UNIVERSIDADE DE CAXIAS DO SUL

CENTRO DE CIÊNCIAS EXATAS E DA TECNOLOGIA

CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

RICARDO PASIN ANDRADE

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL COM OCR E RECONHECIMENTO DE VOZ PARA LEITURA DE CONSUMO DE ÁGUA E GÁS EM CONDOMÍNIOS

CAXIAS DO SUL

2016

RICARDO PASIN ANDRADE

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL COM OCR E RECONHECIMENTO DE VOZ PARA LEITURA DE CONSUMO DE ÁGUA E GÁS EM CONDOMÍNIOS

Trabalho de Conclusão de Curso para obtenção do Grau de Bacharel em Sistemas de Informação da Universidade de Caxias do Sul.

Orientador: Prof. Alexandre Erasmo Krohn Nascimento.

CAXIAS DO SUL

2016

Dedico este trabalho aos meus pais, irmão, namorada e amigos que sempre me apoiaram e para esta nova conquista não foi diferente. Dedico também a todos os colegas e docentes que estiveram ao meu lado nessa caminhada e que contribuíram para o meu crescimento e aprendizado.

RESUMO

Este trabalho apresenta como cenário o processo de coletas de medidas de consumo de gás e água na Administradora de Condomínios Solução, localizada em Caxias do Sul. Neste cenário existem algumas dificuldades vistas pela organização, onde o processo atual é feito de forma manual, acarretando em possíveis erros humanos e maior tempo de coleta. Utilizando tecnologias atuais, através de dispositivos móveis, o presente trabalho tem como objetivo o desenvolvimento de um aplicativo para a realização das coletas acima descritas de uma forma mais automatizada e organizada do que a atual, visando redução no número de erros de leitura bem como diminuição do tempo realizando a tarefa em questão. Para isso, foram utilizados recursos de reconhecimento de caracteres e também reconhecimento de voz. A solução proposta, que foi desenvolvida utilizando framework híbrido com tecnologias de desenvolvimento web, é utilizada pelos leituristas da empresa e foi testada através de entrevistas em pesquisa de satisfação com os mesmos.

Palavras-chave: Aplicativo Móvel. OCR. Reconhecimento de voz. Condomínios. Consumo de Gás. Consumo de água.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API	*Application Programming Interface*
ASR	*Automatic Speech Recognition*
CLI	*Command-Line Interface*
CSS	*Cascading Style Sheets*
CSV	*Comma-separated values*
ERP	*Enterprise Resource Planning*
GPU	*Graphics Processing Unit*
HP	*Hewlett-Packard*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IHC	Interface Humano–Computador
JSON	*JavaScript Object Notation*
OCR	*Optical Character Recognition*
PHP	*Hypertext Preprocessor*
REST	*Representational State Transfer*
SGBD	Sistema de Gerenciamento de Banco de Dados
SOA	*Service-Oriented Architecture*
SQL	*Structured Query Language*
STT	*Speech to Text*
UML	*Unified Modeling Language*
W3C	*World Wide Web Consortium*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO DO ESTUDO

Segundo o DMAE (Departamento Municipal de Água e Esgotos) de Porto Alegre, no ano de 2015 foi registrada uma média de cerca de 2,5 mil reclamações mensais referentes a erros de leitura (ZERO HORA, 2015). Essa estatística levantada pelo DMAE, por ter características muito semelhantes, estende-se às cobranças de consumo de gás, que acabam sofrendo os mesmos problemas citados.

Analisando esse cenário, pode-se evidenciar que as questões das leituras de consumo merecem uma atenção especial, pois altos valores estão envolvidos e a redução da incidência de erros nas mesmas é algo crítico e possível de ser realizado através de tecnologias hoje disponíveis para o desenvolvimento de ferramentas.

Atualmente, no Brasil, cálculos, medições e coletas de consumo de água e gás das residências brasileiras são realizados de maneira desatualizada em relação às inovações tecnológicas já existentes, sendo assim é possível a criação de novos sistemas que visam aperfeiçoar essas ações e garantir a confiabilidade dos dados (ROSÁRIO, 2005).

Essa situação ocorre em administradoras de Condomínios e distribuidoras de energia, gás e água, ou seja, em todas as organizações que repassam os custos dos consumos citados aos consumidores finais.

Algumas das tecnologias disponíveis que podem ajudar na redução dos erros humanos na leitura dos hidrômetros de água e relógios de gás, são as seguintes: OCR (*Optical Character Recognition*) e reconhecimento de voz. OCR nada mais é que o reconhecimento automático de caracteres por parte de uma máquina. Já o reconhecimento de voz, é a capacidade da máquina em interpretar a fala e convertê-la em texto. Além disso, podem ser utilizadas as mais diversas técnicas IHC (Interface Humano-Computador) existentes que visam tornar a experiência do usuário o mais agradável possível.

1.2 PROBLEMA DE PESQUISA

A Solução Condomínios, empresa onde o autor deste trabalho é colaborador, atua do ramo de Administração de Condomínios, na cidade de Caxias do Sul – RS, nos casos que não há individualização dos hidrômetros de água e relógios de gás, possui o compromisso de comparecer aos Condomínios todos os meses para realizar a leitura de consumo dos mesmos. Após estudo realizado na empresa, constatou-se que as reclamações referentes a essas coletas têm um efeito muito negativo, pois afetam a confiança do cliente. Por esse motivo a empresa entende que deve tomar atitudes visando reduzir a ocorrência dessa situação.

O problema das coletas de dados e comparecimento mensal aos Condomínios pode se aplicar também às empresas do ramo de distribuição de gás, essas responsáveis pelas coletas somente quando os relógios de gás são individualizados, pois neste caso, as mesmas que são responsabilizadas pelas coletas. Como o projeto, nesse trabalho descrito, é para os fins de utilização em administradoras de Condomínios, o mesmo contextualizará o caso da empresa Solução Condomínios.

Atualmente na Solução, as coletas são anotadas manualmente, sendo utilizada uma planilha de papel com as unidades descritas na mesma e uma caneta para efetuar as anotações. No processo descrito, além do erro humano na cópia dos dígitos, podem ocorrer dificuldades no entendimento da caligrafia do leiturista, há também um retrabalho na cópia dos dados da planilha para o sistema de gestão, contando ainda com possibilidade de erro na cópia dos mesmos.

A organização realizou uma pesquisa de mercado com os equipamentos e *softwares* existentes no segmento de coletas de dados de consumo e constatou que não havia algo que a satisfizesse. A empresa levantou que, apesar de existir certa evolução tecnológica, as soluções disponíveis esbarravam em problemas de usabilidade, pois a mesma considera coletores de dados de difícil operação e com sistemas operacionais não livres para esse tipo de equipamento. Além do que foi levantado durante a pesquisa, mais um fator veio a corroborar com o não agrado da empresa pelas soluções existentes, contatou-se que seguiria sendo uma leitura visual, onde o leiturista digitaria os valores dos medidores no sistema.

1.3 OBJETIVO DO TRABALHO

O objetivo geral do trabalho é desenvolver um aplicativo intuitivo através de técnicas de IHC, principalmente o de usabilidade, para o gerenciamento de coletas de leituras de consumos de água e gás na Solução Condomínios. Posteriormente, avaliar se houve redução da incidência de erros humanos através da utilização de tecnologias automatizadas de OCR e reconhecimento de voz para realizar esse processo.

Através das técnicas e tecnologias acima descritas, o usuário do aplicativo poderá realizar checagens e comparar dados históricos, tudo isso visando à diminuição da incidência dos erros que hoje ocorrem nas leituras de gás e água.

1.4 ESTRUTURA DO TEXTO

O trabalho está dividido em seis capítulos, sendo o primeiro a introdução, o qual apresenta uma base de onde o problema está inserido, bem como o objetivo a ser atingido. No segundo está à conceituação dos temas abordados neste trabalho. Já o terceiro expõe as formas possíveis de se desenvolver um aplicativo para dispositivos móveis. O quarto mostra o projeto com a proposta de solução e toda a engenharia do software a ser desenvolvido. O quinto capítulo descreve um roteiro de como foi à implantação do *software* na empresa. No sexto capítulo são apresentadas as considerações finais do trabalho desenvolvido. No sétimo e último capítulo são apresentadas as referências bibliográficas utilizadas para desenvolver este trabalho.

O capítulo subseqüente deste trabalho é o referencial teórico e segue descrito adiante.

2 REFERENCIAL TEÓRICO

Neste capítulo, são apresentadas as tecnologias que estão presentes no trabalho com o objetivo de tornar o aplicativo integrado a sistemas heterogêneos e o mais automatizado possível. A plataforma em que o aplicativo será executado é o sistema operacional Android, as principais tecnologias que irão compor o *software* são o OCR, reconhecimento de voz e *Web Services*. As tecnologias apresentadas seguem descritas nas seções seguintes, sendo iniciadas as apresentações com o OCR.

2.1 OCR (*OPTICAL CHARACTER RECOGNITION*)

Para EIKVIL (1993, p. 11), o princípio básico para o reconhecimento automático de caracteres é ensinar a máquina padrões que podem ocorrer e como eles se assemelham. No caso do OCR, os padrões são letras, números e alguns símbolos especiais como vírgulas, pontos de interrogação, entre outros. O ensino da máquina é feito através de amostragem de exemplos de textos com as devidas correspondências.

Com base nesses exemplos a máquina constrói um protótipo ou uma descrição de cada classe de caracteres. Então, durante o reconhecimento, os caracteres desconhecidos são comparados com os anteriormente aprendidos, e então se atribuiu a melhor correspondência (EIKVIL, 1993).

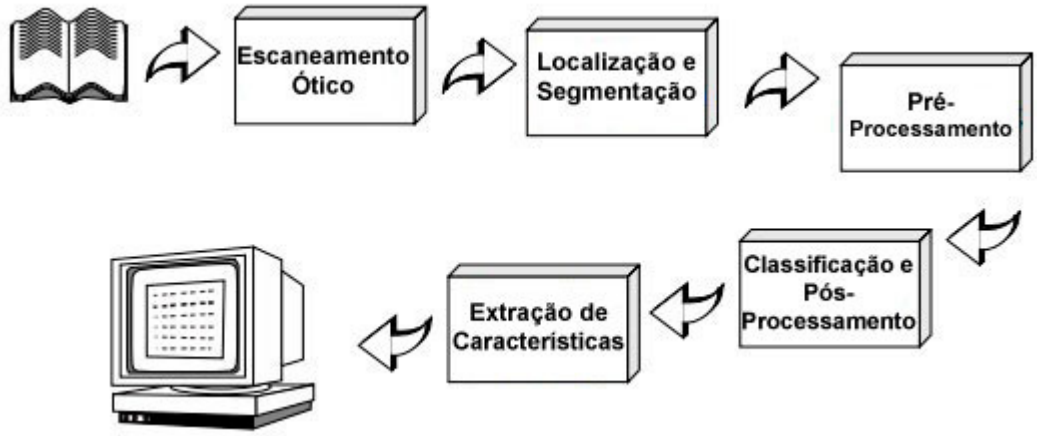
2.1.1 Componentes de um sistema de OCR

Segundo Eikvil (1993, p. 11), um sistema de OCR é composto pelos seguintes componentes: escaneamento ótico, localização e segmentação, pré-processamento, extração de características, classificação e pós-processamento.

O processo pode ser resumido da seguinte forma: um escâner ótico digitaliza um determinado documento, após esse procedimento, ocorre à busca por partes do documento que contém textos, sendo assim extraídos os símbolos. Cada um dos símbolos extraídos passa por um processamento que objetiva eliminar todo o ruído possível visando facilitar a visualização das características do símbolo. Neste momento o aprendizado de máquina citado anteriormente acontece, onde o símbolo

obtido é comparado às amostragens e a melhor correspondência é resultante. Finalmente, o símbolo encontrado é convertido no seu texto correspondente. Esse processo segue ilustrado na Figura 1.

Figura 1 – Componentes de um Sistema OCR.



Fonte: EIKVIL, 1993.

2.1.1.1 Escaneamento Ótico

Escaneamento ótico nada mais é do que a transformação de um documento em uma imagem digitalizada, onde um escâner ótico irá realizar esse processo. Um escâner ótico funciona baseado num mecanismo de transporte, acrescido de um dispositivo que converte a intensidade da luz em tons de cinza. Normalmente documentos impressos estão em preto e branco, quando estes estão em múltiplos níveis, ou seja, coloridos, o escâner executa um processo conhecido como limiar ou *thresholding* que realiza essa conversão para escala de cinza, tendo como objetivo diminuir o esforço computacional e a redução na utilização da memória (EIKVIL, 1993).

O processo de limiarização ou *thresholding*, apesar de muito simples é muito importante e determinante no sucesso do processo de reconhecimento. É definido um valor limiar fixo, onde tonalidades de cinza abaixo deste valor são convertidas em preto e níveis acima são convertidas em branco. Em documentos cujos contrastes entre preto e branco são grandes e bem definidos, um valor fixo de limiar é suficiente para resolver a questão da limiarização. Porém, poucos documentos possuem essa característica e para estes casos o valor de limiar é alterado

dinamicamente tornando o processo mais sofisticado. Entretanto, nesse caso a exigência de memória e capacidade computacional aumenta consideravelmente tornando essa prática de limiarização com a variável dinâmica pouco utilizada em OCR (EIKVIL, 1993). As diferenças de escaneamento podem ser percebidas quando há valores fixos de limiar e variáveis, essa situação é ilustrada nas Figuras 2, 3 e 4.

A Figura 2 representa uma imagem escaneada com certos ruídos de iluminação.

Figura 2 – Imagem escaneada com ruídos de iluminação.

Fonte: Página do OpenCV (http://docs.opencv.org).

A Figura 3 expõe a mesma digitalização da Figura 2, porém com um determinado valor fixo de limiar.

Figura 3 – Imagem escaneada com valor fixo de limiar.

Fonte: Página do OpenCV (http://docs.opencv.org).

A Figura 4 mostra essa digitalização com um valor variável de limiar calculado, sendo muito mais nítida e clara que a Figura 3.

Figura 4 – Imagem escaneada com valor variável de limiar.

Fonte: Página do OpenCV (http://docs.opencv.org).

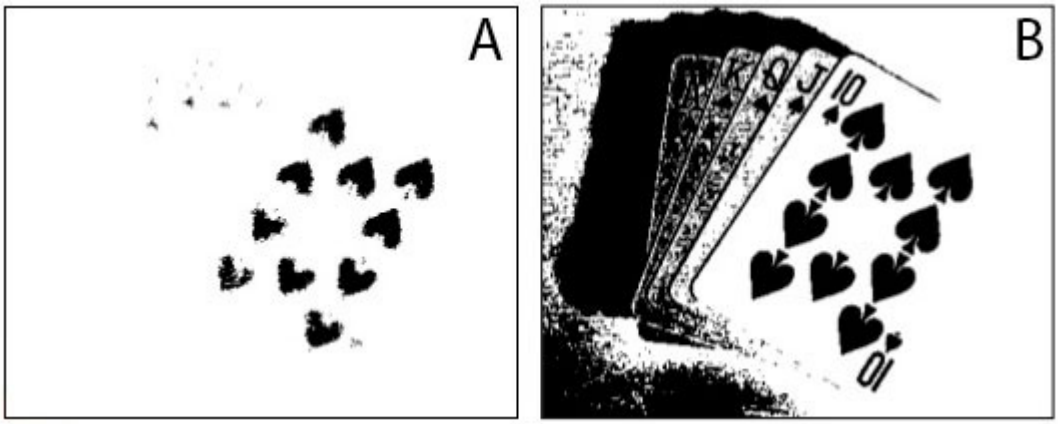
2.1.1.2 Localização e Segmentação

A segmentação de uma imagem é, basicamente, dividir uma imagem nas áreas significativas, ou seja, nas partes relevantes da mesma. É uma tarefa simples de ser descrita, porém é das mais difíceis de ser implementada (KHOSHAFIAN & BAKER, 1996).

A maior dificuldade de um algoritmo de segmentação é encontrar um meio de medir a diferença entre os pixels que pertencem ao texto e os pixels do fundo (CHEN, LUETTIN, SHEARER, 2000).

O processo procura diferir textos de números e gráficos. Quando a segmentação é aplicada ao texto, ela isola caracteres ou palavras, o grande problema que ocorre no processo é que muitas vezes acontece uma confusão por conta de textos e gráficos com caracteres muito unidos. Se o documento foi escaneado com o limiar muito baixo, perda de caracteres e espaços pode ocorrer, se o limitar for demasiadamente alto, o sistema de OCR também pode ficar confuso durante a segmentação e ignorar espaços, além de unir caracteres aos gráficos (MITHE, SUPRIYA, DIVEKAR, 2013). O problema da segmentação de caracteres, gráficos e espaços relacionados ao valor do limiar pode ser visto na Figura 5.

Figura 5 – Imagem escaneada com baixo valor limiar (A) e alto valor limiar (B).



Fonte: KULATHILAKE, H. Compute Graphics & Image Processing - Segmentation Techniques, 2013.

2.1.1.3 Pré-Processamento

Conforme Leondes (1998), a imagem resultante do processo de análise contém sempre certa quantidade de ruído e o processo de pré-processamento é necessário para reduzir o seu efeito. Ruído pode ser descrito como qualquer coisa que impede um sistema de reconhecimento de padrões cumprirem o seu papel, não importando como este "ruído" afeta os dados. Algumas propriedades relevantes dos dados podem também ser reforçadas com o pré-processamento antes que os dados continuem a ser alimentados no sistema de reconhecimento.

O pré-processamento é nada mais que um método de filtragem simples sobre os dados. No reconhecimento de caracteres, a imagem escaneada pode ser filtrada para remover os ruídos que podem dificultar o processo de segmentação.

Segundo Khoshafian & Baker (1996), a imagem original terá uma qualidade inferior à resultante desse processo realizado na imagem digitalizada.

2.1.1.4 Extração de Características

Para Eikvil (1993), o objetivo principal da extração de características é encontrar as propriedades essenciais dos símbolos. Se aceita que este é um dos problemas mais difíceis para os padrões de reconhecimento. A rasterização, que é a tarefa de converter uma imagem vetorial em uma imagem raster, com pixels ou pontos para a saída de vídeo ou impressora, é a forma mais direta de descrever um

caractere. Outra forma de realizar essa descrição é através da extração de certas características que possibilitem a identificação do símbolo e a exclusão de outras menos relevantes. As técnicas para extração de características são divididas em três grupos, sendo elas: distribuição dos pixels, transformações e expansões em série e análise estrutural.

Na distribuição dos pixels, as características são buscadas na distribuição estatísticas dos pontos, onde há tolerância a eventuais distorções e variações de estilo.

As transformações e expansões em séries têm como principal foco a redução da influência das rotações e translações que podem haver. As principais transformações utilizadas são as de Fourier, Walsh, Haar, Hadamard, Karhunen-Loeve, Hough, entre outras. Em sua maioria, se baseiam nas curvas que fazem o contorno dos caracteres, fazendo com que sejam muito sensíveis aos ruídos nas mesmas.

A análise estrutural se detém a descrição da forma geométrica e topológica dos caracteres, de posse dessas informações, é possível identificar pontos de término, intersecções de linha, ranhuras, entre outras características. Em contraponto às outras técnicas anteriormente apresentadas, essas fornecem características tolerantes a ruídos e variações de estilo. No entanto, quando há translações e rotações não são tão efetivas quanto às demais.

Os grupos de características podem ser avaliados conforme sua sensibilidade ao ruído e a deformação, assim como facilidades na sua implementação e utilização (EIKVIL, 1993). As sensibilidades aos ruídos das características são apresentadas na Tabela 1. Os critérios utilizados nesta avaliação são os seguintes:

Tabela 1 – Sensibilidade de ruídos nas características.

Robustez	1) Ruídos: Sensibilidade aos segmentos desconectados de linha, colisões, lacunas, etc; 2) Distorções: Sensibilidade a variações locais como cantos arredondados, saliências impróprias, dilatações e encolhimento; 3) Variação de estilo: A sensibilidade à variação no estilo como a utilização de diferentes formas de representar o mesmo símbolo ou o uso de serifas, inclinações, etc; 4) Translação: A sensibilidade ao movimento de todo o caractere ou seus componentes; 5) Rotação: A sensibilidade para a mudança em orientação dos caracteres.
Uso Prático	1) Velocidade de reconhecimento; 2) Complexidade da implementação; 3) Independência.

Fonte: EIKVIL, 1993.

Cada uma das técnicas é avaliada e comparada na Tabela 2:

Tabela 2 - Avaliação das técnicas de extração de funcionalidade.

Técnica de extração de características	**Robustez**					**Uso Prático**		
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**
Correspondências de modelos	◉	◉	○	○	○	○	●	○
Transformações	○	●	●	●	●	○	○	◉
Distribuição de pontos: zoneamento	○	◉	○	○	◉	●	●	○
momentos	◉	◉	○	●	●	○	◉	○
n-lista	◉	○	◉	○	◉	●	●	◉
características	○	●	●	●	◉	●	●	○
cruzamentos	○	●	●	●	◉	●	●	○
Características de estrutura	○	●	●	●	◉	●	○	●

Avaliação quanto à sensibilidade ao ruído, deformações e facilidade de aplicação.

● alta ou fácil ◉ média ○ baixa ou difícil

Fonte: EIKVIL, 1993.

2.1.1.5 Classificação

Para Eikvil (1993), a classificação é o processo de identificação de cada caractere onde é atribuída a ele a classe de caracteres correta. A seguir serão mencionados dois métodos diferentes para a classificação no reconhecimento de caracteres. Primeiramente o reconhecimento com teoria da decisão é tratado. Esse método é utilizado quando a descrição dos caracteres pode ser representada numericamente em um vetor de características. Também podem ter características padrões derivadas da estrutura física do caractere, que não são tão facilmente quantificadas. Nestes casos, a relação entre as características podem ser importantes ao decidir sobre a escolha da classe. Como exemplos pode-se citar uma linha vertical e uma horizontal que se encontram, podendo ser tanto um "L" quanto um "T", e a relação entre os dois traços é indispenável para distinguir os caracteres. Nesse caso então, é necessária uma abordagem estrutural.

a) Métodos de teoria da decisão

Segundo Eikvil (1993), as principais abordagens para o reconhecimento utilizando teoria da decisão são: distância, classificadores mínimos, classificadores estatísticos e redes neurais. Cada uma dessas técnicas de classificação são brevemente descritas na seqüência:

- Classificação por correspondência

A correspondência utiliza técnicas com base em medidas de similaridade em que a distância entre o caractere e a sua classe correspondente é calculada. Podem ser aplicadas medidas diferentes, mas o comum é a distância métrica. Este classificador de distância mínima funciona bem quando as classes estão bem separadas, ou seja, quando a distância entre os caracteres é grande em comparação com a expansão das classes.

- Classificação por estatística

Na classificação estatística é aplicada uma abordagem probabilística para reconhecimento. A idéia é a utilização de um esquema de classificação que é ótimo,

no sentido de que, em média, a sua utilização dá a menor probabilidade erros de classificação.

O classificador que minimiza a perda média total é chamado de classificador de Bayes. Dado um símbolo desconhecido descrito pelo seu vetor de características, a probabilidade de que o símbolo pertence à classe “c” é calculado para todas as classes c=1...N. O símbolo é então atribuído a classe que dá a melhor probabilidade.

Para que esse esquema seja ótimo, o conhecimento prévio dos símbolos de cada classe deve ser conhecido, juntamente com as funções de densidade e com a probabilidade de ocorrência de cada classe. A função de densidade é assumida geralmente para ser distribuída, e quanto mais próxima está da realidade, mais o classificador de Bayes tratará de forma ideal.

- Classificação por redes neurais

Recentemente, o uso de redes neurais para reconhecer caracteres (e outros tipos de padrões) ressurgiu. Considerando-se uma rede de repropagação, a mesma é composta de várias camadas de elementos interligados. Um vetor é inserido na camada de entrada, cada elemento da camada calcula uma soma ponderada das suas entradas e a transforma em uma saída por uma função não linear. Durante o processo, os pesos em cada ligação são ajustados até se obter um resultado desejado. Uma desvantagem de redes neurais em OCR pode ser a sua previsibilidade limitada e generalidade, enquanto que uma vantagem é a sua natureza adaptativa.

b) Métodos estruturais

Eikvil (1993) define que dentro da área de reconhecimento estrutural, métodos sintáticos estão entre as abordagens mais utilizadas. Existem outras técnicas, mas eles são menos gerais e não serão expostas neste trabalho.

- Classificação por métodos sintéticos

Medidas de similaridade com base nas relações entre os componentes estruturais podem ser formuladas usando conceitos gramaticais. A idéia é que cada classe tenha a sua própria gramática que defina a composição da gramática do caractere. Esta pode ser representada como árvores, e os componentes estruturais

extraídos de um caractere desconhecido são comparados às gramáticas de cada classe. Suponha que temos duas classes de caracteres diferentes que podem ser geradas pelo G^1 e G^2, respectivamente. Dado um caractere desconhecido, dizemos que ele é mais parecido com a primeira classe, portanto ele deve ser gerado pelo G^1 e não por G^2.

