



UNIVERSIDADE FEDERAL DO MARANHÃO

Curso de Ciência da Computação

Wesley Kelson Ribeiro Figueredo

**Leitura Automática de Consumo de Energia
Elétrica em Aplicativos Móveis**

São Luís - MA

2018

Wesley Kelson Ribeiro Figueredo

Leitura Automática de Consumo de Energia Elétrica em Aplicativos Móveis

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Anselmo Cardoso de Paiva

São Luís - MA

2018

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Figueredo, Wesley Kelson Ribeiro.

Leitura Automática de Consumo de Energia Elétrica em Aplicativos Móveis / Wesley Kelson Ribeiro Figueredo. - 2018.

57 f.

Orientador(a): Anselmo Cardoso de Paiva.

Monografia (Graduação) - Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luis - MA, 2018.


1. Aplicativo Móvel. 2. Leitura Automática da Medição de Consumo Elétrico. 3. Medidores de Energia Elétrica. 4. SILEM. I. Paiva, Anselmo Cardoso de. II. Título.

Wesley Kelson Ribeiro Figueredo

Leitura Automática de Consumo de Energia Elétrica em Aplicativos Móveis

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

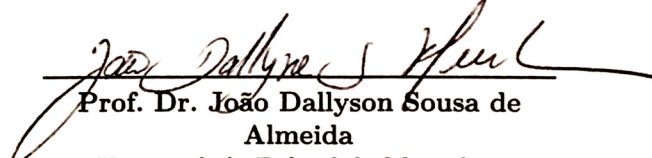
Trabalho aprovado em São Luís - MA, 22 de Janeiro de 2018:



Prof. Dr. Anselmo Cardoso de Paiva
Orientador
Universidade Federal do Maranhão



Prof. Dr. Aristófanis Corrêa Silva
Universidade Federal do Maranhão



Prof. Dr. João Dallyson Sousa de Almeida
Universidade Federal do Maranhão

São Luís - MA

2018

Agradecimentos

Agradeço primeiramente a Deus, aos meus pais, meus irmãos e familiares que sempre me apoiaram e incentivaram nesta caminhada.

Aos amigos que fiz durante o curso, que tanto me ajudaram durante o mesmo e pela companhia no RU.

Aos amigos que conheci durante o intercâmbio, que me apoiaram e compartilharam momentos de alegrias e tristezas.

Aos amigos de longa data que se mantiveram comigo durante todos esses anos.

Aos amigos que fiz durante a execução deste projeto, pelo apoio e pelo conhecimento compartilhado.

Ao professor Anselmo Cardoso de Paiva, pela orientação, pelo suporte e paciência.

Ao professor João Dallyson Sousa de Almeida por ter aceitado compor a banca e pela orientação em projetos.

Ao professor Aristófanês Corrêa Silva ter aceitado compor a banca e pela orientação e suporte.

Ao professor Geraldo Braz Junior, pela orientação durante a iniciação científica e demais projetos.

Aos demais professores e àqueles que me auxiliaram e de alguma forma fizeram parte desta trajetória.

Resumo

O procedimento de leitura da medição de consumo em medidores de energia elétrica, água e gás é feita de forma manual na maioria dos casos. Visando desenvolver uma metodologia alternativa de medição, este trabalho apresenta a arquitetura de um aplicativo móvel, o SILEM (Sistema de Leitura da Medição), voltada à medição automática de medidores eletromecânicos e digitais utilizados em serviços públicos. O objetivo deste está voltado ao setor elétrico, tendo a situação da Companhia Energética do Maranhão (CEMAR) e da Centrais Elétricas do Pará (CELPA) como objeto de estudo. A aplicação desenvolvida busca tornar a atividade de leitura mais simples e confiável, evitando a necessidade de anotar o consumo manualmente, tentando impedir falha humana e reduzir despesas geradas por erros.

Palavras-chaves: SILEM, Leitura Automática da Medição de Consumo Elétrico, Aplicativo Móvel, Medidores de Energia Elétrica.

Abstract

The reading process of electricity is, in general, performed manually. Aiming to develop an alternative reading method, this work presents the architecture of a mobile app, the SILEM (Meter Reading System), directed towards automated reading of electromechanical and digital meters used in public services. It seeks to assist the electric power sector, having the Companhia Energética do Maranhão (CEMAR) and Centrais Elétricas do Pará (CELPA) situation as object of study. The application aims to make the meter reading process simpler and more reliable, avoiding the necessity of manually taking notes, trying to avoid human failure and reduce expenses caused by those errors.

Keywords:: SILEM, Automatic Meter Reading, Mobile App, Electricity Meter.

Lista de ilustrações

Figura 1 – Diagrama de sintaxe de um objeto JSON	17
Figura 2 – Diagrama de sintaxe de um <i>array</i> em JSON	17
Figura 3 – Diagrama de sintaxe valor em JSON	18
Figura 4 – Diagrama de sintaxe de string em JSON	19
Figura 5 – Diagrama de sintaxe de número em JSON	19
Figura 6 – Threads	20
Figura 7 – Arquitetura Android	22
Figura 8 – Exemplo de <i>template</i>	24
Figura 9 – Etapas da metodologia do SILEM	25
Figura 10 – Arquitetura do SILEM	26
Figura 11 – Metodologia utilizada	26
Figura 12 – <i>Template</i> dos dígitos	27
Figura 13 – Tela principal do SILEM	30
Figura 14 – Botão Lanterna	31
Figura 15 – Caixas de texto	32
Figura 16 – Imagem de <i>feedback</i>	33
Figura 17 – Objeto Silem	34
Figura 18 – Telas do MOM.	35
Figura 19 – Exemplo de medidores	36
Figura 20 – Tela do SIMU	37
Figura 21 – Interface do SILEM	37
Figura 22 – Telas do SILEM durante o <i>tracking</i>	38
Figura 23 – Tela de espera	38
Figura 24 – Tela com resultado do processamento	39
Figura 25 – Primeira interface do SILEM	42
Figura 26 – Gráfico 1 - Primeira avaliação	43
Figura 27 – Gráfico 2 - Primeira avaliação	43
Figura 28 – Gráfico 3 - Primeira Avaliação	44
Figura 29 – Gráfico 4 - Primeira avaliação	44
Figura 30 – Gráfico 5 - Primeira avaliação	45
Figura 31 – Gráfico 6 - Primeira avaliação	45
Figura 32 – Gráfico 7 - Primeira avaliação	46
Figura 33 – Gráfico 8 - Primeira avaliação	46
Figura 34 – Gráfico 9 - Primeira avaliação	47
Figura 35 – Gráfico 1 - Segunda avaliação	48
Figura 36 – Gráfico 2 - Segunda avaliação	49

Figura 37 – Gráfico 3 - Segunda avaliação	49
Figura 38 – Gráfico 4 - Segunda avaliação	50
Figura 39 – Gráfico 5 - Segunda avaliação	50
Figura 40 – Gráfico 6 - Segunda avaliação	51
Figura 41 – Gráfico 7 - Segunda avaliação	51
Figura 42 – Gráfico 8 - Segunda avaliação	52
Figura 43 – Gráfico 9 - Segunda avaliação	52

Lista de tabelas

Tabela 1 – Versões do Android	23
Tabela 2 – Requisitos	29
Tabela 3 – Requisitos não funcionais	29
Tabela 4 – Resultados da Metodologia	40
Tabela 5 – Respostas ao questionário	53

Lista de abreviaturas e siglas

SILEM	Sistema de Leitura da Medição
CEMAR	Companhia Energética do Maranhão
CELPA	Centrais Elétricas do Pará
NCA	Núcleo de Computação Aplicada
P&D	Pesquisa e Desenvolvimento
UFMA	Universidade Federal do Maranhão
ANEEL	Agência Nacional de Energia Elétrica
Wh	Watt-Hora
KWh	QuiloWatt-Hora
TWh	Terawatt-Hora
AMR	Leitura Automática de Medidores
CMOS	Semicondutor de Metal-Oxido Complementar
SMS	Serviço de Mensagens Curtas
GSM	Sistema Global para Comunicações Móveis
PDE	Plano Decenal de Expansão de Energia
UML	Linguagem de Modelagem Unificada
JSON	Notação de Objetos JavaScript
PDA	Assistente Pessoal Digital
IDE	Ambiente de Desenvolvimento Integrado
RE	Engenharia de Requisitos

Sumário

1	INTRODUÇÃO	12
1.1	Objetivo	13
1.2	Trabalhos Relacionados	14
1.3	Organização do Trabalho	15
2	MATERIAIS E MÉTODOS	16
2.1	Xamarin	16
2.2	JSON	16
2.3	Threads	18
2.4	Programação assíncrona	20
2.5	Android	21
2.5.1	Arquitetura	21
2.5.2	Versões	22
2.6	Template Matching	23
2.7	Limiar Adaptativo	24
3	METODOLOGIA	25
3.1	Reconhecimento dos dígitos	25
3.2	Aplicação	28
3.2.1	Requisitos	28
3.2.2	Requisitos Funcionais	29
3.2.3	Requisitos Não Funcionais	29
3.2.4	Interface	30
3.2.5	SILEM	32
4	RESULTADOS	40
4.1	Avaliação	41
5	CONCLUSÃO	54
	REFERÊNCIAS	55

1 Introdução

Os processos de medição do consumo de consumo é feito de forma manual. Um agente da empresa prestadora do serviço dirige-se até a unidade consumidora (residência ou empresa do cliente) e coleta o valor consumido no mês através da leitura do consumo registrada no medidor, equipamento responsável por mensurar o consumo mensal do cliente, instalado no local. Apesar de já existir o processo remoto de medição, a forma como é feita essa medição é manual na maioria das residências e empresas, o que pode comprometer a confiabilidade do valor. Este trabalho busca automatizar parte deste processo tendo o setor elétrico como objeto de estudo.

O consumo de serviço do setor elétrico vem crescendo anualmente juntamente com o número de domicílios e medidores instalados, que de acordo com o Plano Decenal de Expansão de Energia (PDE) será de 75 milhões domicílios em 2020 e o consumo de energia elétrica irá subir cerca 4,8% ao ano (BRACIER, 2011). O consumo domiciliar chega a 30% do consumo total de energia elétrica do país.

Segundo Vidinich e Nery (2009), o setor elétrico ainda sofre com perdas de watt-hora (Wh), chegando a um total de 52 terawatts - hora (TWh) anualmente. As perdas são classificadas em técnicas e não técnicas. As perdas técnicas são ocasionadas no transporte e transformação da tensão, e também nos aparelhos de medição. Já as perdas não técnicas correspondem a erros de medição, unidades consumidoras sem medidor, furtos ou fraudes nos equipamentos de medição. As perdas não técnicas equivalem a 44% do total e seu custo anual é de aproximadamente R\$ 55 bilhões sem os acréscimos de atributos.

A Companhia Energética do Maranhão (CEMAR) e a Centrais Elétricas do Pará (CELPA) são as empresas responsáveis pela distribuição de energia elétrica nos estados do Maranhão e Pará, respectivamente. Ambas as empresa utilizam o medidor de energia como equipamento mensurador do consumo mensal de energia do cliente. Ele armazena, acumula e mostra o valor consumido de QuiloWatts-hora (KWh) durante o período de um mês. A acumulação mensal serve como meio comparativo entre o consumo do mês anterior com o atual.

A medição é feita por um leiturista, como é chamado o funcionário responsável pela medição, que dirige-se até o medidor presente na casa ou empresa do cliente e faz a anotação do consumo mensal. Durante o processo de anotação do valor consumido, o funcionário deve digitar o valor mostrado pelo medidor no aplicativo MOM, aplicativo responsável por gerar o faturamento de ambas as companhias previamente citadas, instalado em seu PDA (*Personal Digital Assistants*, em português Assistente Pessoal Digital) e imprimir a fatura a ser entregue ao cliente. Porém durante o processo de anotação podem ocorrer

erros de digitação da leitura. Quando o valor está fora da média de consumo ou alguma outra anormalidade seja notada pelo sistema, este solicita ao leiturista que adquira uma imagem do medidor. O sistema das empresas CEMAR e CELPA recebe cerca de 30 mil imagens diariamente originadas de erros detectados pelo sistema. Tais erros exigem muitos recursos de ambas as empresas.

A metodologia manual ainda é eficiente no atual cenário do sistema de distribuição de serviços públicos, porém o custo originados com erros humanos ainda é alto. A medição remota é uma possível solução para este problema, entretanto, o custo da substituição ou adaptação de todo um sistema também é muito elevado (HAMERSCHMIDT, 2012) (GONÇALVES, 2016).

Visando a criação de um módulo de leitura automática da medição de consumo, um projeto de Pesquisa e Desenvolvimento (P&D), sob contrato das empresas CEMAR/CELPA (ANEEL PD-00371-0029/2016) está sendo desenvolvido pelo Núcleo de Computação Aplicada (NCA) da Universidade Federal do Maranhão (UFMA) para agilizar e facilitar a rotina de trabalho do leiturista em campo. Este projeto propõe a criação de um Sistema de Leitura da Medição (SILEM) para reduzir erros associados ao processo de digitação da leitura e para que o trabalho manual executado pelo leiturista possa ser substituído por um método automático de medição do consumo de energia.

Também com o objetivo de automatizar o procedimento de aquisição do valor consumido através de um aplicativo móvel, este trabalho procura melhorar e tornar o trabalho do leiturista mais rápido. Sendo necessário somente que o funcionário capture uma imagem da região do display ou da área que delimita os dígitos a serem lidos. E então o aplicativo, que utiliza técnicas de processamento de imagem, retorna ao usuário o valor capturado na foto.

Entretanto, é necessário atender alguns requisitos de aquisição tais como distância entre o medidor e a câmera, visibilidade dos números e iluminação do ambiente para que o método funcione como esperado. Evitando focos de luz sobre os dígitos e, também, a captura com grande inclinação em relação à área dos dígitos.

Durante este trabalho pretendeu-se desenvolver um aplicativo móvel que, através da captura da imagem do medidor, seja capaz de identificar os números presentes no display referentes ao consumo. O que permitirá maior agilidade na coleta do consumo e maior confiabilidade do serviço prestado.

1.1 Objetivo

O objetivo deste trabalho é desenvolver um método para realizar a leitura automática do consumo de energia elétrica a partir de imagens de medidores digitais, baseando-se em

imagens de equipamentos de medição digitais em dispositivo móvel.

Esta metodologia foi implementada no Sistema de Leitura da Medição, o SILEM, através do qual os usuários podem executar seu trabalho de anotação do consumo de energia de maneira mais eficiente. Também é objetivo da metodologia implementada em dispositivo móvel aumentar a confiabilidade da medição, impedindo a falha humana e reduzindo custos. Portanto, pretende-se auxiliar as concessionárias de distribuição de energia elétrica a reduzir o custo gerado por perdas não técnicas, ou comerciais.

1.2 Trabalhos Relacionados

A busca por métodos automáticos capazes de lerem o consumo em medidores de energia elétrica, água e gás é um estudo que vem sendo desenvolvido há alguns anos. Métodos de Automáticos de Medição (*Automatic Meter Reading - AMR*) tem sido amplamente estudados nos Estados Unidos e países da Europa. Os AMR são voltados principalmente a métodos de acesso remoto, porém existem outras abordagens como a utilização de dispositivos móveis para capturar o consumo através de imagens. Estes podem ser abordados juntamente com outras tecnologias tais como frequência de rádio e rede de dados.

[Lei et al. \(2010\)](#) propuseram um método utilizando um dispositivo móvel. O sistema utiliza a câmera do dispositivo para capturar a imagem do medidor, e então processá-la e mostrar o consumo ao usuário. Neste trabalho são citados alguns problemas enfrentados em relação a aquisição da foto: baixa resolução, baixa ou alta iluminação, ou algo que obstrua a visualização dos dígitos.

[Zhang et al. \(2016\)](#), [Parthiban e Palanisamy \(2013\)](#) e [Kompf \(2015\)](#) desenvolvem métodos que consistem em fixar uma câmera em frente ao medidor, com foco na área dos dígitos. Após ser capturada, a imagem é processada, gerando o valor consumido em kwh. Enquanto [Parthiban e Palanisamy \(2013\)](#) e [Kompf \(2015\)](#) dão enfoque ao método de reconhecimento dos dígitos do medidor, [Zhang et al. \(2016\)](#) desenvolve também um método de envio automático de consumo gerado. A metodologia por ele encontrada faz uso de redes wireless ZigBee para enviar o consumo para coletores de dados.

Já em [Zhao et al. \(2009\)](#) é utilizado um sensor de imagem CMOS (*complementary metal-oxide-semiconductor*, em português semicondutor de metal-óxido complementar) para a captura da imagem e envia valor da medição através de redes wireless ZigBee.

[Ali et al. \(2016\)](#) também utilizam uma câmera fixada e envia o valor já calculado da fatura ao cliente através de SMS (*Short Message Service* ou Serviço de mensagens curtas, em português) via Sistema Global para Comunicações Móveis (*Global System for Mobile Communications - GSM*).

[Prapasawad, Pornprasitpol e Pora \(2012\)](#) e [Ali et al. \(2012\)](#) baseiam-se no uso de

medidores inteligentes. Estes fazem uso de equipamentos que leem o consumo e a enviam automaticamente em forma de mensagens através de meios como a frequência de rádio e redes de dados.

Apesar de desenvolverem aplicações voltadas à medição automática de medidores de consumo de serviços públicos, os trabalhos supracitados focam, principalmente, no processo de reconhecimento dos dígitos. Desenvolvendo técnicas de processamento de imagem, os autores descrevem minuciosamente passo-a-passo de quais métodos foram desenvolvidos por eles para a resolução do problema apresentado.

