UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

CONTROLLER: REGISTRO DE CONSUMO DE ÁGUA EM CONDOMÍNIOS UTILIZANDO RECONHECIMENTO POR IMAGEM

DAIANE LUÇOLI SCHVEPE

BLUMENAU
2015

2015/2-02

DAIANE LUÇOLI SCHVEPE

CONTROLLER: REGISTRO DE CONSUMO DE ÁGUA EM CONDOMÍNIOS UTILIZANDO RECONHECIMENTO POR IMAGEM

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Aurélio Faustino Hoppe , Mestre - Orientador

BLUMENAU
2015

2015/2-02

CONTROLLER: REGISTRO DE CONSUMO DE ÁGUA EM CONDOMÍNIOS UTILIZANDO RECONHECIMENTO POR IMAGEM

Por

DAIANE LUÇOLI SCHVEPE

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Aurélio Faustino Hoppe, Mestre – Orientador, FURB
Membro:	Prof. Dalton Solano dos Reis, Mestre – FURB
Membro:	Prof. Matheus Carvalho Viana, Doutor – FURB

Blumenau, 10 de dezembro de 2015

Dedico este trabalho às pessoas que sempre me apoiam e permanecem ao meu lado.

AGRADECIMENTOS

A Deus, por tudo que tem feito por mim, a Ele toda a honra e glória!

Ao meu amado esposo Claudio Schvepe por me ajudar e compreender.

Aos meus amigos que sempre estão ao meu lado apoiando em horas boas e ruins.

Ao meu orientador Aurélio Faustino Hoppe, pela paciência e ajuda.

Aos professores do departamento de sistemas de informação pelo ensino.

E ao meu amigo Alexandre Felix de Souza por me ajudar no tema do TCC.

O maior inimigo do conhecimento não é ignorância, mas a ilusão do conhecimento.

Stephen Hawking

RESUMO

Este trabalho apresenta o desenvolvimento de um protótipo para reconhecimento de caracteres em imagens, especialmente de hidrômetros, utilizando as bibliotecas Tesseract e OpenCV. O protótipo foi desenvolvido seguindo a arquitetura cliente-servidor, na qual a aplicação cliente realiza a captura e o envio das imagens do hidrômetro para a aplicação servidora. A aplicação servidora permite o cadastro de unidades habitacionais, apartamentos, usuários e proprietários, assim como, realiza a consulta das leituras efetuadas, a identificação da área de interesse e a interpretação dos caracteres existentes nas imagens do hidrômetro. A partir dos experimentos realizados, comprovou-se que o protótipo desenvolvido é capaz de reconhecer os dígitos contidos nas imagens do hidrômetro. Porém, não alcançou-se os resultados esperados.

Palavras-chave: Processamento de imagem. Hidrômetro. Optical Character Recognition - OCR.

ABSTRACT

This work presents the development of a prototype for character recognition in images, especially of water meters, using the Tesseract and OpenCV libraries. The prototype was developed following the client-server architecture, where the client application performs the capture and transmission of meter images to the server application. The server application allows the registration of housing units, apartments, users and owners as well as performs the query of readings made, identifying the area of interest and interpretation of existing characters in the hydrometer images. From the experiments, it was shown that the developed prototype is able to recognize the digits contained in the meter images. However, it did not reach to the expected results.

Keywords: Image processing. Hydrometer. Optical Character Recognition - OCR.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface*

JPEG – *Joint Photographics Experts Group*

PDI – Processamento De Imagens

PNG – *Portable Network Graphics*

MVC – *Model-view-controller*

OPENCV– *Open Source Computer Vision Library*

OCR – *Optical Character Recognition*

RGB – Vermelho (*Red*), Verde (*Green*) e Azul (*Blue*)

RF – Requisito Funcional

RNF – Requisito Não Funcional

ROI – *Region Of Interest*

UML – *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

Gobbi (2015) descreve que, em 1940, apenas 31% da população brasileira vivia em cidades. Na década seguinte, o processo de urbanização se intensificou e, a partir de 1970, mais da metade dos brasileiros já se encontrava em áreas urbanas. Ainda segundo Gobbi (2015), fazendo uma comparação nos últimos 40 anos a população rural aumentou cerca de 12%, enquanto que a população urbana passou de 13 milhões de habitantes para 138 milhões, um aumento de mais de 1.000%.

A forma de moradia da população vem modificando a paisagem ao longo dos anos, transformando as imagens de casas familiares em enormes edifícios e, junto com essa mudança, surgem novas regras e novas obrigações. Ao que parece, no momento em que vivemos muitas pessoas optam por morar em condomínio pelas facilidades e segurança. No entanto, os condomínios com muitas unidades tornam-se mais complexos e necessitam de uma quantidade maior de atividades e rotinas a serem executadas. Além disso, gerenciar um condomínio é uma responsabilidade que necessita de tempo e transparência na sua administração (SILVA, 2010, p.16 e 17).

No cenário atual a distribuição de despesas desses condomínios que envolve serviços, manutenção e produtos é feita de forma a somar as despesas e dividi-las entre a quantidade de apartamentos (MAIA, 2011). Dalmut (2013) defende a ideia de que é mais correto se ter medidor individual e somente a água utilizada nas áreas comuns deve fazer parte do rateio das despesas. Uma conta simples que se transforma em motivos de brigas por conta do critério de ser mais ou menos justa a divisão do consumo de água sem medidores individuais.

Assim, o justo para uns nem sempre é o justo para outros. Se não houver um medidor individual deve-se levantar um estudo por unidade habitacional levando em consideração o tamanho da unidade, o número de pessoas que habitam no apartamento e os hábitos dos moradores. Pois, todos querem uma coisa só: pagar o que efetivamente gastou (DALMUT, 2013).

Com base neste cenário, este trabalho de conclusão de curso propõe o desenvolvimento de um protótipo que efetue a captura, o reconhecimento e a interpretação de imagens de hidrômetros e, em seguida, apresentar o resultado em formato de texto, de modo a facilitar a coleta dos dados para apuração dos valores individuais.

1.1 OBJETIVOS

O objetivo geral do trabalho é a construção de um protótipo de *Optical Character Recognition* (OCR) para reconhecimento de informações sobre imagens de hidrômetros.

Os objetivos específicos do trabalho são:

a) capturar a imagem utilizando a câmera do *smartphone*;
b) analisar e identificar na imagem a região onde se encontram os dígitos que representam o consumo de água;
c) utilizar técnicas de processamento de imagens, visão computacional e aprendizado de máquina para extrair e apresentar os dígitos existentes na imagem de forma textual.

1.2 ESTRUTURA

No primeiro capítulo tem-se a introdução, a justificativa e os objetivos deste trabalho. O segundo capítulo apresenta a fundamentação teórica com os conceitos de OCR, processamento de imagem e a apresentação de trabalhos correlatos. A seguir o terceiro capítulo apresenta o desenvolvimento do sistema iniciando-se com o levantamento de informações, tendo na sequência a especificação, a implementação e, por fim, os resultados e discussões. No quarto capítulo tem-se as conclusões deste trabalho bem como apresentam-se sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo está organizado em 4 seções. A Seção 2.1 apresenta o conceito de OCR. A Seção 2.2 descreve sobre processamento de imagens, Já a Seção 2.3 aborda o reconhecimento de imagens com Tesseract. Por fim, a Seção 2.4 descreve os trabalhos correlatos.

2.1 OPTICAL CHARACTER RECOGNITION

OCR é uma tecnologia utilizada para reconhecer caracteres a partir de um arquivo de imagem, sejam eles escritos à mão ou digitados. Essa tecnologia faz uso de dois métodos básicos: o ajuste de matriz[1] e o de extração de característica[2]. Algumas das razões para justificar o uso de OCR, são a redução nos erros de entrada de dados, humanamente legível, digitalização de correções, entre outros (SHIGEHARU JUNIOR, 2014).

Os sistemas que utilizam OCR fazem uso de entrada de dados para documentos de negócios, reconhecimento de número de placas, tornando pesquisáveis as imagens eletrônicas de documentos impressos. Esta tecnologia já faz um pré-processamento da imagem por meio da binarização e segmentação, tornando as chances de reconhecimento maiores (SHIGEHARU JUNIOR, 2014).

Shigeharu Junior (2014) descreve que, os fornecedores da tecnologia OCR estão efetuando ajustes para melhor lidar com tipos específicos de entrada de dados. Essas alterações estão apresentando maior eficiência no reconhecimento dos caracteres na expressão padrão, ou informações contidas nas imagens coloridas. Essa nova versão é chamada de OCR orientada a aplicação ou OCR personalizado, e vem sendo aplicada em notas fiscais, capturas de telas, carteira de habilitação entre outros. Para Eikvil (1993), o OCR possui quatro etapas:

a) digitalização de um documento por meio de *scanners*, leitores ópticos, câmeras digitais, dentre outros;
b) extração de padrões dos caracteres da imagem digitalizada (reconhecimento dos caracteres);
c) classificação dos padrões dos caracteres obtidos;
d) conversão da imagem obtida na primeira etapa em documento de texto em formato eletrônico.

---

[1] Ajuste de matriz compara o que o scanner OCR vê como um personagem com uma biblioteca de matrizes ou modelos de caracteres;
[2] Extração de característica, utiliza técnicas gerais de visão computacional o que que é comumente visto em reconhecimento "inteligente" de escrita e nos mais modernos OCR (DATAID, 2003).

Eikvil (1993), descreve que o processo de digitalização consiste geralmente em um mecanismo de transporte, acrescido de um dispositivo que converte a intensidade da luz em níveis de cinza, por isso, é comum converter a imagem de vários níveis em uma imagem de dois níveis de preto e branco. Quando acontece o reconhecimento de caracteres é aplicado a segmentação que é um processo que determina os componentes de uma imagem. A segmentação faz o isolamento dos caracteres, e estes são reconhecidos individualmente.

2.2 PROCESSAMENTO DE IMAGENS

A área de processamento de imagens permite viabilizar as aplicações em duas categorias, sendo aprimoramento de informações pictóricas para interpretação humana e a análise automática por um computador de informações extraídas de uma cena. Os elementos do Processamento De Imagens (PDI) baseiam-se em aquisição (imagem capturada), processamento (computador), armazenamento (discos ópticos e discos magnéticos) e saídas (monitores e impressoras) (FILHO; NETO, 1999, p. 02).

O PDI é expresso normalmente sob forma algorítmica, com base nisto, exceto as etapas de aquisição e exibição, as funções para o processamento de imagens podem ser implementadas via software (FILHO; NETO, 1999). Para maior abrangência do assunto nas próximas subseções serão abordados os assuntos: região de interesse e aquisição, realce e segmentação.

2.2.1 Região de interesse / Aquisição

Marques Filho e Vieira Neto (1999, p. 21) descrevem em seu livro que a parte da aquisição da imagem a ser processada, tem como função converter uma imagem em uma representação numérica adequada para o processamento digital subsequente. Dentre os dispositivos para aquisição de imagens existentes hoje no mercado, os mais comuns são câmeras de vídeos, *scanners* entre outros. Cada um com suas características de resolução, velocidade de operação, precisão e custo.

Ainda segundo Marques Filho e Vieira Neto (1999, p. 21), os dispositivos de aquisição de imagens consistem em de uma matriz de células semicondutoras fotossensíveis, que atuam como capacitores, armazenando uma carga elétrica incidente. Este sinal elétrico produz circuitos eletrônicos especializados, produzindo à saída um sinal composto de vídeo analógico e monocromático

Segundo Albuquerque e Albuquerque (2000), entende-se como Região de Interesse (*Region Of Interest* - ROI) a região definida automaticamente a partir de parâmetros obtidos

na aquisição da imagem, onde nesta região estará o foco totalmente concentrado para o processamento.

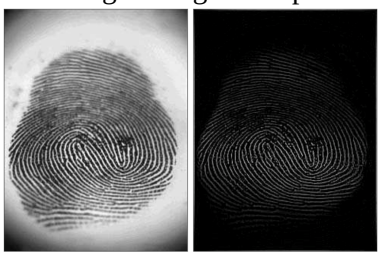
Os autores supracitados exemplificam que para definir uma região de interesse que se sabe por antecedência que a iluminação de fundo é constante ou foi corrigida, normalmente nas técnicas de processamento de imagem o problema estará nas bordas.

2.2.2 Realce

O principal objetivo das técnicas de realce é o de destacar detalhes finos na imagem. Para isso, podem ser utilizados os métodos de imagens no domínio espacial, a saber: filtro passa-altas básico, realce por diferenciação e ênfase em alta frequência (MARQUES FILHO; VIEIRA NETO, 1999, p. 95).

O filtro passa-altas (filtro que permite a passagem das frequências altas com facilidade) deve ser tal que a máscara correspondente apresente coeficientes positivos nas proximidades de seu centro e negativos longe dele. Isto significa projetar uma máscara com pixel central positivo e todos seus oito vizinhos negativos. A Figura 1 mostra um exemplo de resultado de aplicação da máscara monocromática (MARQUES FILHO; VIEIRA NETO, 1999, p. 95).

Figura 1 - Imagem original e após filtragem



Fonte: Marques Filho e Vieira Neto (1999).

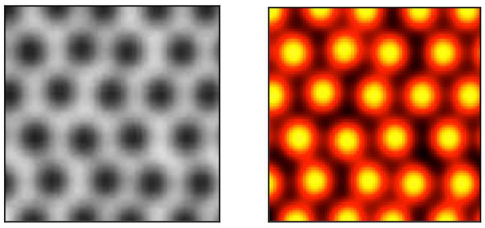
O realce por diferenciação é realizado com base no cálculo da média dos pixels existentes em um trecho de imagem que produz como efeito, a remoção de seus componentes de alta frequência. O conceito de média é análogo à operação de integração e produz o efeito oposto o mais usual em aplicações de processamento na qual a imagem é gradiente (MARQUES FILHO; VIEIRA NETO, 1999, p. 96).

2.2.3 Segmentação

Albuquerque e Albuquerque (2000), descrevem em seu artigo que a segmentação consiste na realidade em dividir a imagem em diferentes regiões, que serão analisadas posteriormente por algoritmos especializados.

A imagem ilustrada na Figura 2, por exemplo, poderia ser segmentada em duas regiões: aquelas pertencentes às células e aquelas pertencentes ao fundo da imagem. A imagem obtida neste caso é composta por apenas duas regiões: a branca (fundo) e outra preta (células/objeto). A imagem com níveis de cinza (esquerda) é conhecida como imagem binária, pois este tipo de imagem é frequentemente utilizado devido a grandes facilidades na manipulação e no processo de tratamento da informação.

Figura 2 - Imagem estrutura hexagonal e com novas cores



Fonte: Albuquerque e Albuquerque (2000).

Ainda segundo Albuquerque e Albuquerque (2000), existem diversas técnicas de segmentação de imagens, mas não existe nenhum único método que seja capaz de segmentar todos os tipos de imagem. Essas imagens segmentadas podem ser consideradas entre as semelhanças entre os níveis de cinza ou somente pelas suas diferenças. Um exemplo de caso usando só as diferenças é a utilização de um contorno de objeto por meio de matrizes do tipo Passa-Alta. Neste caso é segmentando as imagens em regiões que pertencem à borda do objeto.

2.3 RECONHECIMENTO DE IMAGENS COM TESSERACT

Lanhellas (2012) apresenta a biblioteca Tesseract para uso de reconhecimento de caracteres ópticos. Esta biblioteca suporta muitos idiomas como inglês e português. A *Application Programming Interface* (API) Tesseract não foi desenvolvida em Java, mas em C e em seguida migrado para C++. Para fazer uso de recursos em Java é necessário o Java Native Access (JNA), que funciona como um *wrapper* que possibilita o uso dos métodos. Junto com a biblioteca do Tesseract são necessárias as bibliotecas `ghost4j-0.5.1`, `jai_imageio`, `jna-4.1.0`, `jnuit-4.10`, `log4j-1.2.17` e `tess4j` (LANHELLAS, 2012).

Segundo Souza (2013), a primeira parte do Tesseract consiste em obter possíveis palavras a partir da entrada da imagem, depois para a mesma é realizada a binarização por meio de um *thresholding* adaptativo para evitar danos feitos por irregularidades na iluminação. Em ressumo, durante todo este processo o Tesseract utiliza algoritmos para:

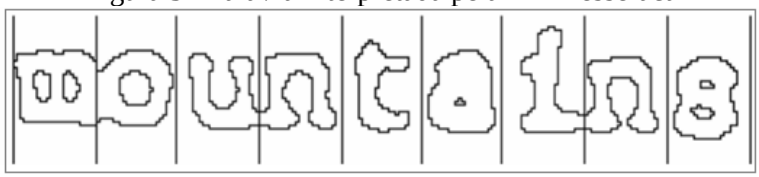
a) detecção de linhas de texto de uma página distorcida;
b) detecção da proporção das palavras e símbolos;

c) cortar caracteres unidos;
d) analisador linguístico para identificar as palavras mais utilizadas;
e) classificador de caracteres: um estático, e um adaptativo.

