Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Estratégias para a análise do consumo de energia de aplicações móveis

Isabela Cristina Oliveira Ribeiro

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Euler Horta Marinho

Dezembro, 2019
João Monlevade–MG

Isabela Cristina Oliveira Ribeiro

Estratégias para a análise do consumo de energia de aplicações móveis

Orientador: Euler Horta Marinho

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Dezembro de 2019

R484e Ribeiro, Isabela Cristina Oliveira.
Estratégias para a análise do consumo de energia de aplicações móveis [manuscrito] / Isabela Cristina Oliveira Ribeiro. - 2019.

45f.: il.: color; grafs; tabs.

Orientador: Prof. MSc. Euler Horta Marinho.

Monografia (Graduação). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Departamento de Computação e Sistemas de Informação.

1. Computação móvel. 2. Aplicativos móveis. 3. Androide (Recurso eletrônico). 4. Baterias elétricas. I. Marinho, Euler Horta. II. Universidade Federal de Ouro Preto. III. Titulo.

CDU: 004.9

Catalogação: ficha.sisbin@ufop.edu.br

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Isabela Cristina Oliveira Ribeiro

Estratégias para a análise do consumo de energia de aplicações móveis

Membros da banca

Euler Horta Marinho - Mestre - UFOP
Diego Zuquim Guimarães Garcia - Doutor - UFOP
Daniela Rodrigues Dias - Mestre - Doutoranda em Educação UFOP

Versão final
Aprovada em 12 de dezembro de 2019

De acordo

Professor (a) Orientador (a)
Euler Horta Marinho

Documento assinado eletronicamente por **Euler Horta Marinho**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 13/01/2020, às 11:36, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0031759** e o código CRC **A38C1A48**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: - www.ufop.br

Este trabalho é dedicado aos meus irmãos Priscila e Gabriel, por acreditarem no meu potencial, me fazendo forte todos os dias para vencer os obstáculos que surgem.

Agradecimentos

Agradeço primeiramente ao meu Deus por toda sabedoria, paciência e sustento concedidos durante essa caminhada de estudo.

Agradeço ao meu professor e orientador Euler Horta Marinho, por cada direção e conselho, compartilhando seus conhecimentos, para que eu pudesse alcançar essa vitória.

Agradeço aos meus pais, Jairo e Mariane, e aos meus irmãos, Priscila e Gabriel, por todo apoio, carinho e amor, sempre me incentivando a correr atrás dos meus sonhos e acreditando no meu potencial, muito obrigada por tudo o que fizeram e fazem por mim. Agradeço a Jefte por toda força nessa caminhada, me mostrando que eu posso ser melhor a cada dia, pela paciência, cuidado e presença em todos os anos da minha graduação, você foi essencial para que eu chegasse até aqui. Amo vocês!

Agradeço aos meus amigos, que conquistei durante essa caminhada acadêmica, por não me deixarem desistir, por acreditarem na minha capacidade, me incentivando a continuar estudando. Principalmente as minhas amigas, Flávia, Larissa e Tais, que faziam o meu dia melhor, me incentivavam a continuar e superar as dificuldades. Jamais me esquecerei de vocês.

Muito obrigada a todos que me ajudaram direta ou indiretamente a alcançar este sonho, vocês fazem parte desta vitória.

"Deus quer, o homem sonha, a obra nasce"

— Fernando Pessoa.

Resumo

A evolução dos *smartphones* trouxe uma variedade de recursos que são disponíveis para serem usufruídas pelas aplicações móveis, como consequência surgiu um importante requisito não funcional, a eficiência energética. O presente trabalho tem o objetivo de mostrar os problemas que são enfrentados para conquistar esse requisito, que apesar de ser difícil é possível, com auxilio de ferramentas e estratégias que serão apresentadas, de forma a preparar o desenvolvedor para lidar com o assunto. Foi realizado um questionário com desenvolvedores para entender como o consumo de energia é tratado por eles e conhecer suas expectativas sobre características que pretendem encontrar em ferramentas para apoia-los. Com base nos estudos, foi possível observar que falta conhecimento sobre a eficiência energética das aplicações por parte dos desenvolvedores e muitas vezes há falta de preocupação em relação a esse requisito não funcional. Espera-se que esse estudo possa conscientizar os desenvolvedores sobre a importância de utilizar recursos e realizar o consumo de bateria de forma eficiente.

Palavras-chaves: Consumo de bateria. Android. Eficiência energética. Aplicações móveis.

Abstract

The evolution of smartphones has brought a variety of features that are available for mobile applications to use as a consequence of an important non-functional requirement, energy savings. This paper aims to show the problems that are faced to achieve this requirement, which, although difficult, is possible with the aid of tools and statistics that may be lost in order to prepare the developer to deal with the problem. subject matter. A questionnaire was developed with developers to understand how their energy consumption is handled and to meet expectations about the features they want to find in the tools to support them. Based on the studies, it was possible to observe the lack of knowledge about the energy efficiency of applications by developers and there is often a lack of concern regarding a non-functional requirement. It is hoped that this study will make developers aware of the importance of using resources and performing battery consumption efficiently.

Key-words: battery consumption. android. energy efficiency. mobile applications.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ANATEL	Agência Nacional de Telecomunicações
API	*Application Programming Interface*
APK	*Android Application Pack*
CPU	*Central Processing Unit*
GATOR	*Program Analysis Toolkit For Android*
GPS	*Global Positioning System*
IDE	*Integral Development Environment*
JDK	*Java Development Kit*
LCD	*Liquid Crystal Display*
LTS	*Long-Term Support*
mAh	Miliampère-Hora
RAM	*Random Access Memory*
SDK	*Software Development Kit*
SMS	*Short Message Service*
TI	Tecnologia da Informação
USB	*Universal Serial Bus*
WTG	*Window Transition Graph*

Sumário

1 Introdução

O ano de 2007 foi marcado pela chegada do primeiro dispositivo móvel revolucionário. Apresentado por Steve Jobs, o *iPhone* mudou a história da Engenharia de *Software*. O novo dispositivo veio com um sistema chamado iOS, trouxe a interface multitoque e um navegador *web* com recursos bem projetados (CRUZ, 2019).

No ano seguinte, foi anunciada uma plataforma de distribuição digital que permitia adquirir e publicar aplicativos para dispositivos móveis, chamada *App Store*. Os usuários poderiam recuperar, comprar, baixar e instalar qualquer aplicativo em seu *smartphone* utilizando a nova plataforma de distribuição. Desde então, cada vez mais desenvolvedores eram atraídos a começar a criar seus aplicativos. De 500 aplicativos em 2008, a *App Store* cresceu para mais de 2,2 milhões de aplicativos em 2018. Outras plataformas de distribuição digital começaram a chegar no mercado, como a *Google Play*, voltada para a plataforma Android e a partir de 2018 ela era a maior loja de aplicativos com mais de 3,3 milhões de aplicativos voltados a plataforma Android. (CRUZ, 2019).

Segundo a ANATEL (Agência Nacional de Telecomunicações), o Brasil registrou 228,64 milhões de linhas móveis em operação no mês de maio de 2019 (ANATEL, 2019). O uso dos *smartphones* tem crescido de forma exponencial, isso mudou o campo da Computação. O Android é a plataforma dominante nesse espaço e cada vez mais desenvolvedores estão entrando nesse ramo da Computação (WU; YANG; ROUNTEV, 2016). A alta demanda dos usuários incentiva o desenvolvedor a oferecer aplicativos eficientes e com riqueza de funções, estas muitas vezes utilizam serviços que demandam alto poder computacional e, consequentemente, um grande gasto energético, como: o uso de GPS, Internet e *Bluetooth*. Estima-se que nas duas últimas décadas o orçamento de energia do processador aumentou devido ao alto uso de recursos como os citados acima (AHMAD et al., 2017).

O consumo de energia é um tema bastante discutido nas comunidades de desenvolvedores Android (STACKOVERFLOW, 2018). Um desenvolvimento inadequado gera um alto custo de energia e traz várias reclamações de usuários (HU et al., 2018). Como a capacidade de bateria dos *smartphones* é pequena, devido a restrição de tamanho e peso, é importante realizar o gerenciamento do consumo de energia visando a eficiência energética. (CARROLL; HEISER, 2010).

Este trabalho é continuação do trabalho "Levantamento de Abordagens para a análise de Consumo de Energia de Aplicações Móveis: Um Estudo de Caso do Laboratório IMOBILIS"desenvolvido pelo aluno Michel Wagner Ferreira. No referente trabalho, o aluno buscou ferramentas que pudessem auxiliar os desenvolvedores do laboratório a analisar

o consumo de bateria de suas aplicações. Assim o trabalho presente tem como objetivo a identificação de estratégias e ferramentas de suporte que possam ser utilizadas por desenvolvedores de aplicações do Android, na criação de aplicações com consumo eficiente de energia. Portanto, o escopo do trabalho não será restringido a um contexto específico de desenvolvimento.

1.1 O problema de pesquisa

O crescimento do uso de *smartphones* só tem aumentado. Segundo a GSMA, uma empresa de análise que representa o setor mundial de comunicações móveis, em todo o mundo 5,1 bilhões de pessoas usam algum tipo de aparelho móvel. No terceiro trimestre de 2017, havia 690 milhões de conexões móveis na América Latina. Dentre essas conexões, 234,6 milhões eram do Brasil (GSMA, 2017). Em uma pesquisa desenvolvida pela Delloite, os *smartphones* foram os aparelhos mais utilizados por 92% dos brasileiros (DELLOITE, 2019). Segundo uma pesquisa chamada Juventude Conectada, realizada pela Fundação Telefônica, 42% dos jovens afirmam que o aparelho que mais utilizam para acessar a Internet é o celular. Também segundo esta pesquisa o celular tem um papel importante na construção da identidade dos jovens (FUNDACAO, 2014). A pesquisa a *E.life* mostrou que o celular substituiu várias outras tecnologias, como o *videogame* portátil; em 2016 70% dos brasileiros entrevistados utilizavam o celular para jogos, também substituiu o DVD e os jornais impressos, 76% liam as notícias pelo celular (ELIFE, 2016).

Um dos requisitos mais cobiçados é a duração da bateria, o que traz um grande desafio para os desenvolvedores, já que os aplicativos mais utilizados são aqueles com maior gasto computacional, como jogos *multiplayers* e os que dependem de localização e Internet; segundo a Pesquisa Gamer Brasil, 84% dos brasileiros utilizaram os *smartphones* para jogos em 2018, sendo essa a plataforma de preferência para tal atividade (RESEARCH; SIOUX, 2018).

