência do mecanismo quando aplicado ao cenário de usuários reais.

4.6 Ameaças à validade

Segundo Wohlin (2000), durante a validação da proposta, existem quatro principais formas de ameaça à validade que devem ser observadas, a saber: conclusão, interna, construção e externa.

As ameaças validade conclusão e interna dizem respeito a garantia da existência de uma covariação entre as variáveis independentes e dependentes, de forma que essa variação não seja aleatória e que ela corresponda à causalidade do efeito da primeira variável sobre a segunda. Dessa forma, a presente validação, por abranger estudos de casos que consideram a adoção de uma nova metodologia, MyBatRecommender, utilizou-se de métodos estatísticos, a saber Ryan-Joiner, teste para verificação da distribuição normal das amostras similar ao Shapiro-Wilk e Teste U Mann-Whitney. Além disso, durante os experimentos realizados, cenários controlados foram utilizados, e, quando experimentado com usuários reais, foram selecionados participantes que já possuíam uma rotina de utilização de *smartphones* diária, evitando que fosse forçado algum tipo de comportamento. Também garantiu-se que não houvesse nenhum tipo de interferência em relação a coleta dos dados necessários para as análises, uma vez que a mesma foi realizada de forma automática e sem conhecimento dos usuários, através do MyBatLogger e através de capturas da tela quando em cenários controlados, sendo possível generalizar as análises ao conceito do estudo, que diz respeito ao consumo de energia em *smartphones*, referido pelo ameaça de construção.

Por fim, a ameaça validade externa diz respeito a capacidade de generalizar os resultados para um contexto mais amplo. Para isso, como citado anteriormente, usuários que já estavam acostumados com a rotina diária dos *smartphones* foram considerados como participantes dos experimentos, uma vez que são representativos para o contexto de estudo do presente trabalho. Além disso, os experimentos contaram, quando necessário, com o tempo mínimo para que o mecanismo proposto pudesse apresentar o seu funcionamento completo, desde a coleta de dados até a recomendação do perfil, de forma que as inferências obtidas podem ser generalizadas para outros contextos.

4.7 Considerações finais

Este capítulo apresentou a proposição e execução de experimentos com o objetivo de validar o mecanismo proposto por esta dissertação. Esses experimentos compreenderam estudos de casos com cenário controlado e também com usuários reais, voluntários da Universidade Federal de São Carlos - *campus* Sorocaba e do Venturus Centro de Inovação Tecnológica.

Os resultados mostraram que a implementação do mecanismo proposto para o sistema operacional Android alcançou uma eficiência de aproximadamente 30% quando comparado ao mesmo cenário sem nenhum mecanismo de otimização e gerenciamento de energia instalado. Observou-se também, através dos resultados, que o mecanismo proposto pela dissertação foi tão eficiente quanto a outro mecanismo similar, denominado BatteryDoctor. Por fim, o mecanismo, quando experimentado com usuários reais não alcançou uma eficiência significativa em termos de economia de energia.

5
Conclusões e trabalhos futuros

O presente trabalho propôs um mecanismo, chamado MyBatRecommender, para prover uma solução ao problema existente em relação ao consumo da energia em *smartphones* Android. Esse mecanismo tem como principal função gerenciar e otimizar os estados dos componentes do *smartphone* a fim de economizar a energia do mesmo ao evitar que esses componentes permaneçam em estados indesejados e consumindo energia em momentos desnecessários.

Através de análises feitas de soluções semelhantes existentes no mercado, coletadas através do principal veículo de publicação de aplicativos Android, o Google Play, notou-se que a grande parte das propostas utilizam um método que não é automatizado, uma vez que exigem interação com os usuários para funcionar. Notou-se também que as soluções não são personalizáveis, uma vez que não coletam nem consideram nenhum tipo de informação do usuário para exercerem a sua funcionalidade, adaptando-se àquele usuário. Considerando isso, o mecanismo MyBatRecommender foi proposto e posteriormente implementado para o sistema operacional Android. O MyBatRecommender trabalha de forma a gerenciar e otimizar o uso de energia em *smartphones* Android de forma dinâmica e automatizada. Ou seja, a forma de gerenciar os estados dos componentes é baseada no histórico de utilização do usuário com o *smartphone*, obtida através da coleta de diversas informações relativas ao mesmo. Além disso o gerenciamento é automatizado uma vez que após instalar o aplicativo em seu *smartphone* o usuário não necessita executar nenhum outro tipo de ação para o que o mesmo funcione.

Para validar a abordagem proposta foram elaborados e executados experimentos através de cenários controlados. Verificou-se que o mecanismo proposto pelo presente trabalho mostrou uma economia de aproximadamente 32% quando comparado ao uso do *smartphone* sem nenhum mecanismo de economia de energia instalado. Além disso, comparou-se a eficiência do MyBatRecommender com a eficiência do principal aplicativo Android de mesma categoria, o BatteryDoctor, e, após os testes, chegou-se a conclusão que em ambas soluções possuem desempenho parecido.

Além dos casos de uso em cenários controlados a abordagem proposta foi testada com usuários reais: alunos de graduação da UFSCar - *campus* Sorocaba e também colaboradores do instituto Venturus Centro de Inovação Tecnológica utilizaram o mecaismo proposto por um período de duas semanas, sendo uma sem o sistema proposto e outra com o sistema proposto instalado. Após esse período análises foram

feitas e observou-se que o mecanismo proposto não produziu um resultado satisfatório em relação à economia de energia para esse experimento.

As principais contribuições desse trabalho são: a proposição de um mecanismo automatizado para o problema do consumo de energia em *smartphones* além da sua implementação para o sistema operacional Android, além dos experimentos elaborados e executados para sua validação. O trabalho também contribuiu através de um estudo bibliográfico sobre o assunto de gerenciamento e otimização de energia em *smartphones* em nível de software, conduzido através do processo de revisão sistemática.

5.0.1 Limitações e trabalhos futuros

O presente trabalho, em sua forma implementada, está restrito ao sistema operacional Android. Além disso, apesar do mesmo não estar restrito a determinadas versões desse sistema operacional, notou-se que a implementação pode ter a sua funcionalidade alterada em versões iguais ou superiores ao Lollipop, uma vez que a partir dessa versão foram impostos controles pelo próprio sistema operacional que invalidam algumas ações do sistema.

Os seguintes pontos foram levantados como possíveis melhorias a presente proposta e trabalhos futuros:

- Implementar o mecanismo para outros sistemas operacionais, como iOS;
- Considerar outras formas de gerar o perfil do usuário, como algoritmos de aprendizado de máquina, no módulo MyBatServer;
- Considerar formas para criar períodos dinâmicos que se adequem ao cotidiano do usuário ao invés da utilização de períodos estáticos;
- Implementar o serviço de forma incremental, considerando os dados coletados desde a primeira instalação, ao invés de apenas a última semana, como está implementada a atual solução;
- Para a presente implementação, em Android, integrar os dois aplicativos para o MyBatLogger e o MyBatSaver desenvolvidos em apenas um, para que seja mais fácil a sua instalação e utilização pelos usuários. A implementação atual possui um aplicativo para cada elemento pela facilidade de desenvolvimento e melhor organização das validações;
- Considerar novos componentes além de novas variáveis para serem coletadas e analisadas a fim de melhorar a eficiência;
- Considerar os aplicativos em execução no *smartphone* a fim de validar o perfil gerado;

- Avaliar a eficácia através de estudos de consumo de energia por componente;

Referências

ABOGHARAF, A. et al. A methodology for energy performance testing of smartphone applications. In: *Proceedings of the 7th International Workshop on Automation of Software Test*. Piscataway, NJ, USA: IEEE Press, 2012. (AST '12), p. 110–116. ISBN 978-1-4673-1822-8. Disponível em: <http://dl.acm.org/citation.cfm?id=2663608.2663630>. Citado na página 121.