Smith (2007) ainda detalha que na análise de componentes gerados pela binarização, é feita a comparação por um pixel central e seus vizinhos, fazendo a definição se estão compactados. Em seguida é feito a busca por linhas e palavras, após uma análise prévia da imagem, as regiões de texto são encontradas, e há uma eliminação de manchas e ruídos.

Na Figura 3, pode ser visto um exemplo de imagem no qual a API Tesseract testa as linhas de uma palavra para determinar as bordas. A API encontra primeiramente a linha, faz o corte para encontrar as letras e, em seguida, aplica sobre estas letras a etapa de reconhecimento (SMITH, 2007).

Figura 3 - Palavra interpretada pela API Tesseract



Fonte: Smith (2007).

As regiões resultantes são chamadas de *blobs* e são organizados em linhas de texto e, depois, são novamente analisadas. As linhas de texto são divididas em palavras de forma diferente de acordo com o tipo de espaçamento entre caracteres. Depois que as linhas de texto foram encontradas, é aplicada uma curva (*spline*) matematicamente calculada por dois ou mais pontos de controle, que consiste em dividir o intervalo de interesse em vários subintervalos. Isso permite ao Tesseract lidar com páginas com linhas de base curvas (SMITH, 2007).

2.3.1 Biblioteca leptonica

Leptonica é uma biblioteca desenvolvida em C de código aberto para operações de processamento de imagem e análise de imagem eficientes (LEPTONICA, 2011). Esta biblioteca contempla implementações de morfologia binária, isto é, morfologia em tons de cinza, filtros de convolação e ordem de classificação e aplicações tais como processamento de imagem *Joint Bi-level Image Experts Group* (JBIG2) e redução do número de cores da imagem (LEPTONICA, 2011).

A morfologia é baseada na teoria dos conjuntos e no fato de que o conjunto de todos os pixels pretos constituem uma completa descrição de imagens binárias. É executado no

processamento de imagens o aumento de qualidade, restauração, detecção de falhas, análise da textura, análise particular, análise de formas, e compreensão (VALENTE NETO, 2011).

Essa biblioteca ainda possui uma estrutura de dados utilizando o paradigma de orientação a objetos é subdividida em dois segmentos: funções de alto nível e as de baixo nível que utilizam intrinsecamente tipos de dados do C. Quanto aos tipos de imagens com compressão com bibliotecas *open source* se enquadram: o *Portable Network Graphics (PNG)*, *Joint Photographics Experts Group* (JPEG) e o novo (`libvpx`, uma biblioteca de compressão de vídeo) WEBP (KOHATSU, 2012).

A biblioteca Leptonica possui uma classe chamada de `AdaptiveMap`, que contém métodos para mapeamento de imagens adaptativas. Em conjunto com a API Tesseract deverá ser necessário apenas à implementação do método de normalização do fundo de uma imagem. A biblioteca ainda possui a classe `binarize`, que como o nome sugere, faz a binarização de imagens (KOHATSU, 2012).

A parte de segmentação fica sob a responsabilidade da função `otsuAdaptiveThreshold`, que faz a definição dos objetos de uma imagem de seu fundo. A classe `Box` representa um *container* para o *box* nativo da biblioteca Leptonica. Essa contém funções para a criação do container que contem a restrição da porção do texto que envia à API de reconhecimento de caracteres (KOHATSU, 2012).

As imagens são convertidas e o retorno será uma imagem em 8 bits em escala de cinza ou um código de erro, pela classe `Convert`. Enquanto a classe `Enhance` tem métodos para melhorar a nitidez e a função `unsharp masking` que faz a chamada para a função nativa correspondente que criará uma camada sobre a imagem, tratando a nitidez (KOHATSU, 2012).

A classe `JpegIO` trata de entrada e saída de arquivos no formato `Jpeg`, utilizando a função `compressToJpeg` compacta/converte qualquer imagem bitmap para o formato `Jpeg`, com qualidade padrão. Estas classes da biblioteca Leptonica e a própria podem ser obtidas gratuitamente em GoogleCode (KOHATSU, 2012).

2.4 TRABALHOS CORRELATOS

Esta seção destina-se a apresentar alguns dos trabalhos voltados ao reconhecimento de imagens digitais e OCR. Foram encontrados os trabalhos de Nascimento (2012), Kohatsu (2012) e Souza et al. (2013).

2.4.1 Detecção e reconhecimento de placa automotiva com baixo custo

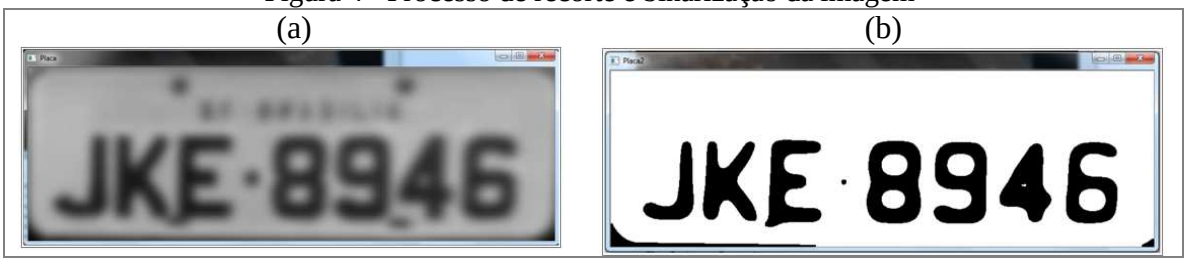
Nascimento (2012) desenvolveu um software que identifique a placa veicular com umas câmeras de baixo custo, utilizando como base as interfaces de programação de aplicativos (API) OpenCV e Tessaract OCR, na linguagem de programação C/C++.

Nascimento (2012) desenvolveu o software realizando as seguintes etapas:

a) captura da imagem: por meio de algum dispositivo que efetue a captura da imagem;
b) pré-processamento: no qual é utilizado tratamento de imagem empregando as técnicas de diminuição de ruído de Gauss, conversão de cores do formato Red (vermelho), Green (verde) e Blue (azul) - RGB, para escala de cinza (único canal), técnicas de diminuição de ruído de Gauss, conversão de cores do formato RGB para escala de cinza (único canal);
c) localização do objeto: para a detecção da borda;
d) validação;
e) segmentação: validação do recorte do objeto localizado;
f) leitura OCR: onde se inicia o processo de leitura da imagem.

No desenvolvimento de seu trabalho, Nascimento (2012) mostra a etapa em que é feito o recorte e dimensionamento da imagem a ser lida, conforme mostra a Figura 4 item (a). Na Figura 4 item (b), é exibido o resultado da transformação, de forma que os caracteres já são claramente visíveis no formato preto e branco.

Figura 4 - Processo de recorte e binarização da imagem



Fonte: Nascimento (2012).

Segundo Nascimento (2012), todos os objetivos do projeto foram alcançados, tanto para os casos de testes de nível de exatidão na detecção da placa do veículo, da performance do algoritmo utilizando as bibliotecas OpenCV e Tesseract OCR, e da exatidão na conversão dos caracteres da imagem para o texto, salvo nos casos particulares, em que foram encontradas situações em que o método de detecção de borda é falha e necessita uma atenção especial nos equipamentos de captura.

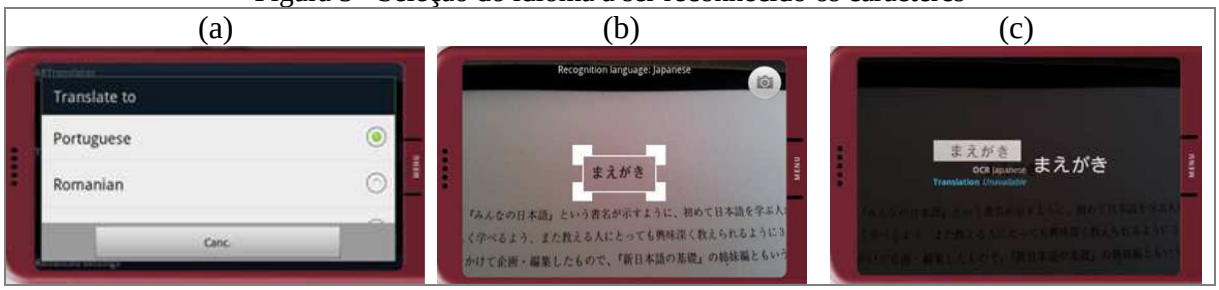
2.4.2 ARTRANSLATOR: Protótipo de um tradutor baseado em técnicas de reconhecimento ótico e realidade aumentada para móveis

Kohatsu (2012) propôs um protótipo que tem como objetivo: empregar técnicas de Realidade Aumentada (RA) para simplificar o processo de visualização de dados para tradução de textos; e permitir a tradução de textos entre diferentes idiomas com entrada de dados por reconhecimento ótico de caracteres.

Para o desenvolvimento do protótipo, Kohatsu (2012) utilizou a arquitetura do Android que permite que os aplicativos sejam desenvolvidos na linguagem de programação Java. O protótipo é formado pelos componentes: biblioteca de visão computacional do Google Leptonica, biblioteca OCR do Google Tesseract, API Camera Manager, API de OCR, API de tradução e API de Realidade Aumentada e componentes externos como o idioma de treinamento do OCR e as mensagens do Google Translate.

A Figura 5 item (a) mostra a lista de idiomas ao qual é permitida a tradução, depois deste passo o usuário determina a palavra a ser interpretada pelo aplicativo Figura 5 item (b). Por fim, o aplicativo apresenta a palavra interpretada e o seu significado, Figura 5 item (c).

Figura 5 - Seleção do idioma a ser reconhecido os caracteres



Fonte: Kohatsu (2012).

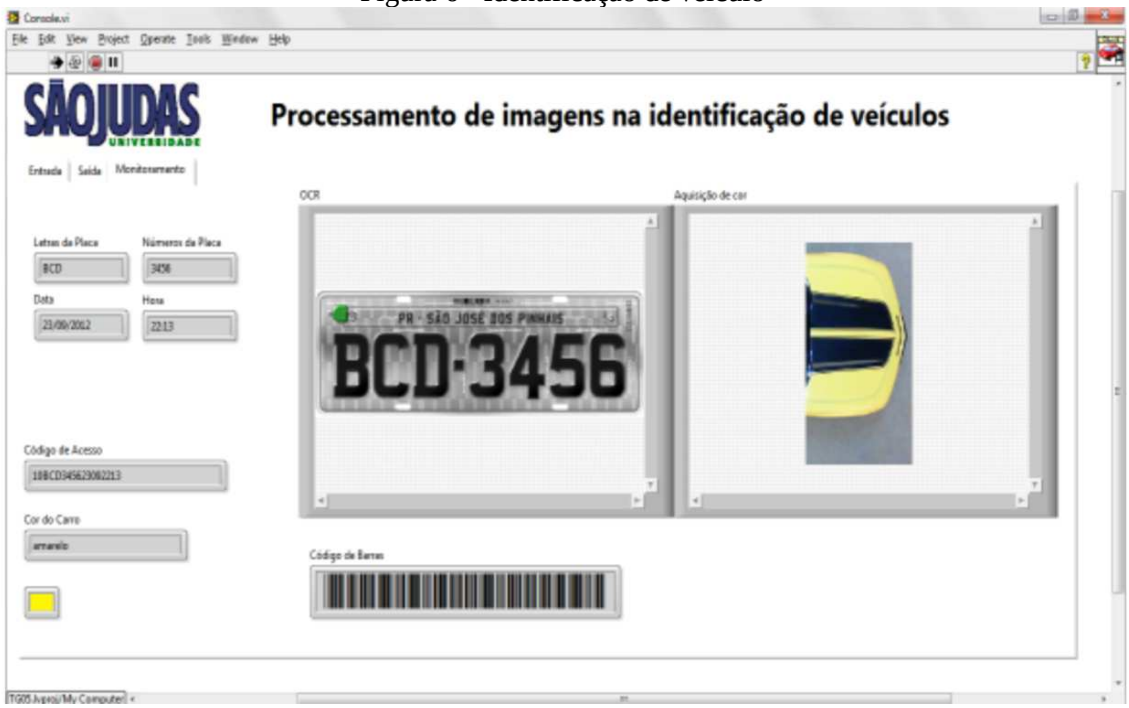
Os resultados alcançados indicam que os meios físicos, impressos possuem estatísticas melhores quanto ao reconhecimento dos caracteres. Foi constatado que os números representam uma estatística maior de acerto sobre uma pequena amostra de testes. Quanto mais próximos os caracteres estiverem da grafia de forma, mais precisas são as chances do reconhecimento dos caracteres (KOHATSU, 2012).

2.4.3 Processamento de imagens na identificação de veículos

Souza et al. (2013) fez um estudo sobre as técnicas para identificação dos caracteres presentes em placas de veículos, com a utilização da tecnologia de OCR. Foi utilizando também o Vision Assitant, no qual são efetuadas as leituras das três letras e, logo depois, dos quatro números. Souza et al. (2013) descreve que para facilitar a identificação dos caracteres

foi utilizada a ferramenta OCR Training, que consiste no treinamento do sistema para reconhecer os caracteres, conforme a Figura 6.

Figura 6 - Identificação de veículo



Fonte: Souza (2013).

Para a identificação das placas de veículos, Souza (2013) fez a utilização da aplicação da linguagem G, que é uma linguagem de programação gráfica de alto nível baseada em fluxo de dados desenvolvido para aplicações que necessitam de interatividade, execução em paralelo e múltiplo processamento. A plataforma de desenvolvimento utilizada foi o LabVIEW em conjunto com o NI Vision Development Module.

O sistema de Souza et al.(2013), ao detectar o veículo gera um código de barras para ser impresso na entrada do estacionamento. No momento da saída, o sistema compara o código de barras com o código de acesso e se for compatível, libera a saída. Do contrário, emiti um alerta para dirigir-se ao caixa.

Souza et al.(2013) conseguiu identificar e registrar cada um dos veículos utilizados e suas respectivas cores. Para verificação da leitura das respectivas placas foi atribuído a cada modelo uma placa fictícia em tamanho real e impressa mantendo a proporção e proximidade de fidelidade de cor suficiente para leitura. Alguns testes foram conduzidos a fim de verificar a possibilidade de inconsistência na relação cor e placa entre as miniaturas utilizadas. Observou-se que o sistema identifica as diferenças de maneira bem sucedida.

2.4.4 Comparativo entre as características dos trabalhos correlatos

No Quadro 1 é demonstrado o comparativo das principais características dos trabalhos citados na Seção 2.4.

Quadro 1 – Comparativo entre os trabalhos correlatos

características / trabalhos correlatos	Nascimento (2012)	Kohatsu (2012)	Souza et al. (2013)
utiliza biblioteca Leptonica	Não	Sim	Não
utiliza API Tesseract	Sim	Sim	Não
utiliza OCR	Sim	Sim	Sim
aplicação em dispositivo móvel	Não	Sim	Não
armazena o histórico	Não	Não	Sim

A partir do Quadro 1, pode-se observar que os três trabalhos correlatos supracitados utilizam os métodos de OCR no processamento de imagem, ambos executaram de formas diferentes para a extração dos dados. Nascimento (2012), usa a linguagem C++ e as bibliotecas OpenCV e Tesseract OCR para captar a imagem para realizar o processametno de imagens.

Kohatsu (2012) utiliza em seu protótipo a API Tesseract a biblioteca Leptonica, a API de realidade aumentada e a biblioteca de visão computacional do Google, sendo desenvolvido na linguagem Java. Segundo o autor, os resultados foram satisfatórios.

Embora Nascimento (2012) e Kohatsu (2012) utilizem plataformas de desenvolvimento diferentes, ambos usam as API´s OCR e API Tesseract para o processamento das imagens. Souza et al. (2013) utilizaram no seu trabalho o OCR juntamente com o Vision *Assitan* para identificação de veículos. Como ferramenta de desenvolvimento, foi utilizada a plataforma utilizada *LabVIEW* em conjunto com o *Vision Development Module*. Em comparação com Nascimento (2012) e Kohatsu (2012), Souza et al. (2013) utiliza o OCR para conseguir realizar o processamento de imagens. Apenas Souza et al. (2013) armazenam um histórico das placas, associados ao código de barras e código de acesso para auditar o sistema.

3 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo são apresentadas as etapas do desenvolvimento do protótipo. A Seção 3.1 traz os requisitos principais do trabalho. Na Seção 3.2 consta a especificação, com modelos e diagramas. A implementação, com técnicas, ferramentas utilizadas e a operacionalidade da implementação estão na Seção 3.3. Por fim, a Seção 3.4 detalha os resultados obtidos.