Pouca atenção foi dada a criação de técnicas, ferramentas e processos que auxiliassem e ensinassem os desenvolvedores a utilizar os recursos energéticos do *smartphone* (PINTO; CASTOR, 2017). Alguns desenvolvedores se preocupam mais com os requisitos funcionais e deixam alguns não funcionais de lado, como a quantidade de energia que o aplicativo gastará. Eles frequentemente têm a consciência da necessidade da eficiência energética, mas muitas vezes não conhecem ferramentas que dão apoio para tratar esse tipo de defeito (FERREIRA, 2017). A maioria dos problemas com relação a energia podem ser reduzidos em dois: a falta de conhecimento e a falta de ferramentas (PINTO; CASTOR, 2017).

Com relação a otimização do consumo de energia, os desenvolvedores, na maioria das vezes, precisam recorrer a comunidades de perguntas e respostas ou vídeos no *YouTube*, isso porque não há informações sobre o assunto nos livros didáticos. Os autores Pinto,

Soares-Neto e Castor (2015) pesquisaram artigos publicados em um período de 10 anos, nas plataformas de Engenharia de *Software*. Somente 20 trabalhos foram encontrados com a palavra "energia"em seus títulos ou resumos e nenhum deles foi publicado antes de 2012. Como consequência dessa carência nos livros didáticos, os desenvolvedores têm dificuldades em criar e/ou manter aplicações que economizam energia. Essa falta de recursos está relacionada com a falta de ferramentas que auxiliam no desenvolvimento de códigos eficientes em relação a energia (PINTO; CASTOR, 2017).

Um dos assuntos mais falados ultimamente é a sustentabilidade. Dentro desse assunto, um dos fatores mais vistos pelas organizações e sociedade é o consumo de energia elétrica. Quanto mais recursos de computação for utilizado, mais bateria é gasta, logo mais energia elétrica é consumida e mais produção da mesma se faz necessária e os meios de produção de energia elétrica agridem muito o meio ambiente (DEVMEDIA, 2013). Por isso é necessário fazer um gerenciamento do consumo de bateria das aplicações, tornando-as um *software* verde. Como os recursos de energia são escassos nos *smartphones*, é de total importância que os desenvolvedores conheçam sobre a eficiência enérgica e trabalhem para que consigam alcançá-la (FERREIRA, 2017). Algumas ferramentas podem ser de grande ajuda, principalmente aquelas que conseguem fazer a análise do código em tempo real, mostrando defeitos que podem gerar alto consumo de energia.

1.2 Objetivos

O objetivo geral deste trabalho é realizar a identificação de estratégias e ferramentas para a análise do consumo de energia de aplicações móveis, buscando entender as necessidades que os desenvolvedores têm para criar aplicações verde, ou seja, que economizam energia.

Este trabalho possui aos seguintes objetivos específicos:

- Pesquisar ferramentas e estratégias para desenvolver aplicativos para a plataforma Android com consumo eficiente de energia.
- Pesquisar as necessidades dos desenvolvedores quanto a criar aplicações móveis com consumo eficiente de energia.

1.3 Metodologia

Os passos para execução deste trabalho são assim definidos:

- Revisão Bibliográfica: estudar as possíveis causas do consumo de energia das aplicações.

- Pesquisa: aplicação de questionário para entender as preocupações e necessidades dos desenvolvedores quanto ao consumo de energia de suas aplicações
- Pesquisa: pesquisar ferramentas que auxiliam os desenvolvedores a criar aplicações para a Plataforma Android com consumo eficiente de energia.
- Análise e discussão: analisar as possíveis soluções e fazer comparações, discutindo os resultados obtidos.

1.4 Organização do trabalho

O restante deste trabalho é organizado como se segue. O Capítulo 2 apresenta a revisão bibliográfica, com alguns temas que serão abordados, como TI verde e sistemas de informação. O Capítulo 3 descreve toda a pesquisa realizada, apresenta os resultados obtidos e as análises realizadas. O Capítulo 4 apresenta as considerações finais deste trabalho. Por fim, o Apêndice A traz o questionário aplicado para pesquisa com alguns desenvolvedores.

2 Revisão bibliográfica

Este Capítulo apresenta uma revisão bibliográfica, bem como trabalhos correlatos.

2.1 Sistemas de Informação

Um sistema de informação é um conjunto de componentes que se relacionam, podendo coletar, armazenar, processar e distribuir informações que podem auxiliar na tomada de decisão da organização, ajudar gerentes e trabalhadores a analisar problemas e criar produtos (LAUDON; LAUDON, 2010).

Os sistemas de informação podem conter informações sobre todas as partes da empresa, ou seja, dados apresentados de forma útil para as pessoas. Eles têm três atividades principais: entrada ou coleta de dados, o processamentos desses dados, tornando-os em informação e a saída, que mostra as informações de forma visual para que possam ser utilizadas por quem necessite delas. Os sistemas de informação usam *feedback* como ajuda para avaliar ou corrigir as entradas (LAUDON; LAUDON, 2010).

Dentro desse sistema, os computadores armazenam e processam as informações e os *softwares* são um conjunto de instruções operacionais detalhadas e pré-programadas que controlam o processamento por computador (LAUDON; LAUDON, 2010). No caso dos dispositivos móveis, as aplicações seriam os softwares que controlam os processamentos dos dispositivos.

2.2 Aplicações Móveis

A chegada das redes 3G e 4G trouxe melhorias consideráveis para a banda larga móvel, permitindo que conteúdos e serviços sejam oferecidos e acessados de qualquer lugar. Aproveitando esse fator, o chamado ecossistema de *software*, criou as lojas de aplicativos móveis (CUADRADO; DUEñAS, 2012). Com as lojas o mercado cresceu cada vez mais e atraiu vários desenvolvedores para a área. As vendas de aplicativos presentes nas lojas *Google Play e App Store* somaram um faturamento de US$ 101 bilhões no ano de 2018 (MOBILE, 2016).

Aplicações móveis são *softwares* desenvolvidos para dispositivos móveis, como *tablets* e *smartphones*. Várias empresas estão buscando incorporar as aplicações móveis no seu dia-a-dia com a finalidade de agilizar os negócios, já que com os dispositivos móveis, elas podem ser acessadas de qualquer lugar e a qualquer momento, sendo assim, as aplicações podem sincronizar as informações diretamente do servidor da organização e

estarem conectadas a todo tempo (LECHETA, 2013).

Os aplicativos móveis exigem requisitos além dos que os *softwares* comum exigem, como: possibilidade de interação com outros aplicativos, acesso a sensores e recursos como: microfone, GPS, segurança e interface com o usuário (WASSERMAN, 2010). As aplicações móveis podem ser desenvolvidas para as plataformas iOS e Android e ainda para a *Web*. A escolha da linguagem de programação depende da escolha da plataforma para qual a aplicação será desenvolvida, por exemplo, caso seja para Android a linguagem comumente utilizada é o Java, caso seja iOS a linguagem seria *Swift*. Um aplicativo pode ser nativo (compatível somente com um tipo sistema), um *Web App* (sites que conseguem se adaptar aos dispositivos móveis) ou ainda híbrido (compatível com um ou mais tipos de sistemas) (CHARLAND; LEROUX, 2011).

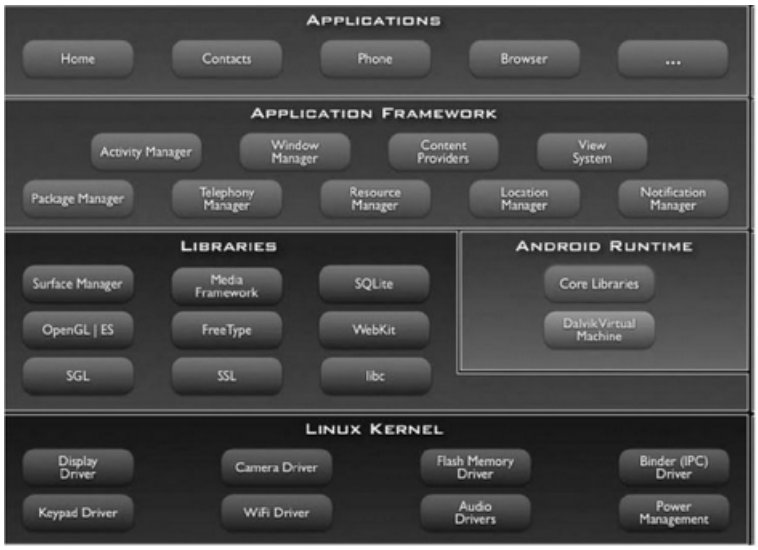
2.3 Android

Criado pela Google, com ajuda de outras empresas, o Android gerou impacto ao ser anunciado, atraindo atenção. Ele é uma plataforma para dispositivos móveis e oferece um sistema operacional, *middleware*, algumas aplicações já instaladas e uma interface com usuário. Sendo uma tecnologia *open source*, ele se baseia no sistema Linux e sua intenção é possibilitar que desenvolvedores criem aplicações móveis que possam usufruir de todos os recursos que os dispositivos oferecem. Para desenvolver aplicações para o Android é necessário o software Android SDK, que contém um emulador que simula celulares, algumas ferramentas e uma API para a linguagem Java (PEREIRA; SILVA, 2009).

A arquitetura do Android é separada em camadas, como mostra a Figura 1. Na camada *Applications* estão todos os aplicativos fundamentais desenvolvidos pela comunidade Android, como agenda, calendário, programas de SMS e gerenciador de contatos. Na camada *Framework* se encontra as APIs e recursos disponíveis para os aplicativos, como o gerenciador de notificações, de pacotes e de atividades que controlam todo ciclo de vida das aplicações e o acesso a navegação entre elas, os elementos dessa camada são: *Activity Manager* que gerencia o ciclo de vida das *Activities*, *Package Manager* utilizado para ler as informações contidas nos APKs, *Window Manager* gerencia as apresentações de janelas, *Content Providers* permite o compartilhamento de dados entre dispositivos, *View System* trabalha com os elementos gráficos como, botões e *layouts*, é nesta camada que também se encontra elementos como *Location Service, Bluetooth Service, Wi-fi Service, USB Service e Sensor Service*, esses são os elementos que disponibilizam os recursos mais utilizados (GPS, *Bluetooth, Wi-fi*, Sensores) pelas aplicações. Na camada *Libraries* estão o conjunto de bibliotecas utilizadas pelo sistema, incluindo aquelas de multimídia, funções para os navegadores, funções de gráficos, entre outros. Na camada de *Android Runtime* está disponível uma instância da máquina virtual *Dalvik* criada para

cada aplicação que é executada no Android, ela tem um melhor desempenho e maior integração com *hardwares* da nova geração. A *Dalvik* tem a finalidade de funcionar em sistemas de baixa frequência de CPU, pouca memória RAM e sistemas operacionais sem espaço de *Swap* e é otimizada para consumo mínimo de bateria. Na camada *Linux Kernel* estão os serviços centrais do sistema como, segurança, gestão de memória e processos. É nessa camada que está o gerenciador de energia, onde o aplicativo faz a requisição de energia para o *driver* (PEREIRA; SILVA, 2009).