ABOWD, G. D. et al. Towards a better understanding of context and context-awareness. In: *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*. London, UK, UK: Springer-Verlag, 1999. (HUC '99), p. 304–307. ISBN 3-540-66550-1. Disponível em: <http://dl.acm.org/citation.cfm?id=647985.743843>. Citado na página 23.

ABREU, D.; VILLAPOL, M. Measuring the energy consumption of communication interfaces on smartphones using a moderately-invasive technique. In: *Global Information Infrastructure and Networking Symposium (GIIS), 2012*. [S.l.: s.n.], 2012. p. 1–6. Citado 2 vezes nas páginas 45 e 121.

ALAM, F. et al. Energy optimization in android applications through wakelock placement. In: *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*. [S.l.: s.n.], 2014. p. 1–4. Citado na página 122.

ANANTHANARAYANAN, G.; STOICA, I. Blue-fi: Enhancing wi-fi performance using bluetooth signals. In: *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*. New York, NY, USA: ACM, 2009. (MobiSys '09), p. 249–262. ISBN 978-1-60558-566-6. Disponível em: <http://doi.acm.org/10.1145/1555816.1555842>. Citado na página 47.

ARDITO, L. et al. glcb: an energy aware context broker. *Sustainable Computing: Informatics and Systems*, v. 3, n. 1, p. 18 – 26, 2013. ISSN 2210-5379. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2210537912000522>. Citado na página 122.

BEDREGAL, J. V.; M, R. A.; GUTIERREZ, E. C. Optimizing energy consumption per application in mobile devices. In: *Information Society (i-Society), 2013 International Conference on*. [S.l.: s.n.], 2013. p. 106–110. Citado na página 121.

BHATTACHARYA, S. et al. Robust and energy-efficient trajectory tracking for mobile devices. *Mobile Computing, IEEE Transactions on*, v. 14, n. 2, p. 430–443, Feb 2015. ISSN 1536-1233. Citado na página 123.

BOLLA, R. et al. Improving smartphones battery life by reducing energy waste of background applications. In: *Next Generation Mobile Apps, Services and Technologies (NGMAST), 2014 Eighth International Conference on*. [S.l.: s.n.], 2014. p. 123–130. Citado na página 125.

BRODKIN, J. *Android*. 2012. Http://arstechnica.com/gadgets/2012/11/on-androids-5th-birthday-5-things-we-love-about-android/. Acessado: 29/12/2015. Citado na página 50.

CAMPS-MUR, D.; LOUREIRO, P. $E^2dwi - fi$: $Amechanismtoachieveenergyefficientdiscoveryinwi - fi.Mobile\ Computing,\ IEEE\ Transactions\ on, v.\ 13, n.\ 6, p.1186 - -1199, June2014.ISSN1536 - 1233.Citadonapágina123.$

CAZELLA, M. A. S. N. N. e. E. B. R. S. C. *A Ciência da Opinião: Estado da arte em Sistemas de Recomendação*. 2010. Http://200.17.141.213/ gutanunes/hp/publications/JAI4.pdf. Acessado: 21/12/2015. Citado 2 vezes nas páginas 49 e 50.

CHEN, X. et al. Demystifying energy usage in smartphones. In: *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*. [S.l.: s.n.], 2014. p. 1–5. Citado 2 vezes nas páginas 23 e 122.

CORRAL, L. et al. A method for characterizing energy consumption in android smartphones. In: *Green and Sustainable Software (GREENS), 2013 2nd International Workshop on*. [S.l.: s.n.], 2013. p. 38–45. Citado 2 vezes nas páginas 51 e 122.

CUERVO, E. et al. Maui: Making smartphones last longer with code offload. In: *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*. New York, NY, USA: ACM, 2010. (MobiSys '10), p. 49–62. ISBN 978-1-60558-985-5. Disponível em: <http://doi.acm.org/10.1145/1814433.1814441>. Citado na página 48.

DATTA, S.; BONNET, C.; NIKAEIN, N. Android power management: Current and future trends. In: *Enabling Technologies for Smartphone and Internet of Things (ETSIoT), 2012 First IEEE Workshop on*. [S.l.: s.n.], 2012. p. 48–53. Citado na página 124.

DEVELOPERS, A. *Android Developers*. 2015. Disponível em: <http://developer.android.com/intl/pt-br/index.html>. Citado na página 74.

DEVELOPERS, A. *Intenções e filtros de intenções*. 2015. Disponível em: <http://developer.android.com/intl/pt-br/guide/components/intents-filters.html#Resolution>. Citado na página 65.

DING, A. et al. Enabling energy-aware collaborative mobile data offloading for smartphones. In: *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2013 10th Annual IEEE Communications Society Conference on*. [S.l.: s.n.], 2013. p. 487–495. Citado na página 124.

DING, F. et al. Monitoring energy consumption of smartphones. In: *Proceedings of the 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*. Washington, DC, USA: IEEE Computer Society, 2011. (ITHINGSCPSCOM '11), p. 610–613. ISBN 978-0-7695-4580-6. Disponível em: <http://dx.doi.org/10.1109/iThings/CPSCom.2011.122>. Citado 2 vezes nas páginas 45 e 121.

DING, R.; MUNTEAN, G.-M. A context-aware cross-layer energy-efficient adaptive routing algorithm for wlan communications. In: *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*. [S.l.: s.n.], 2012. p. 176–179. ISSN 0742-1303. Citado na página 123.

DONOHOO, B.; OHLSEN, C.; PASRICHA, S. Aura: An application and user interaction aware middleware framework for energy optimization in mobile devices. In: *Computer Design (ICCD), 2011 IEEE 29th International Conference on*. [S.l.: s.n.], 2011. p. 168–174. ISSN 1063-6404. Citado 2 vezes nas páginas 46 e 125.

DONOHOO, B. et al. Context-aware energy enhancements for smart mobile devices. *Mobile Computing, IEEE Transactions on*, v. 13, n. 8, p. 1720–1732, Aug 2014. ISSN 1536-1233. Citado 4 vezes nas páginas 13, 46, 47 e 123.

DUAN, R.; BI, M.; GNIADY, C. Exploring memory energy optimizations in smartphones. In: *Green Computing Conference and Workshops (IGCC), 2011 International*. [S.l.: s.n.], 2011. p. 1–8. Citado na página 125.

FEKETE, K. et al. Analyzing computation offloading energy-efficiency measurements. In: *Communications Workshops (ICC), 2013 IEEE International Conference on*. [S.l.: s.n.], 2013. p. 301–305. Citado na página 124.

FERRONI, M. et al. Mpower: Gain back your android battery life! In: *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*. New York, NY, USA: ACM, 2013. (UbiComp '13 Adjunct), p. 171–174. ISBN 978-1-4503-2215-7. Disponível em: <http://doi.acm.org/10.1145/2494091.2494147>. Citado na página 54.

GHERGHINA, A.; OLTEANU, A.-C.; TAPUS, N. Measuring performance and energy consumption when offloading from mobile devices. In: *Systems and Computer Science (ICSCS), 2013 2nd International Conference on*. [S.l.: s.n.], 2013. p. 98–102. Citado na página 124.