3.1 REQUISITOS PRINCIPAIS A SER TRABALHADO

O aplicativo deverá atender aos seguintes requisitos:

a) permitir indicar o número da unidade habitacional de onde será coletada a imagem (Requisito Funcional - RF);
b) permitir ao usuário capturar a imagem enquadrando a ROI (RF);
c) permitir ao usuário enviar as imagens para o servidor via *webservice* (RF);
d) permitir ao usuário do sistema fazer consultas do resultado gerado (RF);
e) realizar a interpretação da imagem utilizando OCR (RF);
f) exibir em texto com o resultado obtido da imagem (RF);
g) armazenar as imagens de acordo com número da unidade habitacional (RF);
h) deverá ser executado na plataforma Android/IOS (Requisito Não-Funcional -RNF);
i) deverá ser desenvolvido em Java utilizando a API Tesseract, a biblioteca Leptonica e biblioteca FileTransfer do PhoneGap (RNF);
j) a aplicação servidora deverá ser desenvolvida em Java utilizando o servidor Tomcat e o banco de dados Postgres (RNF).

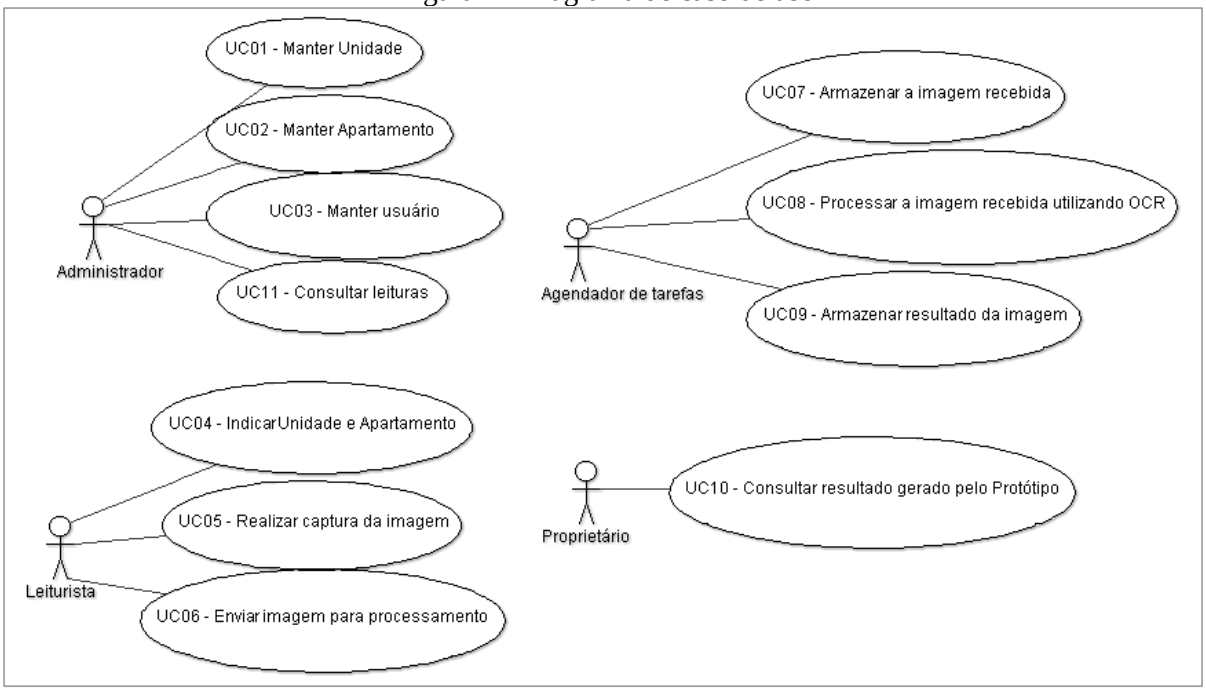
3.2 ESPECIFICAÇÃO

Na especificação do protótipo foi utilizada a ferramenta ArgoUML 0.34. Neste trabalho foram elaborados os diagramas casos de uso, atividades e de classes. O diagrama de Entidade Relacionamento está no Apêndice A.

3.2.1 Diagrama de caso de uso

Nesta seção são apresentados os casos de uso para o desenvolvimento do protótipo, conforme ilustra a Figura 7. Identificou-se os atores, denominado `Administrador`, `Leiturista`, `Proprietário` e `Agendador de tarefa`, o qual utiliza todas as funcionalidades do protótipo.

Figura 7 - Diagrama de caso de uso



O ator `Administrador` tem como objetivo realizar as consultas das leituras e os cadastros de:

a) unidade habitacional através do caso de uso `UC01 – Manter unidade`;

b) apartamento, através do caso de uso `UC02 – Manter apartamento;`

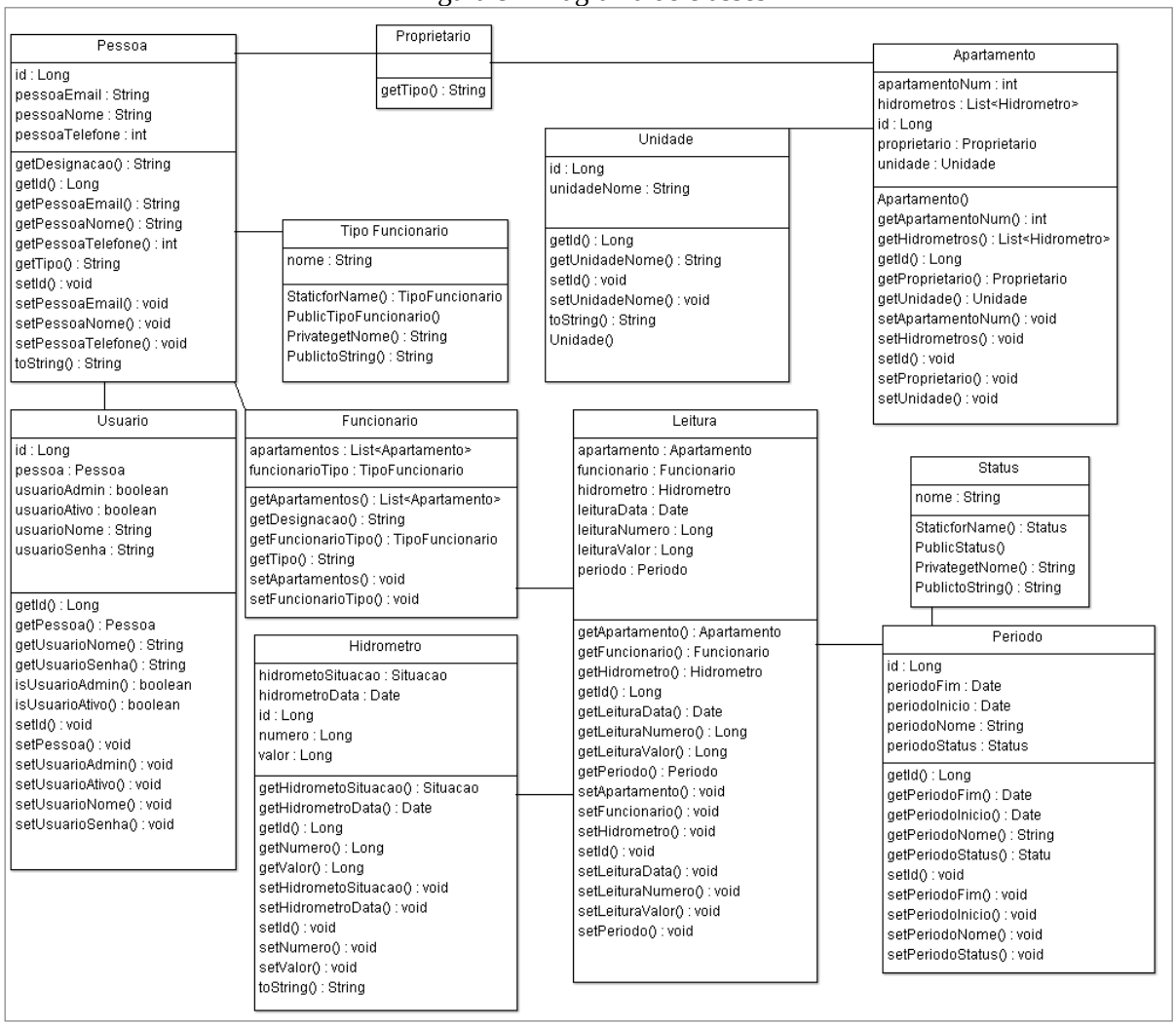
c) usuário/proprietário, utilizando o caso de uso `UC03 – Manter usuário`.

O ator denominado `Leiturista` deve indicar o local da imagem a ser coletada por meio do caso de uso `UC04 – Indicar a unidade habitacional e apartamento`. Em seguida, é feita a captura da imagem do hidrômetro pelo caso de uso `UC05 – Realizar a captura da imagem`, transmitindo para o serviço de reconhecimento os caracteres via caso de uso `UC06 – Enviar imagem para processamento`. O ator `Proprietário` poderá realizar a consulta da leitura do seu hidrômetro, utilizando o caso de uso `UC07 – Consultar resultado gerado pelo protótipo`. O ator `Agendador de tarefas` se refere ao serviço, no servidor, que tem como objetivo processar a imagem recebida pelo caso de uso `UC08 – Processar a imagem recebida utilizando OCR`, armazenando o resultado, que é realizado pelo caso de uso `UC10 – Armazenar resultado das imagens`. Após o `Agendador de tarefas` realizar a leitura e o armazenamento do resultado, o ator `Administrador` poderá consultar as leituras, por meio do caso de uso `UC11 – Consultar leituras`. O detalhamento dos casos de uso encontra-se no Apêndice B.

3.2.2 Diagrama de classe

Nesta Seção constam os diagramas de classe do protótipo, com relacionamentos e estruturas dos objetos. Sendo que do lado servidor será detalhado o módulo administrativo e o de reconhecimento de imagens. Já no lado cliente, não serão apresentadas classes, pois a sua estrutura é construída em HTML, ao qual não define classes. A Figura 8 mostra as classes referentes ao módulo administrativo.

Figura 8 - Diagrama de classes



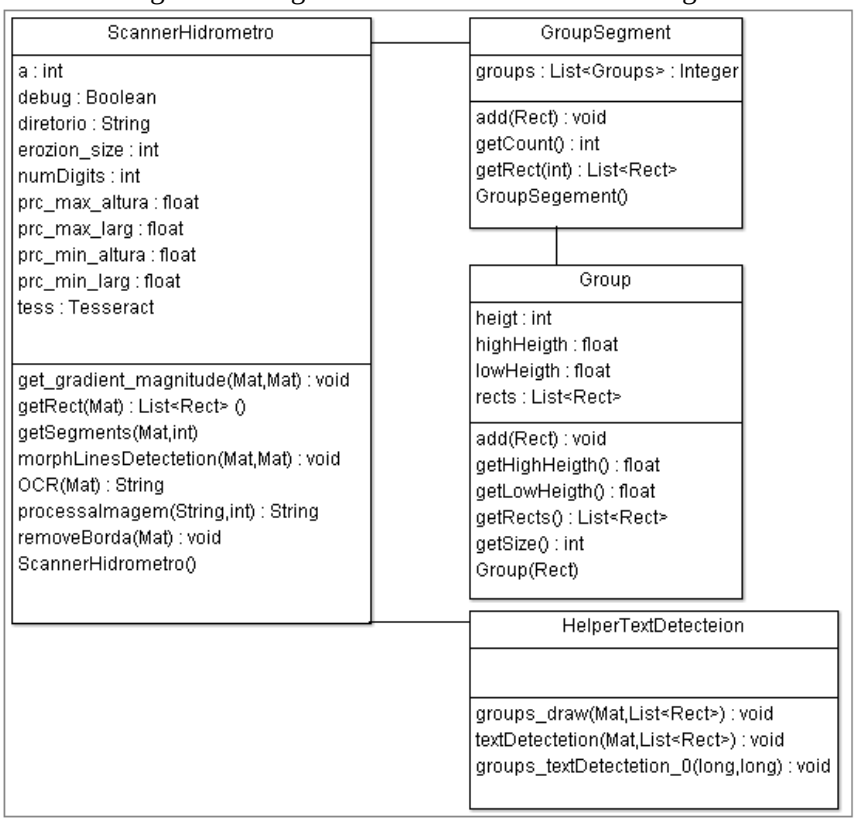
As classes do módulo Administrativo servem para controlar o cadastro e a manutenção das informações que devem ser feitas pelo Administrador, sendo elas:

a) classe `Pessoa` – representa a estrutura do cadastro e tabela `Pessoa`;
b) classe `Funcionário` – classe que determina as características do funcionário;
c) classe `Usuário` – classe que representa a estrutura do usuário do protótipo;
d) classe `Proprietário` – classe que apresenta a estrutura do proprietário;
e) classe `Unidade` – classe que representa a estrutura da unidade habitacional;

f) classe `Apartamento` – classe utilizada para estrutura do cadastro de apartamento;
g) classe `Leitura` (status) – classe responsável pela estrutura de armazenamento e leitura dos períodos para realizar a coleta das imagens;
h) classe `Hidrometro` – classe responsável pela estrutura de armazenamento das informações dos hidrômetros.

Para o módulo de reconhecimento de imagem, é apresentado o diagrama de classe ilustrado na Figura 9.

Figura 9 - Diagrama de classe - Scanner da imagem



A principal classe desse módulo é a `ScannerHidrometro`, que contém os métodos responsáveis pela aplicação de filtros para a imagem (aplicação de tons de cinza, binarização, segmentação e limpeza) e determinação da área de interesse contendo o conteúdo a ser traduzido. A classe `HelperTexDetectetion`, recebe a área de interesse detectada e aplica o `Tessereact` para extrair o conteúdo da região apresentada.

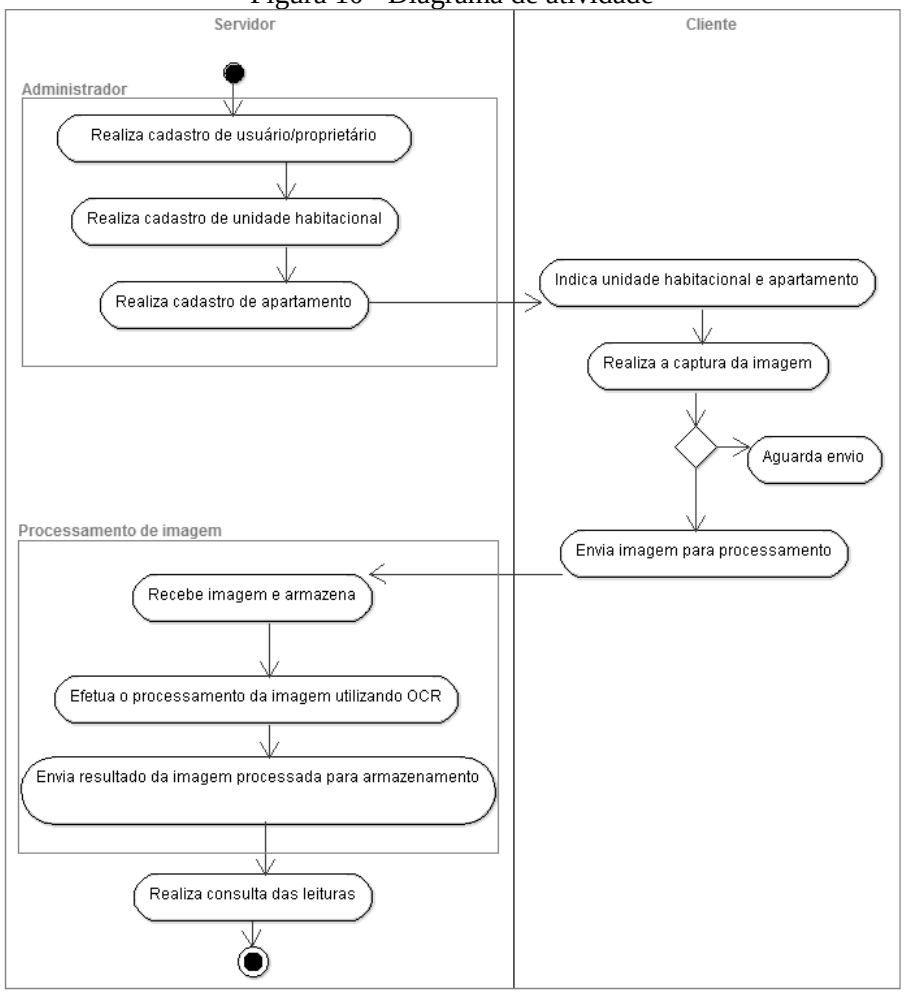
As classes `Group` e `GroupSegment` são utilizadas na etapa da segmentação onde as mesmas formam grupos de segmentos conforme sua altura, variando 20% entre o tamanho máximo e mínimo, ou seja, cada segmento pode variar 20% dentro do grupo da sua altura, caso seja este percentual esteja fora deste padrão, é criado um novo grupo de segmentos. Com base nos parâmetros de altura máxima (98%) e mínima (40%), o algoritmo verifica qual grupo

contém a quantidade de segmentos iguais a quantidade de caracteres a ser reconhecido na imagem (também passado por parâmetro), considerando o grupo em questão valido para a interpretação da imagem.