Figura 1 – Estrutura da Arquitetura Android.



Fonte: Pereira e Silva (2009)

Uma das classes mais importantes do Android é a *Activity*. A *Activity* representa uma tela da aplicação, controla os eventos de tela e define qual *View* desenhará a interface gráfica. A *Activity* é responsável por tratar, por exemplo, quando um usuário clica em algum botão ou quando um item de menu é escolhido (LECHETA, 2013).

2.4 TI e *Software* Verde

Com o crescimento avançado da Tecnologia da Informação (TI) surgiu uma preocupação quanto a questão ambiental devido ao mau uso e ao descarte dos equipamentos eletrônicos. Em 2007 a área de TI representava a terceira maior fonte de consumo de energia, levando em consideração os gastos necessários para manter sua infraestrutura (servidores, computadores, entre outros). No Brasil foram vendidos mais de 13,7 milhões de computadores e a preocupação sobre o descarte aumentou, já que boa parte deles usavam materiais como chumbo e mercúrio em suas fabricações. (LUNARDI; SIMõES; FRIO, 2012).

Visando a sustentabilidade algumas empresas adotaram medidas, como a Dell que fez um planejamento sustentável para ser cumprido até 2020, conseguindo reutilizar plásticos de produtos eletrônicos recicláveis para produzir novos e no ano de 2012 reduziram a intensidade de energia dos servidores em quase 78% (DELL, 2019).

A medida que a tecnologia de *hardware* crescia a influência do *software* no consumo de energia aumentava. Para prover energia aos computadores pessoais, aos *Data Centers* e telefones fixos e móveis foram necessários 8% de toda energia produzida no mundo em 2005 (SCHULZ; SILVA, 2012). Um estudo realizado pelos autores Ardito et al. (2015) mostrou que o consumo de energia de três servidores executando tarefas distintas, pode aumentar até 40%. Analisando diferentes tipos de computadores e tecnologias os autores descobriram que dependendo dos *softwares* utilizados o consumo de energia pode aumentar até 20%. Analisaram também o consumo de energia em alguns dispositivos móveis com a Plataforma Android e perceberam que as aplicações afetam significativamente o consumo de energia dos dispositivos. Por isso, há a necessidade de se preocupar também com o *software* verde, pois o impacto do *software* no consumo de energia é relevante. Usar a energia dos dispositivos de forma consciente pode diminuir o consumo de energia elétrica (ARDITO et al., 2015).

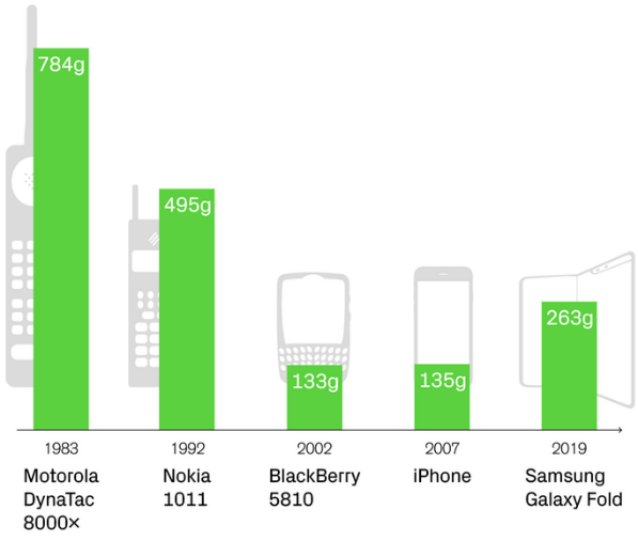
2.5 Evolução dos *Smartphones*

Em 1983 foi lançado o primeiro celular, o Motorola DynaTAC 8000X. Em 1992 o primeiro SMS foi enviado e dois anos depois chegava ao mercado o IBM Simon. Considerado o primeiro *smartphone* o IBM Simon tinha recursos de telefonia, aplicativos e serviços como, *e-mail*, calendário, calculadora, bloco de notas e outros. Era oferecido pela empresa BellSouth e disponível nas regiões sudeste e sul dos Estados Unidos da América. Foram vendidas 50.000 unidades e durou seis meses no mercado. Ele abriu um caminho para a chegada de celulares que ofereceriam funcionalidades semelhantes a de um computador. Em 1996 surgiu o Nokia 9000, primeiro celular com acesso a Internet (ADAM, 2016). Em 2000 surgiram várias inovações no ramo de celulares, o T36 Ericsson sendo o primeiro com *bluetooth*, Samsung Uproar o primeiro com função mp3, o Sharp J-Sh04 primeiro com câmera, o Ericsson R380 foi o primeiro a utilizar o sistema Symbian, sistema operacional mais utilizado antes da chegada do Android e do iOS. O ano de 2007 foi revolucionado pela chegada do iPhone, com poucos botões e sem teclado, foi o primeiro com uma loja de aplicativos (TECNOLOGIA, 2013).

Após a criação do primeiro celular, que tinha 33cm e pesava 794g a tendência era trazer celulares cada vez menores, mais leves e potentes (TECNOLOGIA, 2013). A Figura 2 mostra os celulares cada vez mais leves ao passar dos anos. No início, as telas mostravam apenas informações básicas, como números. À medida que os *smartphones* evoluíam, as

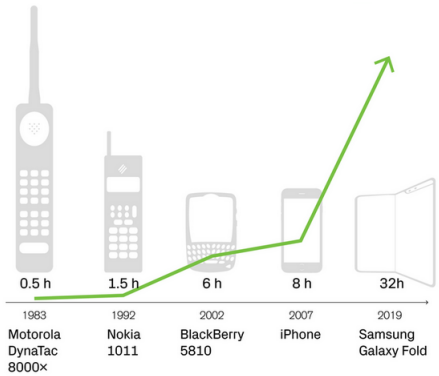
telas ficaram maiores e com resoluções melhores. O primeiro iPhone, lançado em 2007, tinha uma tela de 3,5 polegadas, com sensibilidade ao toque. Ela mostrava todas as funções disponíveis e servia como controlador do dispositivo. Desde então, as telas evoluíram cada vez mais, ficando maiores e com resoluções melhores (TECMUNDO, 2014). Atualmente a tendência é *smartphones* grandes, sem botões, todas as funções são acessíveis pela tela, que ocupa todo o espaço. A cada lançamento, os *smartphones* aumentaram de tamanho, ficaram mais potentes, com um número maior de recursos cada vez melhores, mas para que isso fosse possível houve a necessidade de trazer baterias com maior capacidade energética (MOREIRA, 2017). A Figura 3 mostra a evolução das baterias de acordo com a evolução das telas nos *smartphones*.

Figura 2 – Peso dos celulares durante sua evolução.



Fonte: CreditSuisse (2019)

Figura 3 – Evolução da bateria dos *smartphones* em relação a evolução das telas.

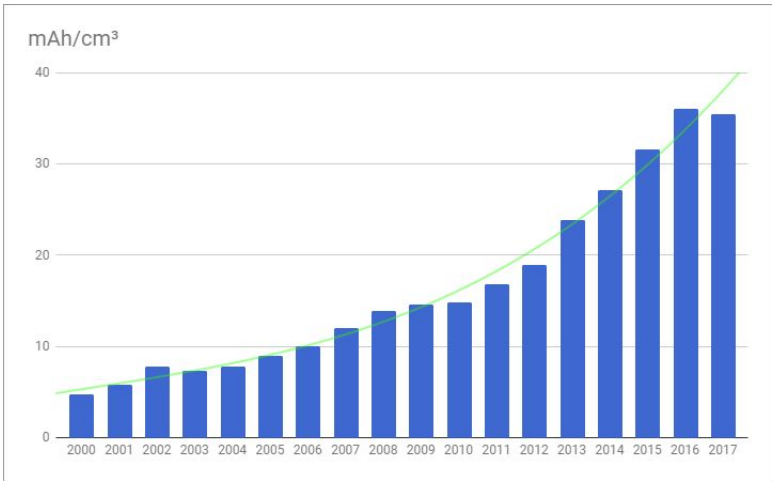


Fonte: CreditSuisse (2019)

Além da evolução das telas, o iPhone trouxe a demanda por aplicativos, sendo o primeiro *smartphone* com uma loja de aplicativos, trazendo a possibilidade de disponibilizar

e adquirir aplicações com diversas funções. Surgiu então a necessidade de aumentar o processamento dos *smartphones* e de aumentar ainda mais a capacidade da bateria. As baterias foram criadas no início dos anos 90 e desde então, há uma dificuldade em aumentar a sua capacidade devido a restrição de tamanho. No final dos anos 90, um celular Nokia tinha uma bateria de 900mAh e era suficiente para manter o celular ligado por vários dias. Atualmente existem baterias de até 3.000mAh, mas dificilmente os aparelhos chegam a dois dias sem precisar de uma nova carga. Isso se dá pelo fato de que os processadores se tornaram muito mais potentes, as telas com resoluções melhores e os *smatphones* com um número maior de funções. A Figura 4 mostra a evolução da capacidade energética das baterias entre os anos de 2000 e 2017 (SEGURO, 2018).

Figura 4 – Aumento da capacidade das baterias.



Fonte: Moreira (2017)

2.6 Consumo de Energia

Com o aumento do uso de dispositivos móveis o consumo de energia se tornou um dos gargalos da computação, tanto fabricantes como os usuários estão preocupados com a vida útil e o consumo da bateria (COUTO et al., 2016). Como esses dispositivos têm tamanho e peso restritos, a capacidade da bateria é restringida, por isso a eficiência energética se torna ainda mais importante nesses tipos de dispositivos, sendo necessário realizar um bom gerenciamento do consumo de energia (CARROLL; HEISER, 2010).