GUPTA, M.; KOC, A.; VANNITHAMBY, R. Analyzing mobile applications and power consumption on smartphone over lte network. In: *Energy Aware Computing (ICEAC), 2011 International Conference on*. [S.l.: s.n.], 2011. p. 1–4. Citado na página 125.

HAMZAOUI, K. et al. Survey on adaptation techniques of energy consumption within a smartphone. In: *Science and Information Conference (SAI), 2014*. [S.l.: s.n.], 2014. p. 247–253. Citado na página 124.

HANS, R. et al. Where did my battery go? quantifying the energy consumption of cloud gaming. In: *Mobile Services (MS), 2014 IEEE International Conference on*. [S.l.: s.n.], 2014. p. 63–67. Citado na página 124.

HAO, S. et al. Estimating mobile application energy consumption using program analysis. In: *Proceedings of the 2013 International Conference on Software Engineering*. Piscataway, NJ, USA: IEEE Press, 2013. (ICSE '13), p. 92–101. ISBN 978-1-4673-3076-3. Disponível em: <http://dl.acm.org/citation.cfm?id=2486788.2486801>. Citado na página 121.

HEIKKINEN, M. V. et al. Energy efficiency of mobile handsets: Measuring user attitudes and behavior. *Telematics and Informatics*, v. 29, n. 4, p. 387 – 399, 2012. ISSN 0736-5853. Green Information Communication Technology. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0736585312000068>. Citado 3 vezes nas páginas 47, 49 e 124.

HERRMANN, R.; ZAPPI, P.; ROSING, T. *Context Aware Power Management of Mobile Systems for Sensing Applications*. 2012. Citado 2 vezes nas páginas 46 e 122.

HYEON, Y. et al. Battery life time extension method by using signalling interval control. In: *Advanced Communication Technology (ICACT), 2012 14th International Conference on*. [S.l.: s.n.], 2012. p. 327–330. ISSN 1738-9445. Citado 2 vezes nas páginas 47 e 123.

ICKIN, S.; WAC, K.; FIEDLER, M. Qoe-based energy reduction by controlling the 3g cellular data traffic on the smartphone. In: *Energy Efficient and Green Networking (SSEEGN), 2013 22nd ITC Specialist Seminar on*. [S.l.: s.n.], 2013. p. 13–18. Citado na página 122.

IDC. *Worldwide Smartphone Growth*. 2015. Disponível em: <http://www.idc.com/getdoc.jsp?containerId=prUS25860315>. Citado na página 24.

INSTRUMENTS, T. *Android Devkit Power Management Porting Guide*. 2016. Http://processors.wiki.ti.com/index.php/Android_Devkit_Power_Management_Porting_Guide. "Acessado: 05/01/2016". Citado 2 vezes nas páginas 13 e 51.

JIAO, L. et al. Cloud-based computation offloading for mobile devices: State of the art, challenges and opportunities. In: *Future Network and Mobile Summit (FutureNetworkSummit), 2013*. [S.l.: s.n.], 2013. p. 1–11. Citado 2 vezes nas páginas 48 e 124.

JOFRI, M.; FUDZEE, M. F. M.; ISMAIL, M. *A survey on energy-aware profiler for mobile devices*. 2015. Citado na página 125.

JUNG, W.; KIM, K.; CHA, H. Userscope: A fine-grained framework for collecting energy-related smartphone user contexts. In: *Proceedings of the 2013 International Conference on Parallel and Distributed Systems*. Washington, DC, USA: IEEE Computer Society, 2013. (ICPADS '13), p. 158–165. ISBN 978-1-4799-2081-5. Disponível em: <http://dx.doi.org/10.1109/.32>. Citado na página 122.

KAMIYAMA, T.; INAMURA, H.; OHTA, K. A model-based energy profiler using online logging for android applications. In: *Mobile Computing and Ubiquitous Networking (ICMU), 2014 Seventh International Conference on*. [S.l.: s.n.], 2014. p. 7–13. Citado na página 121.

KIM, D.; JUNG, N.; CHA, H. Content-centric display energy management for mobile devices. In: *Proceedings of the 51st Annual Design Automation Conference*. New York, NY, USA: ACM, 2014. (DAC '14), p. 41:1–41:6. ISBN 978-1-4503-2730-5. Disponível em: <http://doi.acm.org/10.1145/2593069.2593113>. Citado na página 123.

KITCHENHAM, B. Procedures for performing systematic reviews. *Keele, UK, Keele University*, v. 33, n. 2004, p. 1–26, 2004. ISSN 1353-7776. Citado na página 29.

KOENIG, I.; MEMON, A. Q.; DAVID, K. Energy consumption of the sensors of smartphones. In: *Wireless Communication Systems (ISWCS 2013), Proceedings of the Tenth International Symposium on*. [S.l.: s.n.], 2013. p. 1–5. Citado 2 vezes nas páginas 23 e 122.

KRINTZ, C.; WEN, Y.; WOLSKI, R. Application-level prediction of battery dissipation. In: *Low Power Electronics and Design, 2004. ISLPED '04. Proceedings of the 2004 International Symposium on*. [S.l.: s.n.], 2004. p. 224–229. Citado 2 vezes nas páginas 24 e 125.

LEE, J.; JOE, H.; KIM, H. Automated power model generation method for smartphones. *Consumer Electronics, IEEE Transactions on*, v. 60, n. 2, p. 190–197, May 2014. ISSN 0098-3063. Citado na página 122.

LI, J. et al. Energy efficient wi-fi management for smart devices. In: *Network Operations and Management Symposium (NOMS), 2014 IEEE*. [S.l.: s.n.], 2014. p. 1–9. Citado na página 123.

LI, X. et al. Measurement and analysis of energy consumption on android smartphones. In: *Information Science and Technology (ICIST), 2014 4th IEEE International Conference on*. [S.l.: s.n.], 2014. p. 242–245. Citado na página 122.

LIU, X. et al. Phonejoule: An energy management system for android-based smartphones. In: *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*. [S.l.: s.n.], 2013. p. 1996–2001. Citado na página 122.

LIU, Y.; XU, C.; CHEUNG, S. Where has my battery gone? finding sensor related energy black holes in smartphone applications. In: *Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on*. [S.l.: s.n.], 2013. p. 2–10. Citado na página 121.

LIU, Y. et al. Greendroid: Automated diagnosis of energy inefficiency for smartphone applications. *Software Engineering, IEEE Transactions on*, v. 40, n. 9, p. 911–940, Sept 2014. ISSN 0098-5589. Citado na página 121.

MAN, Y.; NGAI, E. C.-H. Energy-efficient automatic location-triggered applications on smartphones. *Computer Communications*, v. 50, p. 29 – 40, 2014. ISSN 0140-3664. Green Networking. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0140366414001200>. Citado na página 123.

MARIN, R.-C.; DOBRE, C. Reaching for the clouds: Contextually enhancing smartphones for energy efficiency. In: *Proceedings of the 2Nd ACM Workshop on High Performance Mobile Opportunistic Systems*. New York, NY, USA: ACM, 2013. (HP-MOSys '13), p. 31–38. ISBN 978-1-4503-2372-7. Disponível em: <http://doi.acm.org/10.1145/2507908.2507912>. Citado na página 124.

METRI, G. et al. What is eating up battery life on my smartphone: A case study. In: *Energy Aware Computing, 2012 International Conference on*. [S.l.: s.n.], 2012. p. 1–6. Citado na página 121.

MOLDOVAN, A.-N.; WEIBELZAHL, S.; MUNTEAN, C. Energy-aware mobile learning:opportunities and challenges. *Communications Surveys Tutorials, IEEE*, v. 16, n. 1, p. 234–265, First 2014. ISSN 1553-877X. Citado na página 124.