Quanto ao reconhecimento, não foi possível encontrar, em nossos melhores esforços, uma metodologia de reconhecimento de medidores de energia elétrica em situações não controladas, o que não é compatível com o cenário no qual pretendemos trabalhar.

Nos trabalhos encontrados, percebeu-se uma variedade de métodos para a realização da leitura automática. Baseado nestas pesquisas buscou-se criar um comparativo entre os trabalhos apresentados nesta seção e adaptar o conhecimento adquirido para identificar uma alternativa viável para a implementação e implantação de uma aplicação que atenda realidade do sistema de distribuição de energia elétrica da empresas CELPA e CEMAR. No aplicativo, neste descrito, optou-se por usar a câmera do dispositivo móvel para capturar a imagem e enviar o valor reconhecido para o aplicativo da empresa, responsável pela geração da fatura.

1.3 Organização do Trabalho

Esta monografia está organizada em cinco capítulos. O atual capítulo contém a contextualização dos problemas enfrentados, assim como a justificativa para a realização do trabalho e os objetivos gerais e específicos. No segundo capítulo são descritas as tecnologias envolvidas no processo de desenvolvimento do SILEM. O terceiro capítulo apresenta detalhes do desenvolvimento, requisitos, interfaces e outros. Em seguida, o quarto capítulo apresenta a avaliação do sistema e discutimos os resultados. E no último capítulo, discute-se as conclusões obtidas com a realização deste trabalho.

2 Materiais e Métodos

Neste capítulo apresentamos e discutimos os métodos e materiais envolvidos no processo de desenvolvimento da metodologia no aplicativo. Serão descritas características de tecnologias como o próprio sistema Android e a plataforma Xamarin e as técnicas de processamento de imagem envolvidas no reconhecimento.

2.1 Xamarin

Primeiramente, Xamarin é uma empresa voltada para o desenvolvimento de aplicativos móveis. A Empresa foi adquirida em 2016 pela Microsoft. Atualmente, a plataforma está presente em todas as versões do Visual Studio, um ambiente de desenvolvimento (*Integrated Development Environment* - IDE) também pertencente à Microsoft. A Empresa é responsável pela plataforma com o mesmo da companhia (GUTHRIE, 2016) (MICROSOFT, 2017).

Xamarin é uma plataforma de desenvolvimento de aplicativos móveis *cross-plataform* que permite construir aplicações para Android, iOS e Windows em C#, linguagem de programação orientada a objetos também desenvolvida pela Microsoft. Por ser multi-plataforma, o código implementado pode ser reaproveitado entre os três sistemas anteriormente citados sem precisar de reescrever para outra linguagem de programação. Entretanto se faz necessário a criação de uma interface específica para a aplicação de cada sistema, e alterações em partes do código relacionadas à interface. Além disso o Xamarin desenvolve aplicações nativas, ou seja, o C# faz chamadas nativas do sistema operacional móvel, além de ter acesso à todos os recursos que uma aplicação desenvolvida especificamente para uma plataforma. Diferentemente do que acontece em aplicações híbridas, que tentam emular chamadas nativas do sistema.

2.2 JSON

O JSON (*JavaScript Object Notation*) é um derivado da linguagem de programação Javascript, porém JSON não é uma linguagem de programação, mas um formato de intercâmbio de dados. O JSON é independente da linguagem a qual se está trabalhando, pois faz uso de convenções comuns a várias linguagens. A troca de dados pode ocorrer entre aplicações de diversos tipos, como entre um navegador de internet e um servidor, e entre servidores (SMITH, 2015).

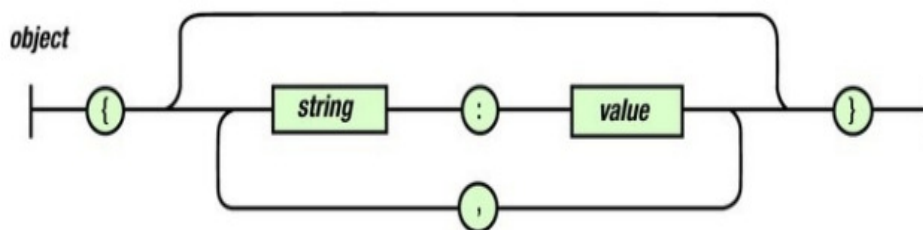
O JSON pode ser estruturado das seguinte formas:

- Uma coleção de pares de nome/valor.
- Uma lista ordenada de valores.

A estrutura de coleção de dados é associada, geralmente, à objetos, *structs*, *hash tablets* e outros. A estrutura de lista representa, em várias linguagens, vetores, arrays, listas, entre outros.

O diagrama da Figura 1 descreve a estrutura de um coleção (objeto) de pares . A coleção é iniciada abrindo uma chave ({) e termina com o fechamento da chave (}). Dentro das chaves, um nome é sucedido por dois pontos (:) e um valor que é associado a este nome. Um nome é apresentado em formato de string. Cada par de nome e valor é separado por vírgula.

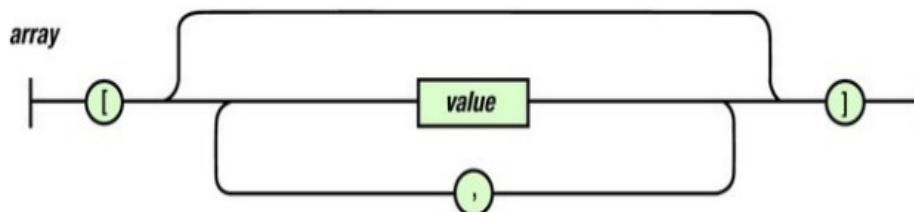
Figura 1 – Diagrama de sintaxe de um objeto JSON



Fonte: [Smith \(2015\)](#)

Uma lista (*array*) ordenada de valores é iniciada abrindo um colchete ([), seguido por 0 ou mais valores e finalizado com o fechamento do colchetes (]) (Figura 2). Cada valor é separado por vírgula.

Figura 2 – Diagrama de sintaxe de um array em JSON

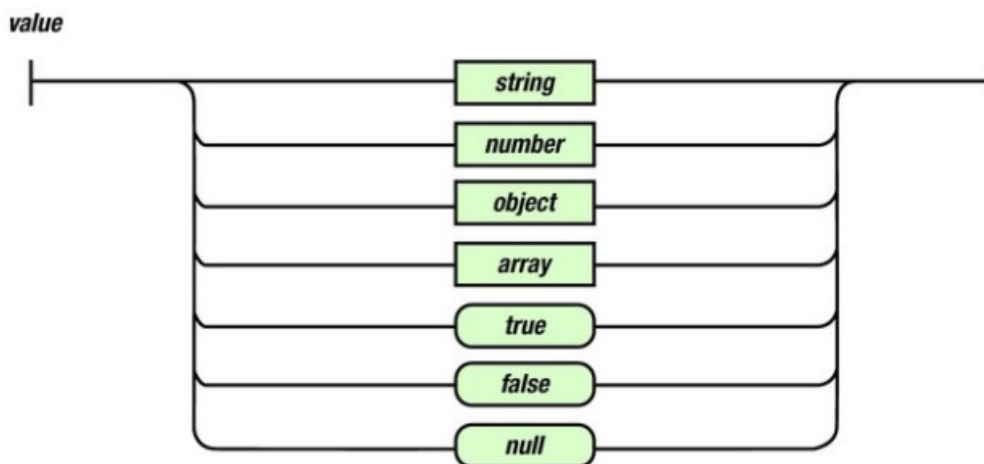


Fonte: [Smith \(2015\)](#)

O valor citado anteriormente e mostrado nas Figuras 1 e 2 pode ser uma string, ou um número, ou um objeto, ou um *array*, ou *true*, ou *false*, ou *null* (Figura 3). É um

requisito que esses três últimos, quando utilizados como valor, sejam mantidos em caixa baixa. Espaços em branco podem ser adicionados antes ou depois dos valores (SMITH, 2015).

Figura 3 – Diagrama de valor em JSON



Fonte: Smith (2015)

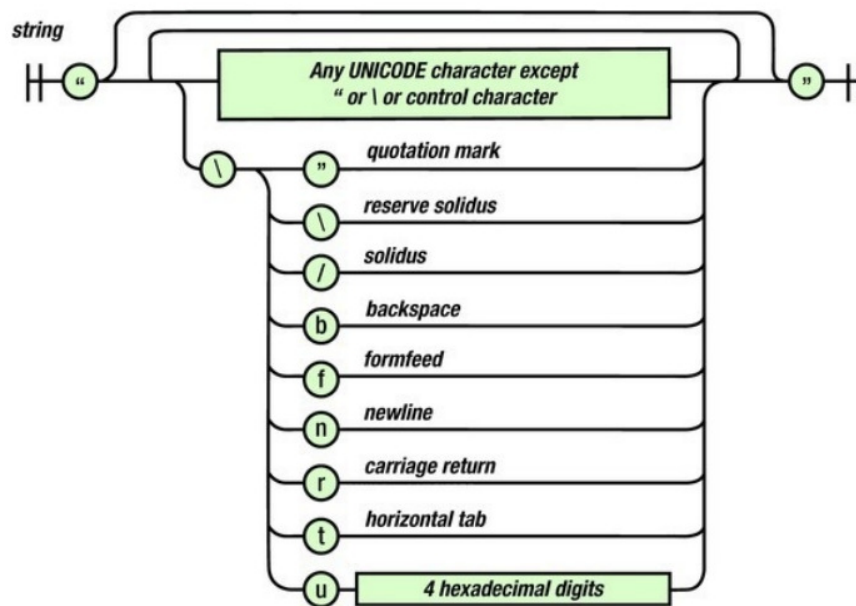
Strings em JSON são representadas por série composta por 0 ou mais caracteres Unicode, encapsulados por aspas duplas (") (Figura 4). A Figura 5 exibe como um número pode ser representado em JSON. Um número pode ser um inteiro, decimal, ou número em formato científico. Porém um número não pode ser em hexadecimal ou octal (CROCKFORD, 2006) (SMITH, 2015).

2.3 Threads

Threads podem ser vistos como funcionalidades de software que executam quase ao mesmo tempo dentro de um mesmo processo ou programa. É como se um processo se dividisse em outros para executar atividades distintas paralelamente. Algumas das características do thread é poder comunicar com outro thread durante sua execução e compartilhar os mesmos recursos, podendo agilizar o processamento de dados. Ao mesmo tempo que os threads podem acelerar a aplicação, eles podem torná-la mais complexa (TANENBAUM, 2010).

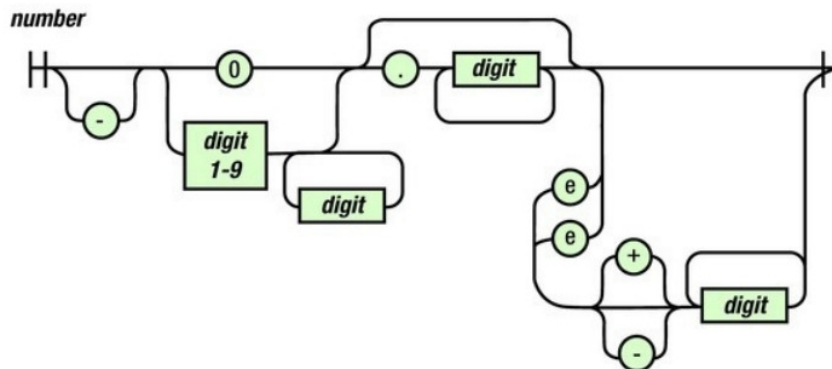
Threads são mini processos que executam quase simultaneamente dentro de outro processo em um mesmo espaço de endereçamento na memória. Um processo, em um sistema operacional tradicional, possui um espaço de endereçamento e um único thread de controle. Porém é comum encontrar situações em que precisamos de múltiplos threads em um mesmo espaço de endereçamento que executem em paralelo (TANENBAUM, 2010).

Figura 4 – Diagrama de sintaxe de string JSON



Fonte: [Smith \(2015\)](#)

Figura 5 – Diagrama de sintaxe de número JSON



Fonte: [Smith \(2015\)](#)

[Tanenbaum \(2010\)](#) cita algumas razões pela qual necessitamos de threads. A primeira razão é o fato de que muitos softwares executam mais de uma atividade simultaneamente. Além de tornar o processo mais simples quando este é decomposto em múltiplos threads.

A segunda razão descrita por [Tanenbaum \(2010\)](#) é a facilidade com a qual se pode criar e destruir threads. Diferentemente de processos, os threads não possuem nenhuma

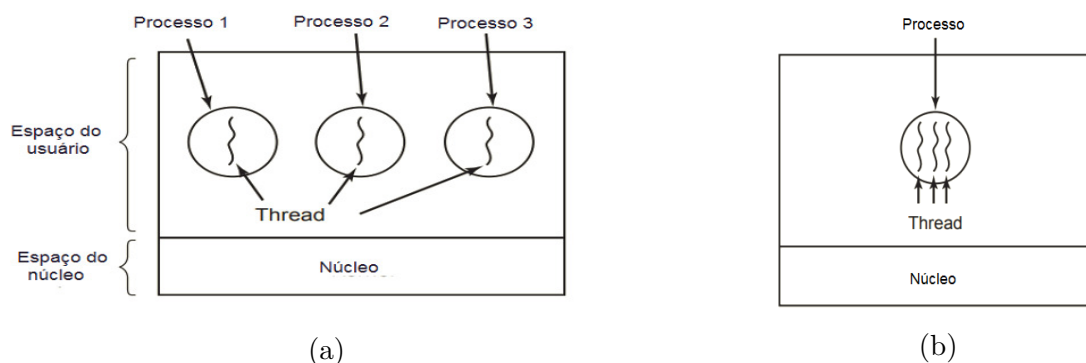
associação com recursos. Sendo uma característica importante quando o número de threads aumenta dinamicamente.

Outra razão citada por [Tanenbaum \(2010\)](#) é o aumento da velocidade de execução da aplicação. Contudo, esse aumento do desempenho é percebido quando existe uma grande quantidade de dados a serem processados, ou seja, há uma necessidade computacional grande. Quando há muito poder de processamento e poucos dados a serem processados, não ocorre o aumento de desempenho.

E o último argumento dado por [Tanenbaum \(2010\)](#) é baseado na utilidade de threads em sistemas com múltiplas CPUs, onde o paralelismo ocorre de fato.

A Figura 6a mostra três processos, e cada um destes possui um espaço de endereçamento com um único thread de controle. Na Figura 6b, há três processos, cada um com seu espaço próprio de endereçamento, e cada um com três threads compartilhando o mesmo espaço ([TANENBAUM, 2010](#)).

Figura 6 – Threads. **(a)**: Representação de três processos, cada um com um thread. **(b)**: Processo com três threads.



Fonte: [Tanenbaum \(2010\)](#)

2.4 Programação assíncrona

O conceito de programação assíncrona é similar ao conceito de thread, ambas servem para executar atividades em paralelo. Entretanto, a manipulação na programação assíncrona é mais simples do que em threads, pois mantêm comunicação direta com o thread principal. No Xamarin, a programação assíncrona é utilizada para retirar um processo da thread principal do Android, chamada de UI thread, e executá-lo em segundo plano. A programação assíncrona é utilizada quando é necessário que o sistema execute uma atividade que exija muito poder computacional, e de alguma forma comprometa o desempenho do aplicativo ([PETZOLD, 2016](#)).

2.5 Android

O Android é um sistema operacional (SO) *open source* (código fonte aberto) do Google desenvolvido para dispositivos móveis como smartphones, tablets e smartwatches. O sistema é baseado no kernel 2.6 do Linux e possui um banco de dados nativo, o SQLite. O sistema é desenvolvido pela *Open Handset Alliance* (OHA), uma aliança originada de um convênio entre várias empresas do ramo tecnológico, dentre elas o Google (LECHETA, 2015).

Por ser *open source*, o sistema está continuamente em desenvolvimento. Existem comunidades de desenvolvedores voltadas a construir suas próprias versões modificadas do Android. Nestes sistemas são adicionadas novas características ao Android e buscam melhorar funcionalidades de versões oficialmente liberadas (PEREIRA; SILVA, 2009).

2.5.1 Arquitetura

A arquitetura do Android está representada na Figura 7 e é dividida nas camadas:

- Kernel do Linux
- Bibliotecas
- Android Runtime
- Application Framework
- Aplicações.

A camada de kernel é o núcleo de um sistema operacional. Responsável por administrar as principais funcionalidades de um SO. No Android, a camada de kernel Linux é responsável pelo gerenciamento de memória, controle de processos, threads, drivers e configurações de segurança.

A camada de Bibliotecas possui um conjunto de bibliotecas, que são utilizadas pelo sistema. Em programação, bibliotecas são uma coleção de funções e rotinas para manipulação de dados. Estas auxiliam no desenvolvimento de softwares e evitam a reescrita de códigos complexos (PEREIRA; SILVA, 2009).

Segundo PEREIRA e SILVA (2009), a camada Android Runtime é uma máquina virtual *Dalvik* criada para cada aplicação executada. Uma máquina *Dalvik* é uma máquina com melhor desempenho e maior integração com hardwares atuais. A máquina foi elaborada com o intuito de executar processos paralelamente, e funcionar em máquinas com pouca memória RAM.

Figura 7 – Arquitetura Android



Fonte: PEREIRA e SILVA (2009)

A camada de framework contém os recursos utilizados por aplicativos. Permite que uma aplicação acesse informação de outra aplicação ou que compartilhem dados entre si (PEREIRA; SILVA, 2009).

E a camada de nível mais alto da arquitetura, a camada de aplicação, é onde se encontram todas as aplicações que são executadas sobre o Android.