Considerando o consumo de energia é importante considerar: o sistema de *software* em execução, o *hardware*, o contexto e o período de tempo. Atualmente qualquer *smartphone* tem suporte para Wi-Fi, 3G e 4G. Em 2017, um estudo mostrou que o 3G pode consumir 1,7x mais energia do celular, enquanto o 4G consome 1,3x mais energia que o 3G, executando a mesma tarefa, na mesma plataforma. O contexto é outro fator importante, pois, o *software* pode consumir mais energia dependendo dos cálculos que realiza e da forma como lida

com as estruturas de dados. Em relação ao tempo, os desenvolvedores podem achar que reduzindo o tempo de execução, diminuem o consumo de bateria, mas isso pode aumentar o número de ciclos da CPU, fazendo com que o consumo de energia aumente (PINTO; CASTOR, 2017). A bateria do dispositivo pode durar até uma hora a mais, caso as aplicações sejam desenvolvidas com práticas de consciência energética (CRUZ, 2019).

Embora a estratégia de deixar a otimização de energia para as camadas inferiores do Android, como para a camada *Android Runtime* seja bem sucedida, pode-se melhorar o consumo capacitando e incentivando os desenvolvedores a participar desse processo. A eficiência energética é um dos principais fatores para o usuário adquirir um aplicativo. Se houver um consumo alto, o usuário pode fazer más avaliações e não recomendar a aplicação (PINTO; CASTOR, 2017).

3 Desenvolvimento e resultados

Este Capítulo descreve a pesquisa realizada com desenvolvedores sobre a preocupação do consumo de energia de suas aplicações, estratégias que auxiliam o consumo eficiente de energia, relata as ferramentas que podem auxiliar os desenvolvedores a analisarem suas aplicações de tal forma que economize energia. Ao final, é discutido os resultados obtidos e comparado com os resultados do trabalho de Ferreira (2017).

3.1 Pesquisa com desenvolvedores

Com a finalidade de compreender o que os desenvolvedores de aplicações móveis pensam e conhecem sobre a importância do consumo de energia de forma eficiente, utilizou-se um questionário que é mostrado no Apêndice A.

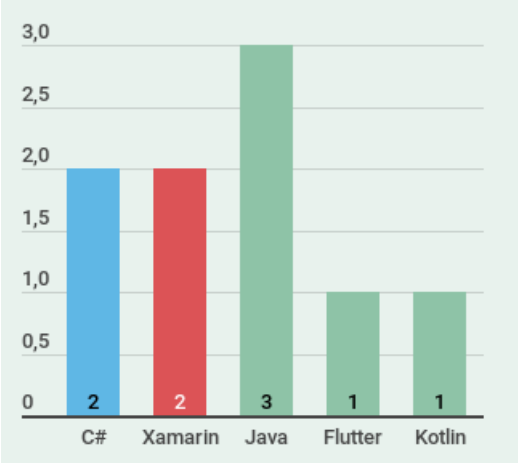
Diferente da pesquisa desenvolvida por Ferreira (2017), que tinha um escopo restringido aos desenvolvedores do laboratório iMobilis, o trabalho aqui presente tem um escopo mais abrangente, pois visa entender as preocupações e necessidades dos desenvolvedores de uma forma mais geral. Sendo assim foram selecionados seis desenvolvedores de aplicações móveis, que fazem parte do Laboratório iMobilis, onde foi aplicado o estudo de caso relatado por Ferreira (2017), dois professores e quatro alunos do campus ICEA/UFOP de João Monlevade e dois profissionais que atuam na área. O questionário foi desenvolvido através da Plataforma Formulários Google, e contou com respostas de seis desenvolvedores, com experiência de 9 meses a 6 anos. Oito dos quatorze desenvolvedores contatados preferiram não participar da pesquisa. O questionário apresentado no presente trabalho têm como foco a preocupação dos desenvolvedores quanto o consumo energético de suas aplicações, diferente da pesquisa realizada pelo Ferreira (2017), onde o questionário foi aplicado somente com desenvolvedores do laboratório iMolibis a fim de descobrir como eles lidavam com os *bugs* de energia de suas aplicações.

O gráfico da Figura 5 é referente a questão 3 do questionário, que tem como objetivo identificar a linguagem de programação mais utilizada pelos desenvolvedores entrevistados. Diferentes linguagens de programação exigem diferentes práticas de codificação que lidam com a eficiência energética. A linguagem escolhida tem um impacto considerável na qualidade do *software*, por exemplo, linguagens que fazem a utilização de *logs* podem consumir mais energia do que as que não utilizam, já que o uso intenso, como taxas acima de uma mensagem por segundo, pode gerar um consumo alto de energia (CRUZ, 2019). Observar-se que 33% utiliza a Linguagem Java, sendo a maioria dos desenvolvedores.

Realizar testes nos *softwares* é uma parte importante do ciclo de vida de criação

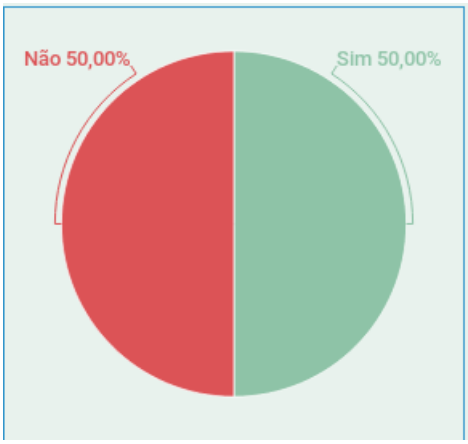
deles, visto que, os testes ajudam na identificação de erros do sistema antes que sejam enviados ao usuário final. Os aplicativos móveis ainda são poucos testados e isso reflete em sua qualidade (CRUZ, 2019). A questão 4 tem o objetivo de entender a preocupação dos desenvolvedores com a qualidade de suas aplicações, visto que o consumo de bateria é um quesito de qualidade. Pela Figura 6 pode-se ver que somente 50% dos desenvolvedores entrevistados realiza a avaliação da qualidade em suas aplicações.

Figura 5 – Pergunta 3, referente a linguagem de programação utilizada.



Fonte: Elaborado pela autora (2019)

Figura 6 – Pergunta 4, referente a realização da avaliação da qualidade.

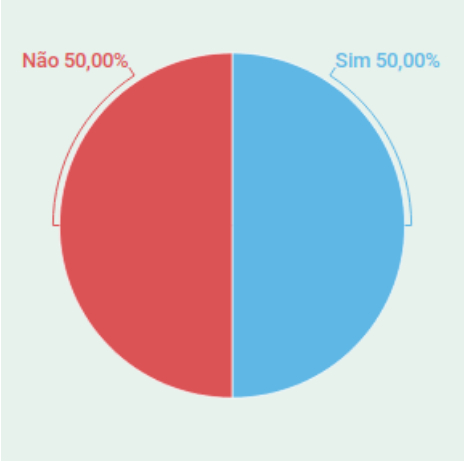


Fonte: Elaborado pela autora

Com o fato de que usuários têm dado importância ao consumo de bateria, é de suma importância que os desenvolvedores conheçam e tratem do assunto. Dessa forma, a questão 5 tem o objetivo de estabelecer o nível de conhecimento dos entrevistados quanto a eficiência energética. 50% informou que não teve contato com o assunto, portanto, não o conhece. Outros 50% tiveram contato por meio de artigos na Internet, quando precisaram utilizar em suas aplicações o sensor de geolocalização presente no *smartphone*.

O *framework* escolhido também influencia no consumo energético da aplicação. Um *framework* pode gastar mais energia do que um outro, realizando as mesmas tarefas. Isso torna-o menos eficiente e traz ao desenvolvedor a necessidade de pesquisar sobre o desempenho energético do mesmo (CRUZ, 2019). Portanto, a questão 6 tem o intuito de identificar se os desenvolvedores se preocupam em pesquisar, sobre o desempenho energético oferecido pelos *frameworks* escolhidos para o desenvolvimento. Por meio do gráfico apresentado na Figura 7 pode-se notar que 50% dos entrevistados inclui o desempenho energético nas pesquisas, enquanto 50% não leva em consideração esse quesito.

Figura 7 – Pergunta 6, referente a eficiência energética dos *frameworks*.



Fonte: Elaborado pela autora

Existem algumas ferramentas que auxiliam os desenvolvedores a encontrar defeitos de código que possam gerar um consumo ineficiente de energia. A questão 7 tem o objetivo de identificar se os desenvolvedores utilizam ou conhecem algum tipo de ferramenta que oferece essa avaliação. 50% dos entrevistados responderam que não conhecem ferramentas com essa funcionalidade e 50% conhece e já a utilizou em suas criações.

A maioria dos defeitos de energia se originam em defeitos de código (HU et al., 2018), por isso, a questão 8 foi aplicada para avaliar a preocupação dos desenvolvedores quanto a corrigir defeitos de código que geram o consumo alto de energia. Como resposta, 50% disseram que não se preocupam e 50% se preocupam, principalmente com a eficiência do uso de recursos computacionais do código desenvolvido. Alguns desses defeitos de código são pelo fato de que os desenvolvedores muitas vezes solicitam recursos que consomem muita energia e, ao terminarem suas funções, não solicitam o cancelamento do recurso (HU et al., 2018). A questão 9 tem o propósito de avaliar se os desenvolvedores utilizam os métodos oferecidos pelas linguagens de forma correta, a fim de que, possam evitar defeitos, como os que consomem energia. As respostas mostram que 33,33% dos entrevistados não se preocupam em consultar a documentação para utilizar os métodos de forma correta. Enquanto 66,67% pesquisam a forma correta de utilizar os métodos.

A fim de descobrir quais características os desenvolvedores julgam importantes estar presentes nas ferramentas, aplicou-se a questão 10. Os entrevistados relataram que tais características são importantes:

1. Identificação dos trechos de código que poderiam ser reescritos de uma forma mais eficiente de forma a melhorar o uso dos recursos disponíveis.
2. Sugestões de como resolver os defeitos detectados.
3. Notificações sobre possíveis problemas em tempo real.
4. Mostrar o tempo de processamento de uma função, com gráficos comparativos.
5. Lista com as linhas de código que apresentam defeitos
6. Filtros para pesquisar por tempo, memória, processamento ou nome de funções.