NEGI, V.; KALRA, M. Framework for energy saving in the android smartphone. In: *Information Management in the Knowledge Economy (IMKE), 2013 2nd International Conference on*. [S.l.: s.n.], 2013. p. 161–165. Citado na página 124.

OSHIN, T.; POSLAD, S.; MA, A. Improving the energy-efficiency of gps based location sensing smartphone applications. In: *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*. [S.l.: s.n.], 2012. p. 1698–1705. Citado 2 vezes nas páginas 47 e 123.

PATHAK, A. et al. What is keeping my phone awake?: Characterizing and detecting no-sleep energy bugs in smartphone apps. In: *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*. New York, NY, USA: ACM, 2012. (MobiSys '12), p. 267–280. ISBN 978-1-4503-1301-8. Disponível em: <http://doi.acm.org/10.1145/2307636.2307661>. Citado na página 25.

PERRUCCI, G.; FITZEK, F.; WIDMER, J. Survey on energy consumption entities on the smartphone platform. In: *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*. [S.l.: s.n.], 2011. p. 1–6. ISSN 1550-2252. Citado 3 vezes nas páginas 44, 53 e 125.

PINTO, G.; CASTOR, F.; LIU, Y. D. Mining questions about software energy consumption. In: *Proceedings of the 11th Working Conference on Mining Software Repositories*. New York, NY, USA: ACM, 2014. (MSR 2014), p. 22–31. ISBN 978-1-4503-2863-0. Disponível em: <http://doi.acm.org/10.1145/2597073.2597110>. Citado na página 122.

POCKETNOW. *Android*. 2016. Http://pocketnow.com/2014/07/28/the-evolution-of-the-smartphone. Acessado: 05/01/2016. Citado na página 32.

QIU, M. et al. Towards power-efficient smartphones by energy-aware dynamic task scheduling. In: *High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on*. [S.l.: s.n.], 2012. p. 1466–1472. Citado na página 123.

ROBINSON, S. Cellphone energy gap: Desperately seeking solutions. *Strategy Analytics*, 2009. Citado na página 24.

SHARMA, R.; PRASAD, S.; BALAJI, S. A tour into ambient energy resources and battery optimization. In: *Signal and Image Processing (ICSIP), 2014 Fifth International Conference on*. [S.l.: s.n.], 2014. p. 343–347. Citado na página 124.

SHEARER, F. Chapter 2 - hierarchical view of energy conservation. In: SHEARER, F. (Ed.). *Power Management in Mobile Devices*. Burlington: Newnes, 2008. p. 39 – 75. ISBN 978-0-7506-7958-9. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780750679589000022>. Citado na página 52.

SHEARER, F. Chapter 5 - batteries and displays for mobile devices. In: SHEARER, F. (Ed.). *Power Management in Mobile Devices*. Burlington: Newnes, 2008. p. 149 – 180. ISBN 978-0-7506-7958-9. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780750679589000058>. Citado na página 52.

SUN, L. et al. Modeling wifi active power/energy consumption in smartphones. In: *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*. [S.l.: s.n.], 2014. p. 41–51. ISSN 1063-6927. Citado na página 122.

TA, T.; BARAS, J.; ZHU., C. *Improving smartphone battery life utilizing device-to-device cooperative relays underlaying LTE networks*. 2014. Citado na página 123.

TRAVASSOS, G. H.; GUROV, D.; AMARAL, E. *Introdução à Engenharia de Software Experimental*. [S.l.]: UFRJ, 2002. Citado na página 83.

VALLINA-RODRIGUEZ, N.; CROWCROFT, J. Energy management techniques in modern mobile handsets. *Communications Surveys Tutorials, IEEE*, v. 15, n. 1, p. 179–198, First 2013. ISSN 1553-877X. Citado 3 vezes nas páginas 23, 48 e 124.

WANG, C. et al. Power estimation for mobile applications with profile-driven battery traces. In: *Low Power Electronics and Design (ISLPED), 2013 IEEE International Symposium on*. [S.l.: s.n.], 2013. p. 120–125. Citado na página 121.

WARREN, C. *Google Play Hits 1 Million Apps*. 2013. Http://mashable.com/2013/07/24/google-play-1-million/#4nflWq.N8PqW. Acessado: 29/12/2015. Citado na página 50.

WOHLIN, C. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic, 2000. (International Series in Engineering and Computer Science). ISBN 9780792386827. Disponível em: <http://books.google.com.br/books?id=nG2UShV0wAEC>. Citado na página 106.

YI, S.-H.; CHO, S.-B. A battery-aware energy-efficient android phone with bayesian networks. In: *Ubiquitous Intelligence Computing and 9th International Conference on Autonomic Trusted Computing (UIC/ATC), 2012 9th International Conference on*. [S.l.: s.n.], 2012. p. 204–209. Citado 2 vezes nas páginas 46 e 123.

ZAHID, I.; ALI, M.; NASSR, R. Android smartphone: Battery saving service. In: *Research and Innovation in Information Systems (ICRIIS), 2011 International Conference on*. [S.l.: s.n.], 2011. p. 1–4. Citado 3 vezes nas páginas 46, 49 e 125.

ZHANG, L. et al. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In: *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2010 IEEE/ACM/IFIP International Conference on*. [S.l.: s.n.], 2010. p. 105–114. Citado na página 125.

ZHANG, Y. et al. Netmaster: Taming energy devourers on smartphones. In: *Parallel Processing (ICPP), 2014 43rd International Conference on*. [S.l.: s.n.], 2014. p. 301–310. ISSN 0190-3918. Citado na página 122.

APÊNDICE A – Lista de artigos classificados por números

Tabela 41 – Lista de artigos classificados por números

Número	Título	Referência	Ano
1	Measuring the energy consumption of communication interfaces on smartphones using a moderately-invasive technique	(ABREU; VILLAPOL, 2012)	2012
2	Monitoring Energy Consumption of Smartphones	(DING et al., 2011)	2011
3	What is eating up battery life on my SmartPhone: A case study	(METRI et al., 2012)	2012
4	A Methodology for Energy Performance Testing of Smartphone Applications	(ABOGHARAF et al., 2012)	2012
5	A model-based energy profiler using online logging for Android applications	(KAMIYAMA; INAMURA; OHTA, 2014)	2014
6	Where has my battery gone? Finding sensor related energy black holes in smartphone applications	(LIU; XU; CHEUNG, 2013)	2013
7	GreenDroid: Automated Diagnosis of Energy Inefficiency for Smartphone Applications	(LIU et al., 2014)	2014
8	Power estimation for mobile applications with profile-driven battery traces	(WANG et al., 2013)	2013
9	Estimating Mobile Application Energy Consumption Using Program Analysis	(HAO et al., 2013)	2013
10	Optimizing energy consumption per application in mobile devices	(BEDREGAL; M; GUTIERREZ, 2013)	2013