2.1.1.6 Pós-Processamento

a) Agrupamento

O resultante de um reconhecimento, nada mais é do que símbolos individuais convertidos em caracteres. Entretanto, esses símbolos carecem de informações suficientes, é necessário associar aos mesmos símbolos que pertencem à mesma cadeia visando torná-los palavras e números. Essa associação de símbolos é, de fato, o processo de agrupamento. O agrupamento se torna muito mais simples e fácil quando o texto apresenta poucas variações de tamanhos de fonte, além de padronização de distâncias entre palavras e caracteres. Quando esses fatores citados não estão bem ajustados, os mesmos podem aumentar muito a complexidade do agrupamento. Outro problema que podemos ter no processo são os caracteres manuscritos e textos inclinados.

b) Detecção e correção de erros

Mesmo os melhores sistemas de reconhecimento não irão obter 100% de exatidão na identificação de todos os caracteres, mas alguns desses erros podem ser detectados ou mesmo corrigidos pelo uso de técnicas (existem duas ténicas principais). A primeira utiliza a possibilidade de seqüência de caracteres que aparecem em conjunto. Isto pode ser feito através da utilização de regras que definem a sintaxe das palavra, dizendo, por exemplo, que após uma lacuna não devem haver uma letra maiúscula. Além disso, para diferentes línguas as probabilidades de dois ou mais caracteres que aparecem numa determinada seqüência pode ser calculado e pode ser utilizado para detectar erros. A segunda abordagem é a utilização de dicionários, esta tem sido a mais eficiente para a detecção e correção de erros. Dada uma palavra, no qual um erro pode estar

presente, a palavra é procurada no dicionário do idioma em questão. Se a palavra não está no dicionário, um erro tem foi detectado, e pode ser corrigida alterando a palavra para a mais semelhante. A palavra pode estar presente no dicionário, mas isso não prova que não houve um erro. Porém, irá sempre direcionar para palavras existentes. A grande desvantagem sobre o método que utiliza o dicionário é a demora que as buscas e comparações implícitas levam.

2.1.2 Bibliotecas OCR

Serão apresentados a seguir alguns dos mais conceituados *softwares* de reconhecimento de caracteres do mercado. Posteriormente à apresentação, os mesmos serão tabulados para comparação.

As ferramentas Tesseract, GNU Ocrad, GOCR e OCROpus são *open-source*[1], ou seja, de licença livre, já os softwares TOCR e Abbyy CLI OCR possuem licença comercial.

Tesseract[2] é uma ferramenta de reconhecimento de caracteres *open-source* desenvolvida pela empresa HP (*Hewlett Packard*). O desenvolvimento foi iniciado em 1985 e seguiu até 1995. Em 2005 a licença Apache foi liberada, tendo o desenvolvimento patrocinado pela Google. Essa poderosa ferramenta pode reconhecer texto de até 60 diferentes idiomas. Porém, necessita que cada um seja baixado e adicionado manualmente. Caso não seja definido um idioma específico, o idioma padrão, que é o Inglês, será utilizado. Tesseract suporta também aprendizagem de máquina que pode ser utilizado quando a fonte ou a língua a ser digitalizada não existe.

Ocrad[3] é uma biblioteca open-source distribuida pela GNU GLP que vem sendo desenvolvida desde 2003 na linguagem de programação C++. É possível se definir qual é o conjunto de caracteres que deseja-se reconhecer, o que ajuda a limitar o que é de interesse.

---

[1] Open Source significa que usuários poderão utilizar, modificar e redistribuir livremente o produto (CAMPOS, 2009).

[2] Tesseract - Disponível em: http://code.google.com/p/tesseract-ocr/

[3] GNU Ocrad – descrição em: http://www.gnu.org/software/ocrad/

GOCR[4] (ou JOCR) é uma biblioteca *open-source* que surgiu no final da década de 90. É distribuída pela GNU GPL. As imagens por vezes podem precisar serem convertidas para serem utilizadas com GOCR, uma vez que apenas essa bilbioteca pode ler poucos formatos de imagem.

Cuneiform[5] é uma ferramenta OCR escrita na linguagem de programação C e C++, é disponibilizada sob a licença Apache, *open-source* a partir de abril de 2008. Foi desenvolvida na Rússia numa parceria da Corel Corporation. Tem apenas dois idiomas disponíveis, que são o russo e o inglês.

Abbyy CLI OCR é uma ferramenta de licença comercial OCR que é baseada no ABBYY FineReader Engine. Ele suporta não só textos horizontais, como verticais, mas isso é especificado pelo usuário durante a execução do programa.

Pode ler 190 idiomas diferentes, e oferece suporte de dicionário para alguns deles. Abbyy[6] oferece suporte de dicionário para sueco e inglês.

2.1.3 Comparação das Bibliotecas OCR

Para uma melhor compreensão das divergências entre as ferramentas, a Tabela 3 expõe as características das mesmas:

Tabela 3 – Comparação das características de ferramentas OCR.

Ferramenta	Fundação	Última Versão	Licença	Linguagem	Idiomas
Tesseract	1985	3.04 (2015)	Apache	C++, C	100+
GNU Ocrad	2003	0.22 (2013)	GPL	C++	Alfabeto Latino
GOCR	2000	0.50 (2013)	GPL	C	Inglês
Cuneiform	1996	1.1 (2011)	BSD	C, C++	23
Abbyy Cli OCR	1989	12 (2014)	Proprietária	C, C++	198

Fonte: O autor, 2016.

[4] GOCR - descrição em: http://jocr.sourceforge.net/

[5] Cuneiform - descrição em: http://cognitiveforms.com/products_and_services/cuneiform/

[6] Abbyy CLI OCR - descrição em: http://www.ocr4linux.com/

Após a apresentação das ferramentas, tornou-se necessário realizar uma pesquisa que exponha o desempenho dessas ferramentas. Após uma extensa busca em artigos, jornais e trabalhos acadêmicos, foi localizada uma pesquisa que atende aos requisitos esperados para a ilustração das mesmas. A pesquisa foi realizada por Andreas Gohr, em 2010. Gohr fez comparações entre as ferramentas apresentadas anteriormente.

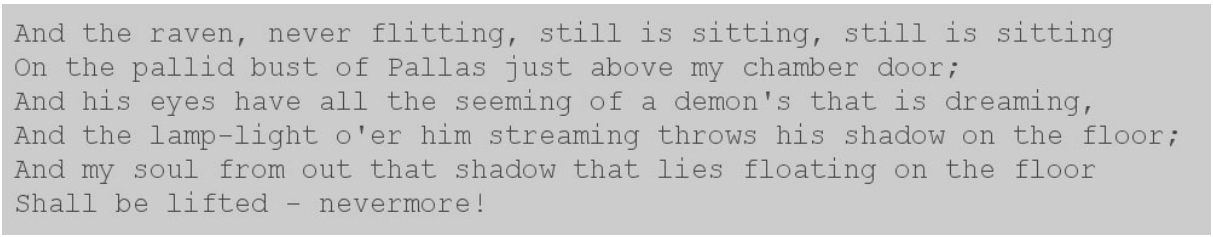
O que motivou Gohr para realizar a pesquisa foi a sua curiosidade quanto às taxas de reconhecimentos das ferramentas em imagens simples e bastante nítidas. O texto escolhido foi convertido em imagens com diferentes fontes, inclusive uma delas lembrando manuscrito. Além das diferentes fontes, foram criadas também versões de fundo em escala de cinza com um menor contraste entre texto e fundo. Na Figura 6 é apresentado o texto original, após, na Figura 7, é exibido um dos exemplos de conversão do texto em imagem com diferentes fontes.

Figura 6 – Texto original.

```
And the raven, never flitting, still is sitting, still is sitting
On the pallid bust of Pallas just above my chamber door;
And his eyes have all the seeming of a demon's that is dreaming,
And the lamp-light o'er him streaming throws his shadow on the floor;
And my soul from out that shadow that lies floating on the floor
Shall be lifted - nevermore!
```

Fonte: Gohr, 2010.

Figura 7 – Fonte Courier (escala de cinza).



Fonte: Gohr, 2010.

Os testes foram realizados utilizando dois critérios: percentual de caracteres corretos e o tempo de execução que cada ferramenta levou para concluir a digitalização. A Tabela 4 lista o percentual de caracteres reconhecidos em cada uma das ferramentas avaliadas.

Tabela 4 – Comparação das ferramentas quanto à taxa de reconhecimento e tempo gasto.

	abbyyocr	cuneiform	gocr	ocrad	tesseract
Courier (preto)	100% (2.92s)	61% (1.11s)	67% (0.09s)	21% (0.02s)	81% (0.63s)
Courier (cinza)	100% (2.85s)		67% (0.09s)	21% (0.03s)	81% (0.63s)
Justy (preto)	11% (3.62s)	3% (1.14s)	31% (0.11s)	1% (0.02s)	15% (0.61s)
Justy (cinza)	14% (3.45s)		31% (0.10s)	1% (0.02s)	15% (0.60s)
Times (preto)	100% (2.80s)	96% (1.07s)	76% (0.16s)	82% (0.03s)	92% (0.74s)
Times (cinza)	100% (2.87s)		76% (0.16s)	82% (0.03s)	92% (0.74s)
Verdana (preto)	100% (2.90s)	95% (1.07s)	98% (0.10s)	98% (0.03s)	98% (0.45s)
Verdana (cinza)	100% (2.85s)		98% (0.10s)	98% (0.02s)	98% (0.46s)

Fonte: Gohr, 2010.

Para Gohr (2010), o *software* Abbyy de licença comercial, se mostrou muito eficiente com fontes impressas. Porém, muito falho na caligrafia de manuscritos. Ele é o mais lento das demais ferramentas testadas, mas tem a vantagem de ler quase qualquer formato de imagem existente. Enquanto, para outras provavelmente será necessário converter o formato da imagem.

Caso o software a ser escolhido seja um *open-source*, o Tesseract se mostra a melhor alternativa. Tesseract foi um produto comercial desenvolvido no início dos anos 90 e mais tarde foi comprado pela Google e disponibilizado de forma livre. O formato de entrada é bastante inflexível. Porém, quando a entrada é correta ele se mostra muito eficiente (GOHR, 2010).

O reconhecimento de escrita funcionou melhor no GOCR. Porém, nas outras imagens de fonte impressa os resultados foram realmente muito ruins (GOHR, 2010).

Na seção 2.2 é apresentada mais uma tecnologia envolvida neste projeto, o reconhecimento de voz.

2.2 RECONHECIMENTO DE VOZ

Segundo Zue (1997), o desenvolvimento de sistemas controlados por voz tem como objetivo suplementar, em alguns casos, as interfaces mais comuns como o uso de teclados para entrada de dados, por exemplo. Dentro desse contexto, o reconhecimento de voz pode se adequar, gerando uma interface muito natural homem-máquina.

Nas últimas décadas as pesquisas em reconhecimento automático de fala ASR (*Automatic Speech Recognition*) têm tido avanços consideráveis. Já existem muitas aplicações desenvolvidas com essa tecnologia, incluindo desde vocabulários reduzidos até compreensão espontânea da fala (ZUE, 1997).

2.2.1 O que é o reconhecimento de voz

O reconhecimento automático de voz nada mais é que o processo no qual um computador mapeia sinais acústicos da fala e converte-os em texto. Após essa conversão, o texto convertido pode ser exibido na tela, ser impresso, enfim, ter uma saída visível ao usuário (CARNEGIE MELLON UNIVERSITY, 1996).

Um sistema de reconhecimento automático de voz pode ser classificado quanto ao seu nível de dependência do usuário locutor, quanto aos vocabulários interpretados e a permissão de fala contínua ou não no processo do reconhecimento (CARNEGIE MELLON UNIVERSITY, 1996).

Um sistema dependente do usuário é desenvolvido para operar normalmente com um único usuário falante que é quem irá treinar o sistema STT (*speech to text*). Estes sistemas são mais simples de serem desenvolvidos, menos custosos de serem adquiridos e mais precisos, porém pouco flexíveis e adaptáveis a diferentes usuários falantes (CARNEGIE MELLON UNIVERSITY, 1996).

Um sistema independente de usuário locutor é desenvolvido para operar em qualquer falante de um idioma, como por exemplo, Inglês-EUA, Português-BR, etc. Estes sistemas são mais complexos de serem desenvolvidos, mais caros de serem adquiridos e sua precisão é menor do que os sistemas dependentes do usuário.

Entretanto, eles são mais flexíveis (CARNEGIE MELLON UNIVERSITY, 1996).

Um sistema adaptativo ao usuário é desenvolvido para adaptar o seu funcionamento às características dos novos usuários. Tanto a sua dificuldade de desenvolvimento quanto custos e precisão são consideradas médias (CARNEGIE MELLON UNIVERSITY, 1996).

O tamanho de vocabulário de um sistema de reconhecimento de fala afeta diretamente a complexidade do mesmo, principalmente nos fatores de requisitos de processamento e níveis de exatidão. Aplicações simples por vezes requerem apenas caracteres numéricos, ou seja, um vocabulário pequeno. Enquanto aplicações complexas utilizam grandes vocabulários (JUANG & RABINER, 2004).

- Vocabulários pequenos - dezenas de palavras;
- Vocabulários médios - centenas de palavras;
- Vocabulários grandes - milhares de palavras;
- Vocabulários muito grandes - dezenas de milhares de palavras.

2.2.2 Assistentes de reconhecimento de voz

Na seqüência, serão apresentadas as três principais assistentes de reconhecimento de voz em dispositivos móveis. Após a apresentação das mesmas, serão expostas duas pesquisas realizadas pelas páginas da web Experts Exchange e Phone Arena.

As assistentes escolhidas são as seguintes: Apple Siri, Google Now e Microsoft Cortana.

2.2.2.1 Apple Siri

Apple Siri[7] é uma assistente de voz que pode ser resumido em três camadas: processamento de voz, análise de contexto e aprendizado. A assistente se comunica ao centro de dados da Apple, dando a resposta ao usuário. Mas a Siri vai muito além de disso, ele inclui compreensão da linguagem, conhecimento de modelagem, aplicação da lógica e aprendizagem de máquina. Não há formas pré-

[7] Apple Siri – Disponível em: http://www.venturewerks.com/Siri-A-Primer.pdf

definidas de solicitar a Siri uma tarefa ou fazer uma pergunta, o mesmo não só entende as palavras faladas, mas também entende contextos. A compreensão de contextos exige decifração de linguagem natural e, após esse processo, é feito o acesso aos recursos disponíveis para realizar tarefas ou responder a questões feitas pelos usuários.

2.2.2.2 Google Now

Google *Now*[8] é um esforço da organização Google em fazer algo produtivo com os dados fornecidos ao longo dos anos pelos usuários do Gmail, Google Buscador, Android e Youtube. É a assistente pessoal de voz padrão de fábrica do Android, sua forma de operar é tentar se antecipar ao usuário e "adivinhar" o que o mesmo quer, enviando sugestões com base em seus históricos de pesquisa, no que foi assistido e o que passa pela caixa de entrada.

Essa assistente tem o poder do reconhecimento de voz com aprendizagem de máquina, além disso, compreende contextos, exibindo os resultados como retorno de pesquisa, Google Maps com rotas traçadas, criando alarmes, entre outros.

2.2.2.3 Microsoft Cortana

A Microsoft Cortana[9] surgiu como uma necessidade da Microsoft para o Windows Phone, visto que iOS tinha Siri e Android Google Now. Mais tarde a empresa lançou versões para Android e iOS da assistente, esse fato de ser multiplataformas é o maior trunfo da Cortana.

É uma assistente que permite a integração de dispositivos móveis com a caixa de entrada do Gmail ou Google Calendar que pode ser aberto em qualquer computador convencional.

---

[8] Google Now – Disponível em: http://www.makeuseof.com/tag/7-siri-alternatives-android-google-now-cortana/

[9] Microsoft Cortana – Disponível em: http://www.makeuseof.com/tag/7-siri-alternatives-android-google-now-cortana/

2.2.3 Comparação dos Assistentes de Voz

Wilfred (2015) publicou sua interpretação sobre a pesquisa realizada pelos especialistas do site Expert Exchanges. Ele explica que a pesquisa reuniu usuários dos sistemas operacionais Apple iOS, Windows Phone e Android, onde os mesmos fariam quatro perguntas e três instruções a seus assistentes de voz.

As perguntas que foram feitas as assistentes são: Primeira – Quem ganhou o *Super Bowl* em 1978? Segunda – Quantas calorias têm um *muffin*? Terceira – Onde posso comprar pneus para meu carro? Quarta, e última – Quando será lançado o “Kung Fu Panda 3”?

Já as instruções são as seguintes: Primeira – Pergunta sobre compromisso marcado na agenda. Segunda – Ativar Wifi do dispositivo. Terceira, e última – Enviar mensagem de texto.

A coleta de dados foi realizada com base no desempenho da interpretação de fala das assistentes sobre as perguntas e instruções citadas anteriormente. Foram levadas em conta as vezes que foram necessárias repetições das instruções ou perguntas, e, a assistente com melhor resultado foi a Apple Siri, seguido por Google Now e Microsoft Cortana.

Foi realizada em paralelo a pesquisa de desempenho, através de uma pesquisa de satisfação dos clientes de cada uma das assistentes, os resultados foram os seguintes: 81% dos usuários da Apple Siri se dizem satisfeitos com a ferramenta, já para os usuários da *Google Now* a satisfação é de 68% e da Microsoft Cortana é de apenas 57%.

Na Figura 8 podemos ver um resumo gráfico dos dados gerados na pesquisa acima descrita:

Figura 8 – Visão geral dos resultados em todas as categorias pesquisadas.

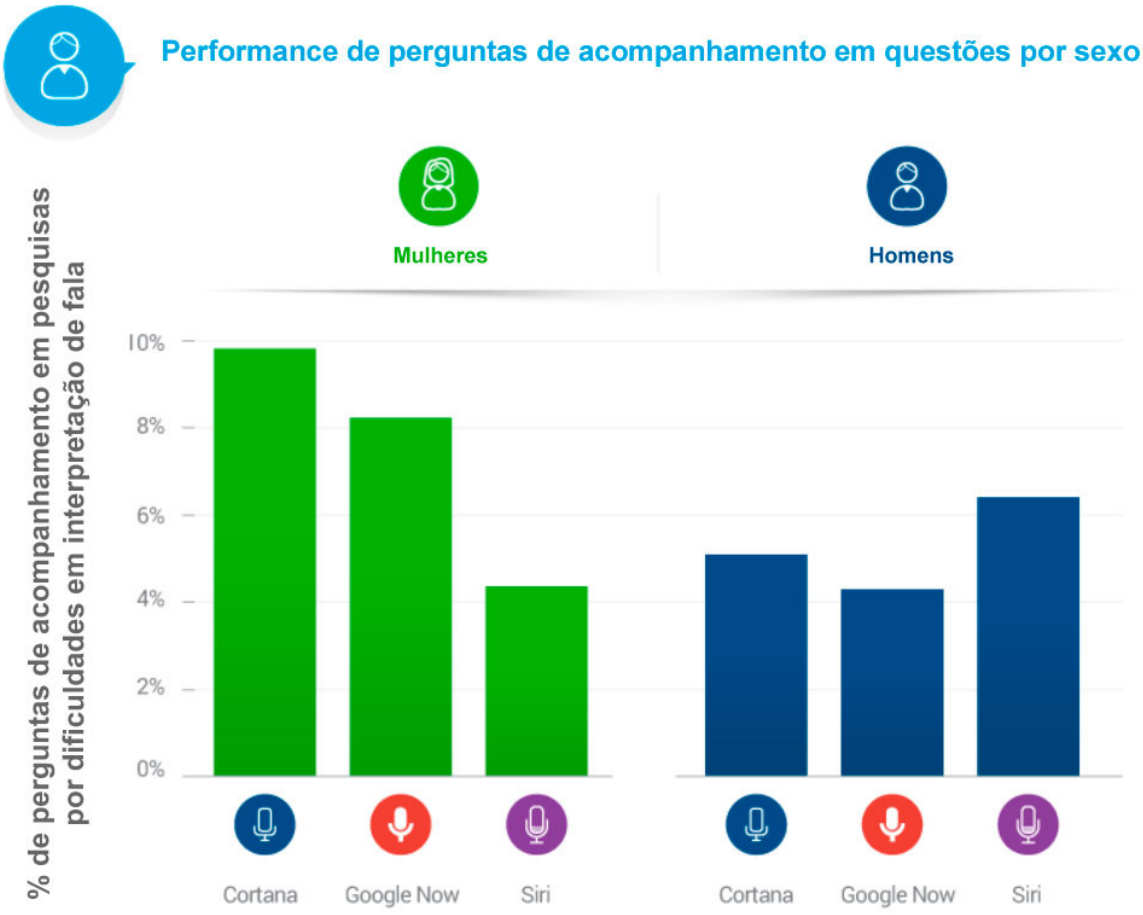
Fonte: Experts Exchange, 2015.

Wilfred com o auxilio da pesquisa realizada pelo Experts Exchange concluiu que a assistente de voz Apple Siri é superior em todos os pontos aos demais, sendo que a Google Now faz uma boa sombra a ele e a Cortana tem muito que evoluir, apesar de estar em um bom nível por ser uma ferramenta relativamente nova às demais na pesquisa expostas.

O site Phone Arena realizou testes de interpretação de fala e conversão para texto de homens e mulheres nas três assistentes de voz, tomando por base perguntas de acompanhamento realizadas quando as assistentes não conseguem interpretar o que foi falado. Nos testes realizados em mulheres, a Siri se mostrou de longe a mais eficiente, por ter menos perguntas de acompanhamento, que nada mais são do que o entendimento do contexto por parte da assistente, seguida pela

Google Now e por fim a Microsoft Cortana. No caso dos homens, os resultados foram mais similares, porém a que melhor se saiu foi a Google Now, seguido pela Cortana e Apple Siri (PHONE ARENA, 2015). A Figura 9 mostra os resultados obtidos nas interpretações de fala dos sexos masculino e feminino com base em perguntas de acompanhamento.

Figura 9 – Percentuais de perguntas de acompanhamento das assistentes nos sexos masculino e feminino.



Fonte: Phone Arena, 2015.

A pesquisa foi concluída tendo a Microsoft Cortana como a melhor assistente no quesito de interpretação de fala, por ter uma diferença grande em mulheres. A segunda melhor opção ficou por conta da Google Now, visto que foi muito melhor que a Apple Siri no sexo feminino e levemente inferior no masculino.

A seguir são apresentadas API's de Reconhecimento de voz, que são conjuntos de rotinas e padrões de programação para acesso a determinados softwares ou plataformas baseados em *Web* a fim de utilizar serviços (TECMUNDO,

2009).

2.2.4 API's de Reconhecimento de Voz

Serão apresentadas na seqüência três API's (*Application Programming Interface*) multiplataformas, onde é possível incorporá-las a diversas linguagens de programação, sendo muito flexíveis e úteis a projetos realizados nas mais variadas plataformas. As API's que serão citadas são a Microsoft Bing API Speech, Google Cloud Speech e IBM *Watson*, essas são responsáveis pelos algoritmos que estão por trás das assistentes de voz apresentadas na subseção anterior.

2.2.4.1 Microsoft Bing API Speech

A Microsoft Bing API Speech[10] é uma API que opera em nuvem e fornece algoritmos avançados para processar linguagem falada. Com esta API, desenvolvedores podem adicionar ações iniciadas através da fala para suas aplicações, incluindo interação em tempo real com o usuário.

A API opera em todos os dispositivos conectados à Internet, não é possível trabalhar *offline*, e é executada nas principais plataformas móveis, como Android, iOS e Windows Phone, além disso a ferramenta oferece suporte à fala para texto, texto para fala, além de capacidades de compreensão de linguagem entregues via nuvem.

A maior utilização da API têm sido para aplicações Windows como Cortana e Skype Tradutor, bem como aplicações Android como Bing Torque para Android Wear e Android Phone.

A API fornece a capacidade de converter áudio falado em texto, enviando o áudio para servidores da Microsoft na nuvem e, para fazer isso, os desenvolvedores têm a opção de usar a API REST (*Representational State Transfer*), que opera apenas *online*, ou a biblioteca do cliente onde é possível a utilização *offline*. Ao utilizar uma API REST, se obtém apenas um resultado de retorno, sem resultados parciais, já utilizando a biblioteca do cliente, os resultados parciais são retornados ao

[10] Microsoft Bing API Speech – Disponível em: https://www.microsoft.com/cognitive-services/en-us/speech-api/documentation/overview

usuário em tempo real.

Quando os aplicativos precisam falar o retorno para os seus utilizadores, essa API pode ser usada para converter o texto gerado pelo aplicativo em áudio que pode reproduzidos para o usuário. A conversão de texto para voz é realizada através de uma API REST.