Com o questionário foi possível detectar que os desenvolvedores que conhecem sobre a eficiência energética se preocupam com a realização de testes em suas aplicações, para avaliar a qualidade. Também se preocupam em pesquisar sobre o desempenho energético dos *frameworks* que utilizam e em corrigir os defeitos do código que possam gerar um consumo alto de energia, cuidando para utilizar os métodos da linguagem de forma correta, bem como conhecem ou utilizam de ferramentas para auxiliá-los. É importante ponderar que, a falta de realização da avaliação da qualidade por 50% dos entrevistados sugere que precisa haver uma conscientização maior sobre a importância dos testes.

3.2 Estratégias para minimizar o consumo de energia

Como consequência de seu tamanho e peso restrito, o *smartphone* tem a capacidade da bateria e da CPU restringidas e a quantidade baixa de Memória RAM. Com a crescente demanda de usuários de *smartphones*, os desenvolvedores se sentiram motivados a criarem aplicações com uma variedade de recursos, para enriquecer a experiência do usuário. Aplicativos que utilizam vídeo sob demanda, jogos *multiplayer*, GPS e sensores estão dentre aqueles que mais consomem bateria, pois aumentam a demanda de energia dos processadores. Nas últimas décadas o orçamento de energia dos processadores aumentou devido ao uso desses e outros recursos (AHMAD et al., 2017).

A falta de livros didáticos que tratam sobre o assunto e a falta de ferramentas que ajudam a gerenciar o consumo de bateria geram a falta de conhecimento dos desenvolvedores quanto a necessidade de se preocupar com o consumo de bateria de suas aplicações e esse requisito tão importante acaba não fazendo parte da cultura dos desenvolvedores. A falta de conhecimento é vista quando soluções são apresentadas como universais, mas na verdade funcionam somente quando aplicada a um certo contexto. Como nem sempre a

otimização de desempenho ajuda na economia de energia, as diretrizes de desempenho nem sempre são úteis (PINTO; CASTOR, 2017). Os efeitos da falta de conhecimento podem ser observados pelos dados coletados no questionário, onde 50% dos entrevistados não faziam avaliação da qualidade das aplicações e não se preocupavam em corrigir os defeitos que geravam consumo de energia alto. A falta de livros didáticos é refletido quando os desenvolvedores que conhecem sobre o assunto, mencionaram o contato por meio de artigos na Internet.

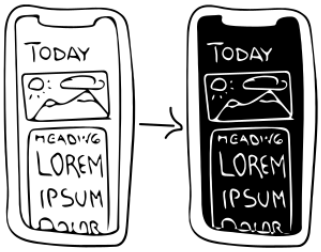
O consumo de energia de uma aplicação é estimada com base no consumo de energia dos componentes que ela utiliza. A taxa do consumo de energia de cada componente é diferente e varia de acordo com o estado de execução durante a execução da aplicação. Por exemplo, o consumo de energia da CPU depende da sua frequência durante a execução, da mesma forma, o consumo de energia do Wi-Fi depende da quantidade de dados transferidos (AHMAD et al., 2017). O modelo de consumo energético precisa levar em consideração os principais componentes de *hardwares* e suas características. Na CPU existe um coeficiente de consumo diferente para cada frequência utilizada pelo processador, esse coeficiente é calculado multiplicando o coeficiente da frequência de uso pela porcentagem da utilização. O consumo da tela LCD depende do nível do brilho da tela. O GPS consome energia dependendo do seu estado (ativo, inativo, desligado), ele possui um coeficiente para quando está ativo e outro para quando está inativo. No componente Wi-Fi o consumo é constante quando seu estado está em baixa potência. Caso a potência seja alta, então o consumo depende do número de dados transferidos (COUTO et al., 2016).

Um dos maiores causadores de gasto energético são os defeitos de código e recursos rodando em segundo plano. Os desenvolvedores solicitam acesso a recursos que consomem muita energia e ao finalizar o uso de tais, não cancelam a operação, fazendo com que os recursos fiquem ativos mesmo quando não há necessidade (HU et al., 2018). Por exemplo, no Android, para obter informações de localização é utilizada uma Interface chamada *location listener*. Através dela, devem ser implementados métodos de retorno de chamada que são acionados quando mudanças relevantes acontecem. Essa Interface pode e deve ser cancelada ao término da atividade para a qual foi solicitada. Para isso, utiliza-se um método chamado *removeUpdates* passando o *listener* que será removido como parâmetro. O padrão de diretrizes para a criação de aplicativos que utilizam GPS, alertam os desenvolvedores para sempre estarem atentos ao fato de que utilizar o *listener* de localização por um longo período de tempo consome muita energia da bateria (WU; YANG; ROUNTEV, 2016). Na pesquisa aplicada pode-se notar que os desenvolvedores que tiveram contato com o assunto de eficiência energética, o fizeram pois se preocuparam com o consumo de bateria do recurso de geolocalização de suas aplicações. Esse fato mostra que o recurso de GPS é um dos que mais consomem energia do *smartphone*, sendo perceptível a desenvolvedores e usuários.

Abaixo são mostrados alguns itens que podem causar o consumo alto de energia das aplicações, bem como estratégias para tratar tais itens, segundo o autor Cruz (2019, p.128-135):

1. **Tela**

Figura 8 – Consumo de energia da tela.



Fonte: Cruz (2019)

- **Causa:** A tela é um dos componentes que mais consomem energia, aplicações que fazem muito o uso dela (ex: aplicativos para leitura) podem diminuir a vida útil da bateria. As telas Amoled consomem energia conforme o número de *pixels* ligados, por isso, o consumo depende das cores que estão sendo exibidas, enquanto nas telas LCD's o consumo depende da intensidade da iluminação.
- **Solução:** Utilize cores de fundo escuras, elas economizam energia. Pode-se permitir que o usuário escolha entre as cores claras e escuras ou utilizar um gatilho para acionar o tema escuro (ex: quando a bateria estiver fraca). Permitir a interação com interfaces alterativas, como áudio, diminuindo o uso da tela.

2. **Transferência de dados**

- **Causa:** Aplicações que fazem transferência de dados, seja coletar ou enviar, podem consumir energia de forma ineficiente, caso o aplicativo tente fazer conexão várias vezes quando o recurso não está disponível.
- **Solução:** Aumentar o intervalo de tempo para fazer tentativas após as falhas de conexão. Esse intervalo pode ser redefinido após uma mudança de contexto (ex: *status* de rede). Reduzir o tempo de transmissão dos dados.

3. **Tarefas que não afetam diretamente o usuário**

- **Causa:** Algumas tarefas têm resultados que não são visíveis para o usuário ou não são relevantes. Quando os aplicativos passam para o segundo plano essas tarefas ainda ficam executando, assim, o *smartphone* utiliza recursos desnecessariamente.

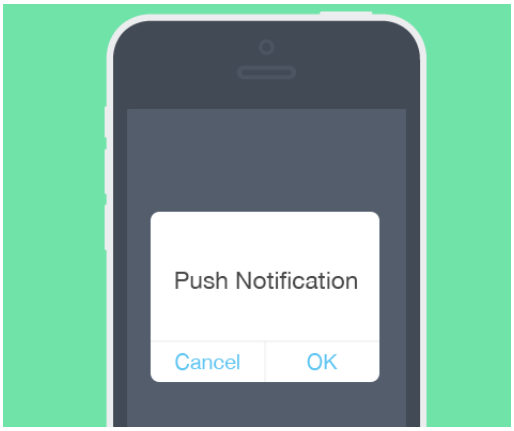
- **Solução:** Selecionar um conjunto de dados que serão apresentados ao usuário, habilitar tarefas que serão necessárias e importantes.

4. **Recursos Inativos**

   - **Causa:** As aplicações podem utilizar vários recursos que podem ser abertos antes do uso, enquanto esses recursos estiverem ativos estarão prontos a responderem solicitações, sendo assim, consomem muita energia.
   - **Solução:** Verificar se existem recursos inativos e, então, fechá-los.

5. **Atualizações**

Figura 9 – Notificações *push*



Fonte: InfiniteLoop (2017)

   - **Causa:** Algumas aplicações precisam receber atualizações dos recursos. Elas os consultam periodicamente para verificar as atualizações, mas isso pode gerar várias solicitações retornando nenhuma atualização.
   - **Solução:** Utilizar notificações *push*; são mensagens enviadas para as aplicações sempre que alguma mudança ocorre.

6. **Experiência do usuário**

   - **Causa:** Alguns recursos são úteis para melhorar a experiência do usuário, como animações, mas consomem muita energia do *smartphone*.
   - **Solução:** Utilizar o modo de economia de energia no aplicativo, fornece a funcionalidade miníma, diminuindo a experiência do usuário e ganhando eficiência energética. Ativar recursos dependendo do *status* da bateria, mesmo quando estiver ligado a uma fonte de energia. Deixar o usuário ciente sobre a troca de recursos por eficiência energética, caso contrário, eles podem pensar que a aplicação não está se comportando corretamente.

7. **Uso de Wi-Fi e uso de Dados Móveis (3G/4G)**

   - **Causa:** Conexão de dados usando a rede do celular (3G/4G) consome mais energia do que as conexões utilizando o Wi-Fi.
   - **Solução:** Adiar a sincronização de dados que não são urgentes, até que uma rede Wi-fi esteja disponível

8. **Registro de log**

   - **Causa:** Os registros de log's podem ser utilizados para garantir o comportamento correto dos aplicativos e simplificar relatórios, mas essas operações sobrecarregam o consumo de energia e não geram valor para o usuário. A linguagem utilizada para o desenvolvimento da aplicação tem influência nesse item, como foi mostrado na pesquisa realizada, podendo aumentar o consumo energético.
   - **Solução:** Evitar o uso do registro de log de forma intensiva.

9. **Conexões com servidor**

   - **Causa:** Geralmente as aplicações apresentam os dados que estão armazenados em um servidor. Coletar os mesmos dados várias vezes pode aumentar o consumo de energia do *smartphone*.
   - **Solução:** Utilizar armazenamento temporário em cache, evitando vários acessos ao servidor em busca de informações repetidas.

10. **Alta resolução**

    - **Causa:** Ao exibir dados, os desenvolvedores se sentem tentados a utilizar uma resolução alta, mas isso requer mais recursos como: memória e processamento. A consequência é o consumo alto de energia.
    - **Solução:** Utilizar a resolução ideal para cada caso, conforme a experiência de usuário que se deseja fornecer.

11. **Sensores**

    - **Causa:** Alguns recursos utilizam a leitura de dados e operações de sensores. Essas operações podem consumir energia de maneira exorbitante
    - **Solução:** Chamar as operações que utilizam sensores um menor número de vezes. Utilizar sensores de baixa potência para verificar se uma determinada operação, que consome muita energia, precisa ser realizada.