2.5.2 Versões

A primeira versão do Android, o Android 1.0, foi lançada comercialmente em 2008. Com lançamentos quase anuais, o Android encontra-se na versão 8.1. A partir da versão 1.5, todas as versões são apelidadas com nomes de doce em inglês. O primeiro dispositivo a ter o Android como sistema foi HTC Dream. A Tabela 1 lista as versões do sistema liberadas, o ano de lançamento e suas principais características (LECHETA, 2015).

O aplicativo Sistema de Leitura da Medição foi desenvolvido para a versão 5.1 até a 6.0 do Android, as mesmas para as quais o MOM foi desenvolvido.

Tabela 1 – Versões do Android.

Versão do Android	Ano	Características
Cupcake (1.5)	2009	Suporte a <i>touch screen</i> ; Suporte a widgets; Teclado virtual;
Donut (1.6)	2009	Suporte a múltiplas resoluções de tela; Pesquisa por voz;
Eclair (2.0/2.1)	2009	Interface mais amigável; Permitiu adicionar mais de uma conta de email ao mesmo tempo; suporte ao HTML 5;
Froyo (2.2)	2010	Sistema mais rápido; Suporte ao Flash;
Gingerbread (2.3)	2010	Possui sensor de movimento; Suporte a dispositivos com câmera frontal;
Honeycomb (3.0/3.1/3.2)	2011	Voltado a tablets; Botões de voltar e voltar para o início (<i>home</i>) fazem parte da tela;
Ice Cream Sandwich (4.0)	2011	Unificou o sistema entre smartphones e tablets;
Jelly Bean (4.1/4.2/4.3)	2012	Melhorias no sistema e na interface;
KitKat (4.4)	2014	Melhorias no sistema e na interface; Permitiu executar o sistema em dispositivos com memória RAM abaixo de 512MB
Lollipop (5.0/5.1)	2014	Barra de status dupla; Múltiplos usuários de sistema
Marshmallow (6.0)	2015	Melhorias no sistema e na interface; Suporte para leitores de impressão digital; Modo não pertube;
Nougat (7.0/7.1)	2016	Melhorias no sistema e na interface; Função de multi-janela;
Oreo (8.0/8.1)	2017	Melhorias no sistema e na interface;

2.6 Template Matching

É uma técnica de processamento de imagem, que consiste em identificar uma região da imagem baseada em imagem pré-selecionada de uma imagem semelhante à mesma região, a qual chamamos de *templates* (figura 8). Este processo é feito através de cálculos de similaridade entre as duas imagens, sendo que a imagem utilizada como *template* passa sobre toda a imagem buscando a região com maior similaridade.

A Equação 2.1 calcula a diferença dos quadrados entre a imagem I e a imagem usada como *template* T para encontrar a combinação (*matching*) entre ambas. Onde x' e

Figura 8 – Exemplo de *template*

Fonte: O autor

y' são as coordenadas no *template* T da imagem.

$$dist(x, y) = \sum_{x', y'} (T(x', y') - I(x + x', y + y'))^2 \quad (2.1)$$

2.7 Limiar Adaptativo

São utilizadas 3 técnicas de limiar adaptativo para redução de ruído na imagem, técnica que consiste em definir um valor limiar e separar os valores de pixels em dois grupos: um com valores de pixel acima do limiar e outro com valores abaixo. O algoritmo proposto por SUAVOLA calcula o melhor corte baseado em no desvio padrão da variação da escala de cinza(S) na Imagem, representada na Equação 2.4:

$$T_{Savvola} = m * (1 - k * (1 - \frac{S}{R})) \quad (2.2)$$

onde m é a media do valor dos pixels, k e r são constantes de 0.5 e 128, respectivamente. Este algoritmo é utilizado quando a uma diferença expressiva entre os valores dos pixels dos dígitos com relação ao display em qual está inserido.

O método proposto por Wolf normaliza o contraste e a média dos tons de cinza através do cálculo (Equação) do melhor limiar:

$$T_{Wolf} = (1 - k) * m + k * Max + k * \frac{S}{Min} (m - Max) \quad (2.3)$$

onde k é uma constante de valor 0.5, Min e Max são, respectivamente, o menor e o maior valor de tom de cinza da imagem e S é o maior desvio padrão de todas as regiões da imagem analisada.

O histograma local dependente calcula o melhor valor de limiar para cada pixel, levando em consideração o valor médio de sua vizinhança e o valor de limiar C , equivalente a 128. Se o valor da imagem no ponto (x, y) é menor que T na Equação, o valor é então é eliminado.

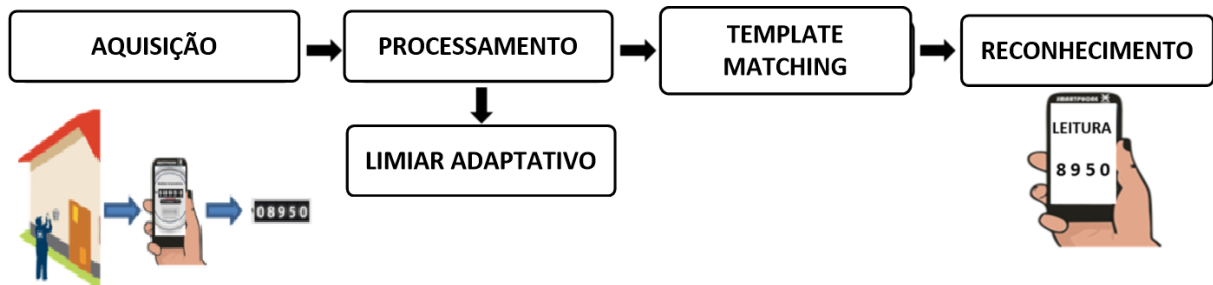
$$T(x, y) = \frac{\sum_{x', y'} I(x', y')}{n} - C \quad (2.4)$$

onde x e y são as coordenadas dos pixels dentro da janela analisada ao redor do pixel.

3 Metodologia

O funcionamento do Sistema de Leitura da Medição, o SILEM, consiste em realizar a captura de uma sequência de imagens buscando a área dos dígitos com maior similaridade a uma imagem pré-selecionada de região de dígitos, carregada pelo sistema, correspondente ao tipo de medidor a ser medido, o *template*, processar esta imagem e então retornar ao usuário o valor consumido pelo cliente. Esse processo de busca pela área dos display é feito também por *template matching* e é chamado de *tracking*.

Figura 9 – Etapas da metodologia do SILEM



Fonte: O autor

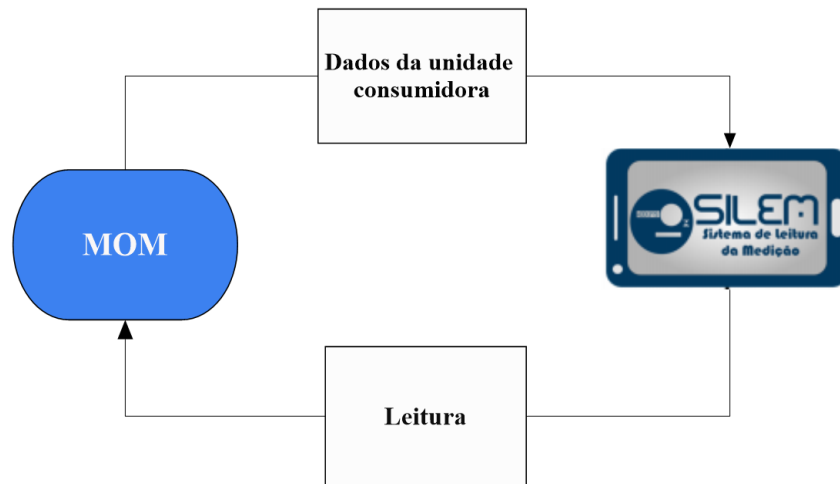
O SILEM foi projetado para ser integrado com a aplicação encarregada pela anotação manual que já é utilizada pelos leituristas. Um dos benefícios dessa integração é proporcionar que o SILEM obtenha informações sobre o aparelho de medição. No nosso caso de estudo, o aplicativo utilizado pela CEMAR e CELPA é o MOM. O MOM é, atualmente, o módulo responsável pela medição e emissão de fatura remota de ambas as companhias. A Figura 10 demonstra o funcionamento da integração do SILEM com o MOM.

Em alguns casos, o funcionário incumbido pela medição não poderá realizar a captura com o aplicativo devido às suas condições de funcionamento, tais como distância e iluminação, por esta razão o objetivo do SILEM não é substituir o processo manual de anotação, mas ser uma alternativa mais rápida e confiável.

3.1 Reconhecimento dos dígitos

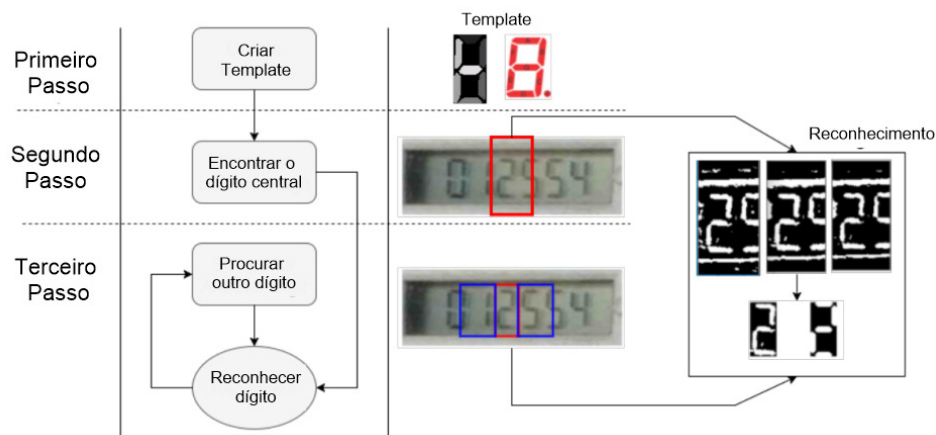
A combinação dos três métodos de limiarização descritos anteriormente são utilizados nesta metodologia de reconhecimento dos dígitos em medidores digitais. A Figura 12 representa a metodologia.

Figura 10 – Arquitetura do SILEM



Fonte: O autor

Figura 11 – Metodologia de reconhecimento de dígitos

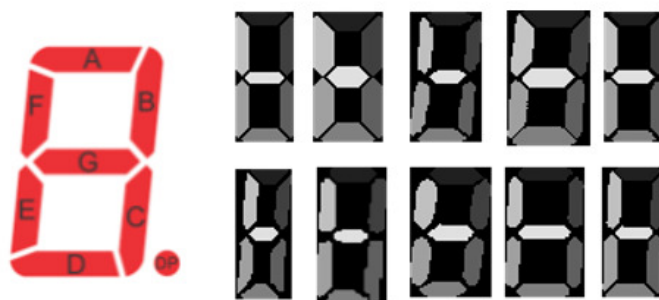


Fonte: O autor

Primeiramente, cria-se *templates* dos dígitos de sete segmentos que serão utilizados no reconhecimento. Cada segmento é nomeado de A a G. Devido a diferenças no formato e orientação dos dígitos, foram criados *template* de dígito para modelo de medidor. Na Figura vemos alguns *templates* utilizados. Com as informações adquiridas do MOM, podemos identificar o *template* a ser utilizado no reconhecimento apropriadamente.

Cada segmento do dígito é codificado com um valor de pixel, começando com valor de 32 para o primeiro segmento e a cada segmento seguinte acrescenta-se 32 ao valor de pixel, assim, o ultimo segmento terá valor de 224. Esse esquema de cor torna o método de *template matching* mais simples.

Figura 12 – Template dos dígitos



Fonte: O autor

Após criar-se cada template, o passo seguinte é encontrar o dígito mais ao centro do display. A partir deste dígito, estimamos a altura e largura dos demais dígitos do display. Para encontrar o dígito central, primeiro temos que achar o pixel central do display baseado na altura e largura do display. Então, a partir deste pixel, cria-se um retângulo em torno deste com largura 2 vezes maior que a largura do *template* e com altura igual a altura do display. Isto é feito porque não sabemos exatamente o tamanho do dígito. O *template* é redimensionado em 60% do seu tamanho e então suas dimensões vão aumentando até encontrar o dígito no tamanho correto, o *template* é redimensionado até 200% do seu tamanho. Então o tamanho do dígito é utilizado nos demais. Estes dígitos são buscados em ambas as direções a partir do dígito central e a busca termina quando não há mais dígitos no display.

Como último passo, aplicamos o limiar adaptativo e comparamos cada dígito do template. O reconhecimento é baseado nos índices abaixo:

- *Matching Index (M)*: mede o quão similar a região que está sendo testada é do *template*;
- *Complementary Matching Index (IM)*: mensura o quão similar é a região testada ao *template* complementar;
- *Noise Index(N)*: verifica o quão similar é a região testada às áreas que não são parte dos dígitos do *template*.
- *Loss Index(L)*: índice de erro de *matching* .

O índices são combinados através da Equação. A razão disso é analisar o grau de similaridade entre a região testada e o *template* considerando as diversos fatores que influenciam o resultado. No fim do processo, o dígito reconhecido é a região que tem maior

similaridade.

$$S = M * (1 - IM) * (1 - N) * (1 - L) \quad (3.1)$$

Se o valor de *matching* for alto mas ainda sim tiver muitas perdas, este é então ponderado para ter a melhor similaridade nas três técnicas de limiar adaptativo. Nós usamos um aumento do algoritmo de conjunto com cortes com intuito de de selecionar o melhor peso de cada índice para melhorar a precisão.

3.2 Aplicação

Nesta seção, descrevemos o funcionamento do SILEM. O SILEM foi implementado no Visual Studio usando com o Xamarin.

3.2.1 Requisitos

Para o desenvolvimento de sistema é essencial definir as premissas e necessidades tanto do sistema quanto do usuário. As transcrições dessas necessidades são chamadas de requisitos do sistema ou do usuário. [Sommerville \(2011\)](#) define requisitos como descrições do que o sistema deve fazer, os serviços que este deve oferecer e suas restrições. Sendo a Engenharia de Requisitos (*requirements engineering* - RE) a responsável por descobrir, analisar, documentar e verificar os serviços e restrições de um sistema.

[Sommerville \(2011\)](#) comenta o fato de que o termo requisito não é usado de forma constante dentro da indústria de software. O termo pode ser usado como um declaração abstrata de forma geral e alto nível do serviço fornecido ou de uma restrição do sistema. Este também pode ser definido de forma mais específica, como sendo um detalhamento formal e minucioso de cada funcionalidade, serviço ou restrição do sistema projetado.

Estes níveis de descrição são distinguidos com os seguintes termos: requisitos de usuários e requisitos de sistemas. Os requisitos de usuário são as declarações de alto nível que dizem ao usuário o que sistema pode oferecer e suas restrições, fazendo uso da linguagem natural e de diagramas ([SOMMERVILLE, 2011](#)). Já os requisitos de sistema, ou especificação do sistema, são os requisitos mais específicos. Estes detalham todas as funcionalidades do projeto a serem desenvolvidas. Sendo esta divisão importante pois, assim, permite que os requisitos passem informações claras e coesas para qualquer leitor.

Além da divisão descrita anteriormente, requisitos também são classificados em requisitos funcionais e requisitos não funcionais.

3.2.2 Requisitos Funcionais

Requisitos funcionais são assertivas que dizem o que o sistema deve fazer, também podem indicar o que o sistema não deve fazer e como o sistema deve se comportar e quando este deve interagir a ações dos usuários (SOMMERVILLE, 2011).

A Tabela ?? mostra os requisitos funcionais de usuário do aplicativo e na Tabela 2 temos os requisitos de sistema que corresponde ao detalhamento do requisito de usuário.

Tabela 2 – Requisitos

1	Capturar imagem com o dispositivo móvel
2	Reconhecer o valor de consumo elétrico na imagem
3	Mostrar ao usuário o valor reconhecido
4	Em caso de erro, permitir que o usuário possa corrigir a leitura antes de sair da aplicação.

3.2.3 Requisitos Não Funcionais

Requisitos não funcionais são requisitos relacionados a restrições ou funções do sistema. Estão ligados à características como usabilidade, confiabilidade, tempo de resposta e restrições de hardware. Diferentemente dos requisitos funcionais, os requisitos não funcionais podem ser aplicados sistema em si, não somente a uma funcionalidade específica(SOMMERVILLE, 2011).

Através de conversas e reuniões com professores, colegas de trabalho, clientes e leituristas, chegou-se aos requisitos não funcionais descritos na Tabela 3. O requisito 1 da Tabela 3 refere-se ao tempo desde o início da aplicação até o momento que o usuário confirma a leitura, considerando-se também o tempo que o sistema leva para adquirir uma imagem apropriada e o de processamento da mesma. O número de cliques no sistema deve ser inferior ou igual ao número de cliques que o leiturista geralmente realiza ao digitar manualmente o valor de consumo, que é equivalente a no mínimo 4 cliques. Com base nesta análise, chegou-se ao requisito não funcional 2.