A Microsoft disponibiliza uma licença gratuita para utilização de um determinado número de interações por mês.

2.2.4.2 Google Cloud Speech

A API Google Speech Cloud[11] permite aos desenvolvedores converter áudio para texto através da aplicação de modelos de redes neurais onde mais de 80 línguas são suportadas. É possível converter áudio ditado ou áudio de arquivos em texto, além de ser possível a realização de comandos de ações por voz. Existe também a opção de ditado, onde o usuário pode digitar um texto e a API lê o mesmo.

Os resultados do reconhecimento são expostos parcialmente em tempo real, assim que estiverem disponíveis, imediatamente após ocorrer uma fala. Como alternativa, a API pode retornar o texto reconhecido de áudio e armazená-lo em arquivos.

Possui a opção de operar *offline*, onde não é necessário que o usuário tenha uma rede disponível para que a conversão de fala para texto seja realizada, além disso, essa ferramenta possui uma tolerância a ruídos, onde o usuário não precisa necessariamente estar num ambiente onde o fundo é totalmente limpo.

São utilizados algoritmos de rede neural com aprendizagem de máquina onde a sua precisão melhora ao longo do tempo à medida que novos termos não incluídos em seu vocabulário passam a ser conhecidos.

O uso da ferramenta é gratuito, porém a Google enfatiza que no futuro haverá custos para a utilização da mesma.

---

[11] Google Cloud Speech – Disponível em: https://cloud.google.com/speech/

2.2.4.3 IBM Watson

O IBM Watson[12] possui duas API's voltadas para o reconhecimento de voz, sendo eles o de texto para fala e de fala para texto. A fim de transcrever a voz humana com previsão, o serviço utiliza aprendizagem de máquina para combinar informações sobre gramática e estrutura da linguagem. O serviço retorna em tempo real o resultado das interações.

Os serviços do Watson são expostos através de uma API RESTful, com isso pode ser facilmente integrado a aplicativos já existentes, sendo estes de diferentes plataformas. Isto com o porém de não operar em modo *offline*, sendo necessário estar conectado à uma rede.

São suportados idiomas como o português do Brasil, japonês, chinês, árabe, espanhol, inglês da Inglaterra e Inglês dos Estados Unidos.

A API possui uma licença gratuita nos primeiros mil minutos do mês, após isso são cobradas taxas adicionais por minutos de utilização.

Após a apresentação das API's de reconhecimento de voz, as tecnologias que visam ser incorporadas ao desenvolvimento proposto foram finalizadas. No capítulo 3, a seguir, serão apresentadas as diferentes formas existentes de desenvolvimento de aplicativos para dispositivos móveis.

[12] IBM Watson (Speech to text and text to speech) – Disponível em: http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/

3 FORMAS DE DESENVOLVIMENTO DE APLICATIVOS

Segundo Mario Korf e Eugene Oksman (2014), aplicativos deixaram de ser uma opção, quando se pensa em desenvolver aplicações *web,* e se tornaram uma necessidade graças ao grande público que hoje utiliza esse tipo de recurso através de dispositivos móveis. Ao definir que essa aplicação será desenvolvida, é necessário avaliar as possíveis formas de desenvolvimento da mesma, levando em conta sempre a habilidade da equipe de desenvolvimento, funcionalidades necessárias para o aplicativo, importância da segurança, capacidade de operar *offline*, entre outros fatores relevantes. É importante salientar que não existe uma escolha única e melhor, isso deve ser avaliado conforme se apresenta cada cenário.

Segundo o site Gartner (2016), o mercado de *smartphones* é bastante heterogêneo, com sistemas operacionais diferentes entre um dispositivo e outro, o que dificulta o trabalha dos desenvolvedores. A Tabela 5 mostra as vendas de dispositivos por sistema operacional na comparação do quarto trimestre de 2014 e 2016:

Tabela 5 – Vendas de *smartphones* por sistema operacional no 4º trimestre 2014/2015.

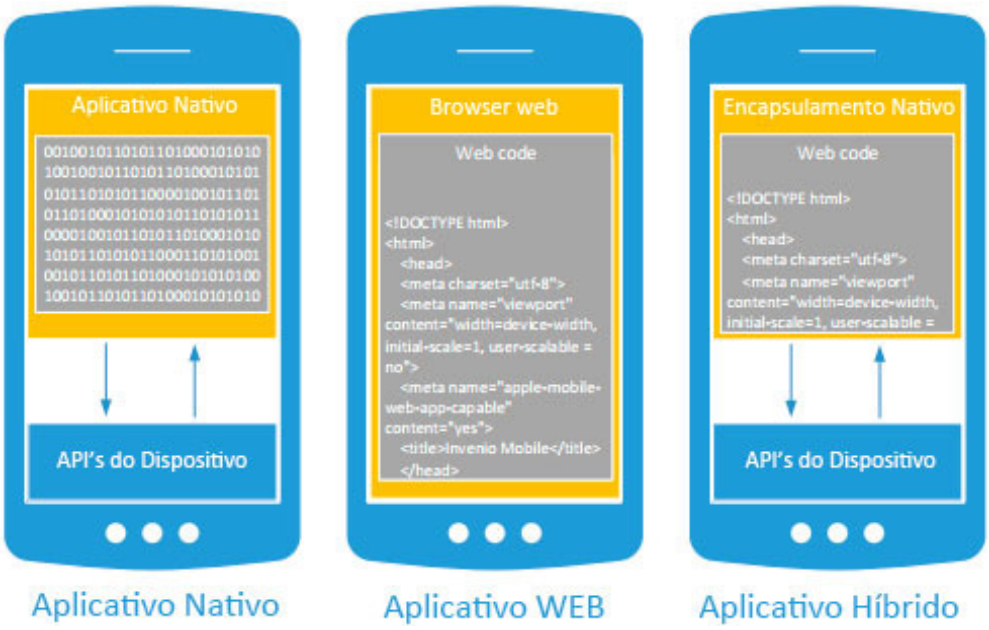
Sistema Operacional	4º trimestre 2014	4º trimestre 2015
Android	76%	80,7%
iOS	20%	17,7%
Windows	2,8%	1,1%
Blackberry	0,5%	0,2%
Outros	0,4%	0,2%

Fonte: Gartner, 2016.

Conforme exposto na Tabela 5, é possível perceber que o sistema Android domina o mercado, mas ainda está longe de ser unanimidade, o que implica em questões para se pensar antes do desenvolvimento ser iniciado.

Há três tipos de aplicativos existentes, são eles: nativos, *web* e híbridos. A Figura 10 ilustra as diferenças dos três tipos de aplicativos citados.

Figura 10 – Diferentes formas de desenvolvimento e tipos de aplicativos resultantes.



Fonte: Tapparel, 2013.

Nas seguintes seções serão conceituados os diferentes tipos de aplicativos apresentados anteriormente e as diferenças no seu desenvolvimento.

3.1.1 Aplicativo Nativo

Segundo Tapparel (2013), os aplicativos nativos são específicos para uma determinada plataforma móvel (como por exemplo, Android ou iOS) e não é possível executá-los em outras, onde o desenvolvimento é feito utilizando as ferramentas e as linguagens da plataforma (por exemplo Xcode e Objective-C para iOS, Eclipse e Java para Android).

Aplicativos nativos possuem pleno acesso às API's de baixo e alto nível, sendo que as de baixo nível são as que estão perto do *hardware* e podem interagir com o dispositivo com rede, áudio, geolocalização, câmera, entre outras. Já as API's de alto nível são os componentes e serviços do Sistema Operacional, tais como contatos, calendário, mensagens de texto, etc.

Desenvolvedores necessitam de um SDK para cada sistema operacional. O SDK fornece ferramentas para escrever o código-fonte e depurar o projeto. Na Tabela 6 podemos ver um resumo das linguagens, ferramentas e lojas de cada sistema operacional:

Tabela 6 – Kit de desenvolvimento para cada sistema operacional.

Sistema Operacional	Linguagens	Ferramentas	Lojas
iOS	Objective-C, C, C++	Xcode	APP Store
Android	Java (alguns C, C++)	Andoid SDK	Google Play
Windows Phone	C#,	Visual Studio	Windows Phone Store

Fonte: Tapparel, 2013.

Cada sistema operacional tem seus próprios elementos de interface com o usuário, tais como: botões, menus, entrada de dados, entre outros. Levando isso em conta, aplicativos que são projetos para operar multiplataformas precisam que o desenho das interfaces siga a familiaridade do sistema operacional.

Utilizando a GPU (*Graphics Processing Unit*), que segundo a NVIDIA (2012) é uma unidade de processamento gráfico que visa acelerar aplicações, os elementos nativos têm um desempenho muito mais eficiente, pois usam o acelerador gráfico.

A grande desvantagem do aplicativo nativo é que seu desenvolvimento se aplica apenas ao sistema operacional para o qual foi concebido, sendo necessário reiniciar o desenvolvimento do início com uma outra linguagem para ser possível executar em outras plataformas. Essa situação descrita resulta no aumento dos custos, do tempo e da quantidade de recursos para a finalização do aplicativo (JUNTUNEN, 2013).

3.1.2 Aplicativo Web

Aplicativos *web* nada mais são que aplicações móveis construídas a partir de linguagens de programação suportadas pelos *browsers*. *Smartphones* e demais dispositivos móveis possuem em sua totalidade *browsers* instalados, fazendo com que esse tipo de aplicação possa ser executado em todos eles. Resumidamente, é um site feito para dispositivos móveis. Os aplicativos da *web* não precisam ser disponibilizados para *download* em lojas de aplicativos ou para serem baixado pelos usuários. Uma grande vantagem desse tipo de aplicação é o fato de utilizar pouco espaço em disco nos dispositivos, além de não serem necessárias atualizações nos

dispositivos, quando necessária troca de versão, a mesma é feita dentro do servidor onde a aplicação está alocada (TAPPAREL, 2013)

Desenvolvedores com experiência em desenvolvimento *web* possuem facilidade para criar esse tipo de aplicação. As tecnologias utilizadas são o HTML (*HyperText Markup Language*) versão 5, CSS (*Cascading Style Sheets*) e JavaScript no lado do cliente, já no lado servidor podemos ter PHP, Java, Python, entre outros (TAPPAREL, 2013).

Não há diferença alguma na aparência de um aplicativo da *web* e um nativo. Em comparação com versões anteriores do HTML, o HTML5 traz novos recursos, tais como: alteração na semântica, trabalhos *off-line*, armazenamento local, conectividade, acesso a recursos do dispositivo, multimídia, efeitos 3D, desempenho e integração a *plug-ins* externos através de *tags* específicas para áudio e vídeo. Detalhes dos novos recursos do HTML5 podem ser consultados no site da W3C (*World Wide Web Consortium*)[13] (TAPPAREL, 2013).

Se na aparência não há diferenças, é importante ressaltar que há diferenças no desempenho, há perdas consideráveis de velocidade na execução de aplicativos web em relação a aplicativos nativos. Considerando que a evolução das tecnologias é constante, a diferença citada tende a ser reduzida com o passar do tempo (ROLNITZKY, 2010).

Aplicativos *web* possuem algumas limitações quanto ao acesso aos recursos dos dispositivos, tais como câmera, acelerômetro, lista de contatos, entre outros. Essas limitações já foram maiores, mas atualmente, contamos com APIs que possibilitam a realização de acessos a alguns recursos dos dispositivos. (ROLNITZKY, 2010).

Para utilização em dispositivos móveis, o HTML5 permite a utilização de API's de geolocalização (GPS, celular e Wi-Fi) e armazenamento local. Na prática, em comparação com aplicações nativas, o HTML5 possui limitações significativas, pois existe um número limitado de API's para serem utilizadas. Além de não ser possível utilizar o acelerômetro, buscar contatos, entre outros. No armazenamento, o HTML5 permite um armazenamento local de 5MB por domínio e 1MB de sessão limitado (GOOGLE, 2012).

[13] W3C: http://www.w3c.br/

O HTML5 tem como objetivo principal facilitar a manipulação dos elementos possibilitando ao desenvolvedor modificar características de objetos de forma simples. Outra característica desta nova versão é a utilização de API's, fazendo com que *websites* e aplicações continuem leves e mais funcionais do que antes, além disso, há novas tags incluídas que facilitam o trabalho do desenvolvedor (W3C, 2016).

3.1.3 Aplicativo Híbrido

Para Tapparel (2013), desenvolvimento híbrido combina a execução nativa com desenvolvimento multi-plataforma de tecnologias. No desenvolvimento híbrido, os aplicativos são construídos em HTML5, CSS e JavaScript e após são convertidos para a linguagem nativa. Não são verdadeiramente nativos nem tampouco puramente *web*, nesta forma de desenvolvimento os aplicativos podem ser escritos em uma linguagem web. Porém, terão acesso aos recursos nativos avançados. Este tipo de aplicação utiliza o motor de renderização do navegador *webkit* como *software* de mecanismo de *layout*. Porém, podem apresentar pequenas diferenças entre sistemas operacionais distintos.

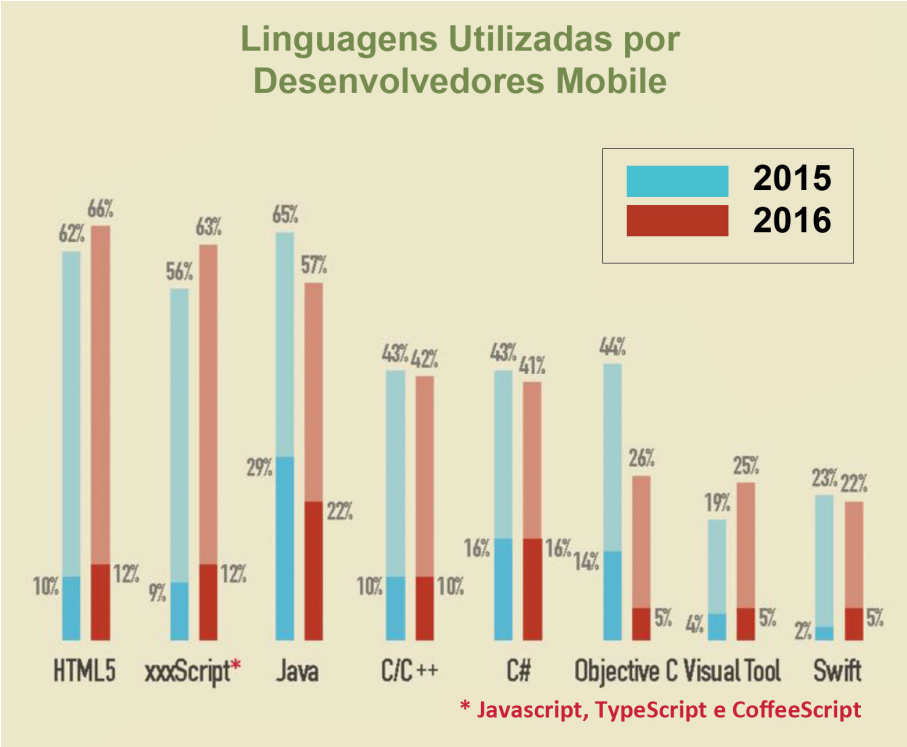
O grande trunfo do desenvolvimento híbrido é ser multi-plataforma, onde o desenvolvedor programa apenas um aplicativo em HTML5, e o mesmo poderá ser executado sem problemas em diferentes sistemas operacionais, podendo ser publicado nas diferentes lojas de aplicativos. A aplicação pode usar as API's e funções nativas sem necessidade de conexão com a internet.

O armazenamento desse tipo de aplicação pode ser realizado de duas diferentes formas (localmente ou em servidor). É comum ser feito localmente, no aplicativo em si como um arquivos de recursos. A forma alternativa é armazenar os arquivos em um servidor, onde nessa forma há a desvantagem da necessidade da conexão com a internet.

Segundo a Vision Mobile (2016), a utilização de linguagens nativas vem caindo enquanto as "híbridas" estão aumentando. O uso do Java, por exemplo, caiu cerca de 8% em apenas um ano, do início de 2015 para o início de 2016, já o HTML5 avançou 4% e o JavaScript 7% neste mesmo período. Esse indicadores são uma evidência clara de que o uso de ferramentas multi-plataforma têm ganho muito

espaço nos últimos anos. A Figura 11 ilustra esse crescimento do primeiro trimestre de 2015 para o mesmo período de 2016.

Figura 11 – Linguagens mais utilizadas para mobile no primeiro trimestre 2015/2016.



Fonte: Vision Mobile, 2016.

Existem diversos *frameworks* para o desenvolvimento de aplicativos híbridos, na seqüência do trabalho serão apresentados alguns dos principais e mais utilizados do mercado.

3.1.3.1 Ionic

É um *framework* de desenvolvimento híbrido de licença *open source* que vem ganhando muito espaço no âmbito deste segmento. Sua equipe de desenvolvedores busca manter o mesmo atualizado acompanhando as tendências e novas tecnologias disponíveis. Possui a vantagem de ser bastante utilizado, fazendo com que seja fácil encontrar materiais interessantes junto a comunidade que aderiu ao Ionic para o desenvolvimento (NOETICFORCE, 2016).

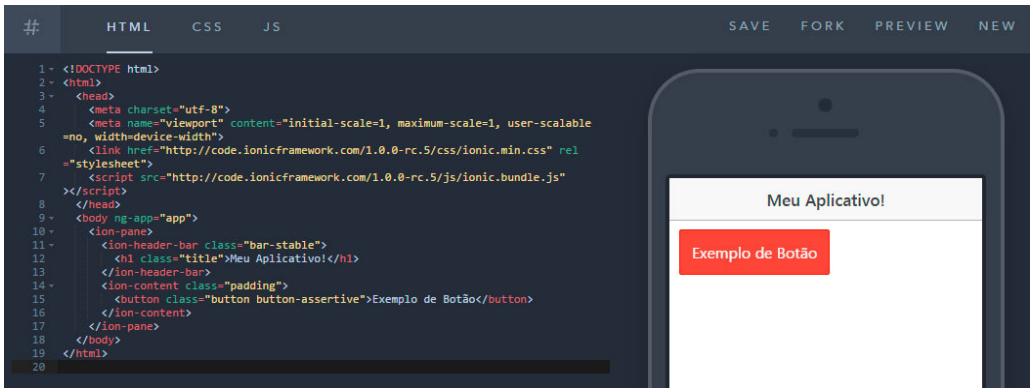
Ionic utiliza o poder do HTML5 junto ao AngularJS, além disso utiliza o Apache Cordova para compilar nativamente em sistemas operacionais como iOS, Android, Windows Phone, entre outros (NOETICFORCE, 2016).

O AngularJS é um *framework* escrito em JavaScript e desenvolvido pela empresa Google, de licença *open source* que funciona como uma extensão do HTML, onde o objetivo da criação do mesmo foi de aumentar o número de aplicativos que podem ser acessados por um *browser* e facilitar o desenvolvimento, tornando o código mais enxuto e direto, procurando poupar linhas de código com funções já desenvolvidas e sendo chamadas sempre o desenvolvedor desejar. Utiliza o padrão MVC (*Model View Controller*) e possui tags específicas para cada função desejada (ANGULARJS, 2016).

O *framework* é limpo e de fácil marcação, junto a ele vem uma biblioteca altamente otimizada para dispositivos móveis de CSS, HTML e componentes JS. Ele também possui ferramentas e atalhos que visam facilitar o desenvolvimento de aplicativos interativos (NOETICFORCE, 2016).

A Figura 12 ilustra o ambiente de desenvolvimento Ionic.

Figura 12 – Exemplo de código-fonte do *framework* Ionic.



Fonte: O autor, 2016.

3.1.3.2 Xamarin

Xamarin é uma solução mista entre a estratégia híbrida e a nativa, o desenvolvimento é realizado utilizando a linguagem C# e o código escrito será reaproveitado entre as diferentes plataformas quase que totalmente, fazendo com que exista unicidade da base de código tornando a experiência como a de se utilizar aplicações nativas (ONCEDEV, 2014).

O código escrito em C# é totalmente reaproveitado entre as plataformas, e isso se torna uma vantagem, pois é justamente a parte responsável por garantir que a aplicação possua a experiência nativa em conformidade com as interfaces e funcionalidades presentes na plataforma em que a mesma está sendo executada. Outra vantagem dessa solução é o fornecimento de espelhos de todas as API's disponíveis no Android e no iOS (ONCEDEV, 2014).

Por não ser uma ferramenta *open-source*, o Xamarin possui a desvantagem de ter custos envolvendo licenciamento, que possuem diferentes valores para estudantes, desenvolvedores e empresas (ONCEDEV, 2014).

A Figura 13 mostra um exemplo de código-fonte utilizando a solução de desenvolvimento Xamarin.

Figura 13 – Exemplo de código-fonte do Xamarin.

```
namespace ExemploArtigoAndroid
 {
     [Activity (Label = "ExemploArtigoAndroid", MainLauncher = true, Icon = "@drawable/icon")]
     public class MainActivity : Activity
     {
         protected override void OnCreate (Bundle bundle)
         {
             base.OnCreate (bundle);

             SetContentView (Resource.Layout.Main);

             Button button = FindViewById<Button> (Resource.Id.myButton);
             EditText editText1 = FindViewById<EditText> (Resource.Id.editText1);
             EditText editText2 = FindViewById<EditText> (Resource.Id.editText2);

             button.Click += delegate {
                 button.Text = OperacaoMatematica.Soma(int.Parse(editText1.Text), int.Parse(editT
             };
         }
     }
 }
```

Fonte: DEVMEDIA, 2016.

3.1.3.3 Phonegap

Phonegap possibilita criação de aplicativos para plataformas iOS, Android e Windows, de diferentes linguagens de desenvolvimento. Isso é possível utilizando tecnologias *web* baseadas em padrões, assim desenvolvendo aplicações *web* para dispositivos móveis (PHONEGAP, 2016).

É uma ferramenta *open source* desde outubro de 2012 quando ficou conhecida sob o nome de Apache Cordova. Tem a vantagem de possuir a iniciativa de ser sempre livre, sendo assim, o desenvolvimento poderá seguir sendo feito pelo

framework, pois permanecerá nessa condição com código aberto (PHONEGAP, 2016).

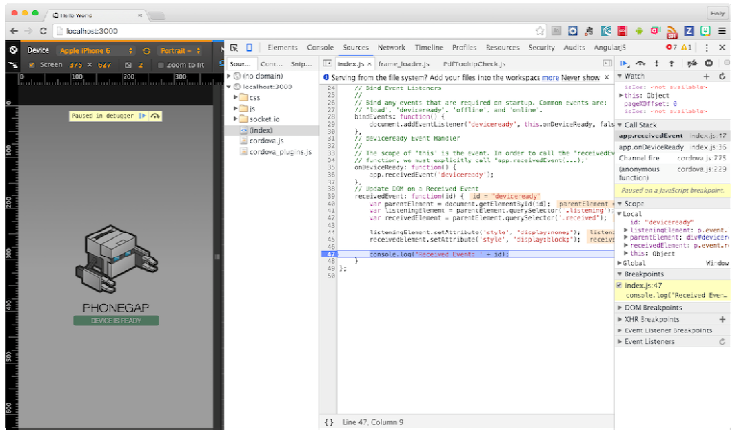
O funcionamento é baseado em HTML5, CSS3 e Javascript. Aplicativos desenvolvidos com esse *framework* são executados simulando ambientes nativos de cada plataforma. Para realizar essa simulação de ambientes nativos, o mesmo utiliza API's que possibilitam o acesso aos recursos dos dispositivos. Nos casos em que as API's existentes não são suficientes para realizar essa comunicação, é possível desenvolver *plugins* para realizar a mesma (PHONEGAP, 2016).

O desenvolvimento multiplataforma ocorre graças ao utilitário Cordova CLI (*Command-Line Interface*), este permite que programação alto nível acesse recursos de baixo nível. Este utilitário cria comandos padrões para pastas criadas para cada plataforma que se deseja executar, ajustando a compatibilidade conforme necessidade do desenvolvedor e gerando os arquivos de execução (PHONEGAP, 2016).

Para possibilitar a utilização do CLI são necessárias duas instalações, sendo o Node.js e um cliente Git. O Node.js é uma uma ferramenta escrita em Javascript orientada a eventos que permite conexões de rede não bloqueantes ao servidor, ou seja, ideal para aplicações em tempo real e com troca intensa de dados através de dispositivos distribuídos (NODEBR, 2013). Já o Git é basicamente um controle de versionamentos de código (GIT, 2016).

A Figura 14 ilustra o ambiente de desenvolvimento do *framework* de desenvolvimento Phonegap.

Figura 14 – Ambiente de desenvolvimento do Phonegap.



Fonte: Phonegap, 2016.