3.2.3 Diagrama de atividade

A Figura 10 apresenta o diagrama de atividades das ações executadas no lado cliente, servidor e as funções do protótipo.

Figura 10 - Diagrama de atividade



Na primeira etapa, o `Administrador` realiza os cadastros básicos, sendo estes de usuário, unidade habitacional e apartamento. Após a realização destas atividades, o usuário classificado como `Leiturista` deve efetuar a coleta das imagens indicando para cada imagem a unidade habitacional e apartamento. O `Leiturista` também tem a função de enviar as imagens para que o servidor realize o reconhecimento dos caracteres. Ao receber as imagens, o servidor executará os algoritmos de `OpenCV` e `Tesseract` para realizar a identificação e tratamento da imagem e em seguida a sua interpretação. Ao final, tanto

`Administrador` quanto o usuário `Proprietário` podem consultar os resultados das leituras realizadas.

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas na implementação do protótipo, a descrição do desenvolvimento do trabalho e a sua operacionalidade.

3.3.1 Técnicas e ferramentas utilizadas

Para o desenvolvimento deste protótipo foi utilizado o ambiente de desenvolvimento Netbeans IDE 8.0.2 e a linguagem de programação Java versão 8.0. Também foram utilizadas as bibliotecas de visão computacional JavaCV, `opencv-300.jar`, `tess4j.jar`, `itext-2.1.7.jar`, `log4j-1.2.17.jar`, `jai_imageio.jar`, `jna.jar`, `commons-io-2.4.jar`, `commons-logging-1.1.3.jar`, `commons-beanutils-1.9.1.jar` e as bibliotecas: `spring-boot-starter 1.2.5` , `sping-security3.2.7`, `postgresql 9.4-1200`, `thymeleaf 2.1.4`, `tomcat-enbed 8.0.23` para desenvolvimento do serviço. A biblioteca JavaCV é uma implementação em Java da biblioteca OpenCV que apenas é utilizável para C e C++. Dessa biblioteca foram utilizadas rotinas como segmentação, operadores morfológicos, conversão para tons de cinza entre outros.

Para o desenvolvimento do aplicativo foi utilizada a ferramenta Phonegap na versão 5. PhoneGap é uma plataforma para desenvolvimento de aplicativos móveis que utiliza tecnologia Apache Cordova para acessar a funções dos aparelhos móveis como câmera e geolocalização, e permite por meio de alguns que os desenvolvedores criem aplicações utilizando HTML5, CSS3 e JavaScript sem a necessidade de depender de APIs específicas. Essa plataforma é compatível com os sistemas operacionais: iOS, Windows Phone e Android.

3.3.2 Etapas do desenvolvimento

O desenvolvimento foi realizado seguindo os princípios da arquitetura cliente-servidor. No lado servidor ficam os módulos administrador e de reconhecimento de imagem e no lado cliente a implementação da interface do aplicativo responsável pela coleta dos dados.

3.3.2.1 Cliente

Esta seção é destinada a mostrar o funcionamento da aplicação. A tela inicial do aplicativo possibilita escolher entre as ações de registrar, enviar dados pendentes e configurações. A opção de registrar a leitura está representada na Figura 11. Nela, o usuário

deve indicar a unidade e o apartamento do qual será coletada a imagem do hidrômetro. Ao acionar o botão Leitura, o protótipo chama a funcionalidade de máquina fotográfica do *smartphone*.

Figura 11 - Efetuando leitura

O Quadro 2 apresenta parte do código JavaScript da página responsável por chamar a câmera do dispositivo móvel para realizar a captura da imagem do hidrômetro.

Quadro 2 - JavaScript para registrar a leitura

```
1   //regisrar. js
2
3   function startScan() {
4        navigator.camera.getPicture(onSuccess, onFail, { quality: 60 ,
5           destinationType : Camera.DestinationType.DATA_URL,
6           sourceType : Camera.PictureSourceType.CAMERA,
7           encodingType: Camera.EncodingType.JPEG,
8           targetWidth :500,
9           saveToPhotoAlbum: false
10       });
```

Todas as imagens capturadas só serão enviadas a partir do momento que o usuário acionar o botão Enviar, conforme mostra a Figura 12.

Figura 12 - Enviar imagens pendentes

Ao acionar o botão enviar, o protótipo busca em uma tabela no dispositivo móvel as imagens armazenadas e, para cada registro encontrado, a imagem é carregada e, passando como parâmetro, conforme mostra o Quadro 3.

Quadro 3 - Enviar imagem

```
1   //busca a imagem na tabela do celular  e prepara mandar
2    this.enviar = function(){
3        function txEnviarSusscess(tx, results) {
4           var len = results.rows.length;
5           for (var i=0; i<len; i++) {
6             var leitura = results.rows.item(i);
7             uploadPhoto(leitura);
8           }
9       }
10
11      function txEnviar(tx){
12        var sql = "select l.apartamento_id, a.numero ,l.data,l.image
        from leituras l INNER JOIN  apartamentos a on l.apartamento_id =
        a.id  ";
13        tx.executeSql(sql,[] ,txEnviarSusscess);
14      }
15      db.transaction(txEnviar, callbackError, callbackSuccess);
16    }
17  // para cada registro que achou carrega a foto...
18  function uploadPhoto(leitura) {
19        var options = new FileUploadOptions();
20        options.fileKey ="file";
21        options.fileName ="leitura_"+leitura.apartamento_id+".jpg";
22        options.mimeType ="image/jpg";
23        var data = b64toBlob(leitura.image,"image/jpg");
24        var params = new Object();
24        params.apartamentoId = leitura.apartamento_id;
25        console.log(usuario);
26        console.log(senha);
27        params.senha   = senha;
28        params.user    = usuario;
28        options.params = params;
30        options.chunkedMode = true;
31
32  var fileURI;
33        requestFileSystem(LocalFileSystem.TEMPORARY,
        0,onFileSystemSuccess, fail);
34        console.log("inicio File System");
35        function onFileSystemSuccess(fileSystem) {
36          console.log("File System"+fileSystem.name);
37
38  fileSystem.root.getFile("temp"+leitura.apartamento_id+".jpg",{create:
        true, exclusive: false}, gotFileEntry, fail);
39        }
40
41        function gotFileWriter(writer) {
42          console.log("inicio Write");
43          writer.write(data);
44          console.log("Fim Write");
45          var ft = new FileTransfer();
46          ft.upload(fileURI.toURL(), url_base+"/mobile/upload", win,
47  fail, options);
48        }
49     }
```

O Quadro 4 apresenta o trecho de código JavaScript onde o usuário da aplicação pode configurar o servidor, o usuário e a senha para envio das imagens.

Quadro 4 - JavaScript das configurações de envio

```
1  //configuracao.js
2
3        Controller.saveConfiguracao($("#urlField").val(),$("#usuarioFie
4  ld").val(),$("#senhaField").val());
```

3.3.2.2 Processamento da imagem e reconhecimento dos dígitos

Para realizar a interpretação da imagem, foram utilizadas as bibliotecas Tesseract que é usada para reconhecer os caracteres na imagem e em seguida fazer o OCR e conversão e, a biblioteca OpenCV para realizar o tratamento da imagem, alterando cores e detectando a ROI. O package `controller`, possui a classe `Controller`, que contém a classe `ScannerHidrometro`. Na classe `ScannerHidrometro` é chamada a classe `HelperTextDetection` e `GroupSegment` que estende a classe `Grupo`.

O algoritmo inicia com um conjunto pré-estabelecido de caracteres possíveis a serem detectados que, no caso deste protótipo, são apenas números entre 0 e 9. Após a determinação dos possíveis caracteres a serem reconhecidos, a imagem é carregada e, a partir disso, inicia-se o processamento de imagens, onde para cada região de texto é verificado se a largura da imagem original é mais larga que a região de texto. Se a imagem inteira for detectada como área de processamento, ela será descartada.

A seguir, o algoritmo determina as regiões de interesse, salva todas as imagens encontradas, converte-as para escala de cinza, em seguida, realiza a binarização. Por fim, efetua a segmentação das regiões.

O algoritmo de detecção de texto traz a ideia de classe específica de extremas regiões, onde são selecionados a partir de toda a árvore de componentes da imagem. O componente árvore é construída por um valor de limiar que vai aumentando passo-a-passo de 0 a 255 e, em seguida, ligando as componentes conexas obtidas a partir de níveis sucessivos numa hierarquia por sua relação de inclusão.

O componente de árvore pode conter um grande número de regiões, mesmo para uma imagem muito simples, este número pode chegar facilmente da ordem de 1 x 10 ^ 6 regiões para uma imagem média de 1 megapixel. A fim de selecionar com eficácia as regiões adequadas.

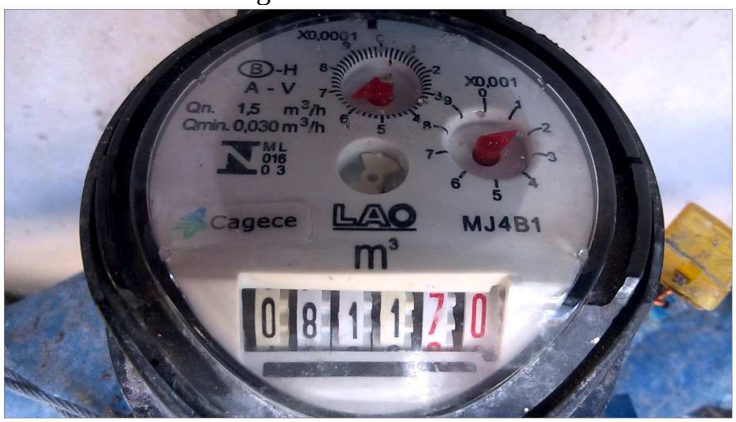
Primeiramente o algoritmo de detecção de texto computa a área, o perímetro, a caixa delimitadora e o número de Euler, que são calculados para cada região e usados como

recursos para um classificador que estima a condicional de classe probabilidade. Apenas as regiões extremas que correspondem ao máximo local da probabilidadesão selecionados (se a sua probabilidade é acima de um limite global e `p_min` a diferença entre o máximo local e mínimo local é maior do que um valor `delta_min`).

Na segunda etapa, as regiões extremas que passaram na primeira fase são classificadas em classes de personagens. O processo de filtragem da região é feito em diferentes projeções no canal de entrada, a fim de aumentar a identificação de caracteres. Depois de realizar a filtragem é feito o agrupamento dos blocos de texto de alto nível.

Para facilitar a compreensão dessas etapas, cada uma das etapas do algoritmo `ScannHidrometro,` estão detalhadas na sequência, tendo a Figura 13 como exemplo.

Figura 13 - Hidrômetro



A primeira etapa o algoritmo detecta e determina as regiões de interesse, conforme apresentado no Quadro 5.

Quadro 5 - Determinar a ROI

```
1   List<Rect> regions = new ArrayList();
2   HelperTextDetection.textDetection(original, regions); //busca regiões de
3   texto
4   Rect regMaior = null;
5
6   int num_region =0;
7   // para cada regiao de texto
8    for (int i = 0; i < regions.size(); i++) {
9     Rect rect = regions.get(i);
10    Mat roi =new Mat();
11  // se a largura da imagem origial e mais que a regiao de texto
12     if (original.cols() >= rect.width+rect.x &&original.rows()>=
13  rect.height+rect.y)
14       original.submat(regions.get(i)).copyTo(roi);
15       //submat faz ROI
16     else
17       // se a imagem toda for determinada a ROI, então faz a cópia
18          original.copyTo(roi);
19      // salvando as regioes encontradas
20        Imgcodecs.imwrite(diretorio+"r"+num_region+".png",roi);
```

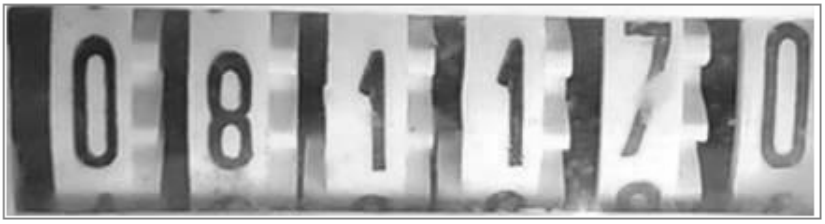
O Quadro 6 apresenta a implementação do tratamento da imagem para converter uma imagem em escala de cinza. Foi utilizada a constante `COLOR_BGR2GRAY`.

Quadro 6 - Aplicar escala de cinza

```
1   private List<Rect> getSegments(Mat src,int id) {
2           // Transform source image to gray if it is not
3           // matriz de pixel
4           Mat gray = new Mat();
5           if (src.channels() == 3) {// se conter 3 canais(RGB)
6                                                   então converte
7               Imgproc.cvtColor(src, gray, Imgproc.COLOR_BGR2GRAY);
8           } else {
9               // se estiver cinza então não muda
10              gray = src;
11          }
```

A Figura 14 ilustra a escala de cinza a partir da imagem original.

Figura 14 - Imagem em tons de cinza



Para a obtenção da imagem binarizada, a imagem em tons de cinza é previamente computada e em seguida submetida à técnica de *Thresholding*, conforme apresentado no Quadro 7. A Figura 15 ilustra o resultado da aplicação da binarização na imagem obtida após a conversão para escala de cinza. Após a aquisição da imagem binarizada é realizada a segmentação dos elementos a serem interpretados, conforme mostra o Quadro 8. A Figura 16 ilustra como o algoritmo que determina a região do caractere.

Quadro 7 - Aplicar a binarização

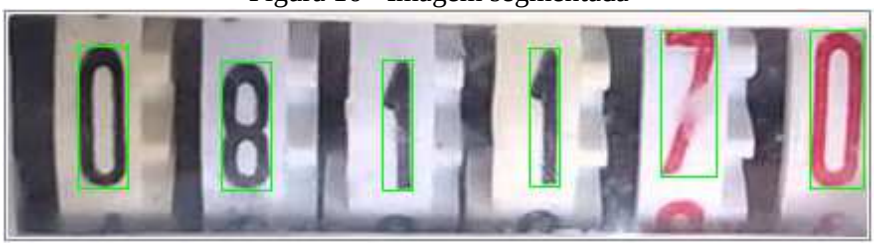
```
1   // Apply adaptiveThreshold at the bitwise_not of gray, notice the ~
2   symbol
3           Mat bw = new Mat();
4           Core.bitwise_not(gray, gray);// 2 -  binariza a imagem
5           Imgproc.adaptiveThreshold(gray,             bw,             255,
6   Imgproc.ADAPTIVE_THRESH_MEAN_C, Imgproc.THRESH_BINARY, a, -2);//olhar no
7   opencv = filtros
```

Figura 15 - Imagem binarizada

Quadro 8 - Realiza a segmentação

```
1         List <Rect> segments = getRect(bw);
2         // Somente para monstrar segmentos selecionados da imagem
3         if (segments !=null){
4         for (Rect rect:segments){
5             //cria o contorno nos caracteres
6          Imgproc.rectangle(src, new Point(rect.x,rect.y),
7                                       new           Point(rect.x+rect.width,
8    rect.y+rect.height),
9                                       new Scalar(0,255,0),1);
10          }
11         Imgcodecs.imwrite(diretorio+"segment"+id+".png",src);
12         return segments;
13        }
14         else
15           return new ArrayList();
16     }
```

Figura 16 - Imagem segmentada



O algoritmo irá realizar os filtros em todas as regiões entornadas e ao qual são possíveis de fazer reconhecimento de caracteres. Assim, consegue-se determinar a região e a limpeza da imagem, conforme apresenta o trecho de código do Quadro 9.