12. **Tarefas que consomem muita energia**

- **Causa:** Algumas tarefas podem consumir muita energia do *smartphone*, mas são estritamente necessárias para a aplicação.
- **Solução:** Permitir que o usuário acione tais tarefas, por exemplo, por meio de um botão.

3.3 Ferramentas para analisar o consumo de energia

Algumas ferramentas podem ser de grande ajuda quando o assunto é análise de código, principalmente aquelas que conseguem fazer a análise em tempo real, mostrando defeitos do código que podem gerar alto consumo de energia. Este capítulo apresentará as ferramentas: GATOR (*Program Analysis Toolkit For Android*) (YANG, 2019), SonarQube (SONARQUBE, 2019) e Android Profiler (ANDROIDSTUDIO, 2019).

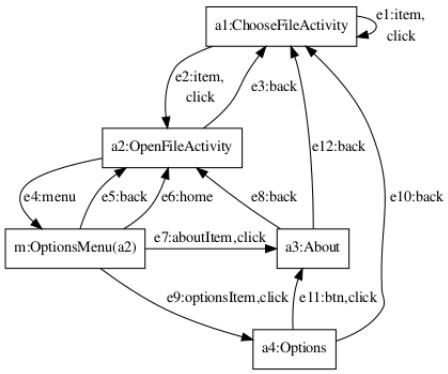
3.3.1 GATOR

Desenvolvido pelo autor Yang (2019) o *Program Analysis Toolkit For Android*, também conhecido como GATOR, é um kit de ferramentas para análise de programas Android. O primeiro componente desse kit é a análise estática de interface (GUI) dos *softwares* Android. Ela modela os objetos relacionados a interface, o fluxo e suas intenções. Por meio de um grafo, modela-se o fluxo, a estrutura hierárquica e os efeitos das ações. A análise de forma estática é fundamental para análises de compiladores. A análise acessa o grafo e percorre os caminhos de fluxo de controle que são compatíveis com o contexto, identificam instruções que podem disparar retornos de chamada e os caminhos que evitam essas instruções. O segundo componente é uma análise de fluxo de controle de retornos de chamada acionados por eventos. Como resultado, o GATOR gera um diagrama que é chamado de diagrama de transição de telas.

O GATOR foi desenvolvido com base no grafo de transição de janelas (WTG) proposto pelos autores Yang et al. (2015). Um exemplo deste grafo é mostrado na Figura 10. A finalidade do WTG é detectar defeitos de energia e auxiliar na gestão de retornos de chamada. Eles mostram a sequência de transição de janelas, seus eventos e retornos de chamada. Os eventos são representados pelos vértices e as arestas representam as ações tomadas/transições de janelas. Com essas informações, pode-se avaliar se algum recurso está ativo sem ser utilizado, trazendo ineficiência energética.

O GATOR é uma ferramenta de código aberto. Informações sobre como obter a ferramenta e configurações necessárias encontram-se na Tabela 1. Informações sobre quais características que os desenvolvedores relataram na pesquisa podem ser encontradas nessa ferramenta, bem como os itens relatados na Subseção 3.2 que podem causar o consumo alto de energia, mas que podem ser solucionados por essa ferramenta, se encontram na Tabela 2.

Figura 10 – Exemplo de Grafo WTG



Fonte: Yang et al. (2015)

3.3.2 SonarQube

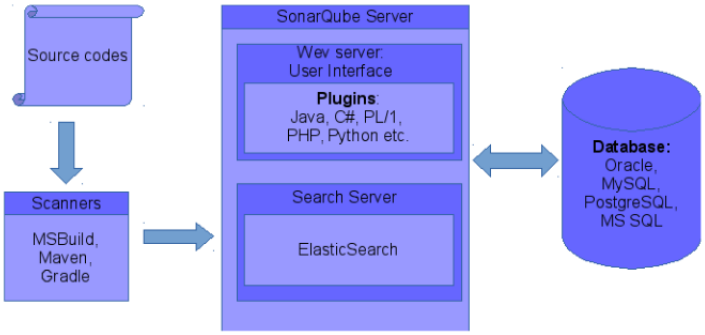
Figura 11 – Logo da Ferramenta SonarQube

Fonte: SonarQube (2019)

A ferramenta SonarQube é uma plataforma com iniciativa de código aberto e de análise estática com objetivo de gerenciar a qualidade de código fonte e foi desenvolvida pela empresa SonarSource. A arquitetura da ferramenta conta com quatro componentes: servidor, banco de dados, *plugins* e *scanner* como mostra a Figura 12. O servidor possui um processo que oferece uma interface com usuário capaz de mostrar os resultados das analises dos projetos e um outro processo com servidor de pesquisa para consultas. O banco de dados armazena as configurações de instalação e as análises realizadas. O *scanner* analisa os projetos linha por linha e envia os resultados para o servidor. Os *plugins* podem ser utilizados para aumentar as funcionalidades da ferramenta (PAANANEN, 2016).

O SonarQube recebe como base o código fonte e é capaz de analisar várias linguagens de programação por meio de regras através dos *plugins*, através deles mais regras podem ser adicionadas. Existem *plugins* para várias IDE's disponíveis no mercado e ele permite que o usuário escrevam seus próprios *plugins*. Com base no código inserido, a ferramenta verifica se as regras estão sendo cumpridas e, ao final, são mostradas informações e sugestões de melhorias. Os problemas que o Sonarqube encontra podem ser classificados em 5 grupos, dependendo de sua gravidade: bloqueador, crítico, principal, pequeno e informações (INFOBIP, 2016). A ferramenta lista todos os erros, cada um pertencente a uma regra e mostra possíveis soluções.

Figura 12 – Arquitetura da Ferramenta SonarQube



Fonte: Paananen (2016)

O Sonarqube é uma ferramenta de código aberto e foi sugerido pelos desenvolvedores que participaram da pesquisa. Informações sobre como obter a ferramenta e configurações necessárias encontram-se na Tabela 1. Informações sobre quais características que os desenvolvedores relataram na pesquisa podem ser encontradas nessa ferramenta, bem como os itens relatados na Subseção 3.2 que podem causar o consumo alto de energia, mas que podem ser solucionados por essa ferramenta, se encontram na Tabela 2

3.3.3 Android Profiler

Figura 13 – Ferramenta Android Profiler do Android Studio.

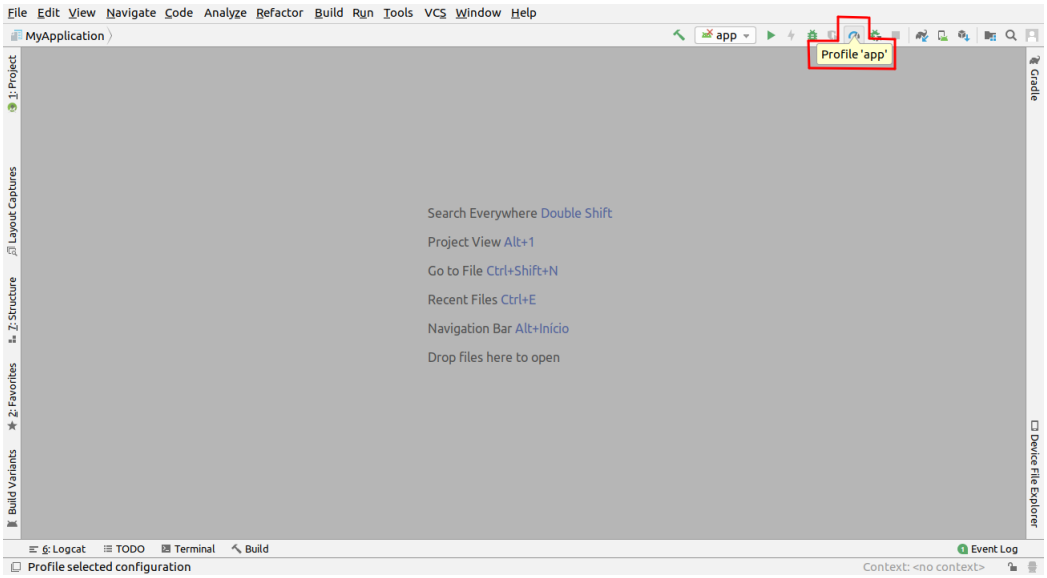
Fonte: Elaborado pela autora

O Android Profiler é uma ferramenta disponível desde a versão 3.0 do Android Studio. Ele cria um perfil do aplicativo que mostra dados em tempo real para auxiliar o desenvolvedor a entender como sua aplicação está utilizando recursos de CPU, memória, rede e bateria.

Para acessar o Android Profiler clique em *Profiler 'app'* na barra de ferramentas, como é mostrado na Figura 14. Caso a caixa de diálogo *Select Deployment Target* apareça, então selecione o dispositivo em que deseja criar o perfil do aplicativo, se for um dispositivo conectado via USB é necessário ativar a depuração de UBS, se estiver utilizando o emulador

do Android, o Profiler listará todos os processos que estão em execução, mesmo os que não são depuráveis.

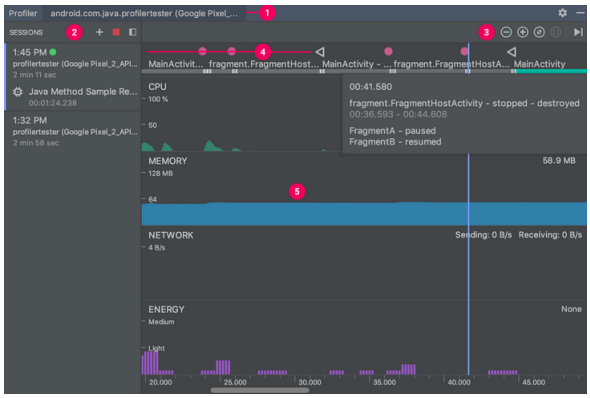
Figura 14 – Acessar Android Profiler



Fonte: Elaborado pela autora (2019)

Na Figura 15 é mostrada a linha do tempo do Android Profiler. No ponto 1, é possível visualizar o processo e o dispositivo que o perfil está sendo criado. No ponto 2, é apresentado o painel de *sessions*, onde pode-se escolher uma sessão para visualizar ou criar uma nova. No ponto 3, está os botões para dar zoom e conseguir controlar a linha do tempo, também encontra-se o botão *Attach to live* que permite acessar as atualizações em tempo real. No ponto 4, é exibida a linha do tempo dos eventos, relacionado com entradas de usuário, atividades do teclado e dos botões de volume e rotação da tela. Por último, no ponto 5, pode-se ver a linha do tempo compartilhada mostrado gráficos de CPU, memória, rede e bateria.