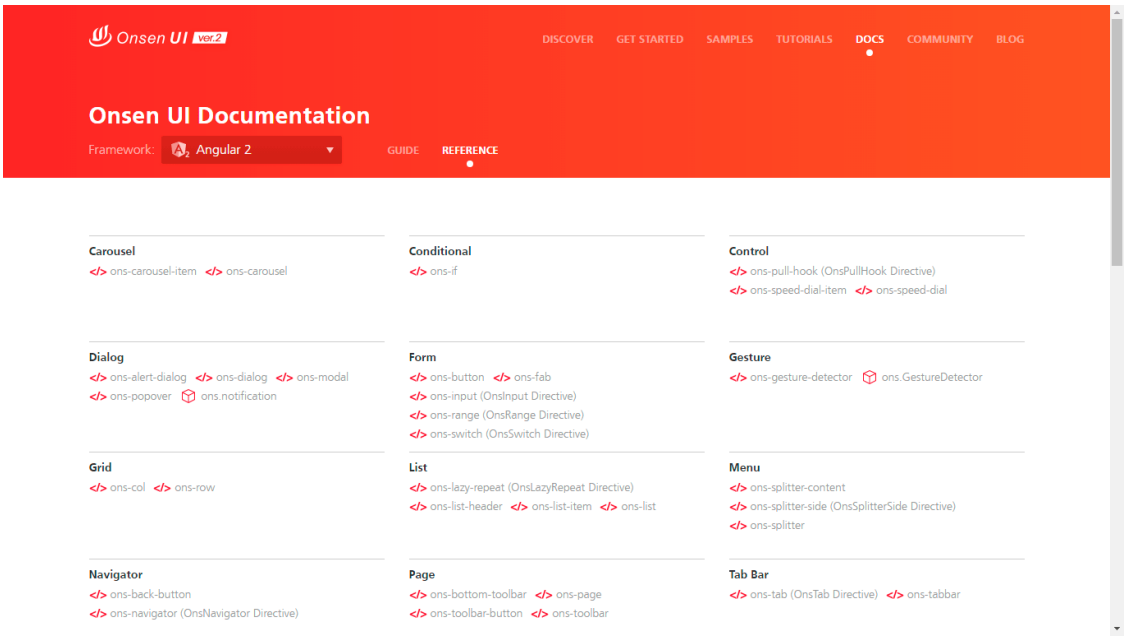
3.1.3.4 Onsen UI

O Onsen UI é um *framework* de desenvolvimento híbrido de licença *open source* semelhante ao Ionic. O desenvolvimento é realizado utilizando HTML5, CSS3 e Javascript. Onsen UI utiliza o Apache Cordova e trabalha bem juntamente com o Phonegap para realizar sua compilação, além de trabalhar com o AngularJS da Google. Seu desenvolvimento pode ser realizado em nuvem graças ao Monaca Cloud IDE que inclui também um debugador.

O principal objetivo para que esse *framework* fosse criado foi o de tornar as aplicações híbridas, que antes eram criadas com PhoneGap puro, mais rápidas, focando em performance, por isso todas as animações que ele contempla foram desenvolvidas visando rapidez de execução, diminuindo a dependência do alto desempenho dos *hardwares*.

Outro fator positivo do *framework* é o fato de conter diversos componentes de interface de fácil implementação, o que torna o desenvolvimento mais ágil e direto ao ponto crítico, que é a regra de negócio propriamente dita, ou seja, o desenvolvedor não perde tempo com desenhos de tela, Onsen UI contém diversas interfaces pré-definidas que podem ser utilizadas livremente além de botões e efeitos de transição de tela. (ONSEN UI, 2016).

Figura 15 – Documentação com componentes de tela do Onsen UI.



Fonte: Onsen UI, 2016.

A figura 16 mostra um trecho de código do próprio *software* apresentado neste trabalho utilizando o *framework* Onsen UI.

Figura 16 – Código fonte do projeto utilizando Onsen UI.

```
<ons-page ng-controller="detalhesController">
        <toolbar></toolbar>
    <div class="cond-list">
        <ons-list class="solucao-list" ng-click="sliding_menu.closeMenu();">
            <ons-list-header class="solucao-list-header detalhes-header" ng-class='
                <div class="cond_img"> </div>
                <div class="cond_info">
                    <h1>{{condominio.condominio}}</h1>
                    <span>cod {{condominio.codcond}}</span>
                </div>
            </ons-list-header>
            <ons-list-item ng-repeat="unidade in clientes | filter: searchQuery">
                <span class="cliente_listing" id="unidade.economia">
                <div  ng-repeat="tipo in tipos" class="unidade solucao-list-item">
                    <span>{{ unidade.economia }} - {{ tipo | tipoLeitura }}</span>
                    <div class="leitura-icon {{tipo}}">
                    </div>
                    <div class="leitura-indices">
                        <div class="indice anterior">
                            <p>Anterior</p>
                            <ons-icon icon="ion-chevron-left"</ons-icon>
                            </ons-icon>
                            <p> {{leituras[unidade.economia + '_' + tipo].anterior
                        </div>
                        <div class="indice menor">
                            <p>Menor</p>
```

Fonte: O autor, 2016.

Após a apresentação das diferentes formas de desenvolvimento e apresentação de ferramentas disponíveis para aplicativos móveis, segue apresentado na seção 2.4 seguinte o conceito de *web services*.

3.2 WEB SERVICES

Web services são soluções utilizadas para realizar integrações de aplicações que operam em plataformas diferentes e não compatíveis. Essa tecnologia surgiu sob a arquitetura SOA (*Service Oriented Achitecture*), que nada mais é do que uma arquitetura orientada a serviços onde as funcionalidade implementadas nas aplicações devem ser disponibilizadas na forma de serviços, e é definida pela W3C como um “sistema de *software* projetado para suportar a interoperabilidade entre máquinas conectadas a uma rede. O mesmo provê uma interface descrita em um formato processável por máquinas” (W3C, 2016).

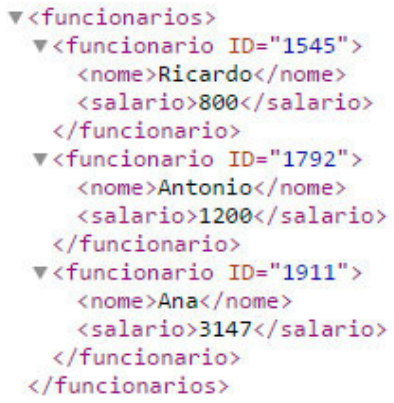
Essa solução utiliza padrões abertos para comunicação tais como XML (Extensible Markup Language), JSON (*JavaScript Object Notation*), CSV (*Comma-separated Values*). Os formatos XML e JSON, que são os mais utilizados formatos de transmissão citados seguem descritos nos subitens conseguintes.

3.2.1 XML

XML é uma linguagem de marcação que utiliza um formato de texto simples e é um subconjunto co SGML (*Standard Generalized Markup Language*). Podem ter estruturas hierárquicas e elementos recursivos. Outra característica marcante dessa linguagem é ela ser legível para humanos, isso contribui e muito para o seu sucesso (W3C, 2008).

A Figura 17 mostra um exemplo de arquivo XML.

Figura 17 – Exemplo de estrutura de um arquivo XML.



Fonte: O autor, 2016.

3.2.2 JSON

JSON é um formato leve para intercâmbio de dados, é uma linguagem independente e de fácil compreensão humana. Seu formato é idêntico ao código para criação de objetos JavaScript e graças a isso pode utilizar funções do mesmo para converter dados JSON em objetos nativos do próprio JavaScript. É semelhante ao formato XML, porém existem algumas particularidades como não utilização de tag de fim, possível criação de *arrays*, além de ser mais curto e rápido para se ler e escrever (W3C, 2016).

A Figura 18 mostra um exemplo de arquivo JSON.

Figura 18 – Exemplo de estrutura de um arquivo JSON.

```
{"funcionarios":[
    {"nome":"Ricardo", "salario":"800"},
    {"nome":"Antonio", "salario":"1200"},
    {"nome":"Ana", "salario":"3147"}
]}
```

Fonte: O autor, 2016.

3.3 FERRAMENTAS E LINGUAGENS DE DESENVOLVIMENTO

3.3.1 PHP

PHP (*Hypertext Preprocessor*) é uma linguagem de script *open-source* voltada para o desenvolvimento *web* e que pode ser combinada dentro do HTML (PHP, 2016).

As páginas PHP contêm HTML mesclado, onde uma instrução PHP é renderizada pelo browser se tornando um HTML puro ao cliente. O código desta linguagem é delimitado pelas tags de início e fim, sendo <?php e ?> respectivamente, o que permite ao desenvolvedor entrar e sair do chamado “modo PHP” (PHP, 2016).

A Figura 19 mostra um exemplo de código da linguagem PHP.

Figura 19 – Exemplo de script PHP.

```
1  <!DOCTYPE HTML>
2  <html>
3      <head>
4          <title>Exemplo</title>
5      </head>
6      <body>
7
8          <?php
9              echo "Olá, eu sou um script PHP!";
10         ?>
11
12     </body>
13 </html>
```

Fonte: Adaptado de PHP, 2016.

A principal diferença do PHP para uma linguagem como o JavaScript é que o mesmo é executado no lado servidor, diferentemente do JavaScript que é no lado cliente (PHP, 2016).

A última versão estável do PHP é a 5.6.9 que foi liberada na data de 14/05/2016, existem também uma versão de testes, que é a 7.0.5 liberada em 31/03/2016 (PHP, 2016).

3.3.2 JavaScript

O JavaScript é uma linguagem de programação da *Web*. Praticamente todos os sites da internet utilizam a mesma, os navegadores mais utilizados possuem interpretadores para ela, sendo utilizada amplamente não são em computadores e *notebook*, assim como em *tablets* e *smatphones* (FLANAGAN, 2014).

JavaScript é uma das tecnologias que todos os desenvolvedores da Web deveriam saber, pois o HTML estrutura o conteúdo das páginas, o CSS faz o estilo das mesmas e finalmente o JavaScript trabalho o seu comportamento (FLANAGAN, 2014).

A figura 20 mostra um exemplo de código da linguagem JavaScript.

Figura 20 – Exemplo de script JavaScript.

```
1  var indexController = function ($scope, $http)
2  {
3      var resultPromise = $http.get("/Home/GetTweets");
4      resultPromise.success(function (data)
5      {
6          $scope.tweets = data;
7      });          © DevCurry.com
8
9      $scope.newTweets = {
10         0: "No new Tweets",
11         other: "{} new Tweets"
12     }
13 }
```

Fonte: AngularJS, 2016.

É uma linguagem que é executada no lado cliente, ou seja, acontece no próprio browser, sem a necessidade de comunicação com um servidor (FLANAGAN, 2014).

3.3.3 MySQL

O MySQL é um servidor e SGBD (Gerenciador de Banco de Dados) relacional que utiliza a linguagem SQL (*Structured Query Language*) e possui licença mista, sendo uma delas *open-source* (MYSQL, 2016).

Inicialmente, o MySQL foi projetado para trabalhar com aplicações de pequeno e médio porte, porém evoluiu com o passar do tempo e passou a atender grandes aplicações, tanto que grandes organizações utilizam esse servidor, tais como: NASA, HP, Nokia, Sony, entre outras (MYSQL, 2016).

Por possuir todos os requisitos que um banco de dados de grande porte necessita é reconhecido como o banco de dados *open source* com maiores condições de concorrer com programas semelhantes de código fechado, tais como o SQL Server e o Oracle (MYSQL, 2016).

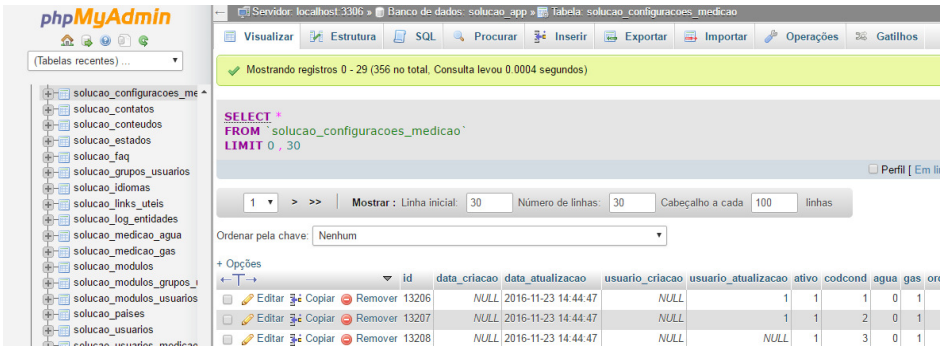
Sua versão estável é a 5.6.12 que entrou em operação no dia 03/06/2013, existem também uma versão em teste que é a 5.7.1, esta lançada em 23/04/2013 (MYSQL, 2016).

3.3.4 phpMyAdmin

O phpMyAdmin é uma ferramenta *web* escrita na linguagem PHP com o objetivo de gerenciar o banco de dados MySQL pela Web. Possui uma interface gráfica que permite ao usuário criar e remover bases de dados, inserir, alterar e deletar tabelas e registros de tabelas, além da possibilidade de executar instruções SQL puras (PHPMYADMIN, 2016).

A imagem 21 mostra a interface do phpMyAdmin.

Figura 21 – Interface do phpMyAdmin.



Fonte: O autor, 2016.

3.3.5 Bootstrap

O Bootstrap é um *framework front-end* criado para facilitar a criação de *websites*, principalmente para dispositivos móveis, sendo muito simples criar telas responsivas, ou seja, que se ajustam a qualquer resolução (BOOTSTRAP, 2016).

Além de ser possível criar com facilidade telas responsivas, o Bootstrap possui uma infinidade de componentes complementares (*plugins*) em JavaScript e jQuery que auxiliam a implementação de efeitos visuais que normalmente são de alta complexidade e consomem muitas linhas de código. Porém, com esses *plugins* essa implementação se torna simples e rápida, sendo necessárias apenas algumas linhas de código para configuração dos mesmos (BOOTSTRAP, 2016).

A Figura 22 mostra um exemplo de *website* desenvolvido com o *framework* Bootstrap e de como o mesmo se ajuda as mais diversas resoluções.

Figura 22 – *Website* com interfaces desenvolvidas com Bootstrap.

Fonte: BootstrapMaster, 2016.

3.3.6 Anyline SDK

O Anyline SDK é um pacote de desenvolvimento de *software open-source* baseado no algoritmo Tesseract, utilizado para reconhecimento de caracteres OCR.

Existem diversas soluções pré-definidas já programadas e prontas para serem incorporadas em projetos, algumas delas são: placas veiculares, números de passaporte, *vouchers* ou código de sorteio, medidores de gás, água e energia, entre outros. É possível treinar o Tesseract do Anyline SDK com fonts específicas definidas pelo desenvolvedor, onde um arquivo de configuração é gerado e deve apenas ser incorporado ao projeto para que o mesmo comece a efetuar os reconhecimentos. (ANYLINE SDK, 2016).

A figura 23 mostra exemplos de reconhecimentos pré-definidos do Anyline SDK.

Figura 23 – Exemplo de aplicações pré-definidas possibilitadas pelo Anyline SDK.

Fonte: Adaptado de Anyline SDK, 2016.

3.4 ENGENHARIA DE *SOFTWARE*

Engenharia de *software* é uma disciplina focada em todos os pontos da produção de *softwares*, desde os levantamentos inicias até questões de manutenção, nos casos em que o mesmo já está sendo utilizado (SOMMERVILLE, 2011).

Há abordagens sistemáticas envolvidas na engenharia de *software*, que são também conhecidas como processo de *softwares*, onde o processo é uma seqüencia de atividades que culminará na produção de um produto de *software* (SOMMERVILLE, 2011).

A Figura 24 mostra que a engenharia de *software* é composta por camadas, onde sua base é a camada de processos, com o foco na qualidade do produto (SOMMERVILLE, 2011).

Figura 24 – Camadas da Engenharia de *Software*.

Fonte: SOMMERVILLE, 2011.

Dentro da engenharia de *software* existem metodologias, ou seja, conjuntos de práticas estruturadas. Uma das metodologias existentes é o ICONIX, esta foi escolhida para a engenharia deste projeto pela sinteticidade que possui, tornando o entendimento do projeto mais simples para o leitor. Subseqüentemente, a metodologia ICONIX é apresentada.

3.4.1 Metodologia ICONIX

A metodologia ICONIX foi desenvolvida por Doug Rosenberg e Kendal Scott por volta do ano de 1993 (ROSENBERG, 2007).

Foi construída baseando-se em um processo simples e unificado dos pesquisadores Booch, Jocobson e Rumbaugh, onde a sua grande vantagem é justamente a sua simplicidade aliada ao alto nível de rastreabilidade dos requisitos (ROSENBERG, 2005).

A metodologia utiliza um subconjunto da UML (*Unified Modeling Language*), sendo utilizados apenas 4 diagramas ao invés dos 14 existentes até o presente momento, sendo eles: diagramas de casos de uso, classes, robustez e seqüencia (ROSENBERG, 2005).

Após a apresentação de todo o referencial teórico do projeto, bem como tecnologias disponíveis para o desenvolvimento do mesmo, será desenvolvida a proposta de solução no capítulo 4, visando apresentar a empresa e todo o processo de desenvolvimento do *software* realizado.

4 PROPOSTA DE SOLUÇÃO E DESENVOLVIMENTO DO *SOFTWARE*

Neste Capítulo é apresentada a proposta de solução para o desenvolvimento e implantação de um aplicativo para dispositivos móveis visando à realização de coletas de consumos de água e gás em Condomínios através dos mesmos numa empresa do ramo de administração de Condomínios.

Na seção 4.1 é apresentada a organização em que se pretende implantar a solução descrita, já na seção 4.2 o processo de coletas de consumos atual é descrito, bem como o posterior a implantação da solução aqui proposta. Nas seções 4.3 em diante, neste mesmo capítulo, é realizada toda a descrição das tecnologias utilizadas, requisitos, casos de uso e modelagem do *software*.

4.1 A EMPRESA

A Solução Serviços Para Condomínios é uma empresa de cultura familiar que atua exclusivamente no ramo de administração de Condomínios, estando no mercado desde 2000, ou, até a presente data, cerca de dezesseis anos. Sua sede está localizada em Caxias do Sul, onde conta com vinte e cinco colaboradores e possui contrato vigente de administração com trezentos e noventa Condomínios.

A Solução tem como característica ser uma empresa que sempre prezou pela automatização dos processos, visando à redução de retrabalhos e diminuição dos erros humanos, tendo isso como vantagem competitiva. Tendo isso em vista, conta com um ERP (*Enterprise Resource Planning*), além de um sistema de apoio *Web* voltado fortemente para a parte comercial da organização.

4.2 O PROCESSO

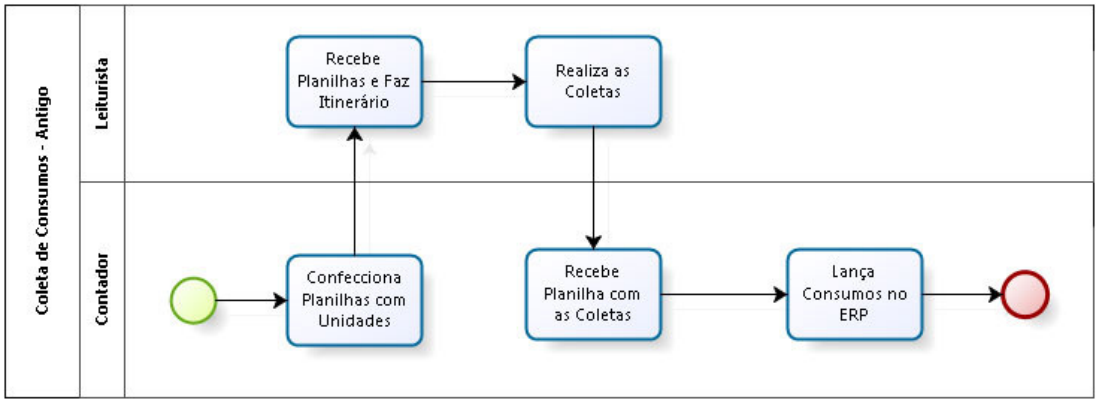
Como descrito no capítulo 1, o processo atual de coleta de consumo de gás e água nos Condomínios da organização em questão é realizado de forma manual e com retrabalhos, além do possível erro humano, que pode ser reduzido.

Estão envolvidos no processo dois atores principais, sendo eles um colaborador da organização que fica no escritório e leituristas que comparecem nos Condomínios para realizar as leituras. Em resumo, o colaborador tem a função de passar os dados fornecidos pelo leiturista para o sistema de gestão da organização,

já o leiturista precisa fazer a coleta dos consumos.

O processo atual pode ser visualizado na Figura 25 que segue.

Figura 25 – Processo de Coletas de Consumo pré Implementação de Solução.



Fonte: O autor, 2016.

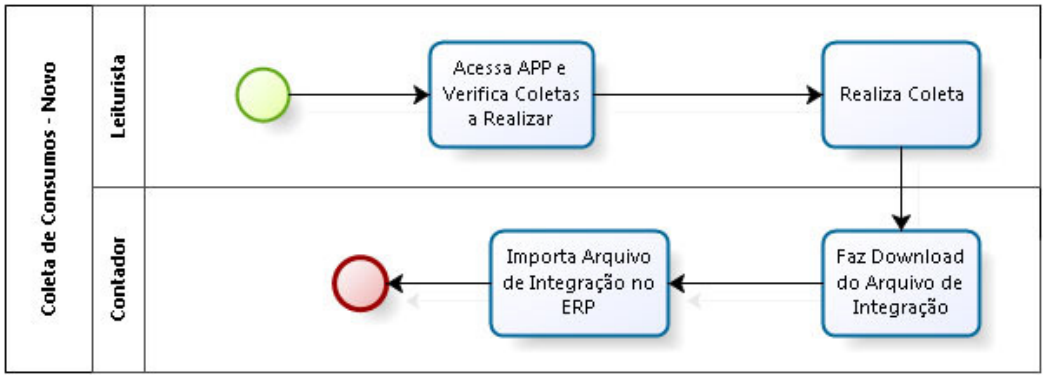
O presente trabalho tem como objetivo realizar o desenvolvimento de um aplicativo que resolva a problemática apresentada, criando um *software* para a coleta de consumos de água e gás em condomínios.

Esse *software* deve ser de fácil operação por parte do leiturista, fazendo com que haja uma redução nos erros de leitura. Para que isso ocorra serão utilizados recursos dos dispositivos móveis, tais como microfone, câmera e lanterna, tornando o processo mais ágil e eficiente que o realizado no presente momento.

Além de a coleta ficar mais rápida e automatizada que a atual, o retrabalho de transferir todos os dados contidos na planilha para o ERP pode ser evitando, fazendo com que o aplicativo gere arquivos que integrem no mesmo.

O processo resultante da solução proposta é ilustrado na figura 26.

Figura 26 – Processo de Coletas de Consumo pós Implementação de Solução.



Fonte: O autor, 2016.

4.3 SOFTWARE PROPOSTO

O software proposto neste trabalho pode ser executado em qualquer dispositivo móvel (*smartphone* ou *tablet*) que possua o sistema operacional Android e uma forma de conexão com a internet, sendo 3 ou 4G. Portando o dispositivo em mãos, o leiturista da Solução irá a campo com o mesmo, onde a rede irá sincronizar o *software* com os dados do ERP da empresa, tais como cadastro de Condomínios, clientes e configurações de medidas de consumo.

Assim que o leiturista acessa o sistema pelo dispositivo, o *software* expõe todo o gerenciamento das coletas, ou seja, são informados quais são os Condomínios de sua responsabilidade, bem como quais são as coletas realizadas e as pendentes dentro do período em questão (mês e ano). O aplicativo também informa dados relevantes da unidade tais como maiores consumos, consumo médio, entre outros. Além disso, percentuais de realização de coletas por bairro dentro do período são exibidos, a fim de situar o operador sobre o panorama atual e o mesmo gerenciar da melhor forma as coletas dentro de seu prazo.

Após a visualização das informações, o colaborador deverá selecionar o bairro que deseja coletar os dados e após isso, selecionar o Condomínio pertencente ao mesmo e assim abrir a planilha com as unidades pertencentes a ele. Nessa planilha são exibidas as unidades que devem ter os consumos coletados bem como os dados históricos de consumo das mesmas, estes visando redução de erros e detecção de vazamentos.

Com a planilha das unidades aberta, o leiturista deve se dirigir aos relógios de gás e/ou hidrômetros de água para coletar de fato os dados. Para fazer isso, o aplicativo proporciona ao usuário três formas de leitura, são elas: reconhecimento de caracteres, reconhecimento de voz e texto. Assim que selecionada a forma de coleta, o sistema irá responder de acordo com a escolha, onde: no reconhecimento de caracteres a câmera é ativada, no reconhecimento de voz o microfone é ativado e por fim, no texto o teclado é ativado. Ao proporcionar a leitura via OCR e via voz, o *software* facilita a execução do processo, fazendo com que o tempo gasto no mesmo seja reduzido. Além disso, pode ocorrer redução na incidência de erros. Outra observação que se pode fazer é que, o leiturista tem a liberdade de escolher a forma de leitura que se sente melhor adaptado, fazendo com que a satisfação na utilização da aplicação aumente.