Tabela 3 – Requisitos não funcionais

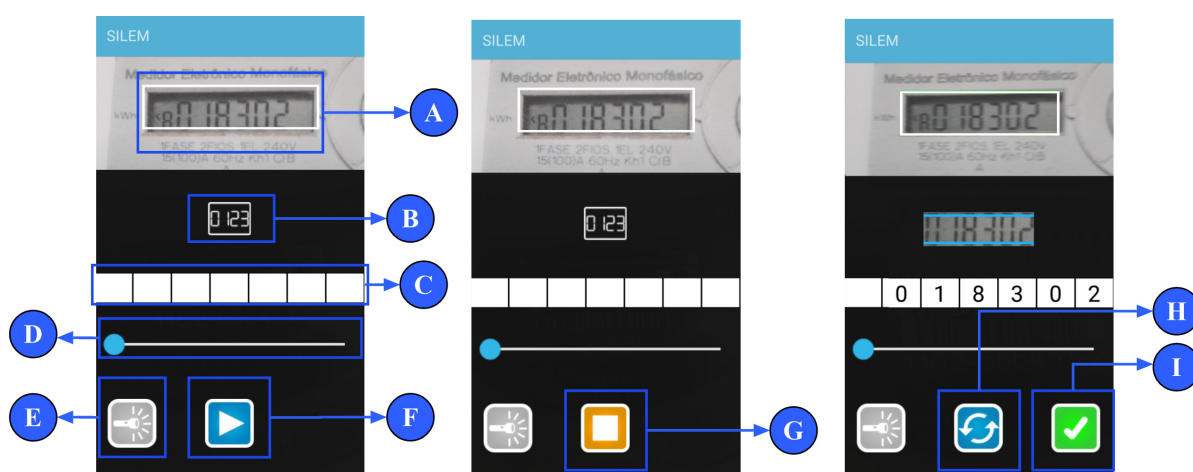
1	O tempo de uso do SILEM não deve ultrapassar 30 segundos
2	Quantidade de cliques para operar o SILEM não deve ser superior a 5
3	Deverá ter uma interface visual compatível com a interface do MOM

3.2.4 Interface

A interface é a parte de um sistema responsável pela interação entre os usuários e o próprio sistema. Isto é, permite a comunicação entre as ambas as partes. Considerando os elementos que fazem parte da interação entre o usuário e sistema, A interface pode incluir tanto o software quanto o hardware (PRATES; BARBOSA, 2003). A interface busca auxiliar o usuário a executar sua tarefa da forma mais confortável possível. A interface do SILEM foi desenvolvido baseando-se nesta premissa.

A interface principal do SILEM é mostrada na Figura 13. Durante o uso do sistema, alguns elementos dessa interface mudam de aparência, com a pretensão de transmitir uma mensagem mais clara ao usuário. Os elementos da interface e as razões pelas quais estes fazem parte da interface são discutidos nesta seção.

Figura 13 – Interface do SILEM



Fonte: O autor

Com a intenção de indicar ao usuário a área de interesse a ser capturada pelo *tracking* do SILEM, uma mira foi adicionada a interface. A mira, em “A” na Figura 13, possui o formato de um retângulo e deve ser posicionada de forma que os dígitos da medição estejam dentro do retângulo. O tamanho do retângulo é baseado no tamanho dos *templates* de imagens de áreas dígitos retirados de imagens adquiridas em campo, que estão presentes no base de imagens enviadas ao Setor de Crítica da CEMAR e da CELPA. Com isto se torna possível encaixar o dígitos no retângulo de forma mais precisa, pois mantêm a proporcionalidade de altura e largura.

Outra vantagem da presença da mira na interface é reduzir o processamento da imagem e tornar o SILEM mais rápido. Caso, a imagem completa capturada fosse enviada ao processamento seria necessário adicionar outras função como segmentação do display e

da área dos dígitos. Com o auxílio da mira, é possível cortar a imagem na área selecionada. Sendo que estas duas teriam um custo computacional grande exigindo mais recursos do dispositivo, que por sua vez possui poder computacional limitado.

Em alguns lugares não é possível aproximar o dispositivo móvel do medidor pois alguns medidores estão dentro de caixas de acrílico, ou podem estar dentro das casas dos clientes, sendo somente visível através de uma janela ou portão, ou podem haver objetos obstruindo a aproximação do leitorista, tornando o processo de captura da imagem mais difícil. Por esta razão a interface do silem possui uma barra de zoom com a finalidade de permitir que o leitorista consiga posicionar a mira no display com mais facilidade em locais onde o acesso ao medidor é restrito, em “D” na Figura 13,.

Além da dificuldade de aproximação entre dispositivo móvel e o medidor, outro problema que ocorre no cotidiano do responsável pela medição de consumo é a falta de iluminação do local onde o medidor foi instalado. Muitos medidores são instalados em subsolos de prédios ou estão dentro de caixas escuras, atrapalhando a visibilidade do medidor. Em vista de contornar situações como estas, também foi adicionado um botão de lanterna. O botão, localizado em "E" na Figura 13, ativa a funcionalidade de lanterna do dispositivo e só é desativada quando o usuário pressiona novamente o botão. Porém nem todo o equipamento disponibilizado pela empresa possui a lanterna como parte de seu hardware.

O Botão de lanterna é representado por um desenho minimalista de uma lanterna na cor branca e é apresentado de duas formas. Na Figura 14, os dois estados do botão são apresentados. Quando não está ativo, a cor de fundo do botão é cinza, e quando o usuário pressiona o botão e inicia a funcionalidade, a cor de fundo torna-se azul.

Figura 14 – Botão que ativa ou desativa a função lanterna. (a): Lanterna ligada. (b): Lanterna desligada.



(a)



(b)

Fonte: O autor

Em “C” na Figura 13, temos 7 (Sete) caixas de textos também foram postas na interface. Cada caixa suporta um número inteiro variando de 0 a 9. Apesar de haverem sete caixas de texto na interface, apenas 6 são habilitadas para a edição. A quantidade disponível foi definido pela CEMAR e CELPA, sendo que a quantidade de dígitos disponíveis varia entre 5 e 6 em equipamentos digitais. O sétimo elemento é inserido na interface caso um

novo medidor que suporte 7 dígitos seja adicionado ao catálogo dos dispositivos utilizados.

Figura 15 – Caixas de texto dispostas na interface. **(a)**: Caixas de texto ao abrir o SILEM. **(b)**: Caixas de texto após o reconhecimento de dígitos digitais.



Fonte: O autor

Quando a aplicação é iniciada todas as caixas estão desabilitadas para edição ou inserção de dígitos. Após a captura e o processamento da imagem cada caixa é preenchida com um dígito referente a sua posição no valor retornado pelo reconhecimento, representadas na Figura 15b. Então elas são habilitadas para que o usuário possa corrigir algum dígito em caso de erro.

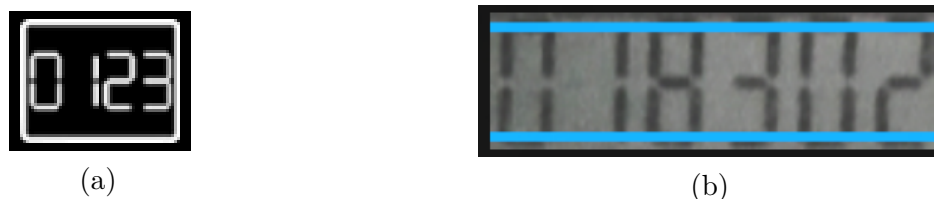
Botão de *play* (“F” na Figura 13), representado por um ícone comum em aplicações de multimídia, é responsável por iniciar o *tracking*. No momento em que o botão *play* é pressionado, ele é prontamente substituído por um botão de *stop* (“G” na Figura 13), também representado por um símbolo considerado padrão. Quando o usuário aperta o botão de parar, o reconhecimento dos dígitos é iniciado. Enquanto o reconhecimento está sendo processado, uma mensagem e uma barra circular de progresso são mostradas ao usuário indicando que ele deve aguardar. Ao término do processamento o botão *stop* é substituído por um botão de *refresh* (“H” na Figura 13), que é responsável por reiniciar o *tracking* sem necessitar reiniciar a aplicação e um botão de confirmação da leitura é mostrado (“I” na Figura 13).

A interface também possui uma imagem representativa do display (Figura 16a), localizada em “B” na Figura 13. Quando o *tracking* é iniciado, essa imagem é substituída pela região mais semelhante ao display na primeira captura (Figura 16b), que serve de *feedback* para o usuário. *Feedback* é uma resposta do sistema ao usuário, serve para indicar o que o sistema está fazendo. Cada vez que o *tracking* encontra uma área com um índice de semelhança maior, a imagem anterior, então, é substituída novamente por esta nova região. Duas linhas horizontais estão dispostas sobre a imagem indicando o tamanho mínimo ideal dos dígitos em medidores digitais para que estes possam ser reconhecidos com precisão. A primeira linha está posicionada na posição equivalente a 10% da altura da imagem, e a segunda linha está localizada no ponto referente à 90% da altura. Sendo esse um requisito para o funcionamento da aplicação.

3.2.5 SILEM

O aplicativo SILEM necessita de alguns arquivos para o seu funcionamento, arquivos para o núcleo de reconhecimento e os *templates* de áreas de dígitos. Estes são copiados

Figura 16 – Imagem de *feedback*. (a): Imagem representando um display. (b): Imagem de *feedback* após encontrar um display.



Fonte: O autor

para a memória interna do dispositivo durante a primeira execução do aplicativo e ficam à disposição do software quando forem requisitados. Estes arquivos são carregados logo após a chamada da aplicação, antes mesmo da interface ser exibida.

Apesar de todos os arquivos serem instalados no dispositivo na primeira execução do sistema, somente arquivos específicos relacionados ao medidor em questão são carregados. Os arquivos a serem carregados são definidos através das informações enviadas do aplicativo que requisitou o SILEM. São estas:

- Valor do código de barras (TAG)
- Modelo do medidor
- Fabricante do medidor
- Nome da companhia.

Essas informações são enviadas da aplicação principal, o MOM, para o SILEM via JSON. Primeiramente, é necessário criar um objeto chamado Silem (Figura 17). O código de barras, o modelo do medidor, o fabricante e nome da concessionária são definidos como atributos do objeto Silem com os nomes de PrefixoMedidor, Modelo, Fabricante e Empresa, respectivamente. Sendo estes os dados necessários para a identificação do medidor.

Para utilizar o MOM juntamente com o SILEM, é preciso entrar no aplicativo. Na tela de login (Figura 18a), é preciso selecionar a empresa, a qual o leitorista representa, selecionar o ambiente (para testes é utilizado o ambiente de qualidade (QAS)), o usuário e senha disponibilizados pela empresa. Após ter acesso ao aplicativo, o MOM mostrará a tela da Figura 18b, que lista as rotas disponíveis, que são os conjuntos de unidades consumidoras que o leitorista deverá anotar o consumo, para este usuário. Depois de selecionar uma rota, o MOM listará as ordens pertencentes a esta (Figura 18c). A seguir, o SILEM exibirá a tela de anotação manual do consumo, além de exibir todas as informações referentes à ordem selecionada (Figura 18d). Nesta tela, temos um botão que representa uma câmera ao lado do campo leitura, este botão é o encarregado por iniciar o SILEM.

Figura 17 – Objeto Silem

```
[Serializable]
public class Silem
{
    public String PrefixoMedidor { get; set; }
    public String Modelo { get; set; }
    public String Fabricante { get; set; }
    public String Empresa { get; set; }

    public Silem()
    {
    }

    public Silem(String PrefixoMedidor, String Modelo, String Fabricante, String Empresa)
    {
        this.PrefixoMedidor = PrefixoMedidor;
        this.Modelo = Modelo;
        this.Fabricante = Fabricante;
        this.Empresa = Empresa;
    }
}
```

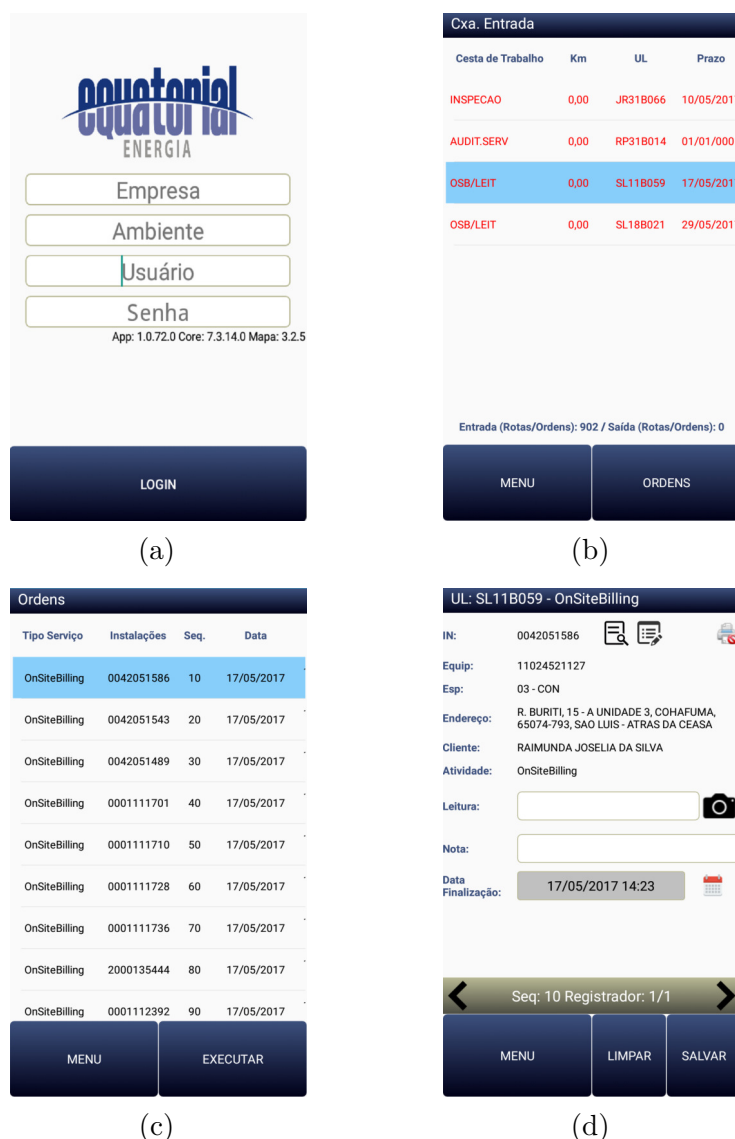
Fonte: O autor

Apesar do SILEM receber o código barras completo, nós precisamos somente de uma parte dele, a qual chamamos de prefixo. O prefixo é o identificador do tipo do aparelho de medição. Na Figura 19a e na Figura 19b, temos exemplos de medidores com os prefixos 1013 e 1102, respectivamente, que podem ser observados abaixo do código de barras. Na maioria dos medidores, o prefixo é composto pelos 4 primeiros dígitos, mas também pode ser a combinação de uma sequência de números com uma sequência de letras no final. Todavia, essa informação ainda não está presente em todos os medidores. A falta do prefixo é uma situação que pode ser contornada obtendo-se o nome do modelo, do fabricante e da companhia. Outro cenário ocorre quando um mesmo prefixo possui dois ou mais modelos de medidores diferentes. Portanto, obter-se todas anteriormente citadas são essenciais para o funcionamento do SILEM.

Para testar a aplicação, foi criado um aplicativo de teste, chamado de Simu, o qual simula a interface do MOM e envia os dados necessários ao SILEM. Diferentemente do MOM, no simulador precisamos digitar as informações referentes a empresa, prefixo, modelo e fabricante. Enquanto no MOM, os dados são adquiridos diretamente do banco de dados da empresa e enviados ao SILEM. A interface, mostrada na Figura 20, possui os campos "Empresa", "Prefixo", "Modelo", "Fabricante", "Leitura" e "Nota". Quando "Leitura" é preenchido com o valor retornado pelo SILEM após o processamento. O campo "Nota" é somente para anotações, mas não é usado de fato. Na Figura 20 exibimos a tela do aplicativo com dados referentes ao medidor 1102.

Após ser chamado, O SILEM buscará o *template* a ser utilizado no *tracking*. A busca pelo *template* é feita pelo nome dos mesmos, que é baseado no prefixo, modelo e

Figura 18 – Telas do MOM. (a): Tela de login. (b): Tela de rotas. (c): Tela de ordens. (d): Tela de anotação do consumo.



Fonte: O autor

fabricante. Os *templates* dos display ficam em pastas nomeadas com o nome da empresa da qual o medidor pertence. Os *templates* são nomeados no seguinte formato: prefixo seguido pelo modelo que é sucedido pelo nome fabricante, sendo que estas palavras são separadas por um *underline* (). Exemplo para o medidor de prefixo 1013: 1013_E22A_LANDYS + GYR. Quando o aparelho de medição não possui prefixo, essa informação é substituída por modelo, ou seja, o modelo é repetido duas vezes no nome do arquivo.

Em seguida, o sistema necessitará de arquivos para o processamento da imagem. Devido ao tamanho destes arquivos, estes são carregados em um thread que é executado enquanto o usuário posiciona a câmera.

Figura 19 – Exemplo de medidores. (a): Medidor com prefixo 1013. (b): Medidor com prefixo 1102.



Fonte: CEMAR

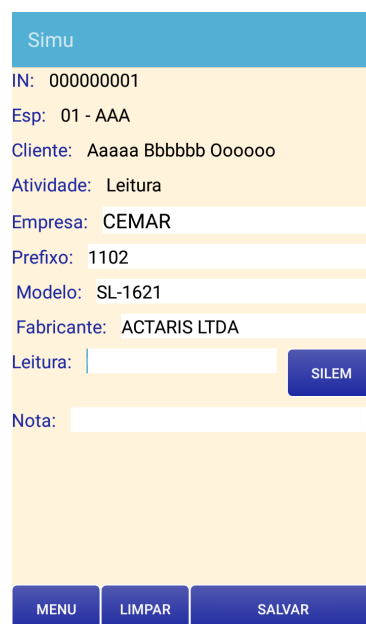
Quando iniciado, o SILEM mostrará a interface referente a Figura 21. A câmera será aberta e o usuário deve posicionar o dispositivo móvel em frente ao aparelho a ser medido de forma que a mira, presente na interface, esteja direcionada na área dos dígitos. O usuário deve estar entre 10 centímetros e 30 centímetros de distância do medidor, podendo aumentar a distância máxima quando utilizar a ferramenta de zoom, ou ligar a lanterna, quando o ambiente estiver escuro. O usuário deverá mover o dispositivo móvel a fim de eliminar qualquer inclinação do medidor. Sendo este o protocolo de aquisição definido.