Quadro 9 - Limpeza da imagem

```
1    //primeiro: se o número de dígitos for diferente desconsidera a
2    região
3    ...
4    // segundo: se a imagem não for mais larga que alta, descarta
5    // terceiro verifica se é vazia
6            if (regions.isEmpty()) {
7                return null;
8            }
9            Mat matROI = new Mat();
10           Rect areaRegiao = regions.get(0);
11           //pega a regiao que sobrou e utiliza com se fosse ROI
12           original.submat(areaRegiao).convertTo(matROI,
13   original.type());
14           //
15           //HelperTextDetection.groups_draw(original, regions);
16           Imgcodecs.imwrite(diretorio + "ROI.png", original);
17           Mat matOCR =new Mat();
18           // repete o processo para a imagem que sobrou escala cinza,
19   binariza e segmenta e descarta conteúdo fora do segmento
20   transformando em preto
21           moprhLinesDetection(matROI,matOCR);
22           Imgcodecs.imwrite(diretorio + "imgOCR.png", matOCR);
23           Imgcodecs.imwrite(diretorio + "roi" + ".png", matOCR);
24           String text = OCR(matOCR);
24           System.out.println("Roi" + ":" + text);
25           return text;
26       }
```

O resultado da limpeza pode ser visualizado na Figura 17. Dessa forma, a imagem está pronta para ser interpretada.

Figura 17 - Imagem OCR

No passo seguinte, é realizado a conversão da área de interesse em texto. Para isso, o algoritmo utiliza o Tesseract (`tess.doOCR`), indicado no Quadro 10.

Quadro 10 - Leitura

```
1   private  String OCR(Mat src){
2           Imgcodecs.imwrite(diretorio + "seg" + ".png", src);
3           MatOfByte bytemat = new MatOfByte();
4           Imgcodecs.imencode(".jpg", src, bytemat);
5           byte[] bytes = bytemat.toArray();
6           InputStream in = new ByteArrayInputStream(bytes);
7           try {
8                  BufferedImage buff = ImageIO.read(in);
9                return tess.doOCR(buff).trim();
10          } catch (IOException ex) {
11              return null;
12          } catch (TesseractException ex) {
13              return  null;
14          }
```

3.3.2.3 Servidor

Nessa seção são destacadas algumas funcionalidades contidas nos casos de uso associadas ao ator `Administrador` e `Agendador de tarefas`.

Para melhor entendimento, a aplicação servidora foi estruturada em pacotes, sendoeles:

a) `webcontroller` com as classes de controle de acesso ao protótipo (`AppConfig.java`, `Apllication.java` e `SecurityConfig.java`) e classe que recebe a imagem a ser processada, conforme apresentado no Quadro 11;
b) `webcontroller.controller`: contém as classes que controlam o fluxo da aplicação WEB;
c) `webcontroller.domain`: contém as classes do modelo da aplicação WEB;
d) `webcontroller.repository`: armazenador de tabelas;
e) `webcontroller.service`: classes de operações do modelo (salvar, editar, etc);
f) `webcontroller.validator`: validações da aplicação WEB;
g) `templates` (`views`): dentro desse pacote possui as classes de visões para as funcionalidades como `templates.apartamento`, `templates.fragments`,

`templates.funcionario`, `templates.leitura`, `templates.periodo`, `templates.proprietario`, `templates.unidade`, `templates.usuario`, para cada uma delas funcionalidade possui o template de consulta (index) e cadastro(edit). Após realizar o login, onde o usuário deve indicar o seu usuário e senha para que o sistema valide as informações de acesso e permissões, o protótipo e redirecionado para a tela inicial do protótipo;

h) `webcontroller.PDI`: package responsável pelo armazenamento das classes de interpretação da imagem.

Conforme apresentado no Quadro 11, o algoritmo recebe os parâmetros da imagem, cria o arquivo e localiza o apartamento em seguida, efetua a busca pelo hidrômetro e efetua a chamada para realizar o processamento da imagem descrito anteriormente.

Quadro 11 - Recebe a imagem

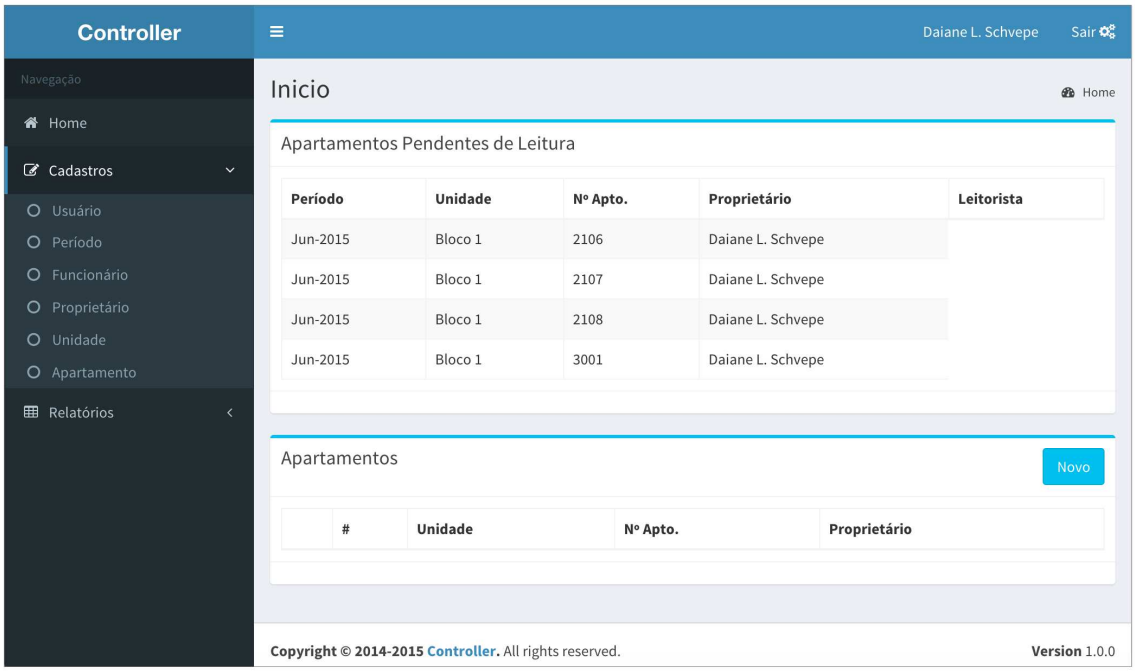
```
1  //receber a imagem
2  @RequestMapping(value = "mobile/upload", method = RequestMethod.POST)
3      public @ResponseBody
4      String handleFileUpload(@RequestParam("apartamentoId") Long
5  apartamentoId,
6                  @RequestParam("user") String user,
7                  @RequestParam("senha") String Senha,
8                  @RequestParam("file") MultipartFile file) {
9
10 //Encontrar Apartamento
11 Apartamento apartamento =
12 apartamentoService.findApartamento(apartamentoId);
13
14 //Buscar Hidrometro;
15 Hidrometro hidrometro = null;
16 for (Hidrometro h : apartamento.getHidrometros()) {
17                     if (h.getHidrometoSituacao() == Situacao.ATIVO) {
18                         hidrometro = h;
19                         break;
20                     }
21                 }
22      //Processar image
23      if (hidrometro != null) {
24      //pegar a ultima leitura
24      Leitura ultimaLeitura =
25      leituraService.getUltimaLeitura(hidrometro);
26      int  numDigitos =hidrometro.getHidrometoNumDigito();
27
28 Periodo periodo = periodoService.findByPeriodoAberto();
28 ScannerHidrometro scan = new ScannerHidrometro();
30 String text  = scan.processaImagem(image.getAbsolutePath(),numDigitos);
31       Leitura leitura = new Leitura();
32       leitura.setApartamento(apartamento);
33       leitura.setFuncionario(null);
34       leitura.setHidrometro(hidrometro);
35
36 leitura.setLeituraData(Calendar.getInstance().getTime());
37    Long numero =Long.valueOf(text);
38    leitura.setLeituraNumero(numero);
39    leitura.setPeriodo(periodo);
40    Long valor = numero - ultimaLeitura.getLeituraNumero();
41                 leitura.setLeituraValor(valor);
42                 leituraService.saveLeitura(leitura);
43                  }
44             }
45         }
46     }
47 }
```

Após os cadastros e realização dos processos de imagens o protótipo apresenta a tela de relatório de todas as leituras realizadas, conforme mostra a Figura 18.

Figura 18 - Relatório de leituras

Os demais cadastros do protótipo são semelhantes as imagens a seguir; o fluxo inicial do protótipo é realizar o login, onde o usuário deve indicar o seu usuário e senha o sistema valida as informações de acesso e permissões e então o protótipo é redirecionado para a tela inicial do protótipo, a Figura 19 demostra a pagina index com o menu para as funcionalidades de consulta ao cadastro e uma região apartamentos que ainda não tem leitura registrada e outra área com a listagem dos apartamentos.

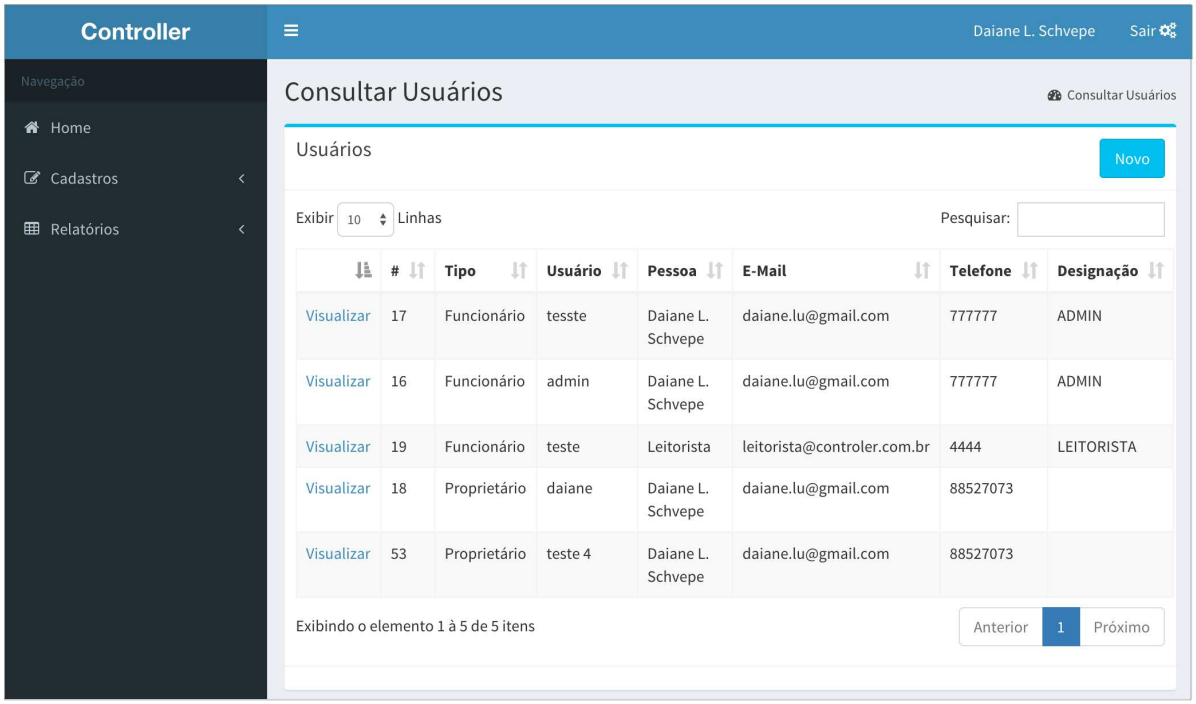
Figura 19 - Página inicial



Como exemplo será apresentado a seguir o cadastro de usuário, as demais telas estão apresentadas no Apêndice C.

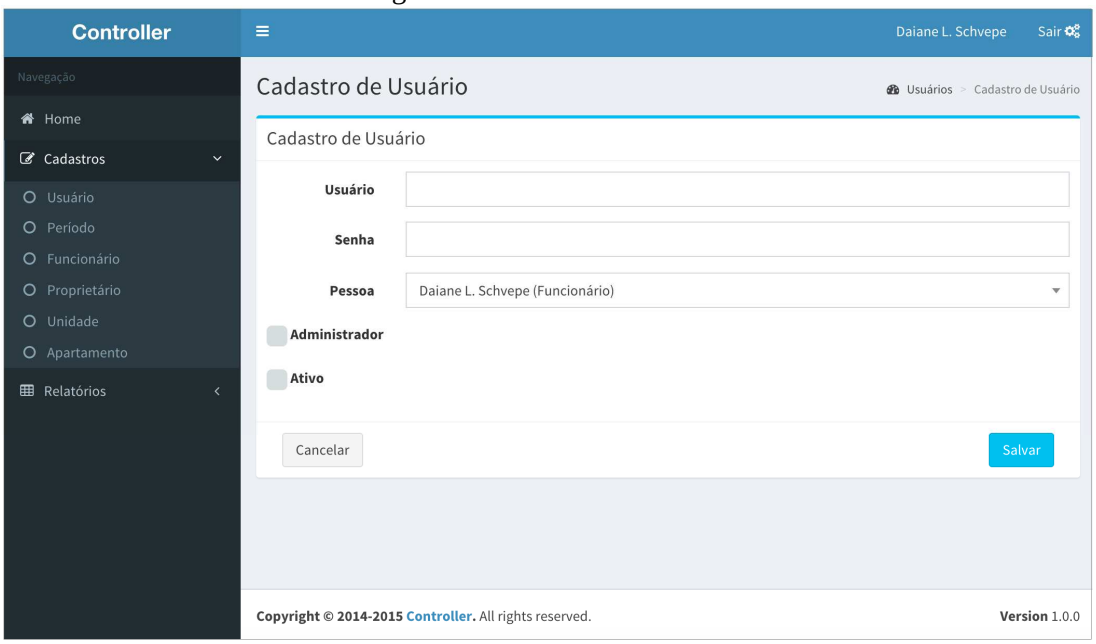
Ao acionar a opção de usuário do menu o protótipo apresenta a tela de consulta de todos os usuários cadastrados, Figura 20.

Figura 20 - Consultar usuários



Ao acionar a opção “Novo” na tela de consulta dos usuários o protótipo apresenta a tela de cadastro de novos usuários, destacado na Figura 21.

Figura 21 - Cadastro de usuário



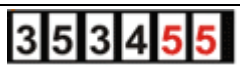
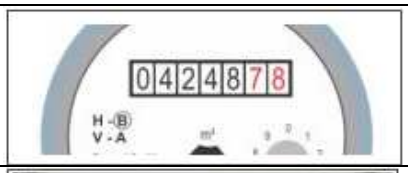
3.4 RESULTADOS E DISCUSSÕES

Esta Seção apresenta dois experimentos, um para validar a técnica de reconhecimento dos dígitos contidos nas imagens dos hidrômetros (Seção 3.4.1) e o outro para verificar a usabilidade do protótipo (Seção 3.4.2).

3.4.1 Experimento 01: validação da técnica de reconhecimento dos dígitos

O objetivo deste experimento é validar os resultados obtidos a partir da aplicação da técnica de OCR. Para isso foram utilizadas 9 amostras de imagens coletadas com diferentes níveis de nitidez. As imagens foram adquiridas a partir de sites de busca ou coletadas a partir de hidrômetros reais. O Quadro 12 apresenta os resultados alcançados.

Quadro 12 - Experimentos

	Imagem	Resultado do reconhecimento
1		5111551
2		0424875
3		0095
4		00005
5		5
6		119 91
7		031120
8		019827

9		1163

A partir do Quadro 12 percebe-se que dentre as 9 imagens submetidas ao reconhecimento, apenas a imagem 8 foi interpretada de forma correta. E, que as imagens 2, 6 e 9 alcançam resultados próximo ao esperado. Sendo que, as imagens 6 e 9 não foram interpretadas de forma correta, pois um dos seus dígitos está passando pelo processo de mudança de número. As demais imagens não alcançaram bons resultados devido à baixa qualidade da imagem, foco, bordas largas, sombreamento e, no caso da imagem 5, não foi possível realizar a interpretação da imagem.

3.4.2 Experimento 02: teste de usabilidade

O objetivo deste experimento é verificar a usabilidade do protótipo e a facilidade do seu manuseio, para isso, foi aplicado um questionário de usabilidade (apresentado no Apêndice C) com 5 pessoas.

O experimento foi feito de forma individual com o celular Iphone 6 e notebook Macbook Pro ambos com sistema operacional iOS. Foi fornecido a cada usuário um questionário de perfil, uma lista de tarefas e um questionário de usabilidade, que estão disponíveis no Apêndice C.

Os participantes inicialmente preencheram o questionário de perfil de usuário e, em seguida, foram orientados sobre o objetivo dos testes de usabilidade e do protótipo. A lista de tarefas foi composta por quatro tarefas, nas quais buscou-se contemplar todas as funcionalidades implementadas e disponíveis no protótipo. Ao finalizar de cada tarefa, foi solicitado que o usuário informasse se a tarefa foi executada com sucesso ou caso contrário, ele deveria relatar o problema ocorrida nas observações da tarefa.

Na primeira etapa, é feita a análise do questionário de perfil de usuário. No Quadro 13 são exibidos os perfis dos usuários envolvidos no teste de usabilidade.