Na seqüência dos processos mencionados até aqui, o aplicativo salva os dados informados na planilha e envia ao servidor, gerando inclusive, um arquivo de integração com o ERP da organização, que posteriormente será importado ao mesmo sem a necessidade do retrabalho de digitação no mesmo.

O aplicativo possui um *backup* de segurança no *cache* do dispositivo a cada consumo de unidade informado, garantindo que, caso a rede não esteja disponível no momento, o que já foi coletado não seja perdido e assim que a rede voltar a ter disponibilidade, os dados são salvos no lado servidor.

4.4 TECNOLOGIAS UTILIZADAS

O desenvolvimento foi realizado utilizando as linguagens de programação PHP e JavaScript. A interface, escrita em HTML e CSS, será complementada pelos *frameworks* Bootstrap e componentes de tela do Onsen UI. Os dados utilizam o SGBD MySQL e o *Web Storage* do próprio HTML5. O aplicativo é híbrido e utiliza o *framework* Onsen UI, juntamente com Cordova e Phonegap, além do complemento do AngularJS em seu desenvolvimento. Como utiliza a linguagem PHP, uma camada de comunicação ao banco de dados com a aplicação híbrida através de *web services* com API's portando arquivos do tipo JSON é utilizada. Para o reconhecimento de voz, foi utilizada a API Google Speech Cloud, já para o reconhecimento de caracteres foi utilizado a API Anyline SDK, que é baseada no algoritmo Tesseract. Todos os conceitos das ferramentas citadas acima estão expostos no capítulo anterior.

Nas seções seguintes são utilizados os artefatos da metodologia ICONIX. Os artefatos da UML que são utilizados neste trabalho serão os seguintes: diagrama de casos de uso, classes, seqüência e de robustez.

4.5 REQUISITOS DO PROJETO

Requisitos são classificados como funcionais e não funcionais. Os requisitos funcionais são declarações dos serviços que o sistema deve fornecer, de como o sistema deve reagir a determinadas entradas e como deve se comportar em situações diversas. Em alguns casos, requisitos funcionais também podem explicar o que o sistema não deve fazer (SOMMERVILLE, 2011).

Os requisitos não funcionais são restrições aos serviços ou funções oferecidos pelo sistema, incluem restrições no processo de desenvolvimento bem como restrições impostas pelas normas (SOMMERVILLE, 2011).

Requisitos de usuário descrevem os funcionais e não funcionais através de linguagem natural, sendo compreensível para os usuários do sistema que não possuem conhecimentos técnicos detalhados. Já os requisitos de sistema são aqueles que descrevem os mesmos de forma mais detalhada, citandos funções, serviços e restrições operacionais do sistema, onde o documento de requisitos do sistema deve fornecer informações exatas do que será implementado (SOMMERVILLE, 2011).

4.5.1 Requisitos Funcionais

A Tabela 7 representa os requisitos funcionais do aplicativo desenvolvido.

Tabela 7 – Requisitos Funcionais.

Código	Descrição
RF-001	O *software* deve manter os usuários.
RF-002	O *software* deve permitir cadastro de leituras.
RF-003	O *software* deve manter configurações dos Condomínios.
RF-004	O *software* deve permitir cadastro de questões em um FAQ para orientar o usuário.
RF-005	O *software* deve permitir cadastro de respostas em um FAQ para orientar o usuário.
RF-006	O *software* deve armazenar dados quando estiver em estado *offline* e sincronizá-los com o servidor quando estiver *online*.

Fonte: O autor, 2016.

Na Tabela 7, o termo “manter” se refere às ações de listar, cadastrar e deletar um objeto.

4.5.2 Requisitos Não-Funcionais

A Tabela 8 representa os requisitos não-funcionais do aplicativo desenvolvido.

Tabela 8 – Requisitos Não-Funcionais.

Código	Descrição
RNF-001	O *software* deve possuir uma interface gráfica de fácil manuseio.
RNF-002	O *software* deve rodar em sistema Android.
RNF-003	O *software* deve estabelecer comunicação com servidor via *Web Service*.
RNF-004	O *software* deve permitir entrada de dados via teclado.
RNF-005	O *software* deve permitir entrada de dados via comandos de voz.
RNF-006	O *software* deve permitir entrada de dados via leitor de caracteres OCR.
RNF-007	Os arquivos de leitura devem obedecer ao layout pré-definido pelo ERP.

Fonte: O autor, 2016.

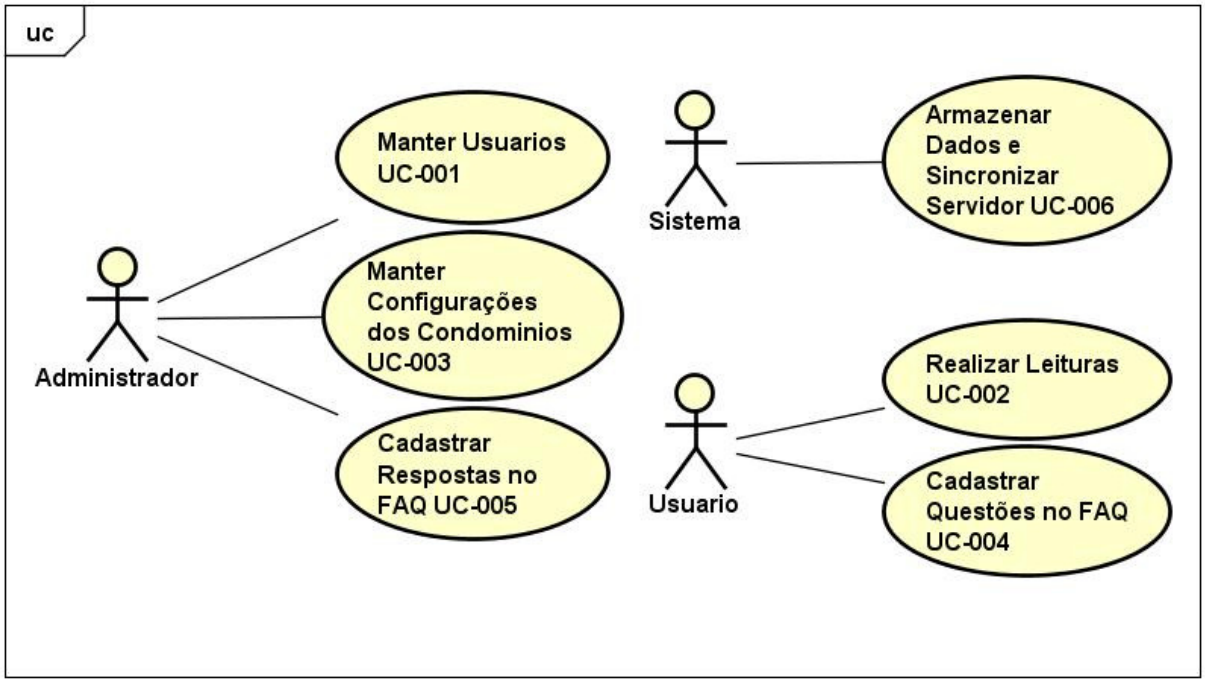
4.6 CASOS DE USO

Os casos de uso são uma técnica de descoberta de requisitos e já se tornaram característica fundamental da linguagem UML. Além da descoberta de requisitos, esse artefato identifica atores envolvidos nas interações (SOMMERVILLE, 2011).

Após ser realizada a apresentação dos requisitos funcionais e não-funcionais que o sistema deverá contemplar, o diagrama de casos de uso pode, enfim, ser criado.

Na Figura 27 estão representados os casos de uso em que os atores “Administrador”, “Usuario” e “Sistema” estão envolvidos no *software*, em seguida, cada caso de uso será descrito com detalhes.

Figura 27 – Casos de Uso que envolvem os atores do *software*.



Fonte: O autor, 2016.

4.6.1 Descrição dos Casos de Uso

Nos itens subseqüentes são expostos os casos de uso baseados a partir dos requisitos funcionais acima mencionados. Na descrição dos casos de uso serão expostas tabelas contendo as seguintes informações:

- Descrição;
- Ator(es);
- Pré-condição;
- Fluxo Principal;
- Fluxo Alternativo e exceções;
- Pós-condições.

Os casos de uso descritos a seguir contemplam todos os requisitos funcionais exibidos na tabela 7, onde em cada um deles é possível visualizar quais são os requisitos envolvidos.

4.6.1.1 Manter Usuários

A Tabela 9 descreve o caso de uso “Manter Usuário”, onde o ator “Administrador” pode cadastrar, alterar e deletar usuários do sistema, a pré-condição é ser um administrador do sistema.

Tabela 9 – Caso de Uso Manter Usuários.

CASO DE USO 001 – MANTER USUÁRIOS (RF-001)	
Descrição	O administrador cadastra usuários para utilização do *software*, sendo possível alterar e remover os mesmos.
Ator	Administrador.
Pré-condição	Ser um administrador do sistema.
Fluxo principal	1. Administrador efetua *login* no sistema 2. Sistema lista todos os usuários cadastrados 3. Sistema exibe formulário para cadastro. 4. Administrador preenche campos obrigatórios. 5. Administrador submete formulário. 6. Sistema retorna lista de usuários.
Fluxo alternativo e exceções	**Item 1:** 1. Administrador não efetua *login* com sucesso. 1. Sistema exibe mensagem de necessidade de *login* correto. **Item 4:** 1. Administrador não preenche todos os campos obrigatórios. 1. Sistema exibe mensagem de erros.
Pós-condição	Usuário Cadastrado.

Fonte: O autor, 2016.

4.6.1.2 Manter Leituras

A Tabela 10 descreve o caso de uso “Realizar Leituras”, onde o ator “Usuário” pode cadastrar as coletas das leituras de gás e/ou água no sistema, a pré-condição é ser um usuário do sistema.

Tabela 10 – Realizar Leituras.

CASO DE USO 002 – REALIZAR LEITURAS (RF-002)	
Descrição	O usuário realiza leituras de gás e/ou água. Para o cadastro de leituras é necessário ter condomínios configurados para o usuário.
Ator	Usuário.
Pré-condição	Ser um usuário do sistema.
Fluxo principal	1. Usuário efetua *login* no sistema 2. Sistema lista Condomínios de sua responsabilidade pendentes de leitura. 3. Usuário acessa cadastro de leitura do Condomínio. 4. Sistema exibe formulário para cadastro. 4.1 Usuário opta por leitura por OCR 4.2 Usuário opa por leitura por Voz 4.3 Usuário opa por leitura por texto 5. Usuário submete formulário. 6. Sistema retorna lista de Condomínios de sua responsabilidade pendentes de leitura.
Fluxo alternativo e exceções	**Item 1:** 1. Usuário não efetua *login* com sucesso. 1. Sistema exibe mensagem de necessidade de *login* correto. **Item 2:** 1. Usuário não possui leitura a fazer no período. 1. Sistema exibe mensagem. **Item 4:** 1. Relógio não tem campos legíveis para leitura OCR. 1. Sistema exibe mensagem. **Item 4:** 1. Voz não é interpretada corretamente. 1. Sistema exibe mensagem.
Pós-condição	Leituras Cadastradas.

Fonte: O autor, 2016.

4.6.1.3 Manter Configurações de Condomínios

A Tabela 11 descreve o caso de uso “Manter Configurações de Condomínios”, onde o ator “Administrador” pode cadastrar, alterar e deletar configurações dos Condomínios no sistema, a pré-condição é ser um administrador do sistema.

Tabela 11 – Caso de Uso Manter Configurações de Condomínios.

CASO DE USO 003 – MANTER CONFIGURAÇÕES DE CONDOMÍNIOS (RF-003)	
Descrição	O administrador informa usuário responsável pelas leituras, se o Condomínio possui água e/ou gás a ser lido e observações relevantes ao mesmo.
Ator	Administrador.
Pré-condição	Ser um administrador do sistema.
Fluxo principal	1. Administrador efetua *login* no sistema 2. Sistema lista todos os Condomínios com suas respectivas configurações. 3. Sistema exibe formulário para alterações. 4. Administrador preenche campos obrigatórios. 5. Administrador submete formulário. 6. Sistema retorna lista de Condomínios.
Fluxo alternativo e exceções	**Item 1:** 1. Administrador não efetua *login* com sucesso. 1. Sistema exibe mensagem de necessidade de *login* correto. **Item 4:** 1. Administrador não preenche todos os campos obrigatórios. 1. Sistema exibe mensagem de erros.
Pós-condição	Configurações Cadastradas.

Fonte: O autor, 2016.

4.6.1.4 Cadastrar Questão FAQ

A Tabela 12 descreve o caso de uso “Cadastrar Questão FAQ”, onde o ator “Usuário” pode cadastrar dúvidas sobre o sistema, a pré-condição é ser um usuário do sistema.

Tabela 12 – Caso de Uso Cadastrar Questão FAQ.

CASO DE USO 004 – CADASTRAR QUESTÃO FAQ (RF-004)	
Descrição	O usuário cadastra perguntas sobre a utilização e funcionamento do *software*.
Ator	Usuário.
Pré-condição	Ser um usuário do sistema.
Fluxo principal	1. Usuário efetua *login* no sistema 2. Sistema lista todas as perguntas e respostas cadastradas. 3. Sistema exibe formulário para cadastro de nova questão. 4. Usuário preenche campos obrigatórios. 5. Usuário submete formulário. 6. Sistema retorna lista de perguntas e respostas.
Fluxo alternativo e exceções	**Item 1:** 1. Usuário não efetua login com sucesso. 1. Sistema exibe mensagem de necessidade de *login* correto. **Item 4:** 1. Usuário não preenche todos os campos obrigatórios. 1. Sistema exibe mensagem de erros.
Pós-condição	Questão Cadastrada.

Fonte: O autor, 2016.

4.6.1.5 Cadastrar Resposta FAQ

A Tabela 13 descreve o caso de uso “Cadastrar Resposta FAQ”, onde o ator “Administrador” pode cadastrar respostas para perguntas pendentes do usuário no sistema, a pré-condição é ser um administrador do sistema.

Tabela 13 – Caso de Uso Cadastrar Resposta FAQ.

CASO DE USO 005 – CADASTRAR RESPOSTA FAQ (RF-005)	
Descrição	O administrador responde as perguntas pendentes de resposta feitas pelo usuário sobre a utilização e funcionamento do *software*.
Ator	Administrador.
Pré-condição	Ser um administrador do sistema.
Fluxo principal	1. Administrador efetua login no sistema 2. Sistema lista todas as perguntas pendentes de resposta. 3. Sistema exibe formulário para inclusão de resposta à questão. 4. Administrador preenche campos obrigatórios. 5. Administrador submete formulário. 6. Sistema retorna lista de perguntas e respostas.
Fluxo alternativo e exceções	**Item 1:** 1. Administrador não efetua *login* com sucesso. 1. Sistema exibe mensagem de necessidade de *login* correto. **Item 4:** 1. Administrador não preenche todos os campos obrigatórios. 1. Sistema exibe mensagem de erros.
Pós-condição	Resposta Cadastrada.

Fonte: O autor, 2016.

4.6.1.6 Armazenar Dados e Sincronizar Servidor

A Tabela 14 descreve o caso de uso “Armazenar Dados e Sincronizar Servidor”, onde o ator “Sistema” fará a verificação do status da conexão bem como o armazenamento dos dados locais e dados servidor.

Tabela 14 – Caso de Uso Armazenar dados e sincronizar servidor.

CASO DE USO 006 – ARMAZENAR DADOS E SINCRONIZAR SERVIDOR (RF-006)	
Descrição	O sistema armazena os dados de leitura de consumo localmente e verifica o status da conexão, caso esteja *online* sincroniza com o servidor e limpa os dados locais, se estiver *offline* os mesmos seguem armazenados localmente até que a conexão esteja disponível.
Ator	Sistema.
Pré-condição	Ter leitura de consumos submetida.
Fluxo principal	1. Sistema verifica que leituras foram submetidas 2. Sistema armazena localmente os dados. 3. Sistema verifica status da conexão. 4. Com o status *online* dados são sincronizados com o servidor. 5. Após sincronização de dados com o servidor, sistema limpa dados locais.
Fluxo alternativo e exceções	**Item 4:** 1. Sistema verifica que conexão é *offline*. 1. Sistema roda em *background* tentativas de sincronização com servidor mesmo com aplicativo fechado, aguardando rede disponível.
Pós-condição	Dados Sincronizados com Servidor.

Fonte: O autor, 2016.

4.7 INTERFACES GRÁFICAS

Na seqüência deste trabalho são exibidas as interfaces do sistema, onde estão expostas as telas do aplicativo móvel e as telas da área administrativa do aplicativo móvel. As interfaces do aplicativo móvel se detêm ao gerenciamento e realização das coletas, já as interfaces da área administrativa são responsáveis pelas configurações do aplicativo, tais como configurações dos Condomínios, gerência de usuários, gerência do FAQ, entre outras. É importante mencionar que as interfaces da área administrativa possuem uma versão web e são visualizadas no *browser*, já as interfaces do aplicativo móvel são visualizadas em *smartphones* e *tables*. Em ambas as situação buscou-se utilizar da melhor forma as heurísticas descritas a seguir.

Para parametrizar a qualidade de uma interface, Nielsen (1995) cita 10 heurísticas que este trabalho procurou seguir, são elas:

- Visibilidade de status do sistema: a interface deve informar o usuário a todo o momento sobre o que está acontecendo, onde o *feedback* precisa ser instantâneo.
- Relacionamento entre a interface do sistema e o mundo real: evitar utilização de palavras técnicas que não fazem sentido para o usuário.
- Liberdade e controle do usuário: quando o usuário desejar desfazer alguma ação ou retornar à página anterior, isso deve ser simples.
- Consistência: criar um padrão de linguagem, facilitando as ações do usuário e o familiarizando com o tipo de comunicação utilizada pelo sistema.
- Prevenção de erros: tentar ao máximo prevenir erros, onde o usuário controle as ações.
- Reconhecimento ao invés de lembrança: evitar fazer o usuário acionar sua memória, a interface deve conter informações suficientes para orientar o mesmo.
- Flexibilidade e eficiência de uso: deve ser fácil de ser utilizado por usuários leigos e ao mesmo tempo ágeis aos avançados, para isso pode ser utilizados recursos como teclas de atalho.

- Estética e *design* minimalista: evitar grandes textos explicativos, a tela deve conter apenas informações indispensáveis ao uso.
- Ajude usuários a reconhecer e sanar erros: as mensagens de erro, quando necessárias, precisam ser simples e claras e não intimidá-lo, sugerindo uma saída para tal.
- Ajuda e documentação: através de boas interfaces podemos evitar a utilização de ajudas, porém uma boa documentação do sistema pode ajudar quando o usuário estiver encontrando dificuldades.

Ainda complementando o contexto de usabilidade, Krug (2008) diz que, usabilidade significa fazer algo que funcione bem e que não exija grande experiência e largos conhecimentos do usuário, onde o mesmo consiga utilizar o sistema para sua devida finalidade, sem frustrações durante esse processo. Krug menciona ainda que as interfaces devem ser auto explicativas, sendo fácil navegar nas mesmas.

4.7.1 Interfaces Gráficas do Aplicativo Móvel

No desenvolvimento das interfaces gráficas do aplicativo móvel, procurou-se aplicar as técnicas citadas nos parágrafos anteriores, buscando-se ser o mais direto e minimalista possível.

As interfaces gráficas do aplicativo móvel contemplam as seguintes telas, que serão apresentadas na seqüência:

- Login do Usuário;
- Tela de Boas-Vindas;
- Lista de Bairros;
- Lista de Condomínios do Bairro;
- Lista de Unidades do Condomínio;
- Menu do Aplicativo;
- FAQ de Perguntas e Respostas.

A Figura 28 é a interface gráfica do login do aplicativo.

Figura 28 – Interface Gráfica de Login do Usuário no Aplicativo.

Fonte: O autor, 2016.

Nessa interface gráfica o usuário realiza sua autenticação no aplicativo de coleta de medidas de gás e água.

Seu funcionamento é simples e baseia-se no preenchimento dos campos “Login” e “Senha”, onde será verificado o correto preenchimento dos mesmos e aberta à sessão do usuário para utilização do sistema. Quando o preenchimento não for correto ou faltarem campos a serem preenchidos, o sistema retornará ao usuário a mensagem adequada.

A Figura 29 é a interface gráfica da tela de boas vindas do usuário ao sistema.

Figura 29 – Interface Gráfica da Tela de Boas Vindas.

Fonte: O autor, 2016.

Nessa interface gráfica o usuário é preparado para a entrada ao sistema, onde um texto de boas vindas é exibido e abaixo dele um botão para continuar é mostrado.

A Figura 30 é a interface gráfica da lista de bairros cadastrados no sistema de gestão da organização. Dentro de cada bairro estão listados os condomínios pertencentes ao mesmo.

Figura 30 – Interface Gráfica da Lista de Bairros.

Fonte: O autor, 2016.

Nessa interface gráfica os bairros são exibidos e, quando clicados, os Condomínios vinculados aos mesmos são exibidos. Além disso, a situação atual do andamento dos bairros no período é exposta ao usuário através de barras de progressão percentuais.

A Figura 31 é a interface gráfica da lista de Condomínios de um determinado bairro com o percentual até então realizado de coletas no período dos condomínios.

Figura 31 – Interface Gráfica da Lista de Condomínios do Bairro.

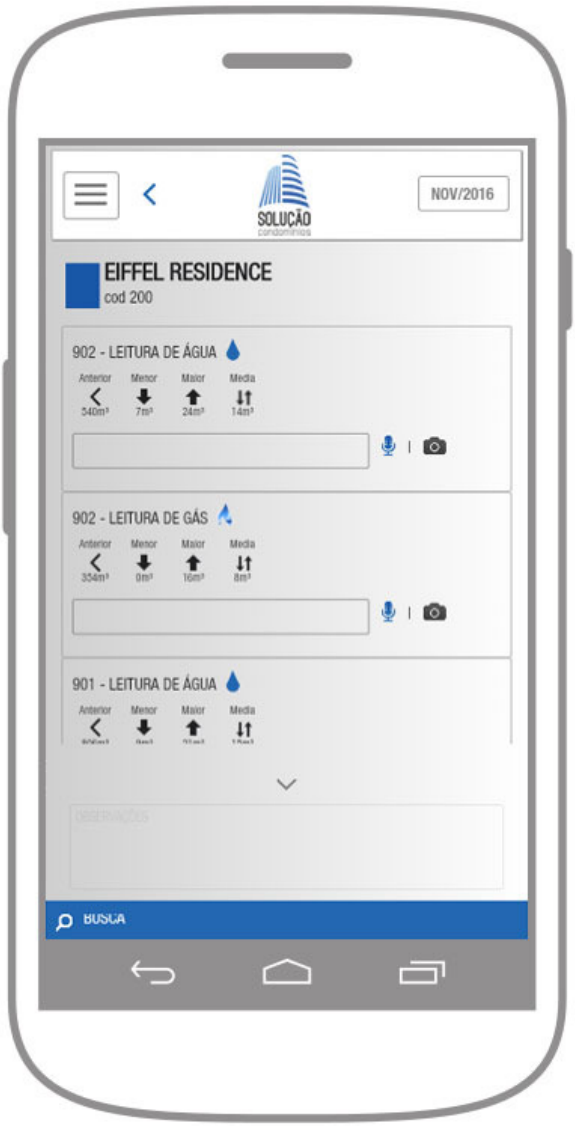
Fonte: O autor, 2016.

Nessa interface gráfica os Condomínios vinculados a um determinado bairro cadastrado no sistema de gestão da organização são exibidos, e a situação atual do andamento dos condomínios no período é exposta ao usuário através de barras de progressão percentuais.

Após a visualização dos Condomínios, o usuário pode clicar nos mesmos e o sistema listará as unidades pertencentes ao Condomínio, sinalizando os realizados e os que ainda necessitam de coletas.

A Figura 32 é o protótipo de interface gráfica da lista das unidades dos Condomínios com coletas a serem realizadas no período atual do aplicativo.

Figura 32 – Interface Gráfica da Lista de Unidades dos Condomínios.