11	Automated power model generation method for smartphones	(LEE; JOE; KIM, 2014)	2014
12	A method for characterizing energy consumption in Android smartphones	(CORRAL et al., 2013)	2013
13	Energy consumption of the sensors of Smartphones	(KOENIG; MEMON; DAVID, 2013)	2013
14	Measurement and analysis of energy consumption on Android smartphones	(LI et al., 2014b)	2014
15	Demystifying energy usage in smartphones	(CHEN et al., 2014)	2014
16	UserScope: A Fine-Grained Framework for Collecting Energy-Related Smartphone User Contexts	(JUNG; KIM; CHA, 2013)	2013
17	Modeling WiFi Active Power/Energy Consumption in Smartphones	(SUN et al., 2014)	2014
18	Context Aware Power Management of Mobile Systems for Sensing Applications	(HERRMANN; ZAPPI; ROSING, 2012)	2012
19	gLCB: an energy aware context broker	(ARDITO et al., 2013)	2013
20	NetMaster: Taming Energy Devourers on Smartphones	(ZHANG et al., 2014)	2014
21	PhoneJoule: An Energy Management System for Android-Based Smartphones	(LIU et al., 2013)	2013
22	QoE-based energy reduction by controlling the 3g cellular data traffic on the smartphone	(ICKIN; WAC; FIEDLER, 2013)	2013
23	Energy optimization in Android applications through wakelock placement	(ALAM et al., 2014)	2014
24	Mining Questions About Software Energy Consumption	(PINTO; CASTOR; LIU, 2014)	2014

25	Improving the Energy-Efficiency of GPS Based Location Sensing Smartphone Applications	(OSHIN; POSLAD; MA, 2012)	2012
26	Battery life time extension method by using signalling interval control	(HYEON et al., 2012)	2012
27	A context-aware cross-layer energy-efficient adaptive routing algorithm for WLAN communications	(DING; MUNTEAN, 2012)	2012
28	Towards Power-Efficient Smartphones by Energy-Aware Dynamic Task Scheduling	(QIU et al., 2012)	2012
29	A Battery-Aware Energy-Efficient Android Phone with Bayesian Networks	(YI; CHO, 2012)	2012
30	Content-centric Display Energy Management for Mobile Devices	(KIM; JUNG; CHA, 2014)	2014
31	Improving smartphone battery life utilizing device-to-device cooperative relays underlaying LTE networks	(TA; BARAS; ZHU., 2014)	2014
32	Context-Aware Energy Enhancements for Smart Mobile Devices	(DONOHOO et al., 2014)	2014
33	Energy efficient Wi-Fi management for smart devices	(LI et al., 2014a)	2014
34	E^{2}D Wi-Fi: A Mechanism to Achieve Energy Efficient Discovery in Wi-Fi	(CAMPS-MUR; LOUREIRO, 2014)	2014
35	Energy-efficient automatic location-triggered applications on smartphones	(MAN; NGAI, 2014)	2014
36	Robust and Energy-Efficient Trajectory Tracking for Mobile Devices	(BHATTACHARYA et al., 2015)	2015

37	Enabling energy-aware collaborative mobile data offloading for smartphones	(DING et al., 2013)	2013
38	Android power management: Current and future trends	(DATTA; BONNET; NIKAEIN, 2012)	2012
39	Energy Management Techniques in Modern Mobile Handsets	(VALLINA-RODRIGUEZ; CROWCROFT, 2013)	2013
40	Cloud-based computation offloading for mobile devices: State of the art, challenges and opportunities	(JIAO et al., 2013)	2013
41	Energy-Aware Mobile Learning: Opportunities and Challenges	(MOLDOVAN; WEIBELZAHL; MUNTEAN, 2014)	2014
42	A Tour into Ambient Energy Resources and Battery Optimization	(SHARMA; PRASAD; BALAJI, 2014)	2014
43	Survey on adaptation techniques of energy consumption within a smartphone	(HAMZAOUI et al., 2014)	2014
44	Energy efficiency of mobile handsets: Measuring user attitudes and behavior	(HEIKKINEN et al., 2012)	2012
45	Where Did My Battery Go? Quantifying the Energy Consumption of Cloud Gaming	(HANS et al., 2014)	2014
46	Reaching for the Clouds: Contextually Enhancing Smartphones for Energy Efficiency	(MARIN; DOBRE, 2013)	2013
47	Analyzing computation offloading energy-efficiency measurements	(FEKETE et al., 2013)	2013
48	Measuring performance and energy consumption when offloading from mobile devices	(GHERGHINA; OLTEANU; TAPUS, 2013)	2013
49	Framework for energy saving in the Android smartphone	(NEGI; KALRA, 2013)	2013

50	Improving Smartphones Battery Life by Reducing Energy Waste of Background Applications	(BOLLA et al., 2014)	2014
51	A survey on energy-aware profiler for mobile devices	(JOFRI; FUDZEE; ISMAIL, 2015)	2015
52	Survey on energy consumption entities on the smartphone platform	(PERRUCCI; FITZEK; WIDMER, 2011)	2011
53	Analyzing mobile applications and power consumption on smartphone over LTE network	(GUPTA; KOC; VANNITHAMBY, 2011)	2011
54	Accurate online power estimation and automatic battery behavior based power model generation for smartphones	(ZHANG et al., 2010)	2010
55	Exploring memory energy optimizations in smartphones	(DUAN; BI; GNIADY, 2011)	2011
56	Android Smartphone: Battery saving service	(ZAHID; ALI; NASSR, 2011)	2011
57	AURA: An application and user interaction aware middleware framework for energy optimization in mobile devices	(DONOHOO; OHLSEN; PASRICHA, 2011)	2011
58	Application-level prediction of battery dissipation	(KRINTZ; WEN; WOLSKI, 2004)	2004

APÊNDICE B – Formulário - Participação em Pesquisa sobre Consumo de Energia para Android

O objetivo da pesquisa é coletar dados de uso dos sensores do celular (WiFi, GPS, etc) que incidam diretamente no consumo de energia de smartphones Android. Os dados coletados serão utilizados no trabalho de mestrado do aluno Marcel Cunha. Os dados coletados serão utilizados de forma anônima.

Você gerencia o estado dos componentes do seu smartphone? *
Por exemplo, desliga Wi-Fi/Bluetooth/outros componentes quando não está conectado, reduz o brilho da tela, etc.

- ○ Sim
- ○ Muitas vezes
- ○ Poucas vezes
- ○ Nunca

Você acredita possuir uma rotina que se repete ao longo dos dias e das semanas? *
Por exemplo, de casa para o trabalho/escola, do trabalho/escola para a academia e depois para casa.

- ☐ Sim, uma rotina diurna
- ☐ Sim, uma rotina vespertina
- ☐ Sim, uma rotina noturna
- ☐ Sim, uma rotina aos finais de semana
- ☐ Não, não possuo uma rotina

Indique, para o período diurno, quais os componentes que você acredita que mais utiliza *
Por exemplo, para o Wi-Fi, considerar que ele é utilizado quando ele estiver conectado a uma rede

- ☐ Wi-Fi
- ☐ Redes Móveis
- ☐ Bluetooth
- ☐ GPS
- ☐ Other: _____

Indique, para o período vespertino, quais os componentes que você acredita que mais utiliza *
Por exemplo, para o Wi-Fi, considerar que ele é utilizado quando ele estiver conectado a uma rede

- ☐ Wi-Fi
- ☐ Redes Móveis
- ☐ Bluetooth
- ☐ GPS
- ☐ Other: _____

Indique, para o período noturno, quais os componentes que você acredita que mais utiliza *
Por exemplo, para o Wi-Fi, considerar que ele é utilizado quando ele estiver conectado a uma rede

- [ ] Wi-Fi
- [ ] Redes Móveis
- [ ] Bluetooth
- [ ] GPS
- [ ] Other:

Quantas vezes, em média, no período diurno, você muda de ambiente? *
Por exemplo, sair de casa para ir ao transporte/trabalho/escola.

Quantas vezes, em média, no período vespertino, você muda de ambiente? *
Por exemplo, sair de casa para ir ao transporte/trabalho/escola.

Quantas vezes, em média, no período noturno, você muda de ambiente? *
Por exemplo, sair de casa para ir ao transporte/trabalho/escola.