Após usuário encontrar a posição mais adequada, a captura poderá ser iniciada. A Figura 22a demonstra como a tela é apresentada após a leitura pressionar o botão de play. Com o início do *tracking*, cada *frame* capturado pela câmera é analisado buscando a melhor área de dígitos para o reconhecimento. Toda vez que uma área melhor for encontrada a imagem de *feedback* é substituída e a cor da mira torna-se verde (Figura 22b).

Assim que o usuário parar o *tracking*, o processo de reconhecimento é iniciado. Entretanto, o reconhecimento só irá começar após o carregamento de todos os arquivos que estão na thread criada no início do aplicativo. que este módulo necessita. O processamento também em feito utilizando programação assíncrona. Por ser um etapa que consome muito poder de processamento do dispositivo, o reconhecimento é retirado da UI thread do aplicativo. Sendo assim, é possível realizar algumas atividades no thread principal do aplicativo, e impedir que a aplicação fique travada durante o processo de reconhecimento de dígitos. Enquanto o valor é processado, a tela da Figura 23 é exibida.

Após o processamento, o valor reconhecido é mostrado ao leitorista (vide Figura 24). Nesta etapa, caso o consumo mostrado pelo SILEM não esteja correto, o usuário

Figura 20 – Tela do simulador Simu

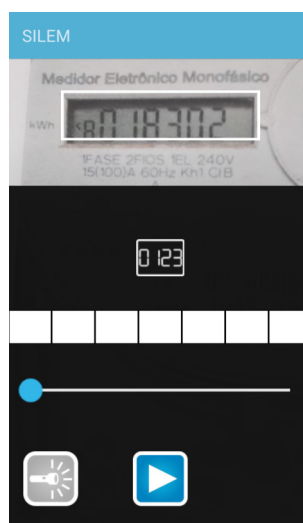


Fonte: O autor

poderá corrigir antes de confirmar. O valor lido de consumo é enviado para o aplicativo da empresa. No MOM, esse valor é processado e, então, a fatura é gerada e entregue ao cliente.

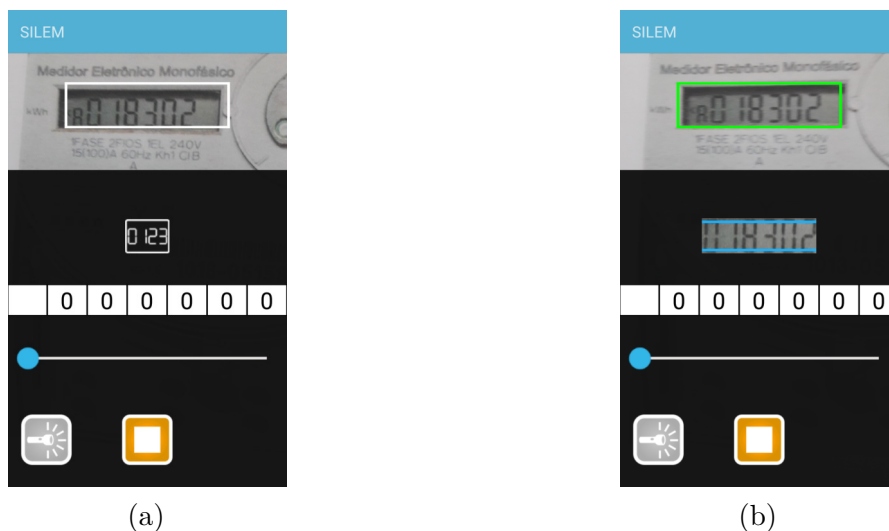
O SILEM foi implementado de maneira que a quantidade de toques na tela seja

Figura 21 – Interface do SILEM



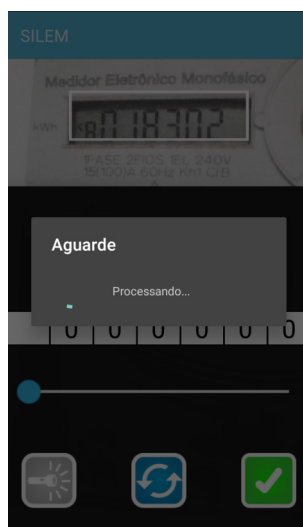
Fonte: O autor

Figura 22 – Telas do SILEM durante o *tracking*. (a): Interface após início do *tracking*. (b): Interface após encontrar área dos dígitos.



Fonte: O autor

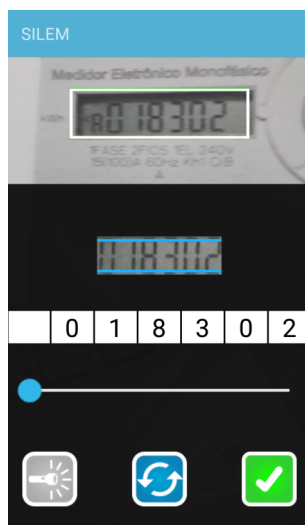
Figura 23 – Tela de espera



Fonte: O autor

menor ou igual ao número de interações que o usuário faz para realizar a anotação manual. Atualmente, na anotação manual, esse número é igual a 4 interações, porém, dependendo do modelo de medidor, o número máximo de toques pode chegar até 6. Considerando que sistema reconheça corretamente todos os dígitos, o total de interações para obter-se a leitura com o SILEM é de 3 toques (o usuário inicia o *tracking*, para o *tracking* e confirma a leitura), assim reduzindo o tempo gasto na tarefa. E com tempo médio de processamento de 5,106 segundos para medidores digitais, o SILEM oferecerá ao leitorista uma forma

Figura 24 – Tela com resultado do processamento



Fonte: O autor

mais rápida de executar sua tarefa.

4 Resultados

Para testar a metodologia testou-se em imagens disponibilizadas pela CELPA e CEMAR. As imagens foram adquiridas do Setor de Crítica. Foi adquirido 2128 imagens de displays LCD de medidores digitais. Cada display foi capturado com o SILEM. Todos os medidores testados possuíam 6 dígitos. Para comparação entre o consumo real e o consumo reconhecido pela metodologia, anotou-se o valor manualmente e associou-se este e o modelo do medidor ao respectivo display.

Os resultado é avaliado pela quantidade de dígitos reconhecidos corretamente. O objetivo principal é reconhecer todos os dígitos do display. Mas também é reduzir o esforço do leitorista ao corrigir os dígitos no SILEM, quando o reconhecimento total não for possível. Então, para análise, os resultados foram divididos em 4 categorias de acertos: todos os dígitos foram reconhecidos (Todos), 5 ou mais (5+), 4 ou mais(4+) e 3 ou mais(+). A Tabela 4 mostra os resultados obtidos.

Tabela 4 – Resultados da metodologia para reconhecimento de dígitos em medidores digitais

Dígitos	Displays	Precisão(%)
All	1536	72.18
5+	1838	86.38
4+	1945	91.40
3+	2017	94.78

A tabela mostra que em 72,18% dos displays todos os dígitos foram reconhecidos. De todas as imagens testadas, em 94,78% dos displays, acertou-se pelo menos 3 dígitos, e em 91,40% reconheceu-se no mínimo 4 dígitos e em 86,38% foram reconhecidos 5 ou mais dígitos. Com, isso percebemos que a metodologia é capaz de minimizar o esforço de correção do consumo de energia elétrica pelo leitorista no SILEM. Também, notamos que a metodologia teve uma acurácia de 90.29% por dígitos, ou seja, foi capaz de acertar 12,768 dígitos de 11,529 dígitos presentes em 2128 displays.

Ao analisarmos as imagens com erros percebemos que alguns ocorrem por erros de inclinação, que apesar de pouca ainda atrapalha os resultados, como acontece na Figura, onde devido a posição do dispositivo móvel no momento da aquisição o dígito ficou conectado a margem do template. Na imagem da Figura temos um exemplo onde todos os dígitos foram reconhecidos corretamente.

4.1 Avaliação

Fazer avaliação do sistema é uma etapa importante do desenvolvimento. É nesta fase que saberemos se o aplicativo está atendendo ao objetivo proposto e quais aspectos devem ser concertados. O usuário é o principal ator aqui. Pois ele é o responsável por indicar as melhorias a serem feitas. O desenvolvedor não deve supor o que o usuário quer ou o que pode ser mais confortável a ele. Também é importante que o software, quando possível, seja testado por mais de um usuário, para que o desenvolvedor tenha um *feedback* sobre a qualidade do sistema consistente. Dito isto, essa avaliação é um dos fatores que permitem a análise de qualidade de software (PRATES; BARBOSA, 2003).

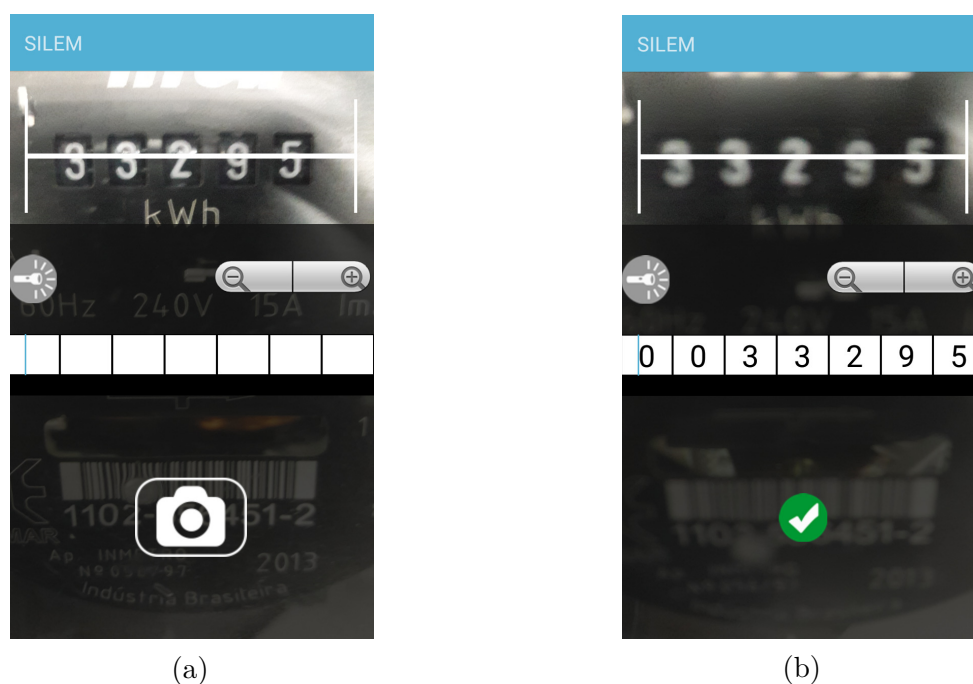
A fim de realizar a avaliação o aplicativo foi construído um questionário constituído de 10 questões sobre a experiência do usuário para com o sistema. Este foi desenvolvido baseado nos princípios de avaliação descritos em Prates e Barbosa (2003). O formulário é formado por 9 questões de múltipla escolha e uma discursiva. Respostas das questões objetivas, possuem repostas que variam de 1 a 5, que dependendo da questão, podem ser classificadas como: variação de muito ruim a muito bom, de discordo fortemente a concordo fortemente, de muito fácil a muito difícil ou, simplesmente, como uma nota. As questões presentes no questionário são:

- A utilização do SILEM foi atrativa e simples?
- Como você avalia a facilidade para aprender a usar o SILEM?
- Como você avalia a facilidade de operação do SILEM?
- Você encontrou dificuldades de entender os ícones, botões ou operações no SILEM?
- A utilização do SILEM durante a leitura auxiliaria no trabalho de medição?
- Você recomendaria o SILEM?
- Como você avalia o tempo que o SILEM demora para enviar o valor de consumo?
- Como você avalia o mecanismo de correção quando o SILEM não consegue reconhecer totalmente o consumo?
- No geral, como você avalia o SILEM?
- Deseja sugerir pontos positivos e negativos? Use esse espaço para sugestões ou críticas ao aplicativo SILEM:

Nas etapas iniciais do projeto, realizou-se uma avaliação preliminar do SILEM na qual uma primeira versão do aplicativo foi testada por três leitores em campo. Nesta versão, o aplicativo possuía uma mira de tamanho único, a qual era composta por

duas linhas verticais e uma linha horizontal, como pode ser observado na Figura 25. Era necessário que os dígitos estivessem entre as duas linhas horizontais e que o centro dos dígitos fosse atravessado pela linha horizontal. Além disso, o processamento era diferente, os arquivos utilizados pelo reconhecimento não eram carregados em threads, o que exigia mais poder computacional do processo principal, e aumentava o tempo necessário de processamento. A imagem de *feedback* também não estava presente. Essa interface inicial pode ser vista na Figura. Além das diferenças na interface, essa versão também possuía diferenças em seu funcionamento. Ao invés de capturar vários *frames* e realizar a busca da melhor área correspondente à área de dígitos do medidor, esta captura apenas um único *frame*, que é processado logo após o usuário pressionar o botão de captura. Ou seja, o aplicativo era mais dependente do usuário.

Figura 25 – Primeira interface do SILEM. (a): Interface para captura. (b): Interface para resultado.



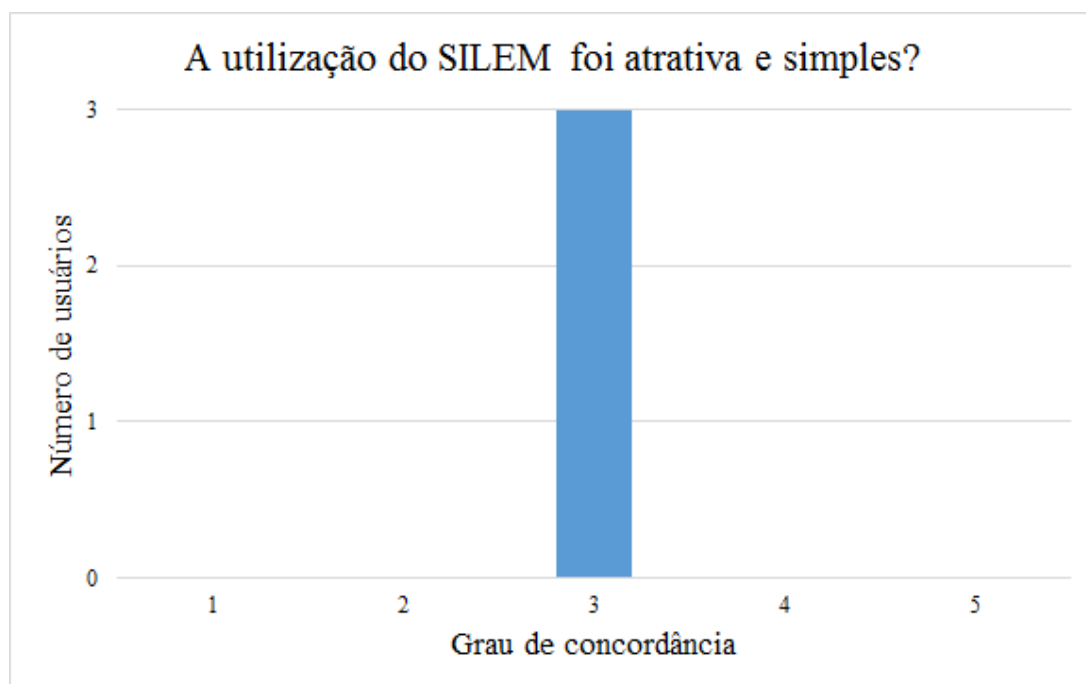
(a)

(b)

Fonte: O autor

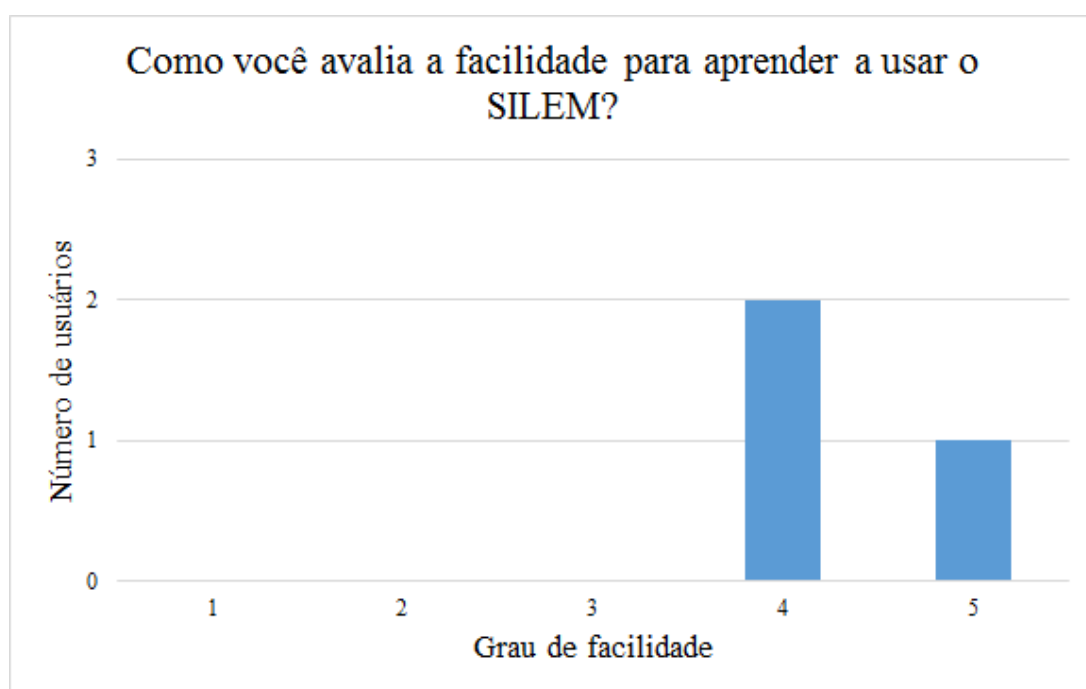
Os resultados desta avaliação são mostrados nos gráficos a seguir. Os gráficos mostrados nesta seção consistem na relação entre o número de usuários que testaram o SILEM e as respostas escolhidas por estes. Isto é, os gráficos demonstram a quantidade de usuários que selecionaram cada opção.