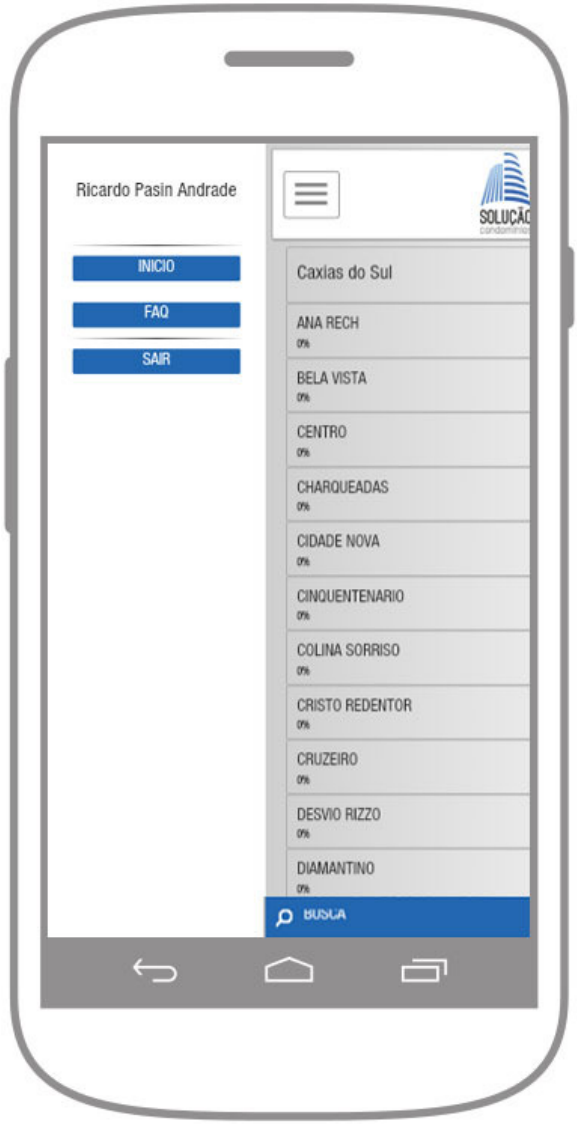
Fonte: O autor, 2016.

Nessa interface gráfica as unidades vinculadas a um determinado Condomínio são exibidas, informando também aos usuários os dados históricos de consumo dessas unidades a fim de situar o leiturista sobre as tendências de consumo das mesmas e ter condições de detectar comportamentos atípicos, tais como vazamentos.

Após a visualização das unidades, o usuário pode informar a coleta das mesmas através do teclado convencional, do comando de voz (utilizando o microfone do dispositivo) ou do leitor de caracteres OCR (utilizando a câmera do dispositivo).

A Figura 33 é o protótipo de interface gráfica de exibição do menu do aplicativo.

Figura 33 – Interface Gráfica do Menu do Aplicativo.



Fonte: O autor, 2016.

Nessa interface gráfica é exibido o menu ao usuário com os recursos disponíveis.

Após a visualização do menu de recursos, o usuário pode navegar entre as telas nessa seção apresentadas.

A Figura 34 é a interface gráfica do FAQ de perguntas e respostas do aplicativo.

Figura 34 – Interface Gráfica do FAQ do Aplicativo.

Fonte: O autor, 2016.

Nessa interface gráfica as perguntas realizadas pelos usuários e respostas inseridas pelo administrador até então ocorridas são listadas a fim de sanar dúvidas sobre a utilização do sistema.

Após a visualização das perguntas e respostas, o usuário pode inserir uma nova questão a ser analisada e respondida pelo administrador do sistema.

4.7.2 Interfaces Gráficas da Área Administrativa do Aplicativo

No desenvolvimento das interfaces gráficas da administração do aplicativo, buscou-se organizar da melhor maneira todas as informações necessárias para a melhor gerência da aplicação.

As interfaces gráficas da área administrativa do aplicativo contemplam as seguintes telas, que serão apresentadas na sequência:

- Login do Administrador;
- Administração dos Usuários do Aplicativo;
- Configurações dos Condomínios do Aplicativo;
- Download dos Arquivos de Integração de Leitura;
- Administração do FAQ do Aplicativo.

A Figura 35 é a interface gráfica do login no sistema já existente na organização que contém o gerenciador do aplicativo.

Figura 35 – Interface Gráfica de Login do Administrador.

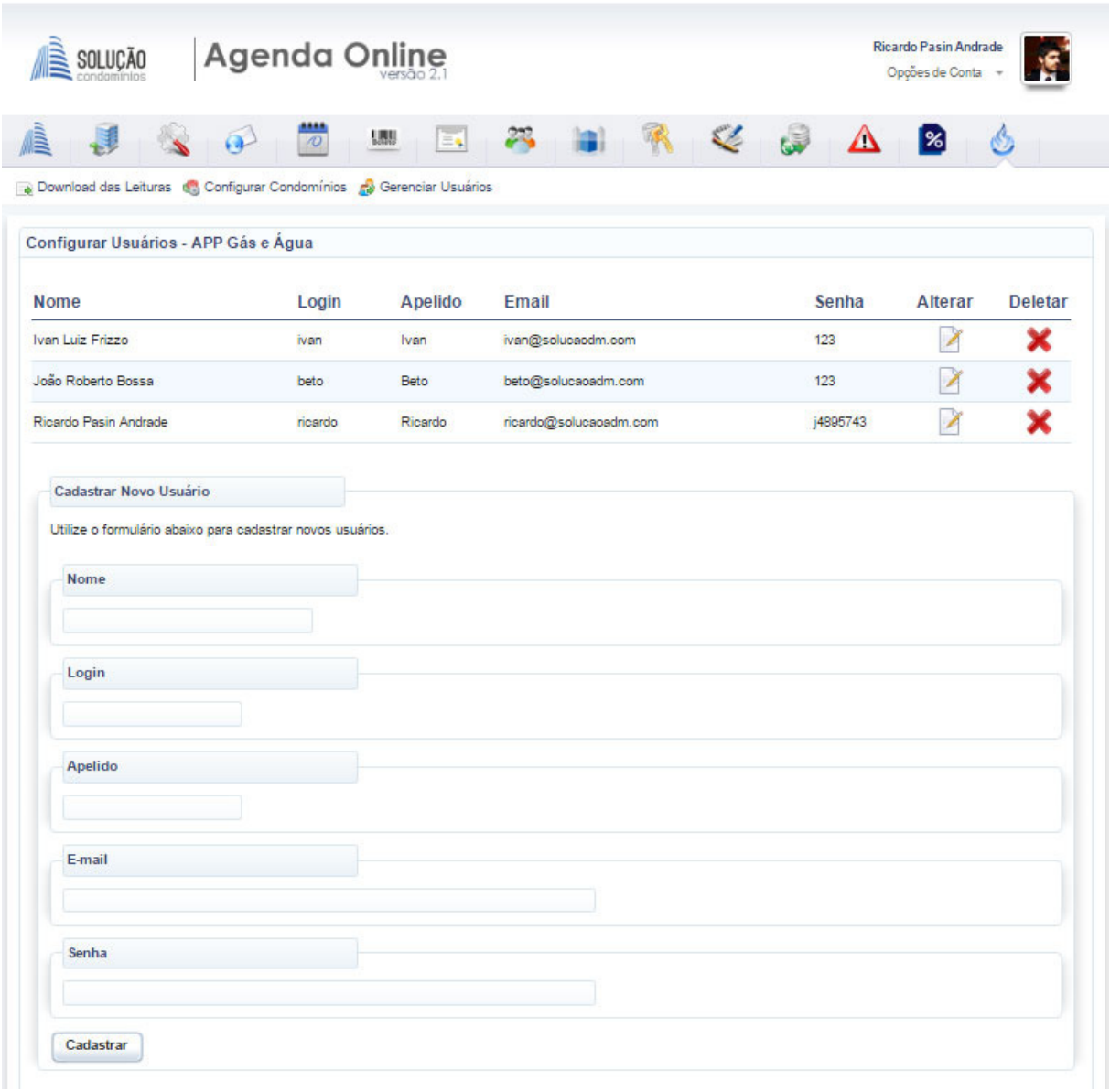
Fonte: O autor, 2016.

Nessa interface gráfica, que é web e não *mobile* assim como em todo o gerenciador de administração do aplicativo, o administrador realiza sua autenticação no sistema comercial da organização, onde dentro dele temos o gerenciador instalado.

Seu funcionamento é simples e baseia-se no preenchimento dos campos “Login” e “Senha”, onde será verificado o correto preenchimento dos mesmos e aberta à sessão do administrador para utilização do gerenciador.

A Figura 36 é o protótipo de interface gráfica das configurações dos usuários do aplicativo.

Figura 36 – Interface Gráfica da Administração dos Usuários do Aplicativo.



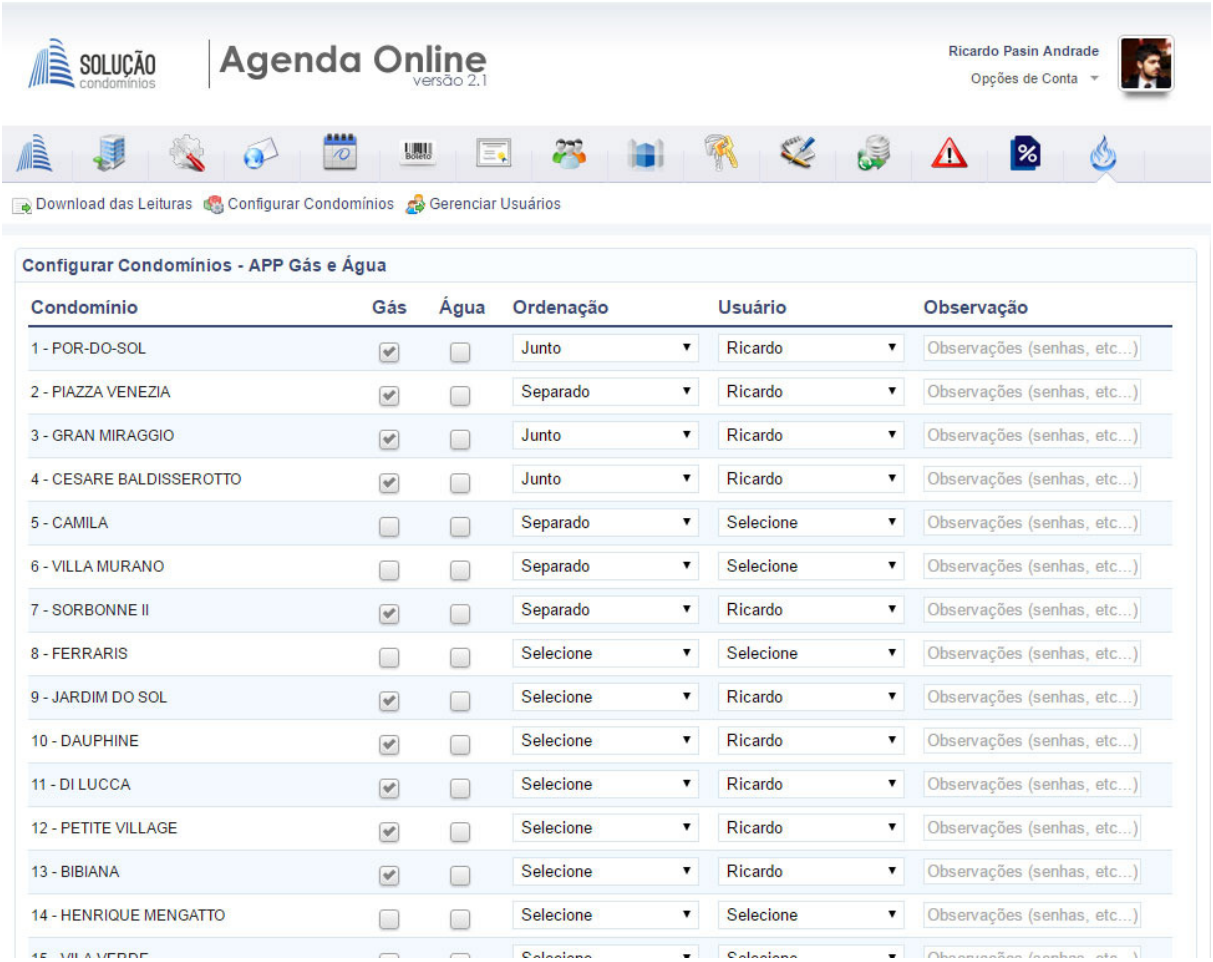
Fonte: O autor, 2016.

Na tela de administração dos usuários, os mesmos são listados para o administrador do aplicativo, sendo possível renomeá-los e excluí-los. Além dessas funções, há também como cadastrar novos usuários.

A lista de usuários com a possibilidade de edições e alterações, assim como todo o gerenciador do aplicativo, são de uso exclusivo do administrador do sistema.

A Figura 37 é a interface gráfica das configurações dos Condomínios.

Figura 37 – Interface Gráfica de Configurações de Condomínios.



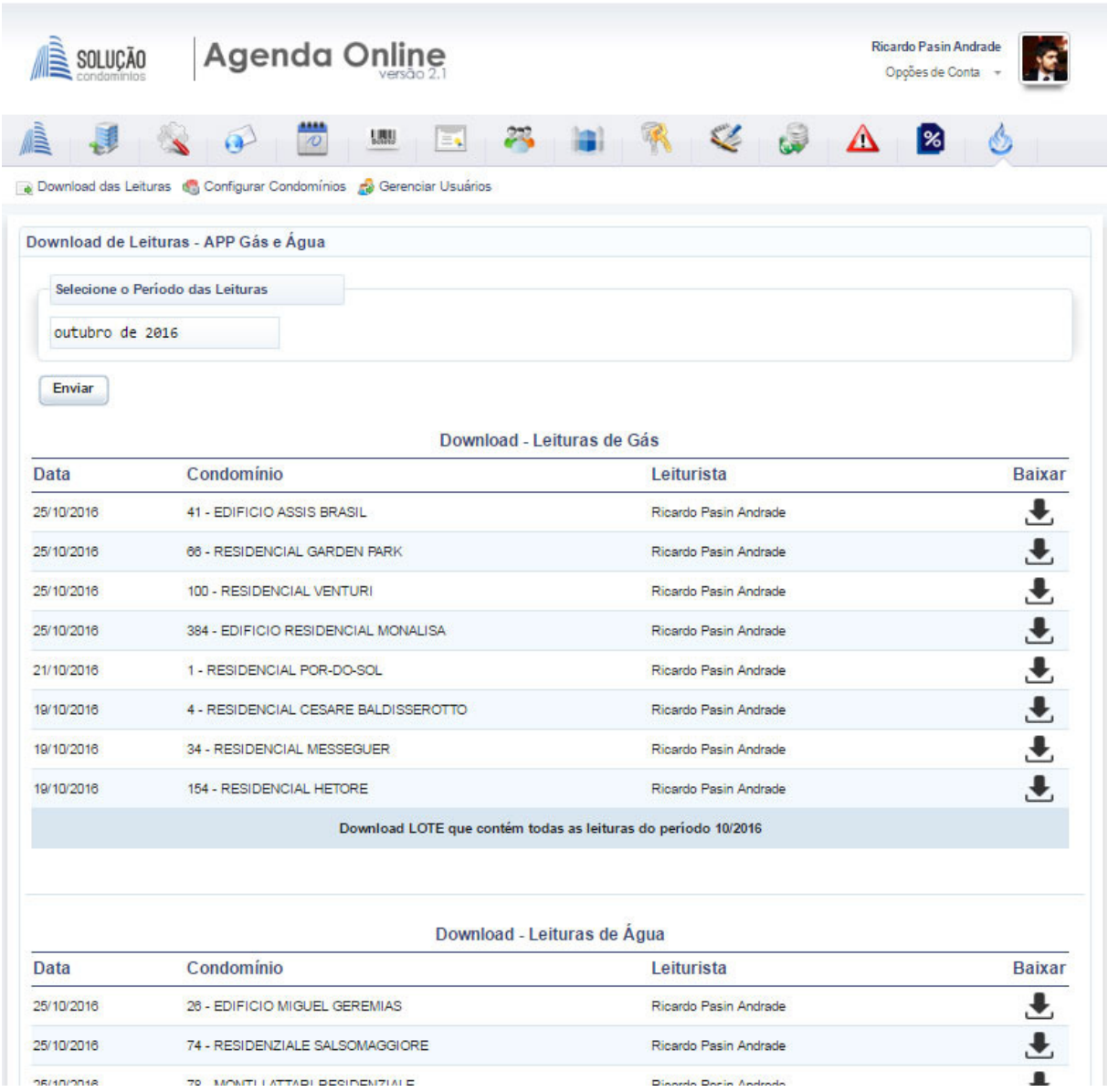
Fonte: O autor, 2016.

Na interface de configuração dos Condomínios, os mesmos são configurados a fim de informar ao sistema se o mesmo possui água ou gás, qual é a ordenação dos relógios, bem como observações relevantes e o responsável pelas coletas.

As configurações dos Condomínios, assim como todo o gerenciador do aplicativo, são de uso exclusivo do administrador do sistema.

A Figura 38 é a interface gráfica onde é disponibilizado o download do arquivo de integração com o ERP das leituras de consumos de gás e água realizadas nos Condomínios em um determinado período.

Figura 38 – Interface Gráfica de Download de Arquivos das Leituras.



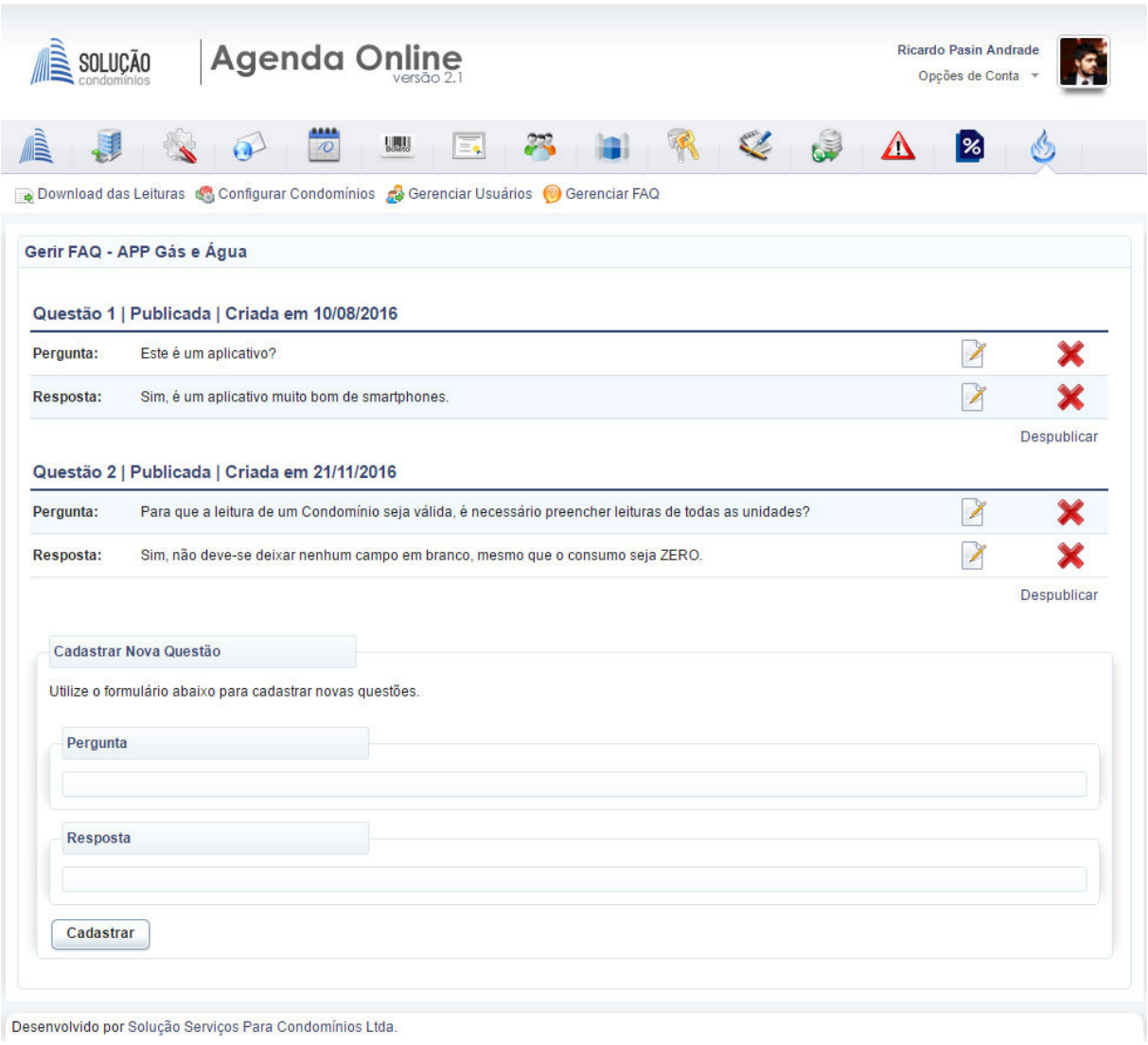
Fonte: O autor, 2016.

Na tela de download dos arquivos de leitura, as leituras de consumo de gás e água são listadas por Condomínio, onde é possível fazer o download dos arquivos de integração com o sistema de gestão da organização. É também possível fazer o download de um arquivo de integração que contém um lote dessas leituras agrupadas, a fim de tornar a integração mais rápida.

A lista de usuários do sistema com a possibilidade de edições e alterações, assim como todo o gerenciador do aplicativo, são de uso exclusivo do administrador do sistema.

A Figura 39 é a interface gráfica onde é gerenciado o FAQ de perguntas e respostas do *software*.

Figura 39 – Interface Gráfica da Gerência do FAQ do Aplicativo.



Fonte: O autor, 2016.

Na interface gráfica de gestão do FAQ, as perguntas e respostas que estão cadastradas no sistema e que contemplam a base de conhecimento do mesmo, são listadas para o administrador do aplicativo, sendo possível alterar, deletar bem como publicar ou despublicar as mesmas. Além dessas funções, há também como cadastrar novas perguntas e respostas.

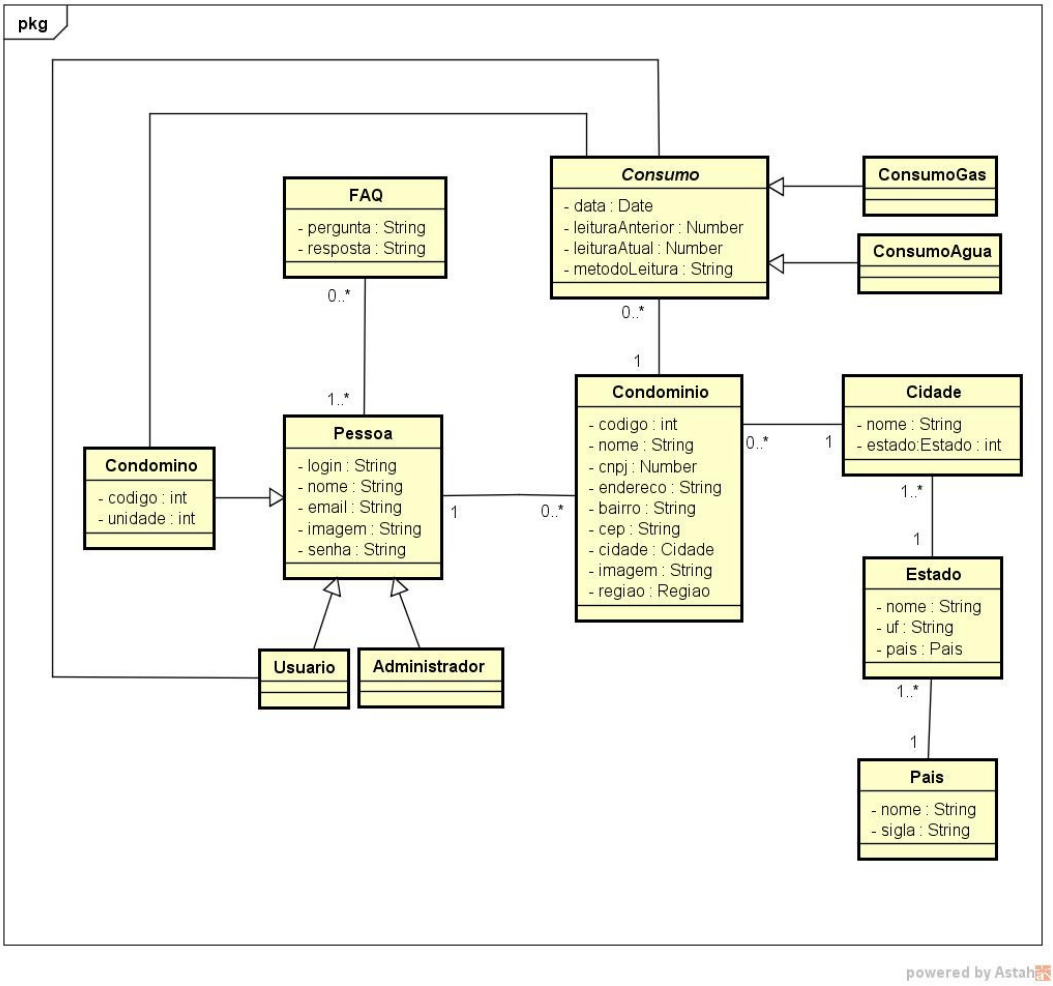
A lista de perguntas e respostas do FAQ com a possibilidade de edições e alterações, assim como todo o gerenciador do aplicativo, são de uso exclusivo do administrador do sistema.