Quantas horas, em média, por dia, você acredita que permanece no mesmo ambiente? *
Por exemplo 12hrs em casa, 12hrs no trabalho, uma média de 12hrs por ambiente.

- ( ) até 1 hora
- ( ) De 1 a 2 horas
- ( ) De 2 a 4 horas
- ( ) 4 ou mais horas

Qual é a versão do Android de seu smartphone? *
Para ver a versão vá em Configuração -> Sobre o telefone -> Versão Android

- ( ) 2.2 (Froyo)
- ( ) 2.3.3 - 2.3.7 (Gingerbread)
- ( ) 3.2 (Honeycomb)
- ( ) 4.0.3 - 4.0.4 (Ice Cream Sandwich)
- ( ) 4.1.X, 4.2.X, 4.3 (Jelly Bean)
- ( ) 4.4 (Kit Kat)
- ( ) 5.0 , 5.1 (Lollipop)

Marca / Modelo do smartphone Android *

Se você já é usuário do aplicativo MyBRecommender, por favor, informe o seu user id *
Para isso, basta abrir o aplicativo e informar o número abaixo da imagem de usuário

APÊNDICE C – Coleta de Dados – Instruções

Olás, tudo bem? Primeiramente gostaria de agradecer a participação de vocês nessa etapa do meu projeto de mestrado, que será dividida em duas fases:

Coleta de dados: Nessa fase, um aplicativo, Logger, será executado com o intuito de coletar os dados de utilização de diversos sensores e componentes do smartphone.

Recomendação de perfil: Nessa fase, um aplicativo, MyBRecommender, será executado com o intuito de gerenciar os estados dos sensores e componentes do smartphone, baseado nos dados coletados na primeira fase a fim de economizar energia.

Desde já gostaria esclarecer que estou à disposição
para sanar qualquer dúvida/reclamação/sugestão que vocês tenham e também gostaria de informar que a qualquer momento vocês podem parar de participar da coleta de dados, por qualquer motivo, bastando apenas desinstalar os aplicativos. Peço apenas que me informer o motivo, se possível, para que isso possa ser usado em prol de melhorias nos aplicativos. A seguir, estão listadas os passos de cada fase:

1) Coleta de Dados
1.1) Informações
Os dados coletados são relativos à diversos sensores e componentes do celular, como: GPS, Wi-Fi, nível de baeria, etc. Os dados não serão tratados de forma nominal, ou seja, serão armazenados de forma que os usuários não poderão ser posteriormente identificados. Os dados são armazenados localmente, em um banco de dados. uando o celular estiver conectado à Internet através de uma conexão Wi-Fi, os dados serão enviados ao servidor remoto e apagados do celular. Por isso, é interessante que você possa se conectar à uma Internet via Wi-Fi ao menos uma vez ao dia, para que os dados sejam enviados ao servidor. Os dados não serão enviados pela rede de dados do celular para não gerar gastos. Caso algum problema com a execução do aplicativo ocasione o envio de dados pela rede móvel, e você se sinta prejudicado, por favor pare a execução do aplicativo e me informe sobre o ocorrido.

1.2) Cronograma
27/09 -> Recebimento e instalação do aplicativo
28/09 à 02/10 -> Execução do aplicativo
10/10 -> Remoção do aplicativo

1.3) Instalando o aplicativo
No dia 27/09, vocês receberão por e-mail um arquivo chamado Logger.apk, que deve ser instalado no smartphone. Esse arquivo deverá ser baixado do próprio smartphone (ou de outra fonte, como um notebook e passado posteriormente para o smartphone de alguma forma: Cabo USB, Micro SD..), e instalado. Para instalar, basta executar o arquivo dentro do celular, lembrando que para isso o smartphone deve estar configurado para aceitar instalar aplicativos de fontes desconhecidas. Para ativar essa configuração, faça : Configuração -> Segurança (Na aba pessoal) -> Fontes desconhecidas (Ativar Checkbox) Feito isso, o aplicativo estará instalado.

1.4) Executando o aplicativo
Para executar a primeira vez o aplicativo, você deve procurar nos aplicativos instalados, através do menu inicial do Android, o aplicativo chamado Logger, e clicá-lo. Ao executar o aplicativo, a seguinte mensagem será exibida na tela : “Service Connected”, e o aplicativo já estará rodando, fazendo a coleta automaticamente. Ao desligar o seu smartphone e ligá-lo novamente, o aplicativo deve executar automaticamente, e a mesma mensagem será exibida. Caso isso não ocorra, favor executar o aplicativo como descrito acima. Caso o aplicativo apresente algum erro e pare de executar, basta iniciá-lo novamente. Favor me comunicar via e-mail para alertar sobre o erro.

1.5) Remoção do aplicativo
Até o dia 02/10 o aplicativo Logger terá a função de coletar os dados que serão utilizados posteriormente para gerar o perfil do usuário. Após isso, o Logger deverá continuar instalado pois ele será utilizado para coletar o nível de energia do celular quando usado com o MyBRecommender, para posterior análise. Por conta disso, ele deverá ficar instalado durante todas as fases. No dia 10/10 o aplicativo já terá coletado dados suficientes. Caso você deseje, basta apenas desinstalar o aplicativo pelo menu inicial do Android, ou através das configurações.

2) Recomendação de Perfil
2.1) Informações

Através dos dados coletados na primeira fase, será gerado um perfil para o usuário. Esse perfil será composto de estados dos sensores e componentes através das horas dos dias. A função do aplicativo MyBRecommender será aplicar essas configurações aos sensores do smartphone a fim de economizar energia. Dependendo da versão do Android e modelo do celular, algumas configurações não poderão ser aplicadas diretamente. Nesses casos, notificações serão exibidas indicando a ação a ser tomada. Pede-se que nesses casos, se possível, o usuário tome as ações descritas na notificação. Você pode entrar no aplicativo a qualquer momento para verificar qual é o perfil atual sendo aplicado, e quais serão os próximos.

2.2) Cronograma
04/10 -> Recebimento e instalação do aplicativo
05/10 à 09/10 -> Execução do aplicativo
10/10 -> Remoção do aplicativo

2.3) Instalando o aplicativo
No dia 04/10, vocês receberão por e-mail um arquivo chamado MyBRecommender.apk, que deve ser instalado no smartphone. Esse arquivo deverá ser baixado do próprio smartphone (ou de outra fonte, como um notebook e passado posteriormente para o smartphone de alguma forma: Cabo USB, Micro SD..), e instalado. Para instalar, basta executar o arquivo dentro do celular, lembrando que para isso o smartphone deve estar configurado para aceitar instalar aplicativos de fontes desconhecidas. Para ativar essa configuração, faça : Configuração -> Segurança (Na aba pessoal) -> Fontes desconhecidas (Ativar Checkbox) Feito isso, o aplicativo estará instalado.