Figura 26 – Gráfico da primeira questão da primeira avaliação



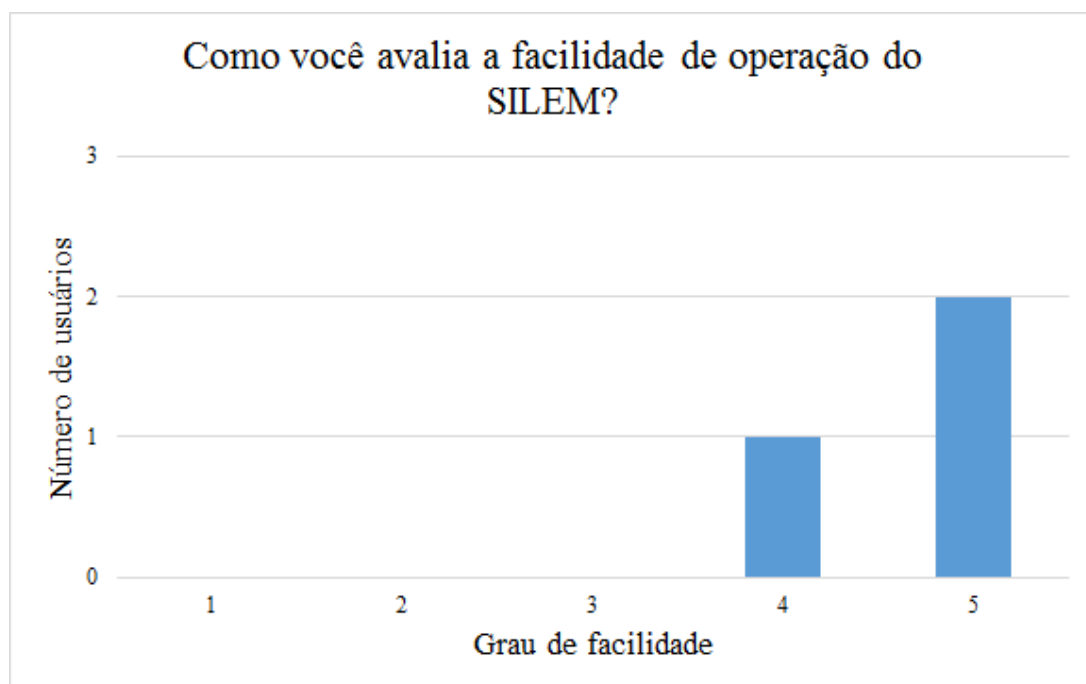
Fonte: O autor

Figura 27 – Gráfico da segunda questão da primeira avaliação



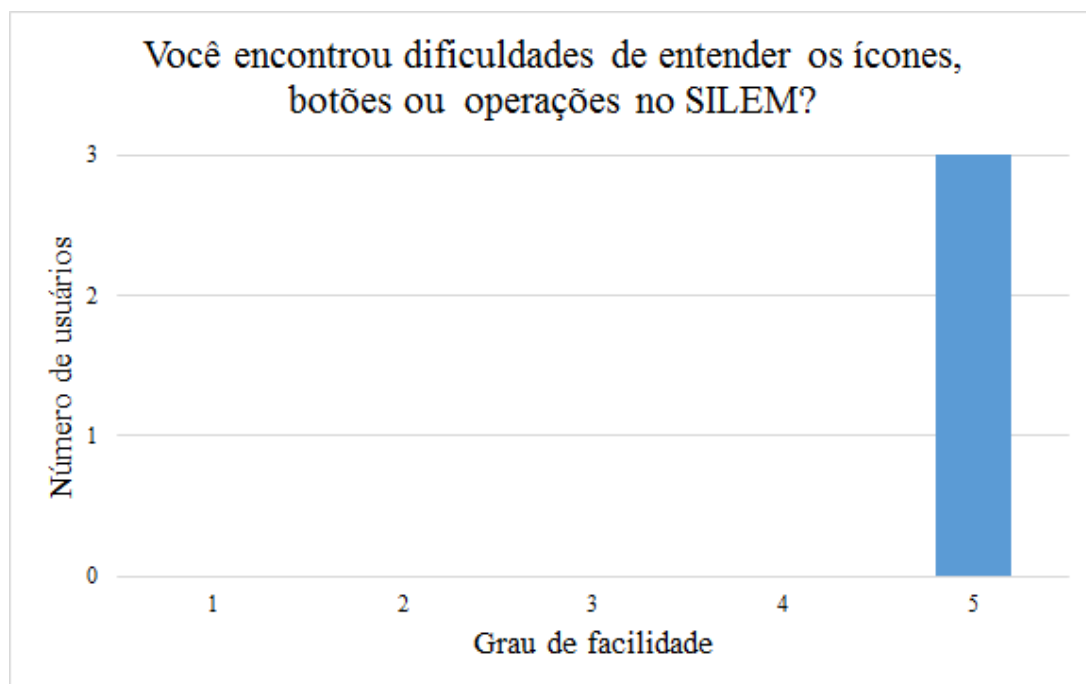
Fonte: O autor

Figura 28 – Gráfico da terceira questão da primeira avaliação



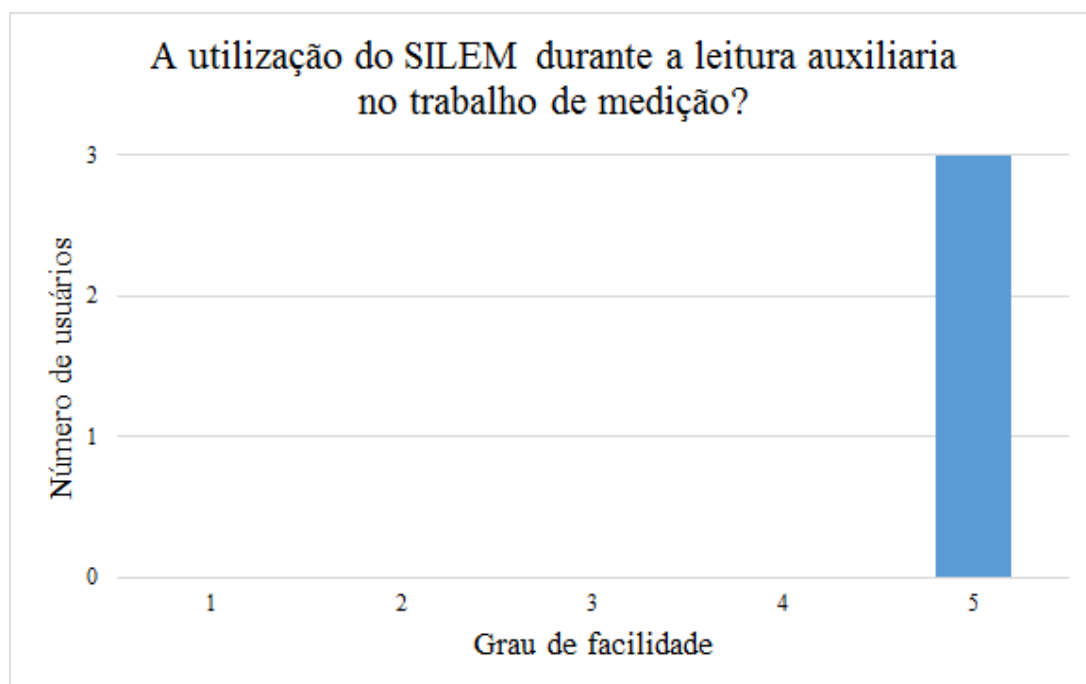
Fonte: O autor

Figura 29 – Gráfico da quarta questão da primeira avaliação



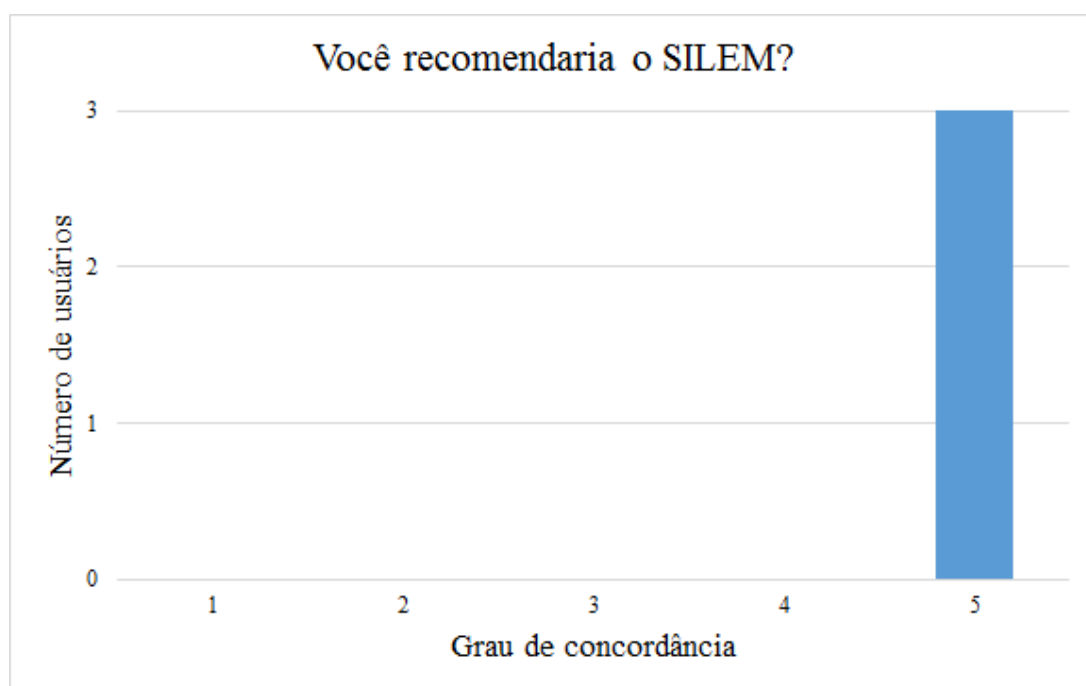
Fonte: O autor

Figura 30 – Gráfico da quinta questão da primeira avaliação



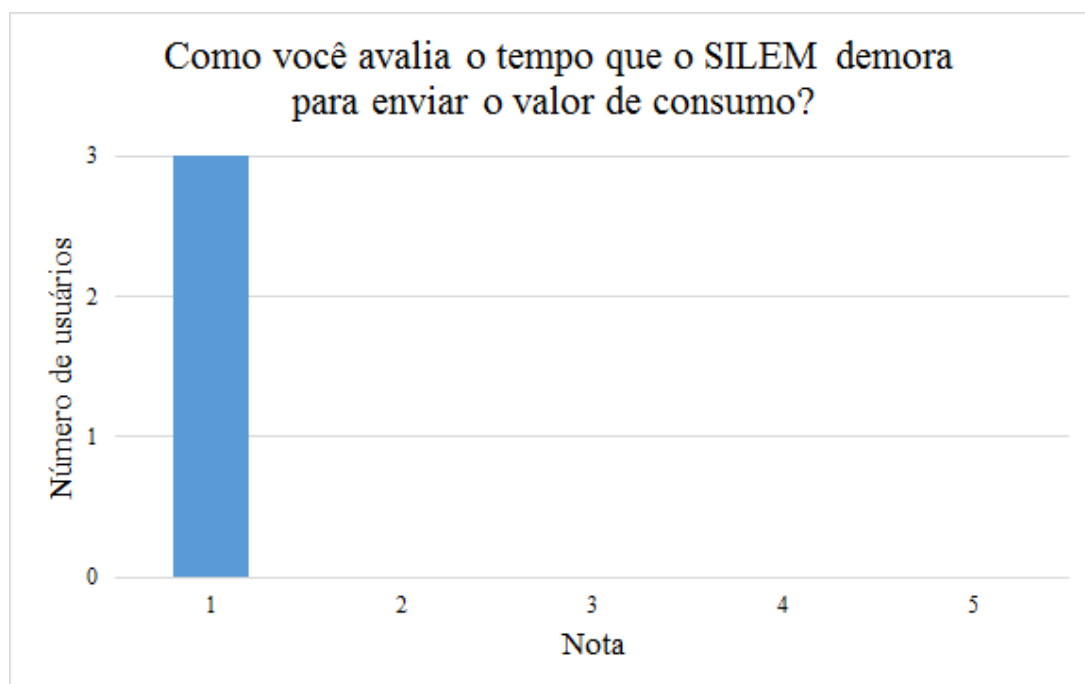
Fonte: O autor

Figura 31 – Gráfico da sexta questão da primeira avaliação



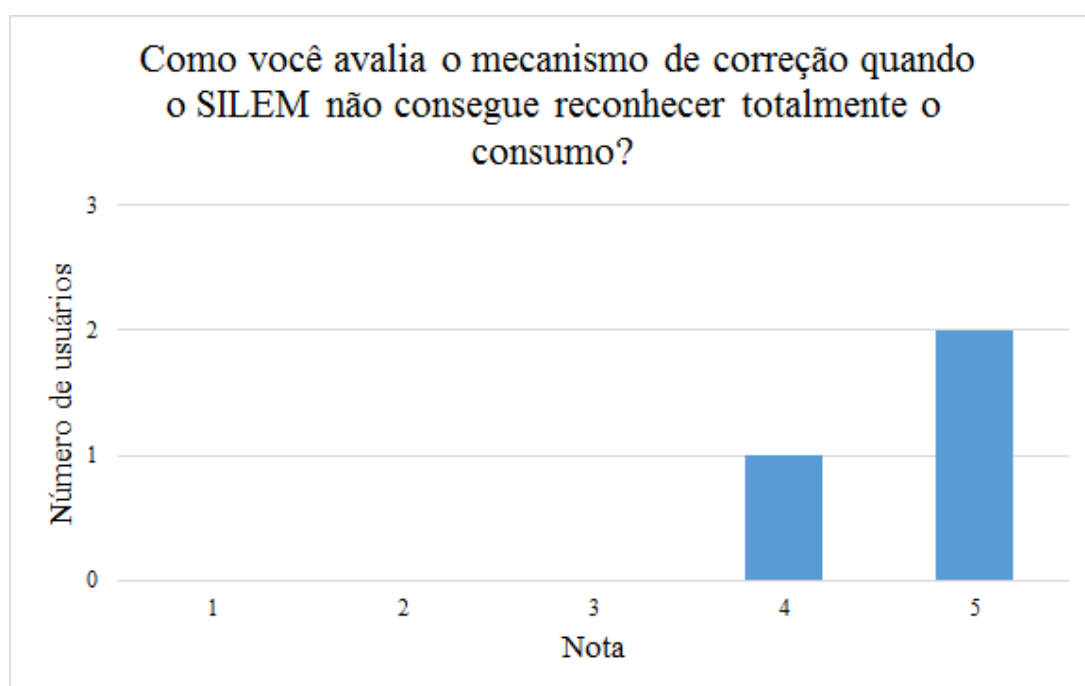
Fonte: O autor

Figura 32 – Gráfico da sétima questão da primeira avaliação



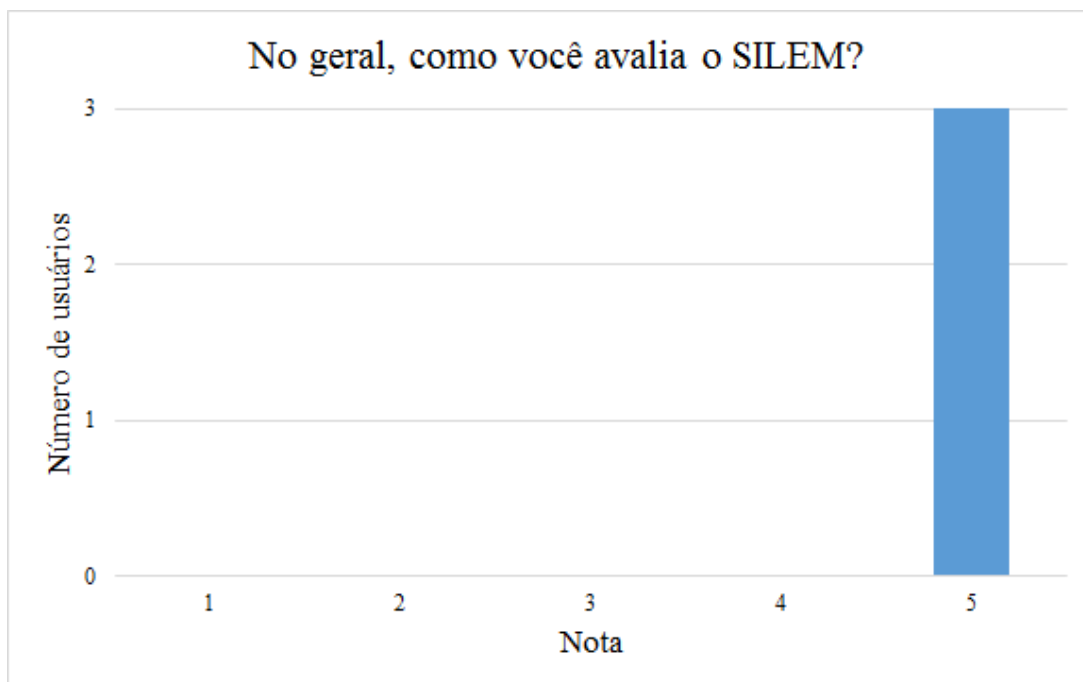
Fonte: O autor

Figura 33 – Gráfico da oitava questão da primeira avaliação



Fonte: O autor

Figura 34 – Gráfico da nona questão na primeira avaliação



Fonte: O autor

Com resultado demonstrado nos gráficos anteriores, podemos perceber que, em geral o aplicativo já estava à caminho de atender seu propósito, porém precisava de ajustes. Como pudemos ver, os três leituristas foram neutros quanto a simplicidade e atratividade do SILEM (Figura 26). Já um deles considerou muito fácil de aprender a utilizar o SILEM, enquanto os demais classificaram como fácil (Figura 27). Quanto a facilidade de operação do SILEM, dois deles classificou este como muito fácil, e o outro como fácil (Figura 28). Todos entenderam claramente os ícones, botões e operações do aplicativo (Figura 29). Todos julgaram que o SILEM será de grande ajuda ao seu trabalho de medição (Figura 30). Todos recomendariam o SILEM para outros usuários (Figura 31). Com tempo médio de 12 segundos de processamento e média de 24 segundos de uso total (tempo desde o início do SILEM até o envio dos dígitos reconhecidos ao MOM), nenhum dos três leituristas aprovaram o tempo (Figura 32). Apesar do tempo está dentro da janela de tempo de 30 segundos definida nos requisitos, os usuários ainda consideraram o processo demorado comparado ao método manual (Figura 33). Quanto ao método utilizado para correção do valor em caso de erro, dois classificaram como muito bom e um como muito bom. Quanto a avaliação geral, todos avaliaram o sistema como muito bom (Figura 34).