4.8 DIAGRAMA DE CLASSE DE DOMÍNIO

Diagramas de classe são utilizados no desenvolvimento de um modelo de sistema orientado a objetos a fim de mostrar as classes de um sistema e suas respectivas associações. Os objetos representam algo no mundo real, como usuário, médico ou paciente e isso facilita a interpretação do desenvolvedor além de enxergar suas relações (SOMMERVILLE, 2011).

Na Figura 40 é apresentado o diagrama de classe de domínio representando todas as classes envolvidas no projeto.

Figura 40 – Diagrama de Classe de Domínio do Projeto.



Fonte: O autor, 2016.

A tabela 15 apresenta cada classe do projeto com sua respectiva descrição.

Tabela 15 – Classes do projeto com suas respectivas descrições.

Classe	Descrição
Usuario	*Usuario* representa os usuários do aplicativo, ou, em outras palavras, os leituristas que comparecerão aos Condomínios mensalmente para efetuar as leituras.
Administrador	*Administrador* representa os administradores do aplicativo, são responsáveis pelo gerenciamento dos usuários do sistema bem como configurações de usuários e Condomínios.
Condomino	*Condomino* representa os condôminos que integram os Condomínios, a unidade em que o mesmo pertence que conterá a leitura de consumos.
Condominio	*Condominio* representa os condomínios da empresa no aplicativo, estes são oriundos do sistema de ERP da empresa e são atualizados por um serviço de WebService que ocorre periodicamente.
Cidade	*Cidade* representa as cidades do país no aplicativo.
Estado	*Estado* representa os estados do país no aplicativo.
Pais	*Pais* representa os países no aplicativo.
ConsumoGas	*ConsumoGas* representa os registros de coletas de medidas de gás realizadas pelos usuários do aplicativo.
ConsumoAgua	*ConsumoAgua* representa os registros de coletas de medidas de água realizadas pelos usuários do aplicativo.

Fonte: O autor, 2016.

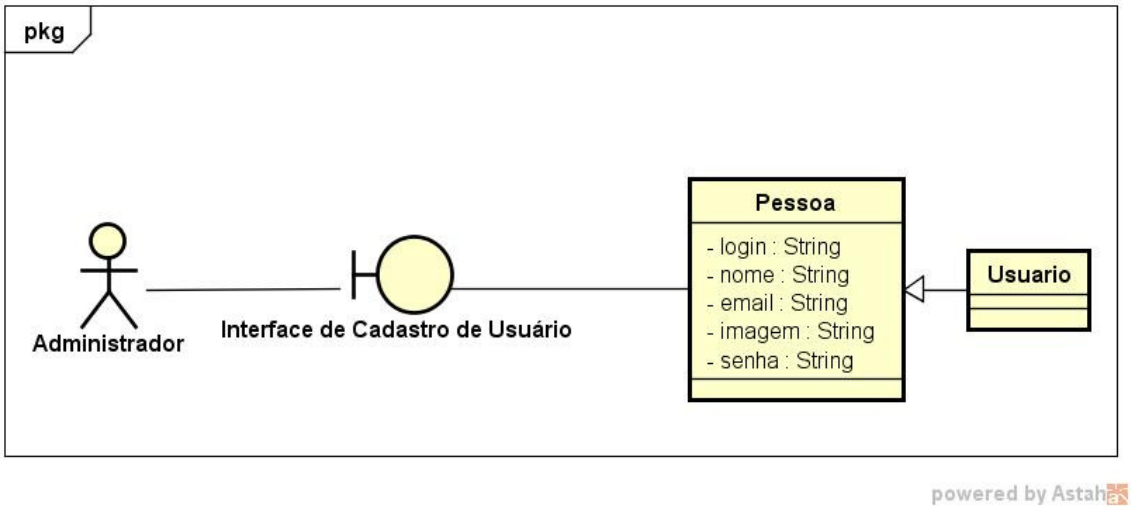
4.9 DIAGRAMAS DE ROBUSTEZ

Para demonstrar a interação das interfaces com os casos de uso e as classes de domínio utilizamos os diagramas de robustez, que seguem nas subseções seguintes.

4.9.1 Cadastro de Usuários

A Figura 41 expõe as associações das classes de domínio na interface gráfica da Figura 36.

Figura 41 – Diagrama de Robustez de Cadastro de Usuários.

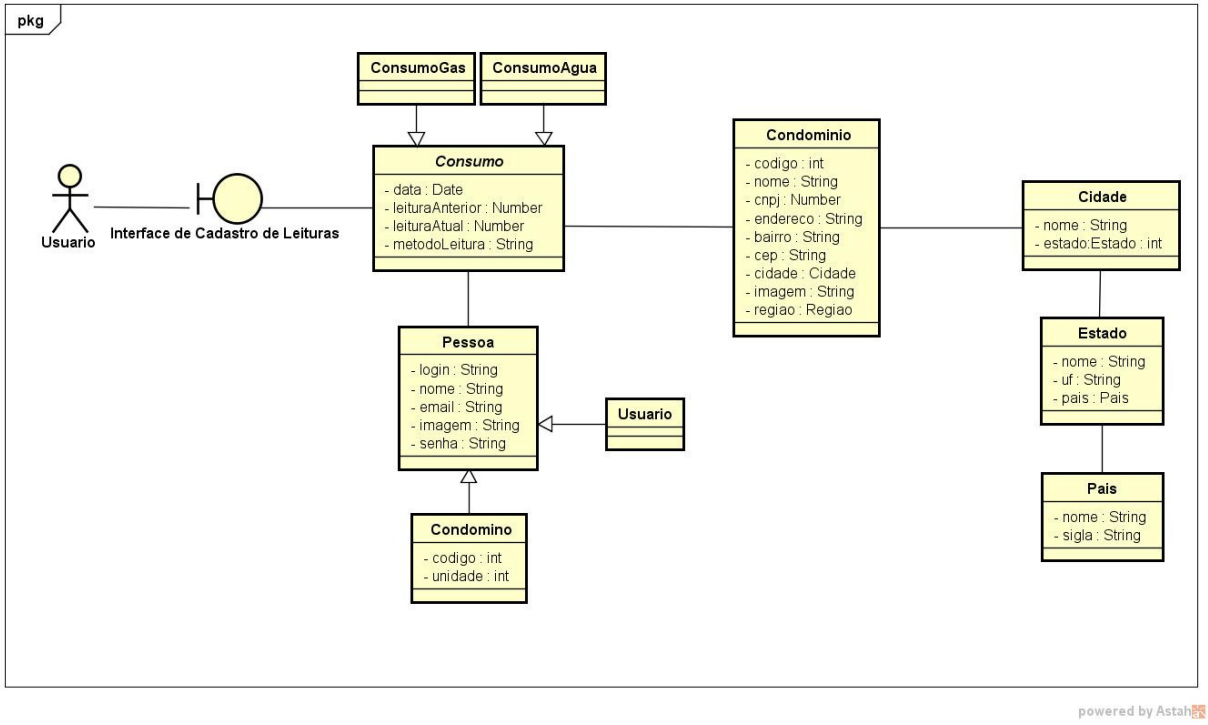


Fonte: O autor, 2016.

4.9.2 Cadastro de Leituras

A Figura 42 expõe as associações das classes de domínio nas interfaces gráficas das Figuras 31, 32 e 33.

Figura 42 – Diagrama de Robustez de Cadastro de Leituras.

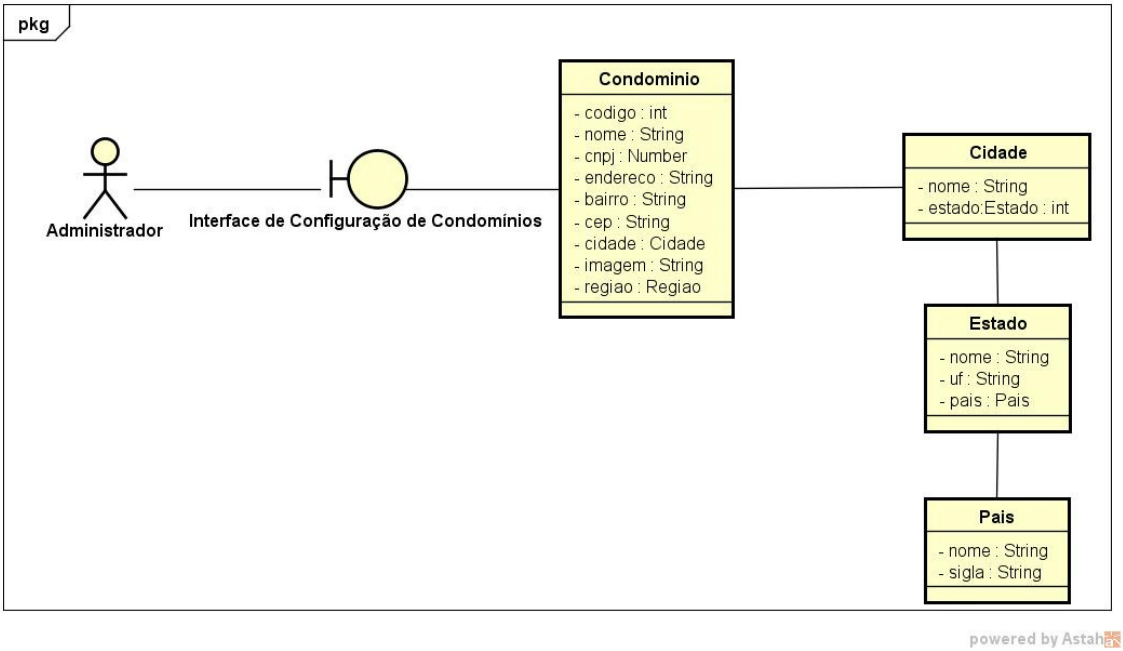


Fonte: O autor, 2016.

4.9.3 Configuração de Condomínios

A Figura 43 expõe as associações das classes de domínio na interface gráfica da Figura 38.

Figura 43 – Diagrama de Robustez de Configurações de Condomínios.

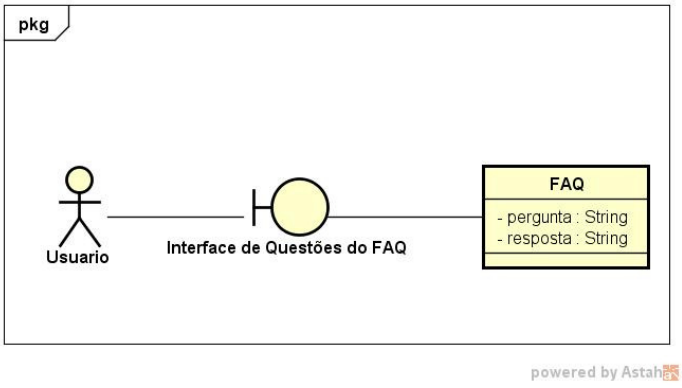


Fonte: O autor, 2016.

4.9.4 Cadastro de Questão no FAQ

A Figura 44 expõe as associações das classes de domínio na interface gráfica da Figura 35.

Figura 44 – Diagrama de Robustez de Cadastro de Questões do FAQ.

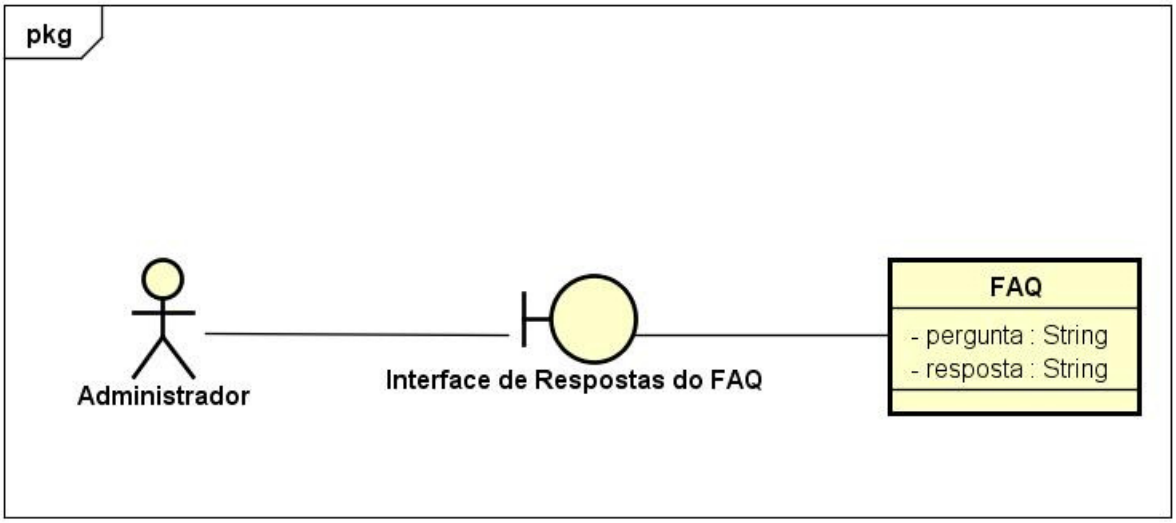


Fonte: O autor, 2016.

4.9.5 Cadastro de Resposta no FAQ

A Figura 45 expõe as associações das classes de domínio na interface gráfica da Figura 35.

Figura 45 – Diagrama de Robustez de Cadastro de Respostas do FAQ.



Fonte: O autor, 2016.

4.10 DIAGRAMAS DE SEQÜENCIA

Para demonstrar a seqüência dos processos dos casos de uso do sistema será utilizado um diagrama de seqüencias para cada caso se uso exposto na subseção 3.6. As seqüências que envolvem o ator administrador, onde a interface é *web* e não dentro do aplicativo, não se utiliza a camada de *WebService*, já nos casos em que a interface é *mobile*, a mesma é utilizada.

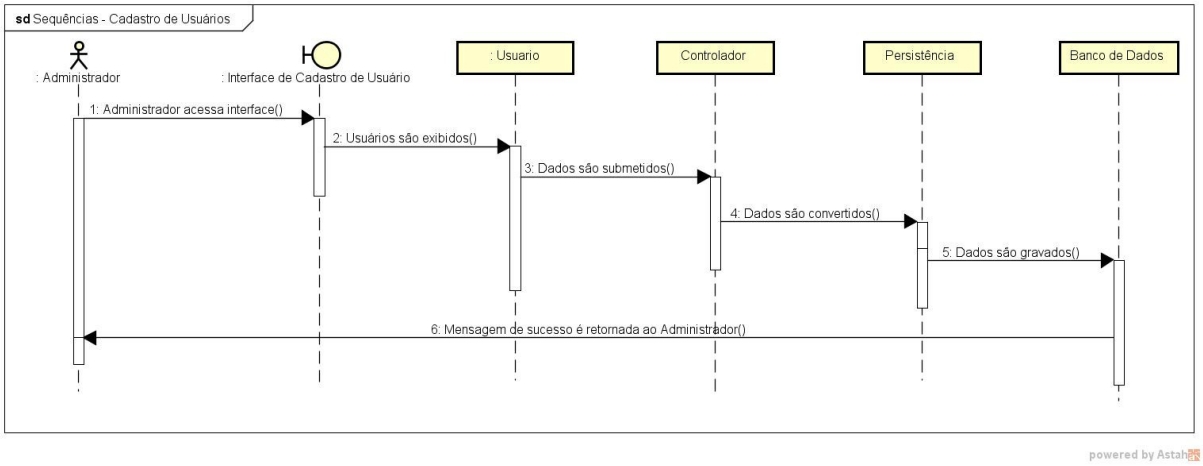
As classes e os métodos representados nos diagramas de seqüência não estão com a nomenclatura literal do código-fonte, essa mudança foi feita visando o melhor entendimento do leitor deste trabalho.

4.10.1 Seqüencia de Cadastro de Usuários

A Figura 46 demonstra o fluxo de cadastro de um novo usuário no sistema. O administrador acessa a listagem de usuários, após isso o sistema exibe o formulário para efetuação de cadastro, onde o ator deverá preencher os campos

obrigatórios e submeter o formulário, o sistema redirecionará o mesmo de volta para a listagem de usuários cadastrados, exibindo inclusive o recém inserido.

Figura 46 – Diagrama de Seqüencia de Cadastro de Usuário.

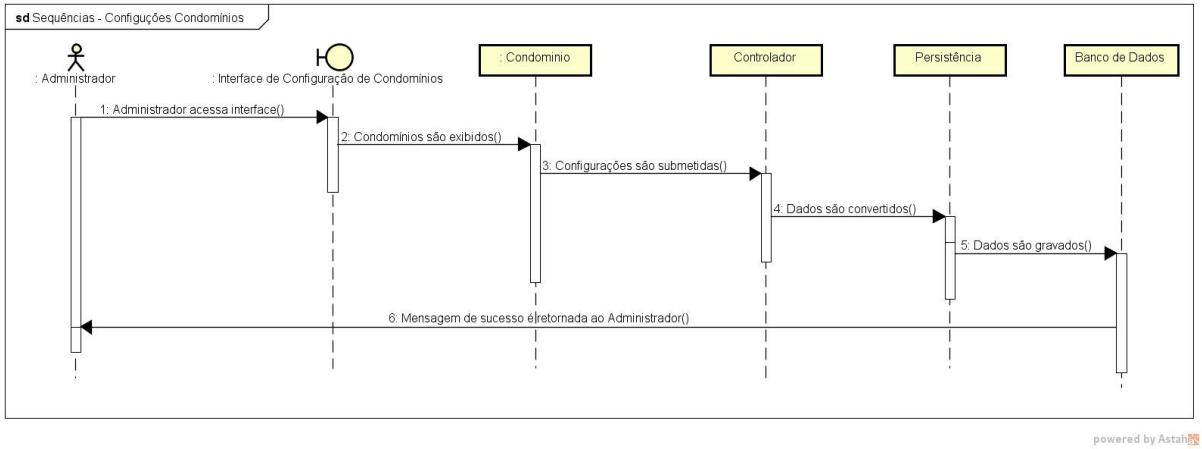


Fonte: O autor, 2016.

4.10.2 Seqüencia de Configuração de Condomínios

A Figura 47 demonstra o fluxo de configurações das características dos condomínios no que diz respeito às leituras de consumo no sistema. O administrador acessa a listagem de Condomínios, após isso o sistema exibe o formulário para alteração das configurações dos mesmos, onde o ator deverá preencher os campos obrigatórios e submeter o formulário, o sistema redirecionará o mesmo de volta para a listagem de Condomínios, exibindo a mesma atualizada.

Figura 47 – Diagrama de Seqüencia de Configuração de Condomínios.

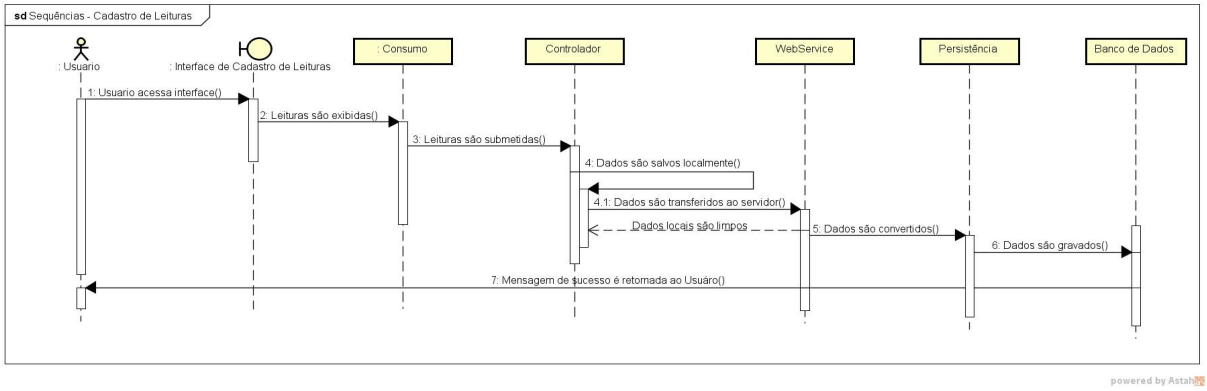


Fonte: O autor, 2016.

4.10.3 Seqüencia de Cadastro de Leituras

A Figura 48 demonstra o fluxo de cadastro de leituras de consumos de água e gás das unidades dos Condomínios cadastrados no sistema. O usuário acessa a interface de coletas de consumo, após isso o sistema exibe o formulário, onde o ator deverá informar os consumos das unidades e submeter o formulário. Após a submissão do formulário, o sistema salvará os dados localmente e fará a verificação da disponibilidade da rede, assim que disponível sincronizará com o servidor e limpará os dados locais. Após os procedimentos descritos, o sistema exibirá ao mesmo a mensagem de sucesso de gravação dos dados.

Figura 48 – Diagrama de Seqüencia de Cadastro de Leituras.

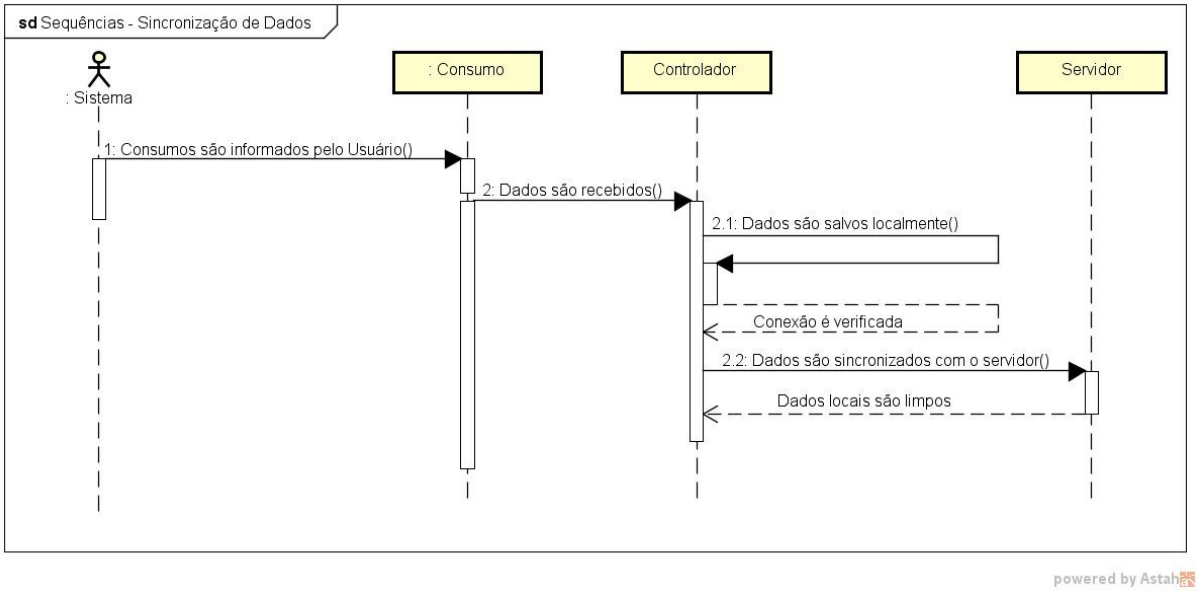


Fonte: O autor, 2016.

4.10.4 Seqüência de Armazenamento de Dados e Sincronização Servidor

A Figura 49 demonstra o fluxo detalhado de parte do que já foi ilustrado no diagrama anterior, onde o sistema após a submissão do formulário de coletas de consumos fará a alocação local dos dados e posteriormente a sincronização com o servidor havendo conexão disponível para tal. Após realizado o processo descrito, os dados locais são limpos.

Figura 49 – Diagrama de Seqüência de Armazenamento de Dados e Sincronização Servidor.

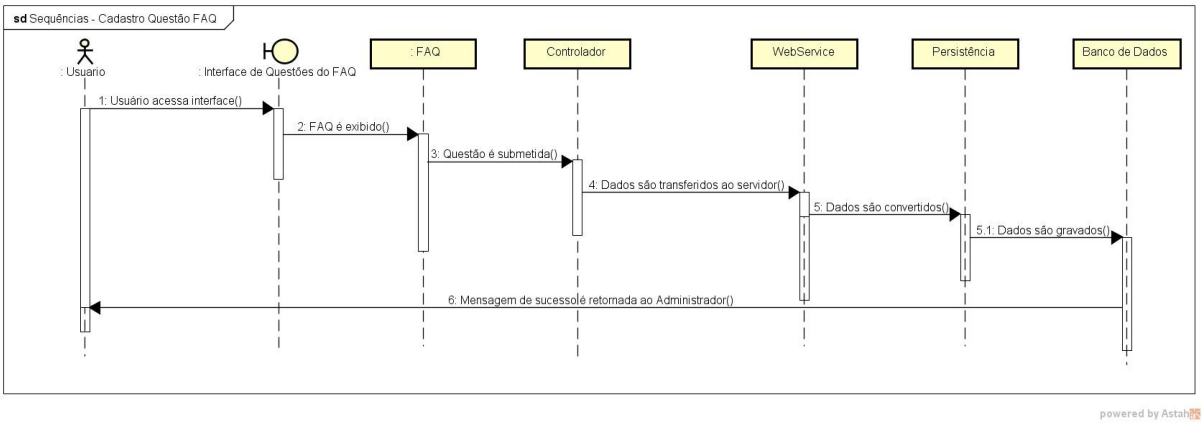


Fonte: O autor, 2016.