Figura 15 – Linha do tempo do Android Profiler



Fonte: Developer (2019)

O Android Profiler é uma ferramenta gratuita e foi sugerida pelos desenvolvedores que participaram da pesquisa. Informações sobre como obter a ferramenta e configurações necessárias encontram-se na Tabela 1. Informações sobre quais características que os desenvolvedores relataram na pesquisa podem ser encontradas nessa ferramenta, bem como os itens relatados na Seção 3.2 que podem causar o consumo alto de energia, mas que podem ser solucionados por essa ferramenta, se encontram na Tabela 2.

Tabela 1 – Ferramentas e suas configurações.

Ferramenta	**Versão**	**Sistemas Operacionais**	**Configurações**	***Link* para *Download***
GATOR	3.8	Linux	Instalação do JDK, Python 3 e Android SDK. Demais configurações podem ser obtidas através da documentação.	Disponível em Yang (2019)
SonarQube	7.9 LTS	Linux, Windows, MacOS	Baixar o SonarQube-Server, adicionar o Sonar nas variáveis de ambiente, instalar o Scanner. Demais configurações podem ser encontradas no *link*: https://dtidigital.com.br/cofiguracao-sonarqube/ ou na documentação.	Disponível em SonarQube (2019)
Android Profiler	3.4.1	Linux, Windows, MacOS	Android Studio acima do 3.0, Android acima do 5.0 (API *level* 21) e configurações do Android Studio necessárias, que podem ser encontradas no *link*: https://developer.android.com/studio/intro/studio-config?hl=pt-br	Disponível em AndroidStudio (2019)

Fonte: Elaborado pela autora (2019)

Tabela 2 – Ferramentas e suas relações com as características requeridas e itens solucionados.

Ferramenta	**Características Presentes**	**Itens que Podem ser Solucionados**
GATOR	1, 5	3, 4, 6, 11, 12
SonarQube	1, 2, 3, 4, 5	3, 6, 8, 9, 12
Android Profiler	4, 5	3, 6, 8, 9, 10, 12

Fonte: Elaborado pela autora (2019)

Pode-se notar que a maioria das características requeridas pelos desenvolvedores entrevistados estão presentes nas ferramentas retratadas, por exemplo, a lista com as linhas de código que apresentam defeitos pode ser obtida em todas as ferramentas. Sendo assim, as ferramentas têm êxito em atender as expectativas dos desenvolvedores e conseguem auxilia-los na criação de aplicações com consumo eficiente de energia, melhorando a

qualidade do código. É importante pontuar que nenhuma das ferramentas descritas traz traz a característica: filtros para pesquisar por tempo, memória, processamento ou nome de funções. Todas as ferramentas podem solucionar os itens: Tarefas que não afetam diretamente o usuário (item 3), experiência do usuário (item 6) e tarefas que consomem muita energia (item 12).

3.4 Resultados

Com base nos estudos realizados pode-se observar que tarefas e recursos em segundo plano são as atividades que mais consomem energia do *smartphone* e na maioria das vezes são irrelevantes para o usuário. O uso de ferramentas de análise, como as descritas neste trabalho, são essenciais para detectar tais defeitos e melhorar a experiência do usuário. Outro fator que precisa ser considerado é a escolha das tecnologias de desenvolvimento, pois elas influenciam no consumo de energia da aplicação. É importante que todos os recursos sejam bem utilizados, seja de *hardware* ou *software*, sendo solicitados somente quando necessários.

A Tabela 3 mostra uma comparação dos resultados obtidos através da pesquisa realizada neste trabalho com a pesquisa de Ferreira (2017). Pode-se notar que nos dois trabalhos houve a percepção da necessidade de difundir, entre a comunidade de desenvolvedores, a importância do requisito de consumo de energia. Sobre a resolução de defeitos, 50% dos entrevistados deste trabalho relataram que não resolvem os defeitos relacionados a energia, enquanto os entrevistados de Ferreira (2017) afirmaram que seus usuários não encontram tais defeitos nas aplicações disponibilizadas, sugerindo que de alguma forma eles trataram os defeitos, mesmo sem saber de sua existência.

Tabela 3 – Comparação dos resultados com os obtidos por Ferreira (2017).

Resultados	**Presente Trabalho**	**Ferreira (2017)**
Entrevistados conhecem o assunto	50%	33%
Entrevistados conhecem ferramentas que os auxiliam	50%	33%
Entrevistados resolvem defeitos de energia	50%	100%
Entrevistados realizam testes de qualidade	50%	17%

Fonte: Elaborado pela autora (2019)

As ferramentas aqui apresentadas oferecem análises de código com funcionalidades que ajudam na correção de defeitos de código que aumentam o consumo energético. A Tabela 3 mostra que somente 33% dos desenvolvedores entrevistados por Ferreira (2017) conhecem alguma ferramenta que tem essa finalidade, contudo o autor mostrou que tais pessoas não souberam esclarecer qual o apoio as ferramentas ofereciam. A pesquisa realizada no presente trabalho mostrou que 50% dos entrevistados conhecem ferramentas, as utilizam

e as indicaram, essas ferramentas contém características que os mesmos propuseram como importantes para auxiliá-los da melhor forma.

Uma fase importante do ciclo de criação de aplicações é a de testes. As duas pesquisas aqui analisadas mostram que poucos desenvolvedores realizam testes nos aplicativos que desenvolvem. Essa é uma das fases mais necessárias quando se trata de qualidade, principalmente na questão do consumo de energia, as ferramentas dão apoio aos testes, mostrando análises e possíveis defeitos.

A capacidade da bateria dos *smartphones* pode ser melhor usufruída e a vida útil pode ser prolongada, conforme sua utilização seja feita de forma eficaz (CRUZ, 2019). Por isso é tão importante que essa questão seja tratada pelas aplicações. Sendo assim, os desenvolvedores precisam estar mais atentos a esse requisito, utilizando ferramentas e estratégias como as citadas neste trabalho.

4 Considerações finais e trabalhos futuros

A evolução dos celulares trouxe funções cada vez mais poderosas e a capacidade de duração da bateria precisou acompanhar essa evolução. O consumo energético de forma eficiente se tornou um dos requisitos não funcionais mais demandados pelos usuários. Medir o consumo é uma tarefa complexa, mas é possível através de ferramentas apropriadas.

O objetivo inicial desse trabalho era mostrar as funcionalidades da ferramenta GATOR, principalmente estudar o diagrama de transição de telas que a mesma propõe como resultado da análise estática e entender como ela auxilia os desenvolvedores a produzir códigos com consumo eficiente de energia e utilizar os recursos da melhor forma possível. Foram encontradas dificuldades para utilizar a ferramenta e entender como ela retornava os resultados. Por essa limitação, a metodologia do trabalho precisou ser adaptada para contemplar outras ferramentas e estratégias que pudessem auxiliar os desenvolvedores.

O presente trabalho apresentou as causas mais comuns de consumo alto de energia dos *smartphones*, em relação a defeitos de código. Também foram apresentadas ferramentas e estratégias que podem ajudar na solução dos problemas, deixando as aplicações mais eficientes na questão energética. Na pesquisa realizada pelo autor Ferreira (2017), os entrevistados relataram que não conheciam ferramentas que pudessem auxiliá-los a tratar a ineficiência energética. Como continuação do referido trabalho, apresentou-se ferramentas com tal objetivo. Dessa forma, o objetivo geral do trabalho, que é estratégias para a análise do consumo de energia de aplicações móveis, foi alcançado.

Por meio da análise qualitativa, obtida através do questionário aplicado, pode-se perceber que é necessário a realização de avaliação da qualidade das aplicações. Nesse sentido, os testes podem facilitar o desenvolvimento de aplicações que utilizem a energia dos *smartphones* de forma eficaz. Ainda, pode-se notar que há falta de conhecimento e preocupação dos desenvolvedores quanto ao requisito consumo de energia. Esse fator contribui para o desenvolvimento de aplicações que utilizam a energia do *smartphone* de forma ineficiente, sendo necessária a realização de conscientização, principalmente em relação ao tema *software* verde.

Como trabalhos futuros há a possibilidade de estudos para utilização das ferramentas descritas em algum contexto de desenvolvimento e avaliar o desempenho de cada uma em determinadas tarefas. Finalizando, espera-se que esse trabalho contribua para a conscientização de desenvolvedores quanto a importância de criar aplicações com consumo eficiente de energia.

Referências

ADAM, P. *A História do Smartphone*. 2016. Disponível em: <http://www.mobileindustryreview.com/2016/10/the-history-of-the-smartphone.html>. Acesso em: 20 de Outubro de 2019. Citado na página 19.

AHMAD, R. W. et al. *A survey on energy estimation and power modeling schemes for smartphone applications*. *International Journal of Communication Systems*, v. 30, n. 11, 2017. Citado 3 vezes nas páginas 12, 26 e 27.

ANATEL. *Brasil registra 228,64 milhões de linhas móveis ativas em maio de 2019* . 2019. Disponível em: <https://www.anatel.gov.br/institucional/noticias-destaque/2310-brasil-registra-228-64-milhoes-de-linhas-moveis-ativas-em-maio-de-2019>. Acesso em: 20 de Outubro de 2019. Citado na página 12.

ANDROIDSTUDIO. *AndroidStudio*. 2019. Disponível em: <https://developer.android.com/studio/?gclid=Cj0KCQiAiNnuBRD3ARIsAM8Kmltgmkx-D7e4WEQ6LE4SHPZGzFXi8R_XoZ-8ybYeyS7DgkxIhq4urikaAmHTEALw_wcB>. Acesso em: 21 de Novembro de 2019. Citado 2 vezes nas páginas 31 e 35.

ARDITO, L. et al. *Understanding Green Software Development: A Conceptual Framework*. In: . [S.l.: s.n.], 2015. p. 1–6. Citado na página 19.

CARROLL, A.; HEISER, G. *An Analysis of Power Consumption in a Smartphone*. In: *Proc. 2010 USENIX conf., USENIX Assoc.* [S.l.: s.n.], 2010. Citado 2 vezes nas páginas 12 e 21.

CHARLAND, A.; LEROUX, B. *Mobile application Development: Web vs. native*. *Communications of the ACM*, v. 54, n. 5, p. 49–53, 5 2011. Citado na página 17.