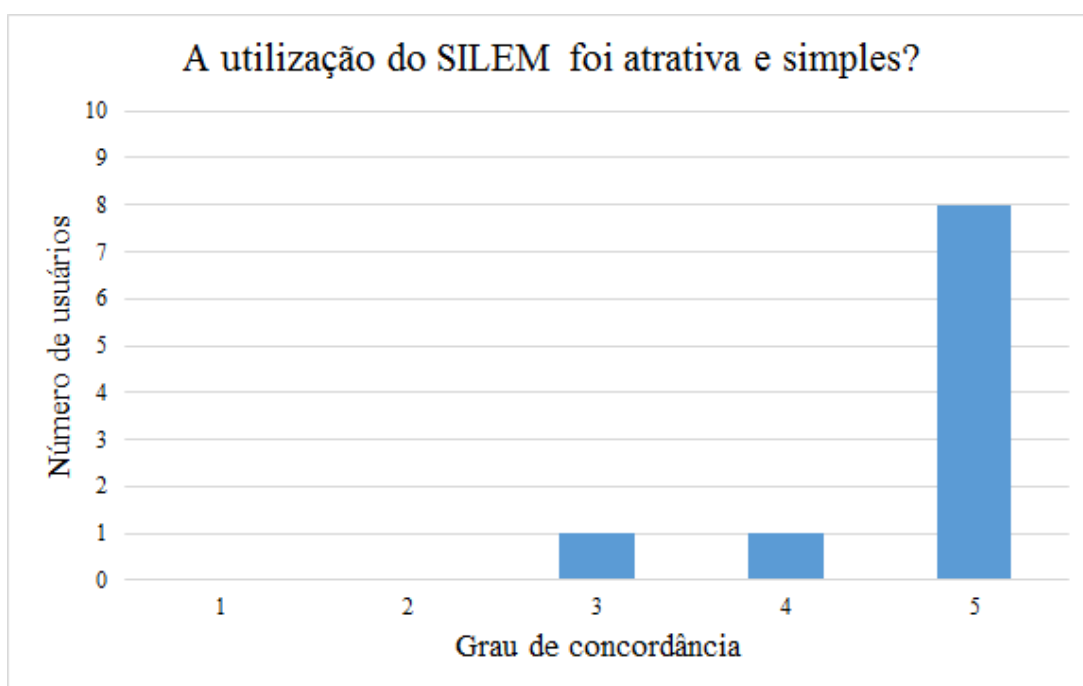
Quando responderam a última questão da avaliação, a qual pedia que eles descrevessem pontos positivos, negativos, críticas ou sugestões, os usuários levantaram os seguinte pontos:

- Redução do tamanho da mira
- Remoção da linha horizontal da mira
- Reduzir tempo de processamento

Durante conversas após o preenchimento do formulário, os leituristas sugeriram que o formato da mira fosse substituído por uma forma retangular, que lhes permitissem encaixar os dígitos mais facilmente dentro da mira. Segundo eles, a linha horizontal dificulta a visualização dos dígitos, impedindo que capturem uma imagem de acordo com o protocolo estabelecido.

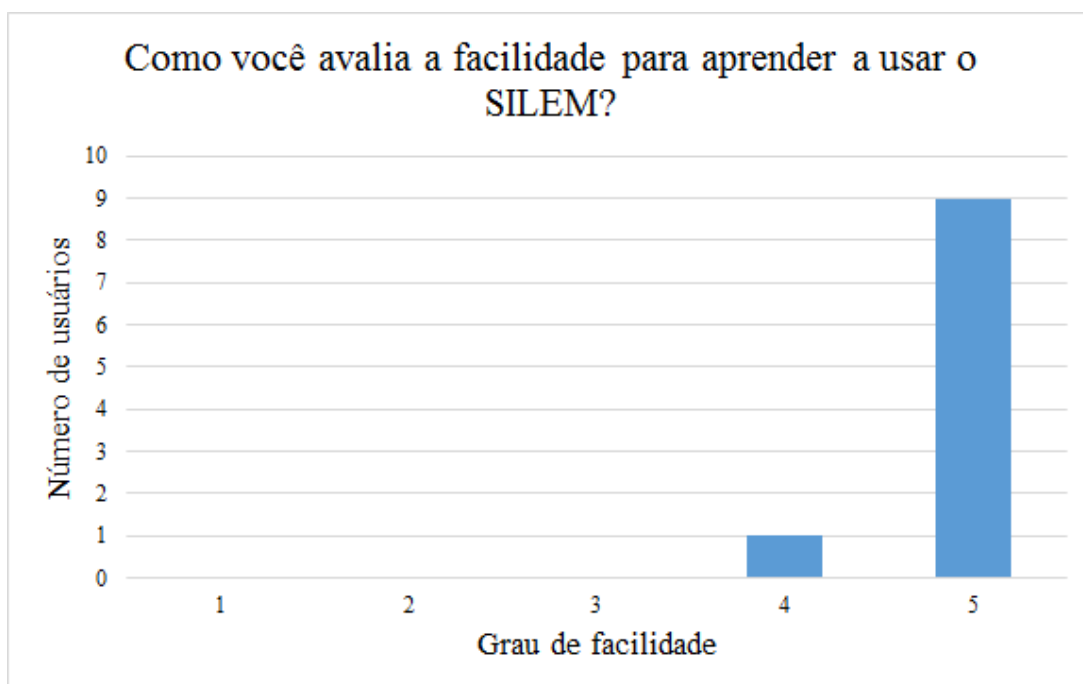
Com todo esse *feedback* adquirido diretamente dos usuários, foi possível alterar a interface e o processamento chegando-se a versão do SILEM descrita nesta monografia. Após as modificações feitas baseadas no formulário e a adição do reconhecimento digital, o mesmo formulário foi aplicado novamente. Desta vez, devido a incompatibilidade nas agendas, o aplicativo foi testado com dez usuários que não são leituristas. Cada usuário foi introduzido ao aplicativo, explicando sua proposta e suas funcionalidades. Os gráficos abaixo nos mostram o resultados obtidos.