Quadro 13 - Perfil dos usuários envolvidos no teste de usabilidade

Sexo	60% masculino 40% feminino
Idade	20% entre 18 e 25 anos 80% mais de 35 anos
Nível de escolaridade	40% ensino superior incompleto 60% ensino superior completo
Utilização de *smartphone*/*tablet*	100% sim

A partir do Quadro 13 percebe-se que todos os usuários voluntários que participaram do experimento possuem familiaridade com *smarphones*/*tablets*. Isso é bom, pois facilitou na realização das tarefas solicitadas no aplicativo cliente. No Quadro 14 estão apresentados os resultados quanto às questões envolvendo o cenário do aplicativo.

Quadro 14 - Perguntas quanto à lista de tarefas

Perguntas/Respostas	Sim	Não
1. Registrar a coleta de uma imagem do hidrômetro	100%	
2. Enviar as imagens para o servidor fazer o reconhecimento	100%	
3. Acessar a aplicação servidora	100%	
4. Verificação se a imagem foi transcrita para texto.	100%	

A partir do Quadro 14 percebe-se que todos os usuários conseguiram realizar todas as tarefas corretamente e, que as funcionalidades desenvolvidas para o protótipo se mostraram adequadas e funcionais. Porém, para verificar se o protótipo é fácil de usar, foi aplicado o questionário de usabilidade com questões fechadas, que estão disponíveis no Apêndice D. O Quadro 15 mostra os resultados obtidos do questionário de usabilidade aplicado.

Quadro 15 - Respostas do questionário de usabilidade

Perguntas	Respostas
1. Das tarefas solicitadas, quantas você conseguiu executar?	100% Todas
2. De um modo geral, você achou o protótipo intuitivo e fácil de usar?	100% Sim
3. Você achou fácil visualizar o resultado da imagem capturada?	60% Concordo parcialmente 40% Concordo totalmente
4. Qual é a sua avaliação do protótipo?	20% Muito bom 80% Bom

A partir do Quadro 15 pode-se observar que todos os usuários acharam o protótipo intuitivo e fácil de usar. Porém, também percebe-se que alguns usuários tiveram um certo desconforto em relação a visualização do resultado da imagem capturada. Isso pode ter ocorrido porque o usuário talvez tinha a expectativa de visualizar o resultado do reconhecimento no próprio aplicativo.

4 CONCLUSÕES

O objetivo principal do trabalho, que era realizar a interpretação da imagem tornando-a em caracteres do tipo texto, foi alcançado. O objetivo foi alcançado com o uso do Tesseract e OpenCV, comprovando que é possível obter caracteres a partir de uma imagem coletada.

O protótipo permite ao usuário realizar a indicação do número da unidade habitacional e apartamento de onde está sendo feita a coletada da imagem. Também permite ao usuário capturar a imagem enquadrando a ROI e, posteriormente, enviando-a para o servidor. Após o usuário realizar o envio da imagem ao servidor, o protótipo realiza a interpretação da imagem utilizando OCR e na sequência apresenta de forma textual o resultado obtido. O protótipo também armazena as imagens de acordo com número indicado para a unidade habitacional.

O uso do Tesseract não garante que a eficácia da interpretação seja perfeita mesmo que a área de interesse esteja devidamente tratada. Em imagens em que há poucos caracteres o índice de acerto é baixo, visto que imagens com caracteres maiores ou com uma boa iluminação são fáceis para realizar o reconhecimento dos dígitos.

A partir dos testes com usuários, constatou-se que a aplicação pode ser útil para fazer a leitura dos hidrômetros mesmo não tendo alcançado bons resultados na maioria dos casos de testes. Contudo, a ideia é válida. Porém, é necessário realizar mais estudos afim de aprimorar o reconhecimento dos dígitos.

4.1 EXTENSÕES

Dentre as extensões possíveis destaca-se:

a) aprimorar o algoritmo de reconhecimento dos caracteres;
b) possibilitar o uso do protótipo para outras plataformas de dispositivo móvel (Androide e Windows Phone);
c) ampliar a leitura também para relógios de energia.

REFERÊNCIAS

ALBUQUERQUE, M. P.; ALBUQUERQUE, M. P. **Processamento de imagens:** métodos e análises. Centro Brasileiro de Pesquisas Físicas MCT. Rio de Janeiro, 2000. Disponível em: < http://www.cbpf.br/cat/pdsi/pdf/ProcessamentoImagens.PDF> Acesso em: 03 abr. 2015.

DATAID. **O que é OCR?**. [S.l], [2003?]. Disponível em: <http://www.dataid.com/aboutocr.htm> Acesso em: 04 abr. 2015.

DALMUT, R. T. L. [S.l], 2013?. **Rateio de água. Critério mais justo**. Disponível em: <http://www.telecondo.com.br/index.php?option=com_content&view=article&id=34:rateio-de-agua-criterio-mais-justo&catid=6:artigos-e-noticias&Itemid=6>. Acesso em: 04 abr. 2015.

EIKVIL, L. Oslo, 1993. **OCR** Optical Character Recognition. [S.l], 1993. Disponível em: <http://www.nr.no/~eikvil/OCR.pdf >. Acesso em: 01 mar. 2015.

GOBBI, L. D. **Urbanização brasileira**. [S.l], [2015?]. Disponível em: <http://educacao.globo.com/geografia/assunto/urbanizacao/urbanizacao-brasileira.html> Acesso em: 01 mar. 2015.

KOHATSU, O. M. **ARTRANSLATOR**: Protótipo de um tradutor baseado em técnicas de reconhecimento ótico e realidade aumentada para móveis. 2012. 67 f. Monografia (Especialização de Desenvolvimento de Sistemas para Web). Engenharia da Computação, Universidade Estadual de Maringá, Maringá.

LANHELLAS, R. **Aplicando OCR em Java com Tesseract**. [S.l], 2012. Disponível em: <http://www.devmedia.com.br/aplicando-ocr-em-java-com-tesseract/31511>. Acesso em: 08 mar. 2015.

LEPTONICA. **Leptonica.** [S.l], 2011. Disponível em: <http://www.leptonica.com>. Acesso em: 10 maio 2015.

MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. **Processamento Digital de Imagens**, Rio de Janeiro: Editora Brasport,1999.

MAIA, G. **Saiba como calcular as taxas de condomínio**. [S.l], 2011 Disponível em: <http://cja-imobiliario.blogspot.com.br/2011/05/saiba-como-calcular-as-taxas-de.html>. Acesso em: 03 abr. 2015.

NASCIMENTO, J. D. **Detecção e reconhecimento de placa com baixo custo**. 2012. 110f. Trabalho de Conclusão de Curso (Engenharia da Computação). Centro Universitário de Brasília, Brasília.

SILVA, K. K. B. **Sistema Web para gestão de condomínio**. 2010. 83f. Trabalho de Conclusão de Curso (Engenharia da Computação). Universidade do Estado do Amazonas. Chapada.

SHIGEHARU JUNIOR, I. A. **Estudo da viabilidade do uso de um OCR na placa Beagleboard e sua integração no xLupa embarcado**. 2014. 67f. Monografia para conclusão de curso (Ciência da Computação). Centro e Ciências Exatas e Tecnológicas, Universidade Estadual do Oeste do Paraná, Cascavel.

SMITH, R. **An Overview of the Tesseract OCR Engine**. [S.l], 2007. Disponível em: <http://tesseract-ocr.googlecode.com/svn-history/r367/trunk/doc/tesseracticdar2007.pdf>. Acesso em: 06 abr. 2015.

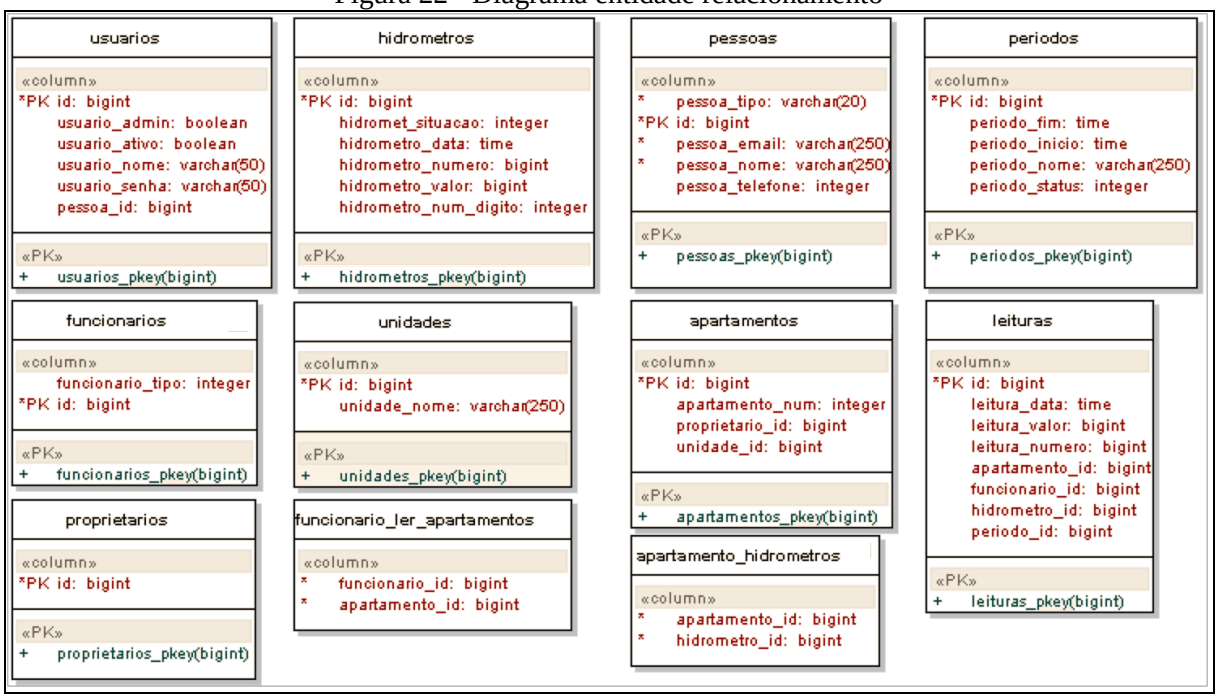
SOUZA, A. F. A. et al. **Processamento de imagens na identificação de veículos**. 2013. 8f. *Paper* de conclusão de curso (Engenharia Elétrica, Ênfase Eletrônica). Universidade São Judas Tadeu. São Paulo.

VALENTE NETO, R. A. **Introdução à morfologia matemática binária e em tons de cinza**. 2011. 22f. Relatório de Iniciação Científica (Departamento de Matemática). Centro de Ciências Exatas, Universidade Estadual de Londrina. Londrina.

APÊNDICE A – Dicionário de Dados

Este Apêndice apresenta da descrição das entidades de relacionamento do protótipo, a Figura 22 representa o diagrama entidade relacionamento do banco de dados e também o modelo utilizado para estruturar as classes.

Figura 22 - Diagrama entidade relacionamento



O diagrama entidade relacionamento apresentado na Figura 22, representa os dados que o protótipo apresentando as tabelas:

a) `pessoa`: entidade responsável pelo armazenamento do cadastro das pessoas, sendo elas usuários, proprietários ou funcionários;

b) `usuario`: entidade que armazena a associação entre a pessoa e os dados adicionais do cadastro de usuário;

c) `proprietario`: entidade que armazena a associação entre a pessoa e os dados adicionais de cadastro de proprietário;

d) `funcionario`: entidade que armazena a associação entre a pessoa e os dados adicionais de cadastro de funcionário;

e) `funcionario_ler_apartamento`: entidade responsável por armazenar o funcionário que efetuou a leitura do apartamento;

f) `periodo`: entidade responsável por armazenar os períodos de leitura dos hidrômetros;

g) `leitura`: entidade que armazena as leituras dos apartamentos e a associação do período da leitura;

h) `hidrometro`: entidade que armazena os dados do hidrômetro;
i) `unidade`: entidade responsável por armazenar os dados da unidade habitacional;
j) `apartamento`: entidade que armazena a associação entre a unidade habitacional e o cadastro dos apartamentos relacionados;
k) `apartamento_hidrometro`: entidade responsável por armazenar a associação entro o cadastro do apartamento e o cadastro do hidrômetro.

APÊNDICE B – Descrição dos Casos de Uso

Neste apêndice apresenta a descrição dos casos de uso conforme previstos no(s) diagrama(s) apresentado(s) na seção 3.2.1 que estão nos quadros de Quadro 16 a Quadro 26.

Quadro 16 - UC01 - Manter Unidade

UC01 Manter Unidade
Caso de uso onde pode-se registrar a unidade habitacional onde será associado o apartamento da imagem a ser coletada.

Ator: Administrador

Constraints:
Pré-condição – O ator deve estar logado no protótipo.
Pós-condição –Unidade habitacional cadastrado no protótipo.

Cenário:
1. O ator acessa o protótipo no formulário de consultas a unidades e aciona a opção de novo registro.
2. O protótipo apresenta o formulário de cadastro.
3. O usuário indica os dados e aciona o processo de armazenamento.
4. O protótipo valida os dados e armazenas as informações.
5. Fim do caso de uso.

Quadro 17 - UC02 - Manter Apartamento

UC02 Manter Apartamento
Caso de uso onde pode-se indicar o apartamento de onde será coletada a imagem

Ator: Administrador

Constraints:
Pré-condição – Execução do caso de uso UC01 – Manter Unidade.
Pós-condição –Apartamentos cadastrados.

Cenário:
1. O ator acessa o protótipo no formulário de consultas a apartamentos e aciona a opção de novo registro.
2. O protótipo apresenta o formulário de cadastro.
3. O usuário indica a unidade, e os demais dados e aciona o processo de armazenamento.
4. O protótipo valida os dados e armazenas as informações.
5. Fim do caso de uso.

Quadro 18 - UC03 - Manter usuário

UC03 Manter usuário
Caso de uso para o cadastro de usuários do protótipo e proprietário dos apartamentos.

Ator: Administrador

Constraints:
Pré-condição – O ator deve estar logado no protótipo.
Pós-condição – Usuário e proprietário cadastrados no protótipo.

Cenário:
1. O ator acessa o protótipo no formulário de consultas a usuários e aciona a opção de novo registro.
2. O protótipo apresenta o formulário de cadastro.
3. O usuário indica os dados e aciona o processo de armazenamento.
4. O protótipo valida os dados e armazenas as informações.
5. Fim do caso de uso.

Quadro 19 - UC04 - Indicar a unidade habitacional e apartamento

UC04 Indicar a unidade habitacional e apartamento
Caso de uso onde pode-se indicar o número da unidade habitacional e apartamento de onde será coletada a imagem

Ator: Leiturista

Constraints:
Pré-condição – O ator deve estar logado no protótipo.
Execução dos casos de uso UC01, UC02 e UC03.
Pós-condição –Indicação do número da unidade habitacional a ser coletada a informação.

Cenário:
1. O ator acessa o protótipo.
2. O ator indica o número da unidade habitacional.
3. O ator indica o número do apartamento.
4. Fim do caso de uso.

Quadro 20 - UC05 - Realizar a captura da imagem

UC05 Realiza a captura da imagem
Permite ao ator a captação de imagens que deseja realizar a conversão para texto.

Ator: Leiturista

Constraints:
Pré-condição – Execução do caso de uso UC04 Indica a unidade habitacional e apartamento.
Pós-condição – Captação das imagens para conversão em texto.

Cenário:
1. O ator posiciona o aparelho celular focando a imagem.
2. O ator capta a imagem a ser convertida em texto. (Caminho alternativo)
3. Fim do caso de uso.

Caminho alternativo:
1. O ator capta metade das imagens em um período e o restante em outro.
2. O protótipo deve compreender qual a última posição captada.
3. Fim do caminho alternativo (volta para o passo 2 do cenário).

Exceção:

Caso ocorra erro na captação o ator deve repetir o processo.

Quadro 21 - UC06 - Enviar imagens para processamento

UC06 Enviar imagens para processamento
Permite ao ator enviar as imagens captadas para processamento.

Ator: Leiturista

Constraints:
Pré-condição – Execução do caso de uso UC05 – Realiza a captura da imagem.
Pós-condição – Envio das imagens.

Cenário:
1. O ator seleciona as imagens.
2. O ator aciona o processo de envio de imagens para conversão.
3. Fim do caso de uso.

Quadro 22 - UC07 - Armazenar as imagens

UC07 Armazenar as imagens
Caso de uso armazenar a imagem obtida.

Ator: Agendador de tarefa

Constraints:
Pré-condição – Execução do caso de uso UC09 Processar imagem recebida.
Pós-condição – Imagens armazenadas.

Cenário:
1. O ator recebe o resultado das imagens “interpretada”.
2. O ator armazena o valor.
3. Fim do caso de uso.