2.4) Executando o aplicativo
Para executar a primeira vez o aplicativo, você deve procurar nos aplicativos instalados, através do menu inicial do Android, o aplicativo chamado MyBRecommender, e clicá-lo. Ao executar o aplicativo, a seguinte tela será exibida:

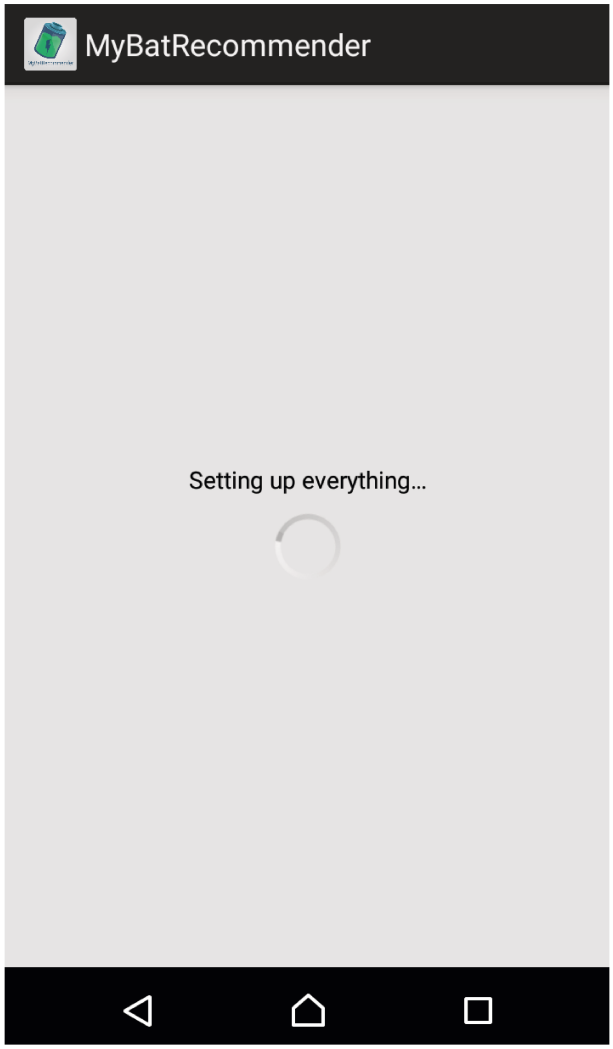
Assim que o download da configuração do perfil do usuário terminar, a seguinte tela será exibida:

A partir desse momento, o aplicativo já estará rodando. Ao desligar o seu smartphone e ligá-lo novamente, o aplicativo deve executar automaticamente. Caso a imagem abaixo seja exibida, signifca que algum erro com a comunição com o servidor ocorreu. Por favor, clique no botão indicado pela seta vermelha, para que o aplicativo tente novamente comunicar-se com o servidor, até que a imagem acima seja exibida com sucesso. Caso isso não ocorra, por favor entre em contato comigo.

Caso o aplicativo apresente algum erro e pare de executar, basta iniciá-lo novamente. Favor me comunicar via e-mail para alertar sobre o erro.

2.5) Remoção do aplicativo
No dia 10/10 o aplicativo já terá aplicado todas as configurações desejadas. Caso você deseje, basta apenas desinstalar o aplicativo pelo menu inicial do Android, ou através das configurações.

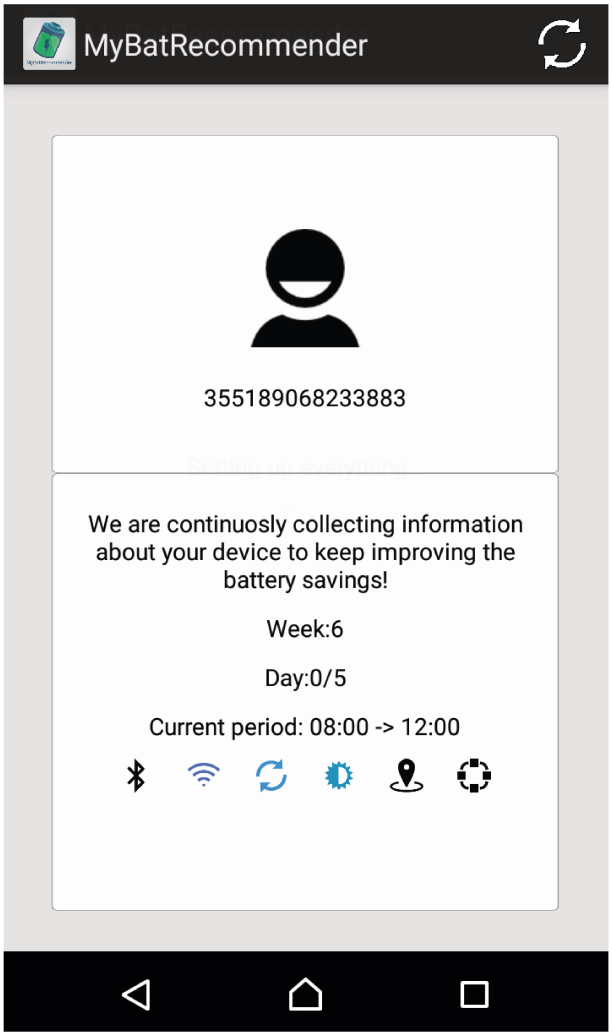
FAQ: 1) Quais informações serão coletadas do celular?
R: Serão coletados dados de diversos sensores do celular, como GPS, Wi-Fi, brilho de tela, etc. Além disso, nomes dos processos rodando no celular também são coletados. Nenhuma informação de dados pessoais será coletada.



2) Os dados serão tratados/rotulados de forma nominal?
R: Não. Os dados são armazenados de forma não nominal, como “user1”, de forma que não é possível identificar o usuário.

3) O Logger/MyBRecommender pode apresentar erros?
R: Sim. Apesar de ter sido testado, o aplicativo não foi testado em todas as versões do Android e dispositivos. Dessa forma, é possível que erros ocorram de forma a interromper a execução do mesmo. Nesse caso, pedimos que o aplicativo seja executado novamente. Caso o erro persista, por favor, contate-me por e-mail para que possa analisá-lo. Além disso, é importante que fique claro que os aplicativos podem ser desinstalados a qualquer momento, interrompendo a sua participação no processo, por qualquer motivo. É importante que a participação no processo não atrapalhe o seu uso do smartphone.

4) O Logger/MyBRecommender pode apresentar um aumento do consumo de energia do celular?
R: Sim. Por ser um aplicativo sendo executado no celular, ele acarreta em consumo de energia, como todos os outros aplicativos. Mas esse aumento do consumo de

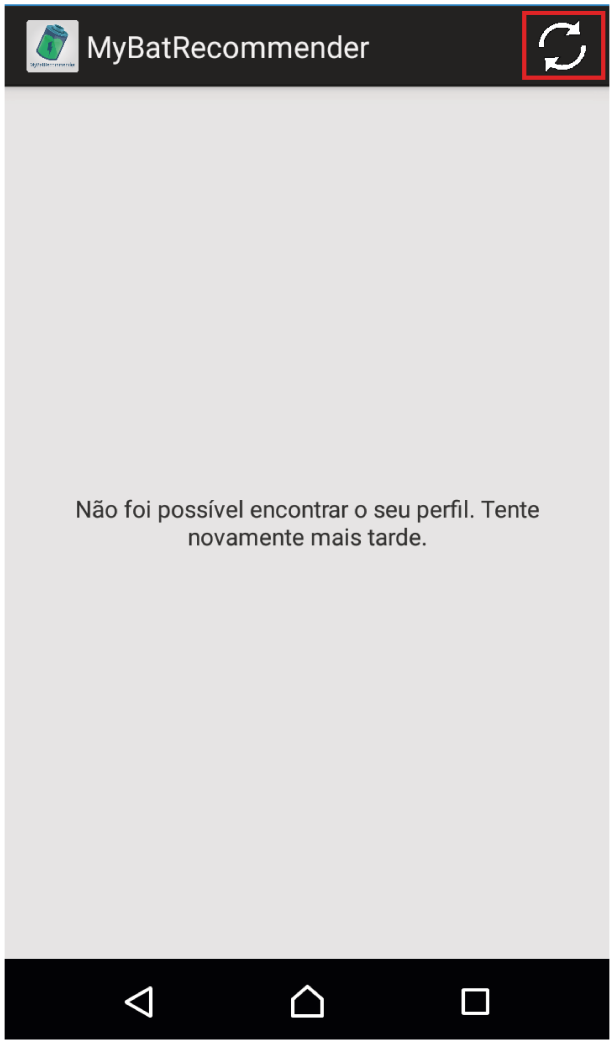


energia não deve ser significativo nem perceptível para o usuário. Com os testes executados, o consumo ficou em níveis entre 2% e 4%. Caso ocorra com você algum valor muito diferente desses, por favor me informe.