COUTO, M. et al. *Analyzing and Classifying Energy Consumption in Android Applications*. In: . [S.l.: s.n.], 2016. p. 1–21. Citado 2 vezes nas páginas 21 e 27.

CREDITSUISSE. *Evolution of the mobile phone*. 2019. Disponível em: <https://www.credit-suisse.com/about-us-news/en/articles/news-and-expertise/evolution-of-the-mobile-phone-201908.html>. Acesso em: 12 de Novembro de 2019. Citado na página 20.

CRUZ, L. M. d. *Tools and Techniques for Energy-Efficient Mobile Application Development*. In: *Universidade do Porto*. [S.l.: s.n.], 2019. Citado 7 vezes nas páginas 12, 22, 23, 24, 25, 28 e 37.

CUADRADO, F.; DUEñAS, J. C. *Mobile Application Stores: Success Factors, Existing Approaches, and Future Developments*. In: *IEEE Communications Magazine*. [S.l.: s.n.], 2012. p. 160–167. Citado na página 16.

DELL. *Relatório de responsabilidade social corporativa do ano fiscal de 2019*. 2019. Disponível em: <https://corporate.delltechnologies.com/pt-br/social-impact/reporting/fy19-csr-report.htm#>. Acesso em: 20 de Outubro de 2019. Citado na página 19.

DELLOITE. *Global Mobile Consumer Survey 2019*. 2019. Disponível em: <https://www2.deloitte.com/br/pt/pages/technology-media-and-telecommunications/articles/mobile-survey.html>. Acesso em: 20 de Outubro de 2019. Citado na página 13.

DEVELOPER. *Medir o desempenho do aplicativo com o Android Profiler*. 2019. Disponível em: <https://developer.android.com/studio/profile/android-profiler?hl=pt-br>. Acesso em: 28 de Novembro de 2019. Citado na página 34.

DEVMEDIA. *TI Sustentável: conceito, soluções e consequências*. 2013. Disponível em: <https://www.devmedia.com.br/ti-sustentavel-conceito-solucoes-e-consequencias/29394>. Acesso em: 20 de Outubro de 2019. Citado na página 14.

ELIFE. *Novo Estudo Hábitos e Comportamento dos Usuários Brasileiros nas Redes Sociais 2016*. 2016. Disponível em: <https://elife.com.br/estudohabitos/>. Acesso em: 12 de Novembro de 2019. Citado na página 13.

FERREIRA, M. W. *Levantamento de Abordagens para a Análise de Consumo de Energia de Aplicações Móveis: Um Estudo de Caso do Laboratório IMOBILIS*. 2017. Monografia (Bacharel Sistemas de Informação), UFOP (Universidade Federal de Ouro Preto), João Monlevade, Brasil. Citado 6 vezes nas páginas 9, 13, 14, 23, 36 e 38.

FUNDACAO, T. *Juventude conectada*. 2014. Disponível em: <http://fundacaotelefonica.org.br/acervo/juventude-conectada/>. Acesso em: 20 de Outubro de 2019. Citado na página 13.

GSMA, L. A. *Smartphones fueling mobile ecosystem growth in Latin America*. 2017. Disponível em: <https://www.gsma.com/latinamerica/smartphones-fueling-mobile-ecosystem-growth-latin-america/>. Acesso em: 20 de Outubro de 2019. Citado na página 13.

HU, Y. et al. *Lightweight energy consumption analysis and prediction for Android applications. Science of Computer Programming*, v. 162, p. 132–147, 2018. Citado 3 vezes nas páginas 12, 25 e 27.

INFINITELOOP. *Confira o que são notificações push para aplicativos e como usá-las*. 2017. Disponível em: <http://www.infiniteloop.com.br/confira-o-que-sao-notificacoes-push-para-aplicativos-e-como-usa-las/>. Acesso em: 20 de Novembro de 2019. Citado na página 29.

INFOBIP. *Melhorando a qualidade do código com SonarQube*. 2016. Disponível em: <https://www.infobip.com/pt/desenvolvedor/melhorando-a-qualidade-do-codigo-com-sonarqube>. Acesso em: 24 de Novembro de 2019. Citado na página 32.

LAUDON, K.; LAUDON, J. *Sistemas de Informações Gerenciais*. 9. ed. [S.l.]: Pearson Education do Brasil, 2010. Citado na página 16.

LECHETA, R. R. *Google Android - Aprenda a criar aplicações para dispositivos móveis com o Android SDK*. 3. ed. [S.l.]: Novatec, 2013. v. 1. Citado 2 vezes nas páginas 17 e 18.

LUNARDI, G. L.; SIMõES, R.; FRIO, R. S. Ti verde: Uma análise dos principais benefícios e práticas utilizadas pelas organizações. In: *Universidade Federal do Rio Grande*. [S.l.: s.n.], 2012. p. 1–30. Citado na página 18.

MOBILE, T. *Apps geram US$ 101 bilhões em receita em 2018*. 2016. Https://www.mobiletime.com.br/noticias/16/01/2019/apps-geram-us-101-bi-em-receita-em-2018/. Citado na página 16.

MOREIRA, E. *Um estudo mostra a evolução das baterias para smartphones*. 2017. Disponível em: <https://www.targethd.net/um-estudo-mostra-a-evolucao-das-baterias-para-smartphones/>. Acesso em: 12 de Novembro de 2019. Citado 2 vezes nas páginas 20 e 21.

PAANANEN, T. *Analyzing Java EE application security with SonarQube*. In: *JAMK University of Applied Sciences*. [S.l.: s.n.], 2016. p. 1–93. Citado 2 vezes nas páginas 32 e 33.

PEREIRA, L. C. O.; SILVA, M. L. d. *Android para desenvolvedores*. [S.l.]: Brasport, 2009. Citado 2 vezes nas páginas 17 e 18.

PINTO, G.; CASTOR, F. *Energy Efficiency: A New Concern for Application Software Developers*. *Communications of the ACM*, v. 60, n. 12, p. 68–75, 2017. Citado 4 vezes nas páginas 13, 14, 22 e 27.

PINTO, G.; SOARES-NETO, F.; CASTOR, F. *Refactoring for Energy Efficiency: A Reflection on the State of the Art*. In: *Universidade Federal de Pernambuco*. [S.l.: s.n.], 2015. p. 1–7. Citado na página 14.

RESEARCH, B. N.; SIOUX. *Game Brasil 2018*. 2018. Disponível em: <https://www.pesquisagamebrasil.com.br/>. Acesso em: 20 de Outubro de 2019. Citado na página 13.

SCHULZ, M. A.; SILVA, T. N. Ti verde e eficiência energética em *Data Centers*. *Revista de Gestão Social e Ambiental - RGSA*, v. 6, n. 2, p. 121–133, 5 2012. Citado na página 19.

SEGURO, B. M. *Baterias de Smartphones: Conheça a Evolução até Aqui e Quais as Projeções Futuras*. 2018. Disponível em: <https://blog.bemmaisseguro.com/baterias-de-smartphone-evolucao-e-projecoes-futuras/>. Acesso em: 12 de Novembro de 2019. Citado na página 21.

SONARQUBE. *SonarQube - Your teammate for Code Quality and Security* . 2019. Disponível em: <http://web.cse.ohio-state.edu/presto/software/gator/>. Acesso em: 21 de Novembro de 2019. Citado 3 vezes nas páginas 31, 32 e 35.

STACKOVERFLOW. *Testing Battery Usage*. 2018. Disponível em: <https://stackoverflow.com/questions/7566910/testing-battery-usage>. Acesso em: 26 de junho de 2019. Citado na página 12.

TECMUNDO. *A evolução das telas em celulares*. 2014. Disponível em: <https://www.tecmundo.com.br/telas/49151-a-evolucao-das-telas-em-celulares-infografico-.htm>. Acesso em: 12 de Novembro de 2019. Citado na página 20.

TECNOLOGIA, U. N. *Do Dynatac ao iPhone, veja a linha do tempo do celular*. 2013. Disponível em: <https://tecnologia.uol.com.br/infograficos/2013/09/02/evolucao-dos-celulares.htm>. Acesso em: 12 de Novembro de 2019. Citado na página 19.

WASSERMAN, A. *Software engineering issues for mobile application development*. In: . [S.l.: s.n.], 2010. p. 397–400. Citado na página 17.

WU, H.; YANG, S.; ROUNTEV, A. *Static Detection of Energy Defect Patterns in Android Applications*. In: *International Conference on Compiler Construction*. [S.l.: s.n.], 2016. Citado 2 vezes nas páginas 12 e 27.

YANG, S. *GATOR: Program Analysis Toolkit For Android*. 2019. Disponível em: <http://web.cse.ohio-state.edu/presto/software/gator/>. Acesso em: 21 de Novembro de 2019. Citado 2 vezes nas páginas 31 e 35.

YANG, S. et al. *Static Window Transition Graphs for Android*. In: *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. [S.l.: s.n.], 2015. p. 658–668. Citado 2 vezes nas páginas 31 e 32.

Apêndices

APÊNDICE A – Questionário sobre Estratégias para a Análise do Consumo de Energia de Aplicações Móveis.

1. **Você tem quanto tempo de experiência?**

   R: _____

2. **Qual a plataforma que utiliza para desenvolver seus projetos?**

   R: _____

3. **Quais linguagens utiliza?**

   R: _____

4. **Você faz avaliação da qualidade das aplicações que desenvolve?**

   R: ( ) Sim

   ( ) Não

5. **A eficiência energética é um tema muito relevante ultimamente. Você já teve conhecimento desse tema alguma vez? Se sim, diga quais foram as suas fontes de informação**

   R: ( ) Não

   ( ) Sim _____

6. **Quando você pesquisa um *framework* para desenvolver o aplicativo, se preocupa com o desempenho dele quanto a eficiência energética?**

   R: ( ) Sim

   ( ) Não

7. **Já pesquisou sobre ferramentas ou técnicas que ajudam encontrar defeitos no código que possam gerar um consumo ineficiente de energia, se sim, qual?**

   R: ( ) Não

   ( ) Sim _____

8. **Se preocupa em corrigir os defeitos que geram consumo excessivo de energia?**

   R: _____

9. **Você procura utilizar todos os métodos da linguagem, de forma correta, pesquisando na documentação como utilizá-los?**

   R: _____

10. **Quais características espera encontrar em uma ferramenta que possa te auxiliar a usar energia do *smartphone* de forma eficiente?**

    R: _____