Quadro 23 - UC08 - Processar a imagem recebida

UC08 Processar a imagem recebida
Permite ao ator executar o processamento da imagem

Ator: Agendador de tarefa

Constraints:
Pré-condição – Execução do caso de uso UC06 Envia imagem para processamento
Pós-condição – Imagem processada.

Cenário:
1. O ator recebe as imagens.
2. O ator executa a “interpretação” da imagem.
3. O ator opte o resultado da imagem em texto
4. Fim do caso de uso.

Exceção:
Em caso de falha no Cenário item 2 , deve repetir o processo até sucesso, então vai para item 3.

Quadro 24 - UC09 - Armazenar resultado da imagem

UC09 Armazena resultado da imagem
Caso de uso onde deve ser apresentado o resultado em texto da imagem obtida.

Ator: Agendador de tarefa

Constraints:
Pré-condição – Execução do caso de uso UC08 Processa imagem recebida
Pós-condição – Resultado em texto da imagem.

Cenário:
1. O ator recebe o resultado do processamento da imagem.
2. O ator armazena a imagem para unidade e apartamento associado.
3. Fim do caso de uso.

Quadro 25 - UC10 - Consultar resultado gerado pelo protótipo

UC10 Consultar resultado gerado pelo protótipo
Permite ao ator enviar o resultado das imagens captadas para processamento.

Ator: Proprietário

Constraints:
Pré-condição – Execução do caso de uso UC09 Armazena resultado da imagem
Pós-condição – Envio das imagens.

Cenário:
1. O ator recebe as imagens, realiza a leitura.
2. O ator aciona o processo de envio de imagens para conversão.
3. Fim do caso de uso.

Quadro 26 - UC11 - Consultar leituras

UC11 Consultar leituras
Caso de uso onde deve ser consultado o resultado em texto da imagem obtida por unidade/apartamento.

Ator: Administrador

Constraints:
Pré-condição – Execução do caso de uso UC09 Armazena resultado da imagem.
Pós-condição – Consulta dos valores do hidrômetro individual.

Cenário:
1. O ator acessa o repositório.
2. O ator executa a consulta.
3. O protótipo apresenta o valor gerado.
4. Fim do caso de uso

APÊNDICE C - Telas do protótipo WEB

O procedimento de cadastro e consulta ocorre para os demais itens do menu de cadastro, tela de consulta de período de leitura, Figura 23, e tela de cadastro de novo registro de período, Figura 24. Tela de consulta a funcionários (Leiturista), Figura 25, e tela de novo cadastro de funcionário apresentado na Figura 26. Tela de cadastro de proprietário, Figura 27 e tela de cadastro de novos proprietários, Figura 28. Tela para cadastro de unidade habitacional, Figura 29 e de novo cadastro, Figura 30. Tela de consulta a apartamentos já cadastrados, Figura 31, e uma tela de cadastro de apartamentos, Figura 32, que liga ao cadastro de unidade habitacional e proprietários já cadastrados.