5) Desejo parar de utilizar o Logger/MyBRecommender (por qualquer motivo), como devo proceder?
R: O Logger pode ser interrompido a qualquer momento, por qualquer motivo. Para parar de executar o Logger, basta desinstalá-lo. Feito isso, não haverá mais coleta de dados. Pedimos apenas que nos mande um e-mail para notificar sobre a decisão. Caso o usuário deseje voltar a utilizar o Logger, basta seguir os passos da instalação novamente, ou, caso o aplicativo não tenha sido desinstalado, apenas executar novamente.

6) O Logger/MyBRecommender parou de funcionar, como devo proceder?
R: Caso o Logger apresente algum erro e pare funcionar pedimos que você execute o aplicativo novamente. Se os erros forem frequentes, por favor me informe.

7) Por quanto tempo devo usar o Logger/MyBRecommender?
R: A ideia é que o Logger/MyBRecommender seja executado por 5 dias após a



instalação e execução dos mesmos. O aplicativo pode ser executado por mais de 5 dias, sem problemas. Caso o usuário queira, o Logger/MyBRecommender pode ser interrompido a qualquer momento, bastando para isso apenas desinstalá-lo.

8) O Logger/MyBRecommender faz uso da minha Internet via plano de dados?
R: Não. Os dados são mandados para o servidor apenas quando o celular estiver conectado na Internet através de alguma rede Wi-Fi. Por isso, é interessante que o usuário conecte-se a alguma rede Wi-Fi sempre que possível.
Lembrando que qualquer dúvida estou a disposição. Mais uma vez muito obrigado pelo seu tempo e ajuda!

Grato, Marcel Cunha.

APÊNDICE D – Cenário para Validação por Simulação

O objetivo desse documento é descrever um cenário que será utilizado durante a validação por simulação para que os aplicativos de recomendação de perfis de bateria possam seguir e serem avaliados da mesma forma.

Pessoa: Usuário de aproximadamente 25 anos, que diariamente vai para o trabalho logo de manhã utilizando transporte próprio. Após o trabalho, no período da tarde, vai para academia, voltando para casa apenas no final do dia. Essa rotina se repete ao longo da semana, e no final de semana passa os dias na sua casa, saindo no final da tarde para outras cidades de carro.

A Tabela 43 mostra o cenário proposto enquanto a Tabela 42 mostra as configurações de cada período desse cenário, referenciado pelo número entre parenteses na tabela do cenário.

Tabela 42 – Estado dos principais componentes para o cenário proposto

	Wi-Fi	***Bluetooth***	**Redes Móveis**	**GPS**	**Áudios**	***Display***
1	Não utilizado	Não utilizado	Não utilizado	Não utilizado	Todos ligados	Não utilizado
2	Não utilizado	Ligado, conectado com o rádio	Utilizada	Não utilizado	Todos ligados	Pouco/não utilizada
3	Utilizado	Não utilizado	Não utilizado	Não utilizado	Todos ligados	Pouco utilizada
4	Não utilizado	Não utilizado	Utilizada	Não utilizado	Todos ligados	Bastante utilizada
5	Não utilizado	Não utilizado	Não utilizado	Não utilizado	Não utilizado	Não utilizado
6	Utilizado	Não utilizado	Não utilizado	Não utilizado	Todos ligados	Bastante utilizada
7	Não utilizado	Utilizado	Utilizada	Utilizada	Todos ligados	Bastante utilizada

Tabela 43 – Cenário da validação

	00h - 07h	**07h - 08h**	**08h - 12h**	**12h - 13h**	**13h - 18h**	**18h - 19h**	**19h - 20h**	**20h - 00h**
Dias de semana	Horário de dormir, permanecendo em casa, mas sem utilizar o celular. (1)	Horário de deslocamento para o trabalho, utilizando o celular para se conectar ao rádio, com as redes móveis ligadas, mas sem interação com o celular. (2)	Horário de trabalho. Utiliza o celular conectado na rede sem fio do trabalho, com interação com o celular, mas não intensa. (3)	Horário de almoço. Possui uma interação maior com o celular, utilizando a internet através de rede móvel. (4)	Horário de trabalho. Utiliza o celular conectado na rede sem fio do trabalho, com interação com o celular, mas não intensa (3)	Horário da academia. O celular não é utilizado. (5)	Horário de deslocamento para a casa, utilizando o celular para se conectar ao rádio, com as redes móveis ligadas, mas sem interação com o celular. (2)	Horário em casa. Utiliza o celular de forma mais intensa, conectado através de redes sem fio. (6)
Finais de semana	Horário de dormir, permanecendo em casa, mas sem utilizar o celular. (1)	Horário de dormir, permanecendo em casa, mas sem utilizar o celular. (1)	Horário em casa. Utiliza o celular de forma mais intensa, conectado através de redes sem fio. (6)	Horário de almoço. Possui uma interação média com o celular, utilizando a internet através de rede móvel. (4)	Horário em casa. Utiliza o celular de forma mais intensa, conectado através de redes sem fio. (6)	Horário de sair. Utiliza o celular na rua, com internet móvel e GPS para navegação. (7)	Horário de sair. Utiliza o celular na rua, com internet móvel e GPS para navegação. (7)	Horário em casa. Utiliza o celular de forma mais intensa, conectado através de redes sem fio. (6)

APÊNDICE E – Termo de Consentimento Livre e Esclarecido - TCLE

TERMO DE CONSENTIMENTO LIVRE

Eu, **NOME DO VOLUNTÁRIO**, aluno do curso de **CURSO DO VOLUNTÁRIO**, da **UNIVERSIDADE DO VOLUNTÁRIO** declaro estar ciente que:

- A coleta de dados está relacionada a um projeto de mestrado.
- Instalarei em meu smartphone Android uma aplicação que coletará os dados de uso dos sensores do dispositivo que envolvem Wi-Fi, GPS, Bluetooth,etc..
- Durante o período de uma semana o aplicativo coletará meu uso em relação aos sensores. Durante esta semana deverei utilizar o dispositivo normalmente.
- Os dados pessoais não serão revelados ou mesmo considerados. Não será coletado sobre hipótese nenhuma senhas. Após a coleta será realizada uma avaliação, aplicando algoritmos de aprendizado de máquina para tentar obter conclusões sobre o consumo.
- Durante a análise não serão usados dados de maneira nominal. O usuário será sempre referenciado de maneira codificada: user1, user2, user3, etc. Nunca por seu nome.
- Caso qualquer informação seja divulgada em relatório ou publicação, isto será feito sob forma codificada, para que a minha identidade seja preservada e seja mantida a confidencialidade.

Eu concordo voluntariamente em participar da coleta.

(Assinatura do participante)

Sorocaba, de de 2013.

APÊNDICE F – Logo do MyBatRecommender

A Figura 23 apresenta o logo desenvolvido para representar o mecanismo proposto, MyBatRecommender.

Figura 23 – Logo do MyBatRecommender