Figura 35 – Gráfico da primeira questão da segunda avaliação



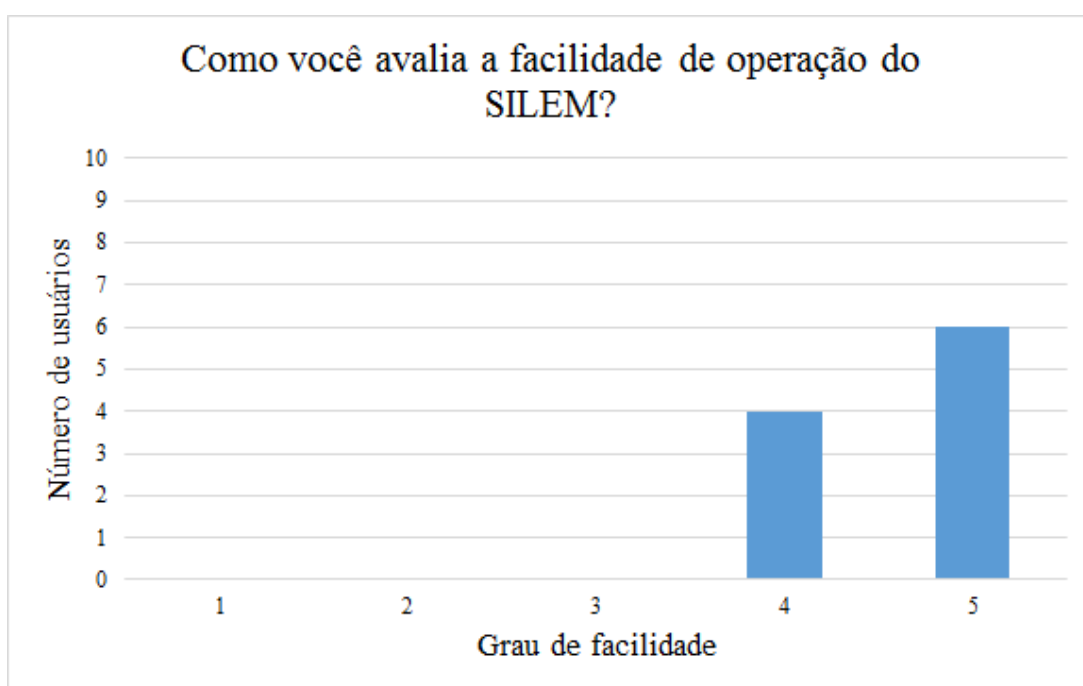
Fonte: O autor

Figura 36 – Gráfico da segunda questão da segunda avaliação



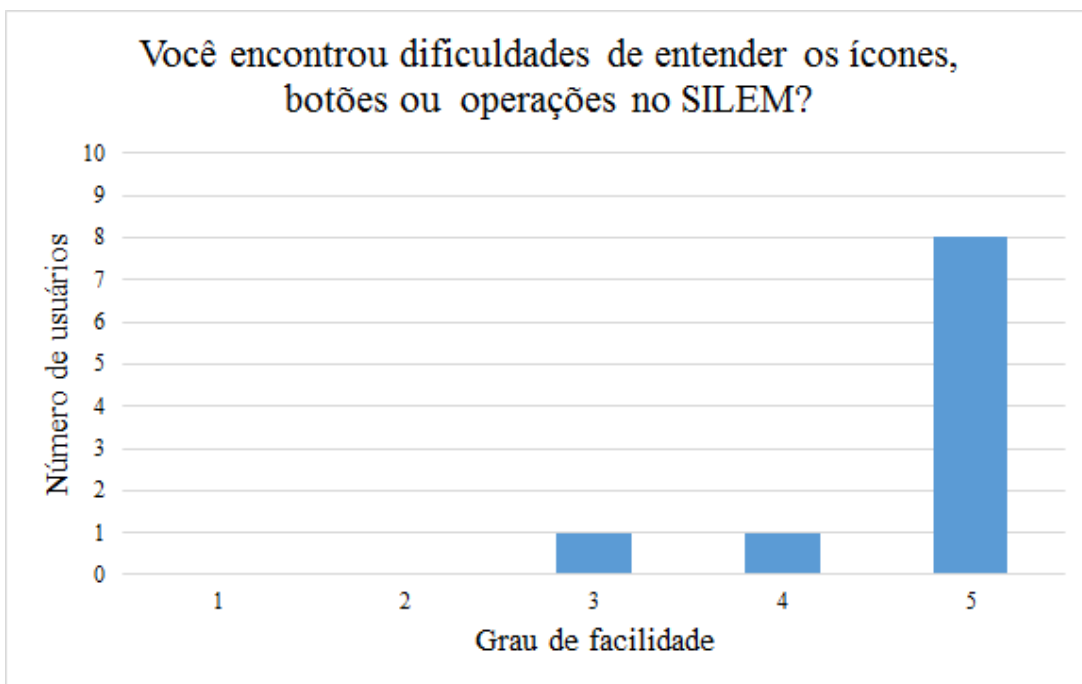
Fonte: O autor

Figura 37 – Gráfico da terceira questão da segunda avaliação



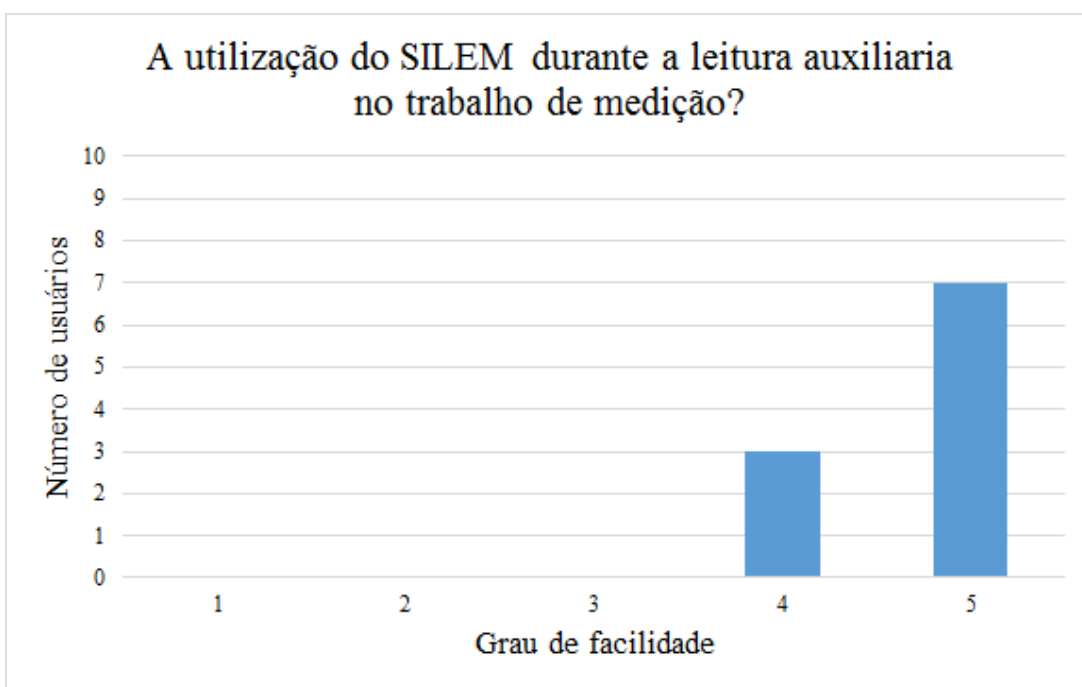
Fonte: O autor

Figura 38 – Gráfico da quarta questão da segunda avaliação



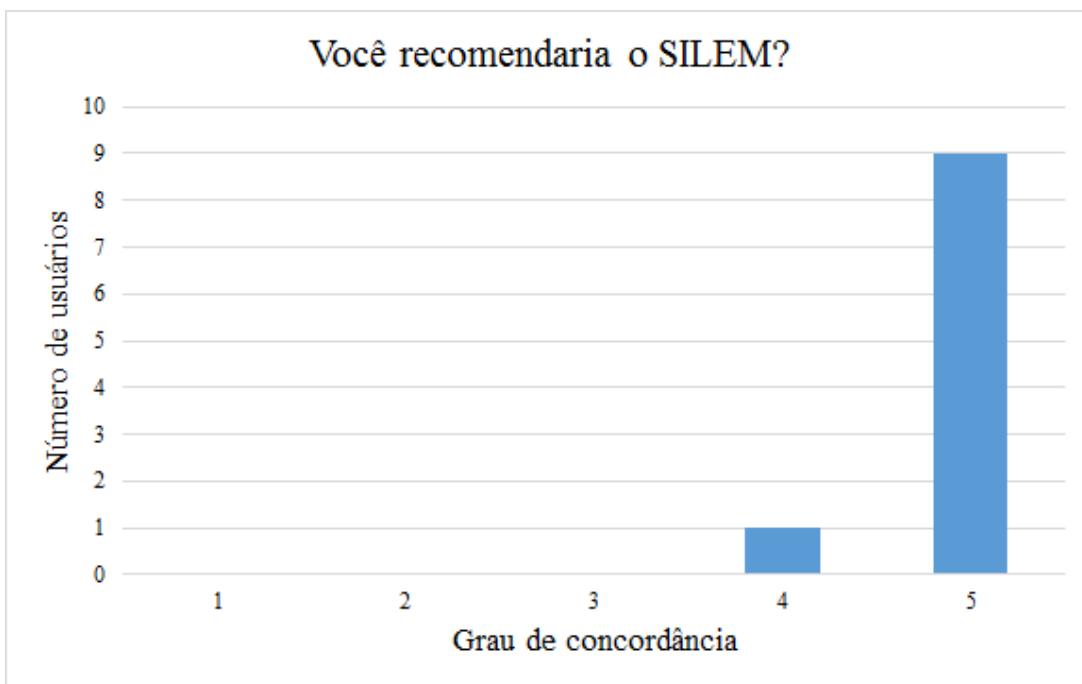
Fonte: O autor

Figura 39 – Gráfico da quinta questão da segunda avaliação



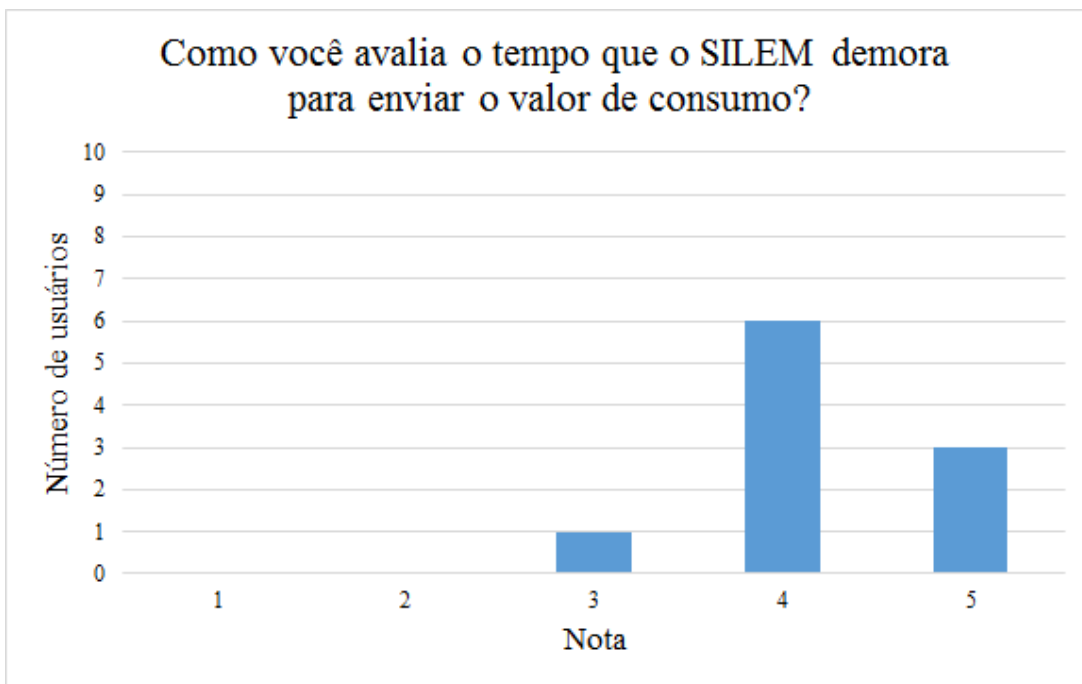
Fonte: O autor

Figura 40 – Gráfico da sexta questão da segunda avaliação



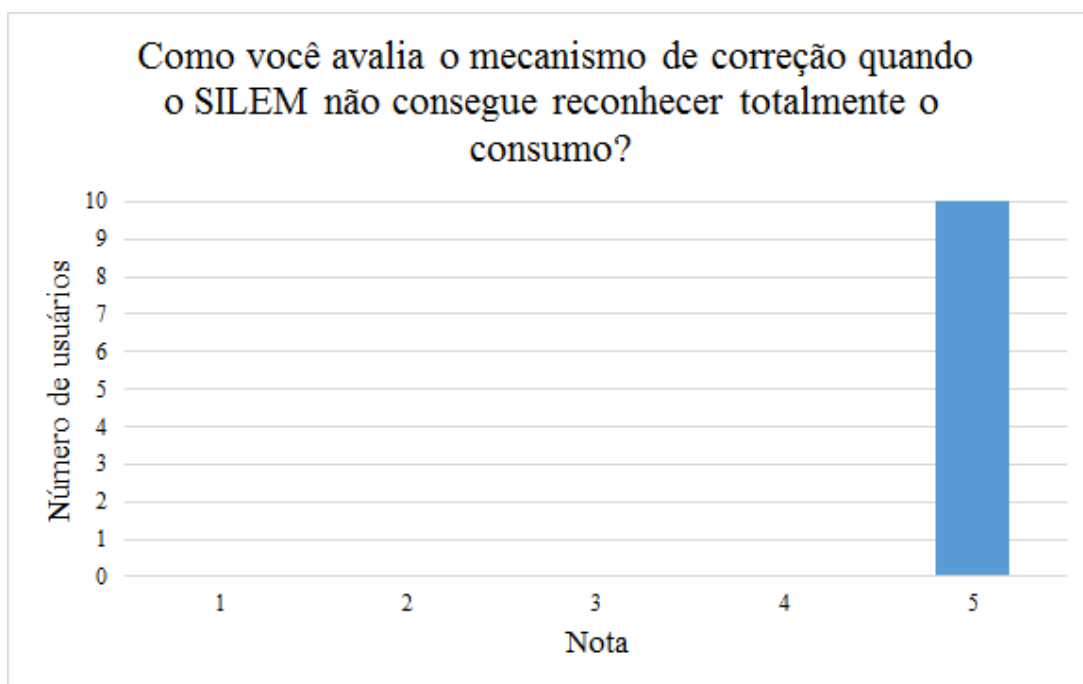
Fonte: O autor

Figura 41 – Gráfico da sétima questão da segunda avaliação



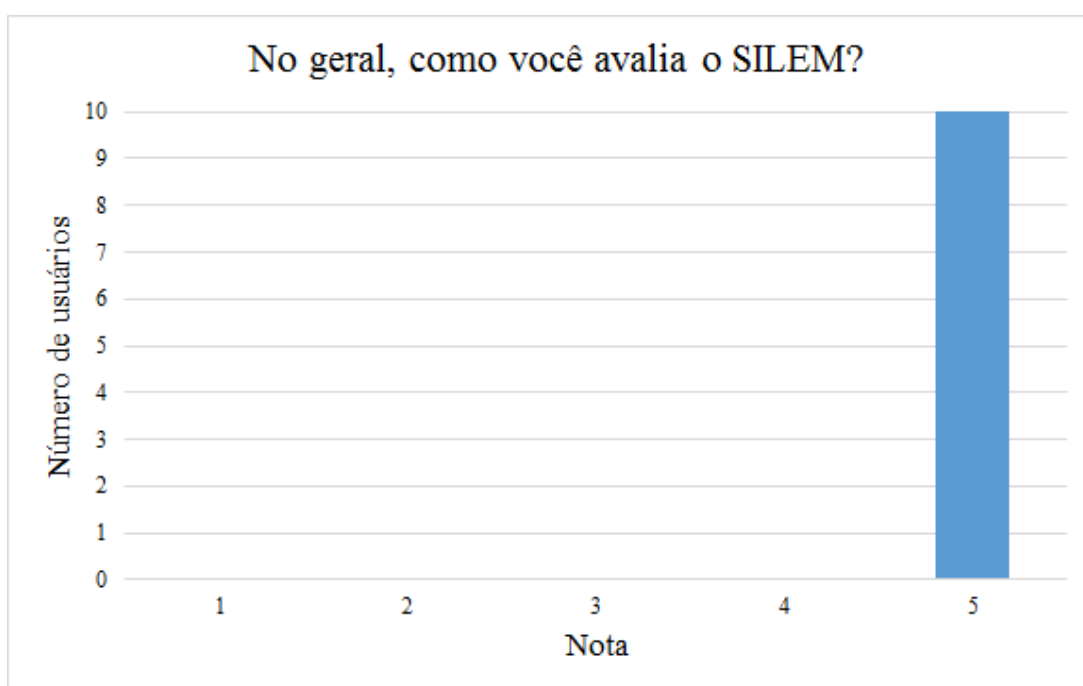
Fonte: O autor

Figura 42 – Gráfico da oitava questão da segunda avaliação



Fonte: O autor

Figura 43 – Gráfico da nona questão da segunda avaliação



Fonte: O autor

Oito usuários concordaram fortemente que a utilização do SILEM é atrativa e simples, um classificou como 4 e o outro como 3 (Figura 35). Nove entre os dez usuários classificaram como muito fácil o processo de aprendizagem de utilização do SILEM, enquanto um deles julgou a atividade como fácil (Figura 36). Seis usuários declararam que a operação do sistema é muito fácil, os demais consideraram este processo fácil (Figura 37). Quanto a dificuldade em entender os ícones, botões ou operações no SILEM, oito usuários classificaram como muito fácil de entender, um julgou fácil e o outro declarou como regular (Figura 38). Apesar de terem sido questionados se o sistema auxiliaria o trabalho em campo, a questão foi desconsiderada na análise de resultados. Ao serem questionados se recomendariam o SILEM, 9 usuários concordaram fortemente e um concordou (Figura 40). Atualmente, o SILEM demora em média cerca de 5,106 segundos para retornar o valor de consumo ao usuário e tem média de tempo total de uso de 21,683 segundos. Sobre este tempo, três deles consideraram o tempo muito bom, seis classificaram como bom e um declarou como regular (Figura 41). Quanto ao mecanismo desenvolvido para correção, todos o avaliaram como muito bom (Figura 42). Em sua visão geral sobre o aplicativo, todos atribuíram nota máxima ao SILEM (Figura 43). Nenhuma sugestão foi declarada na última questão.

A Tabela 5 também demonstra, de forma resumida, os resultados obtidos na segunda avaliação. A tabela consiste em mostrar a quantidade de usuários que selecionaram a mesma resposta para cada questão, em porcentagem. Na tabela, cada questão é representada por um "Q" seguido de um número, que é definido de acordo com a ordem em que foi listada anteriormente nesta seção.

Tabela 5 – Respostas ao questionário.

Resposta	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
1	0%	0%	0%	0%	0%	0%	0%	0%	0%
2	0%	0%	0%	0%	0%	0%	0%	0%	0%
3	10%	0%	0%	10%	0%	0%	10%	0%	0%
4	10%	10%	40%	10%	30%	10%	60%	0%	0%
5	80%	90%	60%	80%	70%	90%	30%	100%	100%

Com resultado obtido, pode-se ter uma dimensão do que o sistema representa e da sua relação com o usuário. Quando comparado ao primeiro teste, percebemos a evolução no sistema. A nova interface agradou aos usuários e o tempo de processamento teve uma significativa redução, assim atendendo as expectativas do usuário. Com 100% dos usuários na primeira avaliação e 90% na segunda indicando que recomendariam o SILEM, e 100% dos usuários, em ambas as avaliações, atribuindo nota máxima na avaliação geral do sistema, percebemos com clareza que o sistema é bem aceito e proverá melhoras ao sistema de anotação do consumo.

5 Conclusão

A constante necessidade de evolução faz com que busquemos novas soluções alternativas para problemas antigos, ou para problemas que já possuem solução, mas necessitam de resoluções modernas e com melhor desempenho. No campo de medição automática, vimos que existem estudos sendo desenvolvidos que utilizam as mais diversas tecnologias e diferentes metodologias para construir um método eficaz de leitura automática.

Este trabalho apresentou a arquitetura do sistema SILEM, que visa a leitura automática de medidores através da captura de imagem de medidores. Apresentou-se, também, as tecnologias envolvidas na construção deste sistema. Com enfoque na aplicação, o trabalho procurou explicar detalhadamente o funcionamento do SILEM, características da interface e os cenários pelos usuários do sistema.

Com interface simples, o sistema se torna uma ferramenta de fácil manuseio para os usuários. Elementos da interface, tais como os botões, foram criados com design simples, visando ter aparência semelhante a elementos de interface de outras aplicações que fazem parte do cotidiano do usuário de dispositivos moveis, ou visando representar objetos reais. Ou seja, a interface como principal elemento da interação entre usuário e sistema foi desenvolvida com o objetivo de ser intuitiva e amigável ao usuário, facilitando a introdução deste aplicativo no dia-a-dia do usuário.

Apresentando um resultado satisfatório quanto a utilização e ao feedback dos usuários, o SILEM trará ao sistema atual de medição uma nova perspectiva de. O SILEM torna-se útil as empresas pois propõe uma possibilidade de redução de falhas humanas, e, por consequência, de custo gerado por perdas não técnicas. Além dos benefícios gerados para a empresa, o aplicativo pretende automatizar e facilitar a tarefa de anotação feita pelos leituristas.

Em meio a várias soluções para desenvolver um método de leitura de medição automática, o SILEM pretende ser uma alternativa viável ao sistema atual de distribuição de energia elétrica do estado do Maranhão e do Pará, e a realidade do cotidiano dos leituristas, evitando despesas com novas metodologias que necessitem da reconstrução do sistema. Sendo assim, o SILEM é uma ferramenta que pretende ser parte do sistema e aprimorá-lo, não substituí-lo.

Referências

- ALI, A.; SAAD, N.; RAZALI, N.; VITEE, N. Implementation of automatic meter reading (amr) using radio frequency (rf) module. In: IEEE. *Power and Energy (PECon), 2012 IEEE International Conference on*. Kota Kinabalu, Malaysia, 2012. p. 876—879. Citado na página 14.
- ALI, A.; SAAD, N.; RAZALI, N.; VITEE, N. Automatic electricity billing. In: IJARCCCE. *International Journal of Advanced Research in Computer and Communication Engineering*. [S.l.], 2016. p. 1048—1049. Citado na página 14.
- BRACIER. *Futuro Iluminado*. 2011. Acessado em 5 de Novembro de 2017. Disponível em: <<https://www.bracier.org.br/o-bracier.html>>. Citado na página 12.
- CROCKFORD, D. *JSON - The x in Ajax*. 2006. Acessado em 25 de Novembro de 2017. Disponível em: <<http://www.json.org/json.pdf>>. Citado na página 18.
- GONÇALVES, J. C. *Reconhecimento de dígitos em imagens de medidores de consumo de gás natural utilizando técnicas de visão computacional*. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2016. Citado na página 13.
- GUTHRIE, S. *Microsoft to acquire Xamarin and empower more developers to build apps on any device*. 2016. Acessado em 28 de outubro de 2017. Disponível em: <<https://blogs.microsoft.com/blog/2016/02/24/microsoft-to-acquire-xamarin-and-empower-more-developers-to-build-apps-on-any-device/>>. Citado na página 16.
- HAMERSCHMIDT, M. B. *telemedições aplicadas ao faturamento de energia elétrica: um diagnóstico de viabilidade técnica e econômica na realidade brasileira*. Dissertação (Mestrado) — Instituto Lactec, Curitiba, 2012. Citado na página 13.
- KOMPF, M. *OpenCV practice: OCR for the electricity meter*. 2015. Acessado em 28 de Novembro de 2017. Disponível em: <<https://www.mkompf.com/cplus/emeocv.html>>. Citado na página 14.
- LECHETA, R. *Google Android 4ª edição: Aprenda a criar aplicações para dispositivos móveis com o Android SDK*. São Paulo: Novatec Editora, 2015. ISBN 9788575224403. Citado 2 vezes nas páginas 21 e 22.
- LEI, H.; ZHANG, P.; ZENG, Q.; XIANYI, L. Numeral recognition of power meter on a handheld terminal. In: *Third International Symposium on Electronic Commerce and Security Workshops (ISECS '10)*. Guangzhou, China: [s.n.], 2010. p. 76–79. Citado na página 14.
- MICROSOFT. *Xamarin e visual Studio*. 2017. Acessado em 28 de outubro de 2017. Disponível em: <<https://msdn.microsoft.com/pt-br/library/mt299001.aspx>>. Citado na página 16.
- PARTHIBAN, K.; PALANISAMY, A. Reading values in electrical meter using image processing techniques. In: IEEE. *Intelligent Interactive Systems and Assistive Technologies*

(IISAT), *2013 International Conference on*. Coimbatore, India, 2013. p. 1–7. Citado na página 14.

PEREIRA, L.; SILVA, M. D. *Android para desenvolvedores*. São Paulo: BRASPORT, 2009. ISBN 9788574524993. Citado 2 vezes nas páginas 21 e 22.

PETZOLD, C. *Creating Mobile Apps with Xamarin.Forms*. Redmond: Microsoft Press, 2016. ISBN 9781509302970. Citado na página 20.

PRAPASAWAD, C.; PORNPRASITPOL, K.; PORA, W. Development of an automatic meter reading system based on zigbee pro smart energy profile iee 802.15. 4 standard. In: IEEE. *Electron Devices and Solid State Circuit (EDSSC), 2012 IEEE International Conference on*. Bangkok, Thailand, 2012. p. 1–3. Citado na página 14.

PRATES, R. O.; BARBOSA, S. D. J. Avaliação de interfaces de usuário–conceitos e métodos. In: *Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação, Capítulo*. [S.l.: s.n.], 2003. v. 6. Citado 2 vezes nas páginas 30 e 41.

SMITH, B. *Beginning JSON*. Berkely: Apress, 2015. Citado 4 vezes nas páginas 16, 17, 18 e 19.

SOMMERVILLE, I. *Engenharia de Software*. São Paulo: Pearson Prentice Hall, 2011. Citado 2 vezes nas páginas 28 e 29.

TANENBAUM, A. *Sistemas operacionais modernos*. São Paulo: Prentice-Hall do Brasil, 2010. ISBN 9788576052371. Citado 3 vezes nas páginas 18, 19 e 20.

VIDINICH, R.; NERY, G. Pesquisa e desenvolvimento contra o furto de energia. In: *Revista Pesquisa e Desenvolvimento da ANEEL–P&D*. [S.l.: s.n.], 2009. p. 15. Citado na página 12.

ZHANG, Y.; YANG, S.; SU, X.; SHI, E.; ZHANG, H. Automatic reading of domestic electric meter: an intelligent device based on image processing and zigbee/ethernet communication. *Journal of Real-Time Image Processing*, Springer, v. 12, n. 1, p. 133–143, 2016. Citado na página 14.

ZHAO, L.; ZHANG, Y.; BAI, Q.; QI, Z.; ZHANG, X. Design and research of digital meter identifier based on image and wireless communication. In: IEEE. *2009 International Conference on Industrial Mechatronics and Automation, ICIMA 2009*. Chengdu, China, 2009. p. 101 – 104. Citado na página 14.