Figura 23 - Consultar período

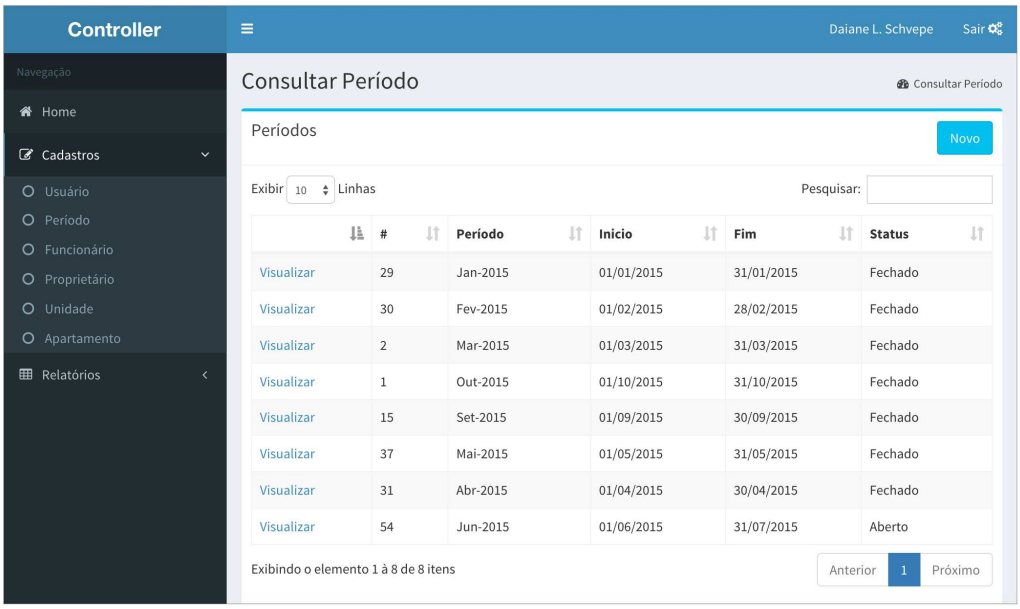


Figura 24 - Cadastrar período

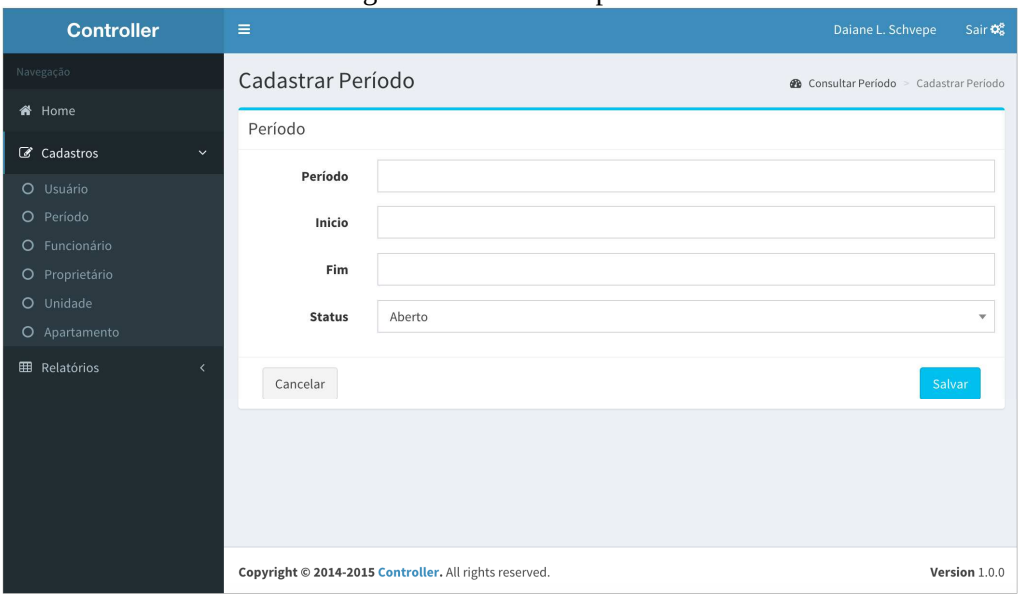


Figura 25 - Consultar funcionário

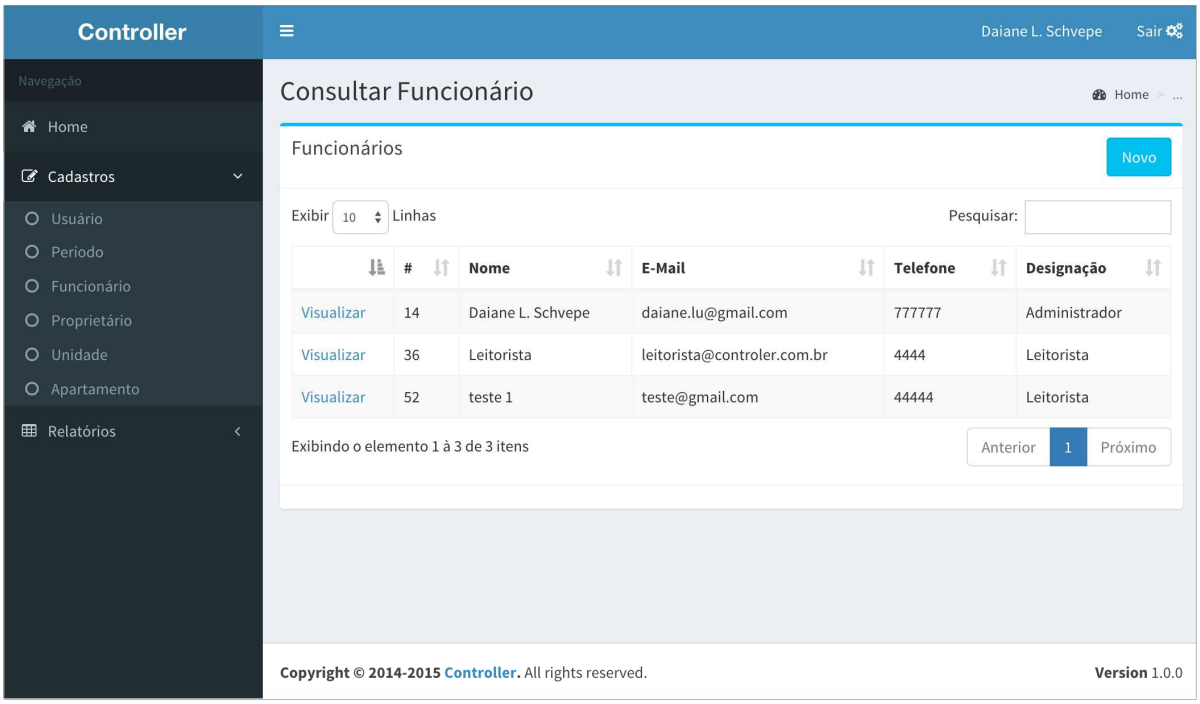


Figura 26 - Cadastro de funcionário

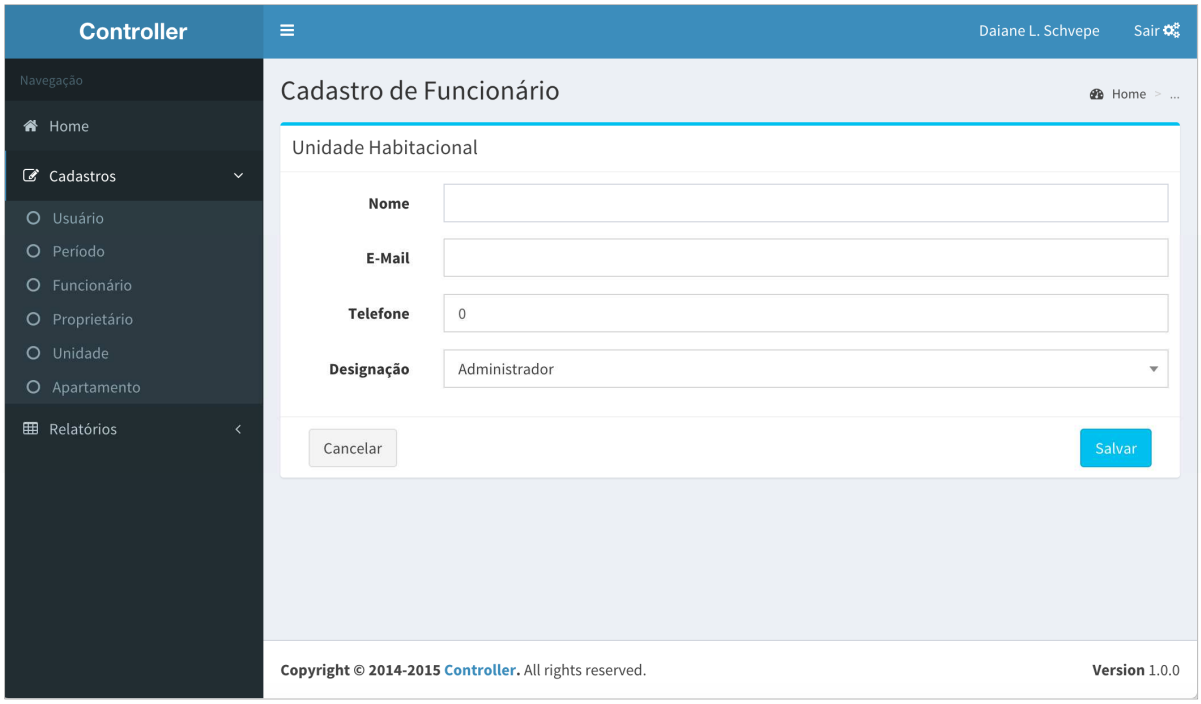


Figura 27 - Consultar proprietários

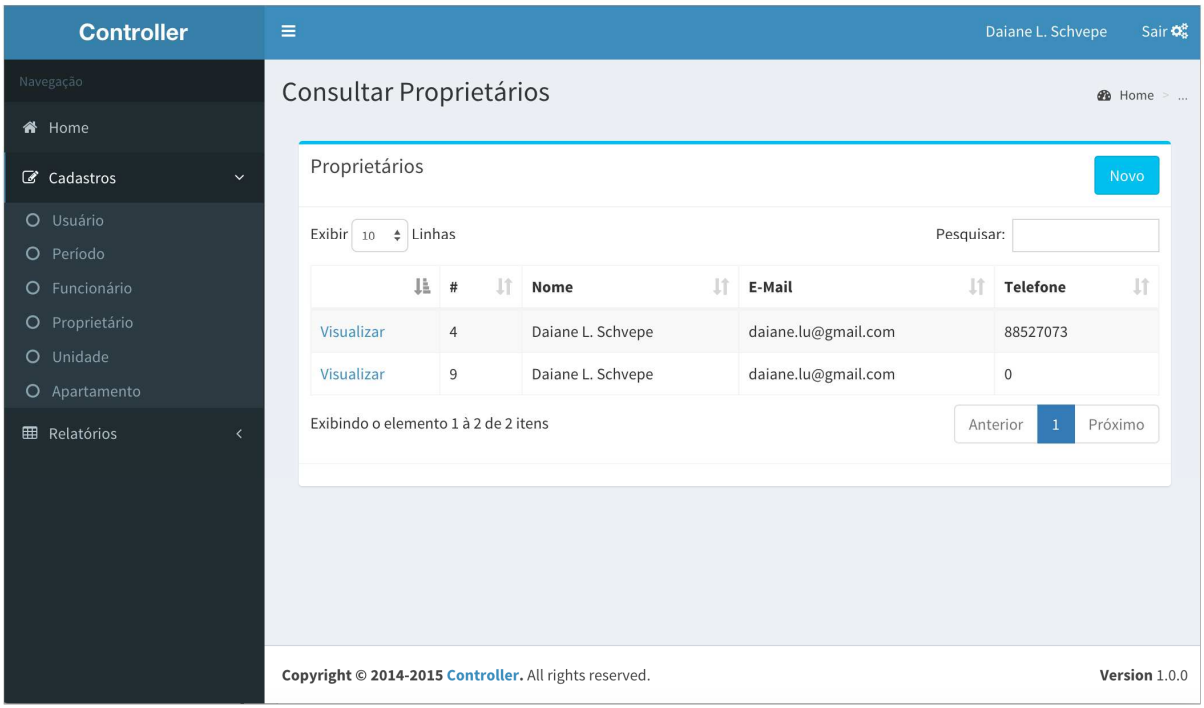


Figura 28 - Cadastro de proprietários

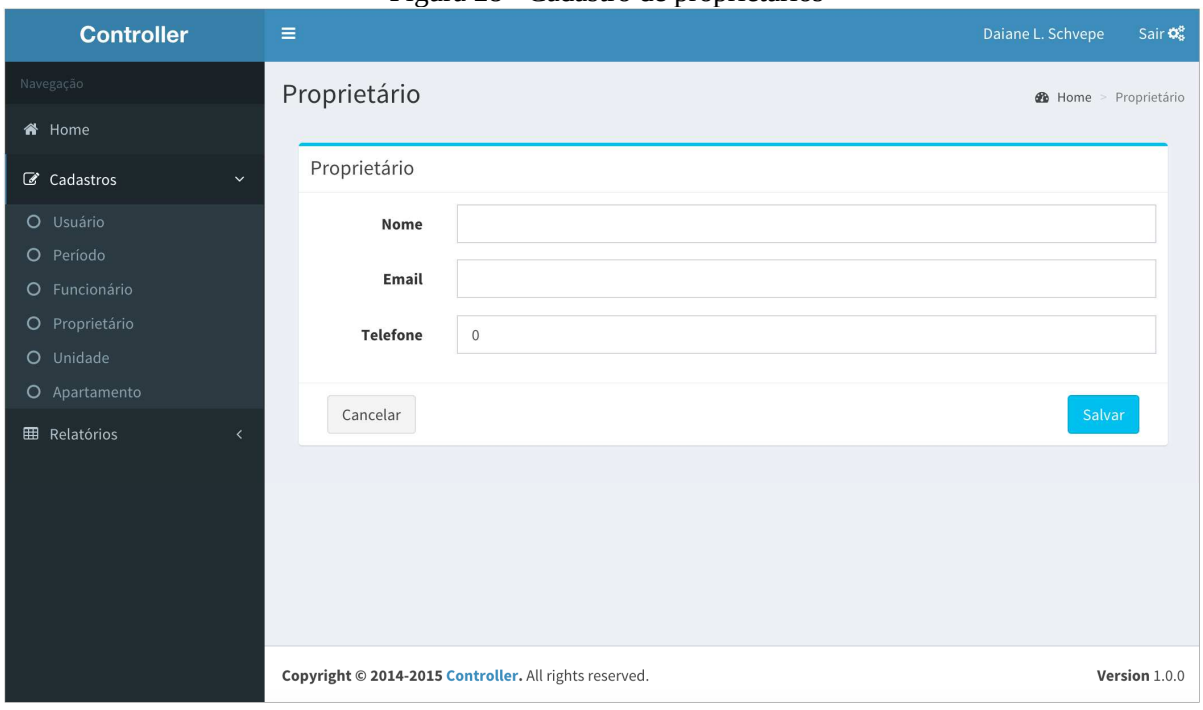


Figura 29 - Consultar unidade habitacional

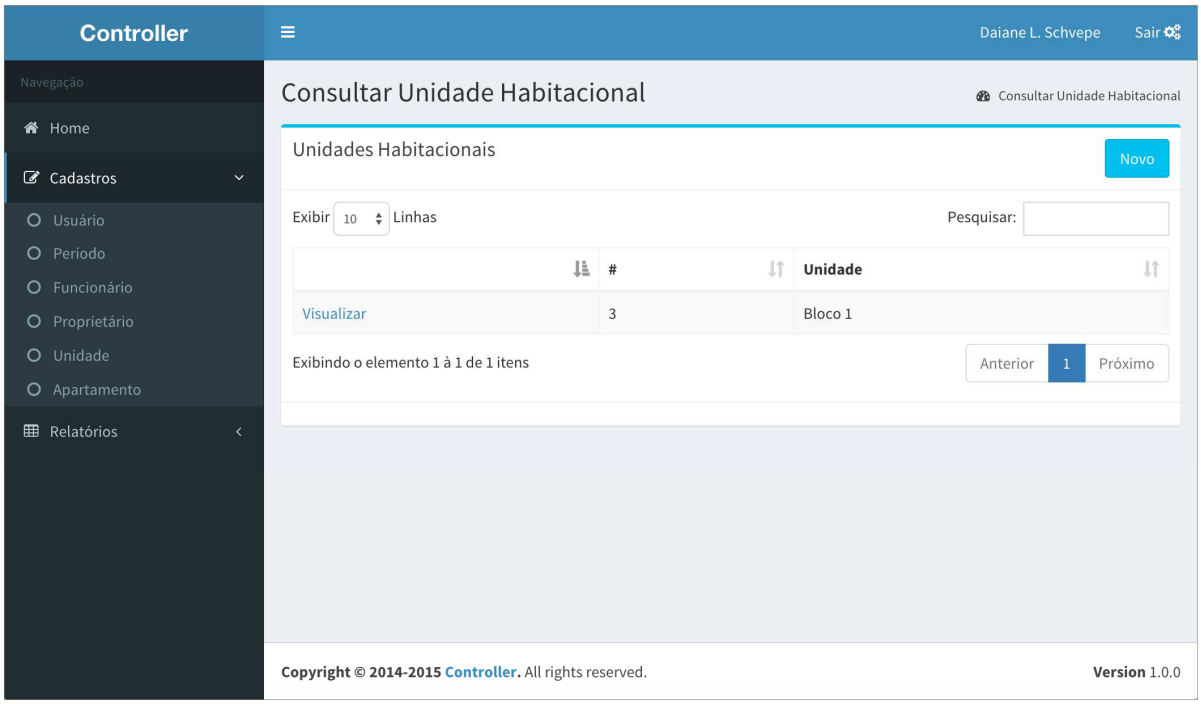


Figura 30 - Cadastro de unidade habitacional

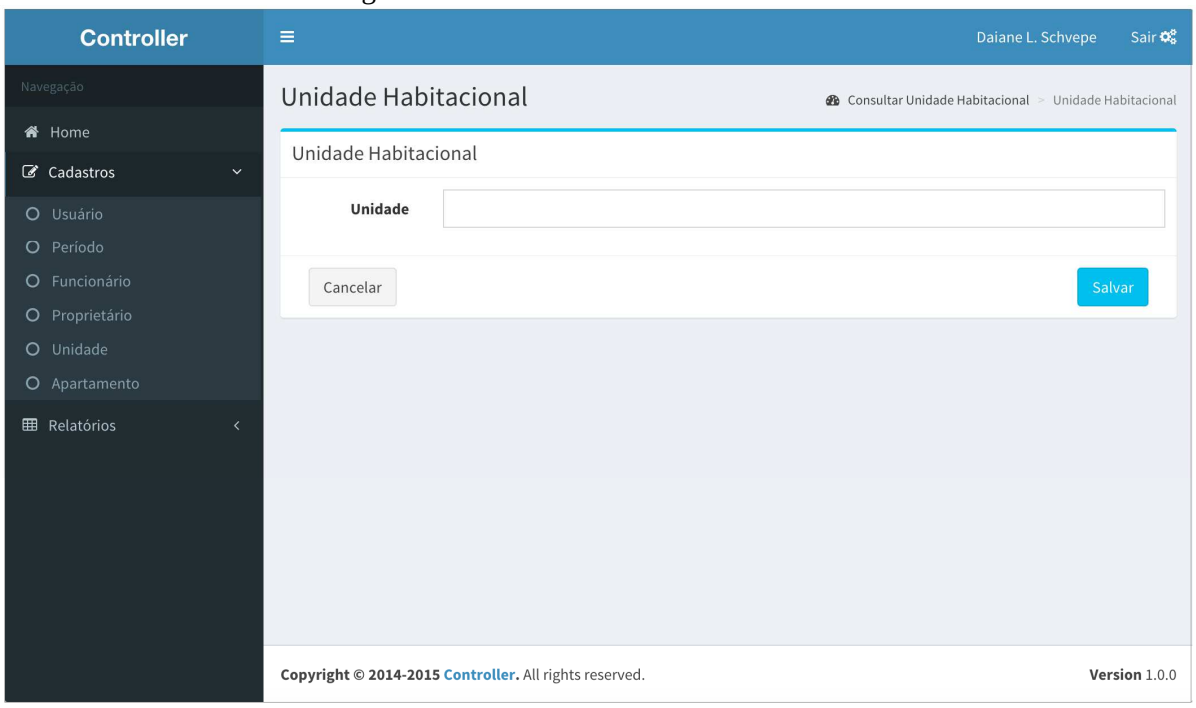


Figura 31 - Consultar apartamentos

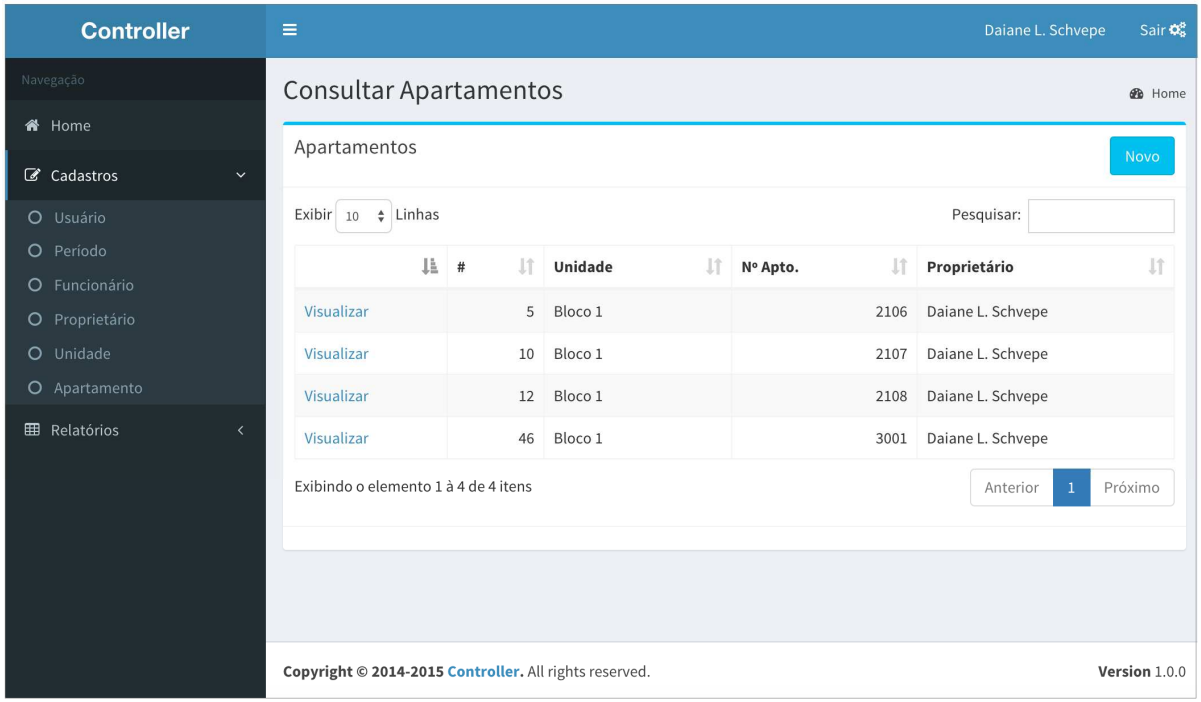
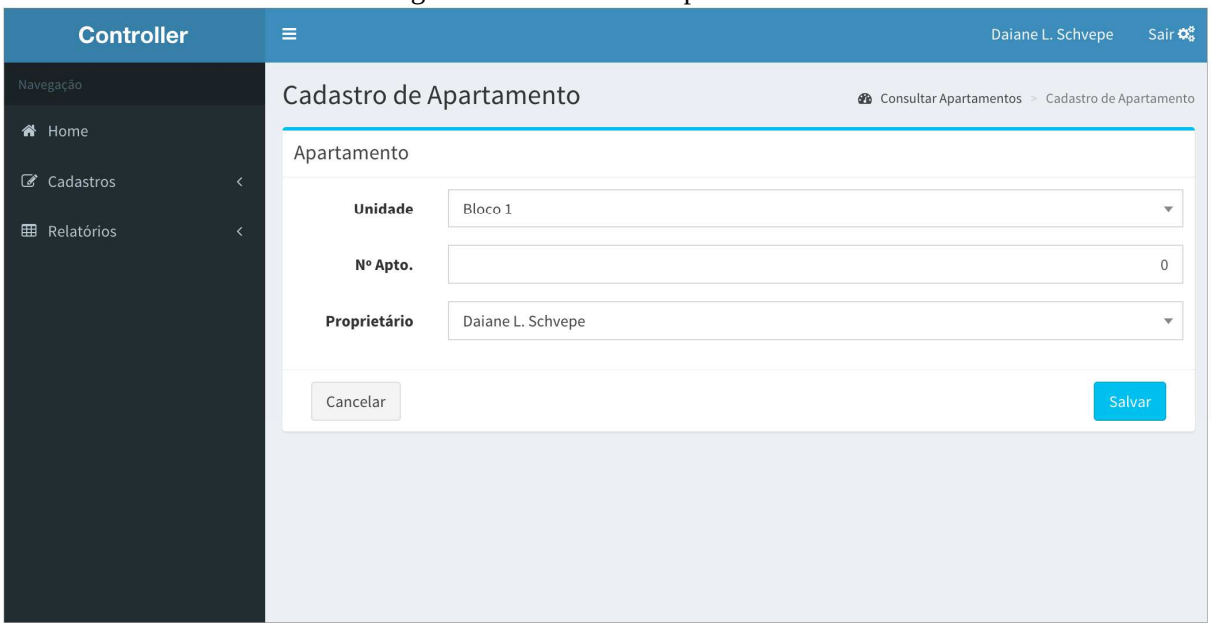


Figura 32 - Cadastro de apartamentos



APENDICE D – Roteiro e questionário de avaliação de usabilidade

Neste apêndice constam o questionário e o roteiro de testes que os usuários seguiram. O questionário de perfil do usuário está no Quadro 27, em seguida contém à lista de tarefas que conduz o usuário, testando as funcionalidades do protótipo no Quadro 28. No Quadro 29 consta o questionário de usabilidade.

Quadro 27 - Questionário de perfil do usuário

PERFIL DE USUÁRIO
Observação: as informações recebidas abaixo serão mantidas de forma confidencial.

Sexo:
( ) Masculino
( ) Feminino

Idade:
( ) Tenho menos de 18 anos
( )Tenhoentre18e25anos
( )Tenhoentre25e35anos
( ) Tenho mais de 35 anos

Nível de escolaridade:
( ) Ensino fundamental incompleto
( ) Ensino fundamental completo - 1o grau
( ) Ensino médio incompleto
( ) Ensino médio completo - 2o grau
( ) Ensino superior incompleto
( ) Ensino superior completo

Você possui algum dispositivo móvel (smartphone/tablet)?
( )Sim
( )Não

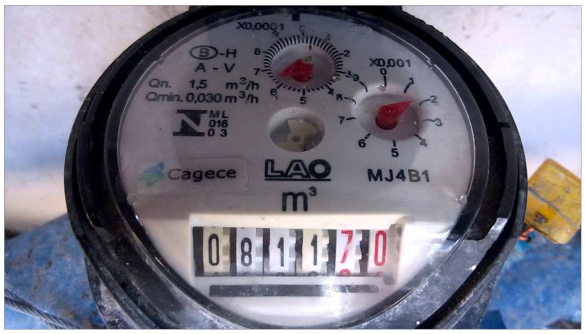
Você possui algum conhecimento na área de informática?
( )Sim
( )Não

Caso sim na pergunta anterior, qual o seu nível?
( )Básico
( )Intermediário
( )Avançado

Quadro 28 - Lista de tarefas

INSTRUÇÕES

Com este questionário buscamos avaliar a utilização do protótipo para realizar a captura de imagens e envio ao servidor. Um dos objetivos do protótipo é realizar a captura da imagem e o seu envio para o serviço de interpretação. Você pode utilizar o protótipo livremente por um período de 5 a 10 minutos para se ambientar. Ao finalizar, solicitamos que prossiga nos testes conforme as orientações abaixo.



Lista de tarefas a serem executadas:

1) Acessar o menu e escolher a opção "Registrar coleta", ao ser apresentada a tela, selecione um número de apartamento e na sequencia acione a opção "Leitura", posicione a área da câmera na imagem e tente centralizar na área dos números. Realize a captura.
   A tarefa foi executada? ( ) Sim ( ) Não
   Observação:_____
   _____

2) Acesse novamente o menu de opções e escolha “Pendentes”, e verifique se o número do apartamento selecionado está associado a imagem na lista de envio. Se sim, acione a opção “Enviar”.
   A tarefa foi executada? ( ) Sim ( ) Não
   Resultado conforme esperado? ( ) Sim ( ) Não
   Observação:_____
   _____

3) No computador, digite o endereço indicado na página de configurações do aplicativo, para usuário utilize “admin”, e a senha “122313” , no menu de opções selecione a opção “Relatórios”, no campo “Pesquisar”, digite o número do apartamento para o qual foi registrado a imagem.
   A tarefa foi executada? ( ) Sim ( ) Não
   Observação:_____
   _____

4) No resultado da pesquisa, verifique se para o apartamento foi registrada a imagem e o seu resultado em forma de texto.
   A tarefa foi executada? ( ) Sim ( ) Não
   Observação:_____
   _____

Quadro 29 - Questionário de usabilidade

QUESTIONÁRIO DE USABILIDADE

1. Das tarefas solicitadas, quantas você conseguiu executar?
( ) Todas
( ) A maior parte delas
( ) Metade das tarefas
( ) Menos da metade das tarefas
( ) Nenhuma tarefa

2. De um modo geral, você achou o protótipo intuitivo e fácil de usar?
( )Sim
( )Não

3. Você achou fácil visualizar o resultado da imagem captada?
( ) Concordo totalmente
( ) Concordo parcialmente
( ) Indiferente
( ) Discordo parcialmente
( ) Discordo totalmente
Observação:_____
_____

4. Qual é a sua avaliação do protótipo?
( ) Muito bom
( ) Bom
( ) Regular
( ) Insatisfatório
Observação:_____
_____

5. Qual foi a sua maior dificuldade utilizando o protótipo?
_____
_____

Muito obrigada pela participação!