4.10.5 Seqüencia de Cadastro de Questões no FAQ

A Figura 50 demonstra o fluxo de cadastro de questões no FAQ de perguntas e respostas do sistema. O usuário acessa a lista de perguntas e respostas, após isso o sistema exibe o formulário, onde o ator deve preenchê-lo com a sua pergunta, o sistema exibirá ao mesmo a lista de questões, inclusive com a recém cadastrada.

Figura 50 – Diagrama de Seqüencia de Cadastro de Questões no FAQ.

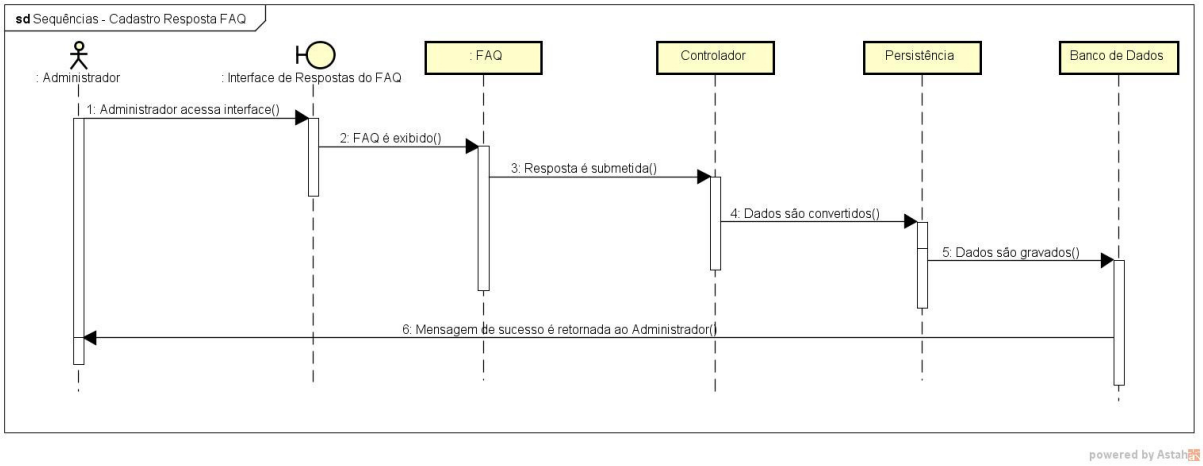


Fonte: O autor, 2016.

4.10.6 Seqüencia de Cadastro de Respostas no FAQ

A Figura 51 demonstra o fluxo de cadastro de respostas no FAQ de perguntas e respostas do sistema. O administrador acessa a lista de perguntas e respostas, após isso o sistema exibe o formulário, onde o ator deve preenchê-lo com a sua resposta, o sistema exibirá ao mesmo a lista de questões, inclusive com a resposta recém cadastrada.

Figura 51 – Diagrama de Seqüencia de Respostas no FAQ.

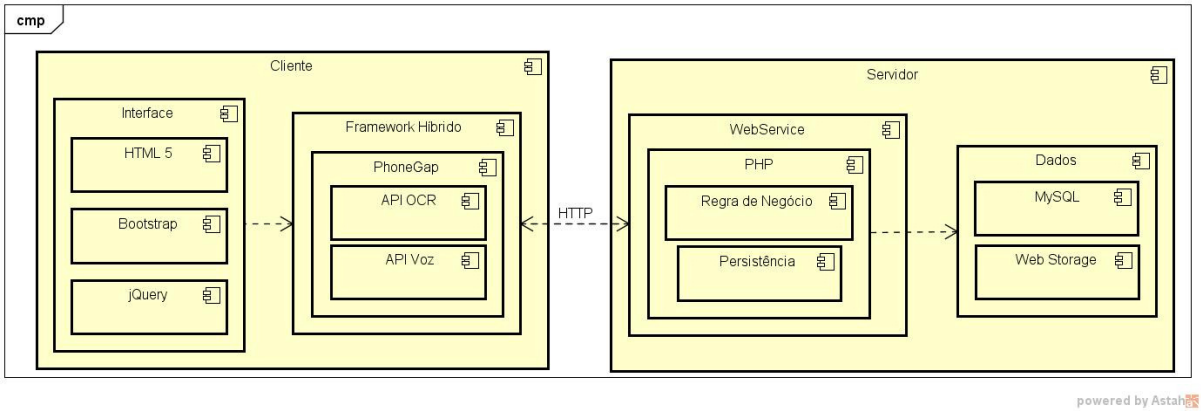


Fonte: O autor, 2016.

4.11 ARQUITETURA DO SOFTWARE

Para o desenvolvimento do sistema foi definida uma arquitetura de software, seu diagrama segue exibido na Figura 52.

Figura 52 – Diagrama de Arquitetura do Software.



Fonte: O autor, 2016.

No lado cliente, temos a camada de interface e a camada de negócio, onde na camada de interface são utilizados o HTML5, o *framework* Bootstrap e Onsen UI, além da biblioteca jQuery, já no negócio temos o *framework* de desenvolvimento híbrido PhoneGap, contendo as API's de OCR e de reconhecimento de voz.

No lado servidor, temos as camadas *WebService* e a camada de dados. Dentro da primeira temos o PHP com a regra de negócio e a persistência, que se comunica com o lado cliente através da utilização de *WebServices* via protocolo HTTP (*Hypertext Transfer Protocol*). A camada de dados, que é responsável pelo armazenamento dos mesmos, contém o SGBD (Sistema de Gerenciamento de Banco de Dados) MySQL que faz a comunicação com o PHP, já os dados locais que visam a efetuação do *backup* de segurança das leituras utiliza o recurso de *Web Storage* do HTML5 e ficam alocadas ali temporariamente, até que os dados sejam remetidos ao MySQL, assim que isso ocorrer o *Local Storage* é limpo.

É importante salientar que os dados dos Condomínios, tais como bairro, unidades e consumos históricos das unidades são provenientes do sistema de gestão da organização. A API de comunicação desenvolvida realiza a integração entre o ERP e o APP desenvolvido.

Após a apresentação da proposta de solução e do desenvolvimento realizado neste capítulo, o capítulo 5, na seqüência, trata do processo de implantação do *software* na organização, apresentando um roteiro da maneira como foi realizada.

5 IMPLANTAÇÃO

A implantação na empresa foi realizada num tempo médio de um mês. Primeiramente, a empresa que desenvolveu o *software* de gestão da Solução foi contatada a fim de acertar detalhes quanto à integração do aplicativo com o ERP. Foi desenvolvida uma interface dentro do sistema de gestão, onde o colaborador que recebe as leituras realizadas via aplicativo pode fazer o *upload* dos arquivos, inclusive em lotes, contendo centenas de Condomínios num só arquivo, e, em poucos segundos descarregar essas leituras dentro do sistema.

A Figura 53 mostra a localização desta interface de integração do ERP com o aplicativo.

Figura 53 – Menu da Integração Dentro do ERP.

Fonte: O autor, 2016.

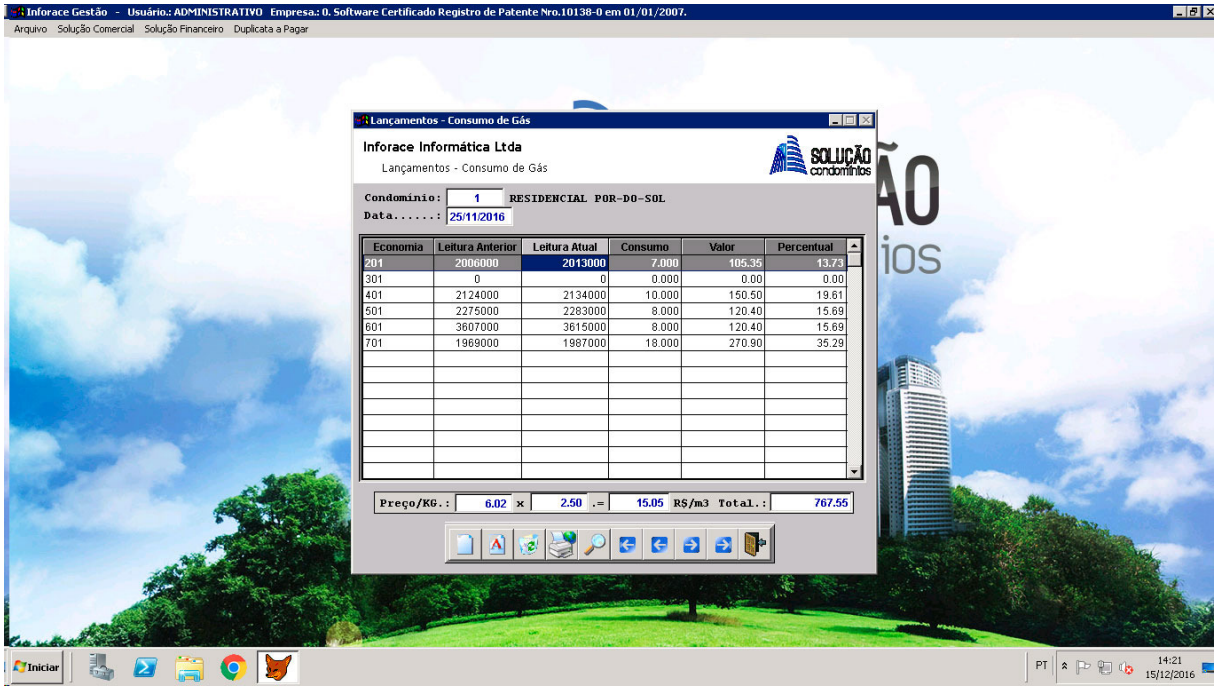
Já a figura 54, mostra o local onde os arquivos de integração gerados pelo aplicativo podem ser depositados.

Figura 54 – Local Onde Arquivo de Integração é Inserido no ERP.

Fonte: O autor, 2016.

Na Figura 55 é possível visualizar uma planilha de rateio de leitura de gás após a descarga de um arquivo de integração.

Figura 55 – Planilha de Rateio de Gás Após Descarga de Arquivo no ERP.



Fonte: O autor, 2016.

Além da negociação com a empresa desenvolvedora do *software* de gestão, esse tempo de implantação foi utilizado para que fossem feitos testes de ajustes no software a fim de deixá-lo o mais funcional possível. Ocorreram alguns contratempos, sendo um dos mais marcantes a listagem de unidades que a API trás, pois a mesma duplicava algumas unidades. O problema foi identificado como sendo unidades que continham inquilinos e corrigido, exibindo apenas uma vez ao usuário.

Após o término do desenvolvimento e dos testes na aplicação, foi realizado um pequeno treinamento com os colaboradores envolvidos no processo, onde foi apresentada a área administrativa ao pessoal do escritório e o aplicativo no dispositivo móvel, já em campo, aos leituristas. O treinamento obteve êxito e rapidamente a equipe conseguiu se adaptar ao sistema em questão.

Além do treinamento e dos testes mencionados acima, foram realizadas entrevistas com os colaboradores envolvidos no processo, onde os detalhes serão ampliados no capítulo a seguir.

No processo de implantação, houve um total apoio da direção da empresa bem como do pessoal envolvido no processo, como colaboradores envolvidos diretamente do escritório e leituristas que irão utilizar a nova aplicação. Houve uma conscientização de que o processo ficará mais ágil e otimizado e que os transtornos comuns de uma implantação de software é justificável pelo resultado final.

Após as percepções da implantação serem expostas, segue na seção 6 a seguir as considerações finais deste trabalho.

6 CONSIDERAÇÕES FINAIS

O presente trabalho inicialmente contribuiu para o entendimento do processo de coletas de medidas de consumos de gás e água nas unidades dos Condomínios da empresa Solução Condomínios. Posteriormente, visando à melhoria do processo, foi proposto um software que realizasse o mesmo de forma mais rápida e visando redução na incidência de erros.

Para viabilizar o desenvolvimento da solução proposta, foram levantadas todas as tecnologias existentes que de alguma forma viessem a contribuir com o sucesso da mesma. Após esse levantamento, foi utilizada uma metodologia da engenharia de software a fim de descrever cada etapa do desenvolvimento do software buscando a qualidade do mesmo, focando-se também nas premissas de usabilidade.

Durante o desenvolvimento do presente trabalho, foram encontradas dificuldades, tais como o desconhecimento de tecnologias de OCR e voz, bem como nenhuma experiência em desenvolvimento de aplicativos para dispositivos móveis, o que acarretou num maior tempo para compreensão dos conteúdos desenvolvidos. Foi necessário também estudar as formas de desenvolvimento de aplicações móveis e as ferramentas livres disponíveis para viabilizar o resultado final.

Pode-se perceber que, após o desenvolvimento do software, e com o sucesso da implantação, o processo ficou totalmente informatizado, diferente do que era praticado até então, além de torná-lo mais ágil e com maiores informações de relevância ao colaborador que efetua as leituras dos consumos.

Em paralelo ao desenvolvimento do aplicativo, a empresa que produziu o *software* de gestão da Solução foi consultada sobre a possibilidade de realizar uma integração entre as leituras realizadas com a nova ferramenta e o ERP. Após algumas conversas via e-mail e reuniões, a desenvolvedora do ERP definiu um layout de arquivo de integração para que a mesma fosse possível, este layout pode ser visto no ANEXO A, ao final deste trabalho.

Ao final do desenvolvimento, foram realizados testes, tanto de usabilidade quanto de integridade das informações exibidas e coletadas, onde se constatou uma melhora significativa na qualidade das informações bem como no tempo de execução do processo de leitura de consumos.

Para se constatar essa melhora citada no parágrafo anterior, foram realizadas entrevistas com um dos leituristas da organização e com um colaborador envolvido no processo de comunicação das leituras realizadas via APP com o ERP da empresa. A primeira entrevista foi realizada visando ter a opinião do leiturista sobre as facilidades e dificuldades do software. Já a segunda, foi realizada visando ter a opinião do leiturista sobre a redução dos retrabalhos e do tempo de lançamento dos consumos no *software* de gestão da empresa. A entrevista com o leiturista está exposta no APÊNDICE A. Já a entrevista com o colaborador está exposta no APÊNDICE B, ambas ao final deste trabalho.

O desenvolvimento deste trabalho teve como objetivo não só a melhora do processo de coleta de leituras de consumo em Condomínios, mas também a aquisição de conhecimento em tecnologias de desenvolvimento móvel bem como reconhecimento de voz e caracteres, visto que são tendências de sistemas inteligentes no futuro.

O autor deste trabalho espera que, com este relatório possa contribuir para que pessoas interessadas pelos temas abordados encontrem as informações que buscam e que agreguem conhecimentos.

7 REFERÊNCIAS

ANGULARJS, Developer Guide. Disponível em: <https://docs.angularjs.org/guide/introduction>. Acessado em 29 abr. 2016.

ANYLINE SDK. Disponível em: <https://www.anyline.io>. Acessado em 03/09/2016.

BOOTSTRAP. Disponível em: <http://www.getbootstrap.com.br>. Acessado em 09 jun. 2016.

BRADSKI, G; KAEHLER, A. Computer Vision with the OpenCV Library, 2008.

CAMPOS, Augusto César. Open Source é..., 2009. Disponível em: <http://www.linuxnewmedia.com.br/images/uploads/pdf_aberto/LM_53_14_15_01_col_augusto.pdf>. Acessado em 05 abr. 2016.

CARNEGIE MELLON UNIVERSITY. What is speech recognition? Disponível em: <http://www.speech.cs.cmu.edu/comp.speech/Section6/Q6.1.html>. Acessado em 05 Abr. 2016.

CHEN, D.; LUETTIN, J.; SHEARER, K.; 2000; A Survey of Texto Detection and Recognition. Disponível em: <http://www.cs.cmu.edu/~datong/survey.pdf>. Acessado em: 27 mar. 2016.

CHERIET, M. Character recognition systems: a guide for students and practitioners. USA: Wiley, 2007. Disponível em: <http://people.mokk.bme.hu/~kornai/OCR/Irodalom/Cheriet_Character_Recognition_Systems__A_Guide_for_Students_and_Practitioners.pdf>. Acessado em: 28 mar. 2016.

EIKVIL, Line. OCR Optical Character Recognition. 1993.

EXPERTS EXCHANGE. Battle of The Virtual Assistants, 2015. Disponível em: <http://pages.experts-exchange.com/virtualassistants.html>. Acessado em 10 abr. 2016.

FLANAGAN, David. JavaScript: o guia definitivo. 4.ed. Porto Alegre: Bookman, 2004.

GARTNET. Gartner Says Worldwide Smartphone Sales Grew 9.7 Percent in Fourth Quarter of 2015, 2016. Disponível em: <http://www.gartner.com/newsroom/id/3215217>. Acessado em 28 abr. 2016.

GIT. Disponível em: <http://git-scm.com/>. Acessado em 02 mai. 2016.

GOHR, Andreas. Linux OCR Software Comparison. Berlim, Alemanha, 2010. Disponível em: <http://www.splitbrain.org/blog/2010-06/15-linux_ocr_software_comparison/>. Acesso em 31 mar. 2016.

GOOGLE DEVELOPERS. Developer's guide - client-side storage (web storage). Technical report, 2012. Disponível em: <https://developers.google.com/web-toolkit/doc/ latest/DevGuideHtml5Storage>. Acessado em 25 de abr. 2016.

Juang, B.H. and Lawrence R. Rabiner. Automatic Speech Recognition - A Brief History of the Technology Development. USCB. 2004. Disponível em: <http://www.idi.ntnu.no/~gamback/teaching/TDT4275/literature/juang_rabiner04.pdf>. Acessado em 05 de Abr. 2016.

JUNTUNEN, Antero; JALONEN, Eetu; LUUKKAINEN, Sakari. HTML 5 in Mobile Devices – Drivers and Restraints in 46th Hawaii International Conference on System Sciences, 2013.

KHOSHAFIAN, S.; BAKER, A. B. Multimedia and Imaging Databases. Morgan Kaufmann, 1996.

KORF, Mario and OKSMAN, Eugene. Native, HTML5, or Hybrid: Understanding Your Mobile Application Development Options, 2015. Disponível em: <https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options> Acessado em 22 abr. 2016.

KRUG, Steve. Não me faça pensar. 2ª edição. Rio de Janeiro: Alta Books, 2008.

LEONDES, C. T. Image Processing and Pattern Recognition, 1998.

MITHE, R.; SUPRIYA, I; DIVEKAR, N. Optical Character Recognition: International Journal of Recent Technology and Engineering (IJRTE), 2013.

MYSQL. Disponível em: <http://www.mysql.com>. Acessado em 19 mai. 2016.

NIELSEN, J. Ten Usability Heuristics. 1995. Disponível em: <http://www.useit.com/papers/heuristic/heuristic_list.html> Acessado em 14 jun. 2016.

NODEBR, O que é Node.js?, 2013. Disponível em: <http://nodebr.com/o-que-e-node-js/>. Acessado em 02 mai. 2016.

NOETICFORCE, 10 Best Hybrid Mobile App UI Frameworks: HTML5, CSS and JS. Disponívell em: <http://noeticforce.com/best-hybrid-mobile-app-ui-frameworks-html5-js-css>. Acessado em 29 abr. 2016.

NVIDIA, What is GPU Accelerated Computing?, 2012. Disponível em: <http://www.nvidia.com/object/what-is-gpu-computing.html> Acessado em 28 abr. 2016.

ONCEDEV, Por que escolher o Xamarin?, 2014. Disponível em: <http://blog.oncedev.com/mobile/2014/06/25/porque-escolher-o-xamarin/>. Acessado em 19 mai. 2016.

ONSEN UI. Disponível em: <https://onsen.io/>. Acessado em 31 ago. 2016.

PHONE ARENA. Testing shows Siri beating out Google Now and Cortana for accuracy, user satisfaction and more, 2015. Disponível em: < http://www.phonearena.com/news/Testing-shows-Siri-beating-out-Google-Now-and-Cortana-for-accuracy-user-satisfaction-and-more_id75601>. Acessado em 10 abr. 2016.

PHONEGAP. Disponível em: <http://phonegap.com/>. Acessado em 02 mai. 2016.

PHP, O que é PHP?, 2016. Disponível em: <http://php.net/manual/pt_BR/intro-whatis.php>. Acessado em 19 mai. 2016.

PHP, Unsupported Historical Releases, 2016. Disponível em: <https://secure.php.net/releases/>. Acessado em 19 mai. 2016.

PHPMYADMIN. Disponível em: < https://www.phpmyadmin.net>. Acessado em 19 mai. 2016.

ROLNITZKY, David. To mobile web app or not to mobile web app?, 2010. Disponível em: <http://www.rolnitzky.com/artifacts/mobile-v-web-app.pdf>. Acessado em 29 abr. 2016.

ROSÁRIO, João Maurício. Princípios de mecatrônica. São Paulo: Patrience Hall, 2005.

ROSENBERG, Doug and STEPHENS, Matt and COLLINS-COPE, Mark. Agile Development with ICONIX Process: People, Process and Pragmatism. 2005.

ROSENBERG, Doug and STEPHENS, Matt. Use Case Driven Object Modeling with UML. 2007.

SOMMERVILLE, Ian. Engenharia de Software. 9ª. Edição. São Paulo: Pearson, 2011.

TAPPAREL, Yannick. Touch-Optimised Mobile Interface for Invenio Digital Library, 2013. Disponível em: <https://cds.cern.ch/record/1596242/files/CERN-THESIS-2013-119.pdf>. Acessado em 22 abr. 2016.

TECMUNDO, O que é API?, 2016. Disponível em: <http://www.tecmundo.com.br/programacao/1807-o-que-e-api-.htm>. Acessado em 13 jun. 2016.

VISION MOBILE, Graphs From the Latest State of the Developer Nation Report. Disponível em: <https://www.visionmobile.com/graphs-from-the-latest-state-of-the-developer-nation-report/>. Acessado em 20 abr. 2016.

W3C, Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008. Disponível em: <https://www.w3.org/TR/REC-xml/>. Acessado em 19 mai. 2016.

W3C, JSON Tutorial. Disponível em: <http://www.w3schools.com/json/>. Acessado em 19 mai. 2016.

W3C. HTML5 Curso W3C Escritório Brasil. Disponível em: <http://www.w3c.br/pub/Cursos/CursoHTML5/html5-web.pdf> Acessado em: 28 abr. 2016.

WILFRED, Ronald. Siri is more accurate than Google Now and Cortana: Report, 2015. Disponível em: <http://mobilesiri.com/siri-is-more-accurate-than-google-now-and-cortana-report/> Acessado em 10 abr. 2016.

ZERO HORA. Por problemas na leitura de hidrômetros, cobrança da água chega a R$ 190 mil em Porto Alegre. Disponível em: <http://zh.clicrbs.com.br/rs/porto-alegre/noticia/2015/07/por-problemas-na-leitura-de-hidrometros-cobranca-da-agua-chega-a-r-190-mil-em-porto-alegre-4812206.html>. Acessado em 03 jun. 2016.

ZUE, Victor. Conversational interfaces: advances and challenges, Proc. EuroSpeech-97, vol. V, pp. KN 9–18, 1997.

LISTA DE APÊNDICES

APÊNDICE A – ENTREVISTA COM LEITURISTA

1) Antes da implementação da coleta dos consumos via aplicativo, a demanda de tempo para a realização do processo era muito grande?

Realizo a leitura de gás e água de cerca de 80 condomínios por mês, costumava levar cerca de 5 a 7 dias de trabalho intenso (manhã e tarde, cerca de 6 horas/dia).

2) Quando as leituras eram realizadas com a planilha de papel, ocorriam muitos erros de leitura?

Havia duas grandes dificuldades, uma delas era a questão de visualização dos relógios, as duas mãos estavam ocupadas e eventualmente a luz era fraca, dificultando a visualização, para poder direcionar uma lanterna ou algo do gênero, era necessário largar a planilha e a caneta em algum local para fazê-lo. A outra dificuldade vinha do pessoal do escritório da Solução, onde muitas vezes não entendiam o dígito escrito corretamente e acabavam impostando no sistema o dado errado, o que gerava reclamações de condôminos.

3) Com o início da utilização do APP de leituras, houve uma redução no tempo das coletas?

Sem dúvidas, a utilização do APP foi muito benéfica para a otimização do processo, acredito que eu esteja levando muito menos tempo para realizar as leituras.

4) As informações históricas de consumos das unidades são úteis para a coleta dos consumos?

Por vezes ficamos com dúvidas sobre o dígito do relógio levemente virado, esses dados históricos fazem com que possamos ver a tendência de consumo da unidade e com isso deduzir da melhor maneira qual seria o consumo correto, acredito que isso faça com que a ocorrência de erros de leitura diminua.

5) O fato de ter no APP os Condomínios divididos por bairros e somente ser listado os que você é responsável facilita a gerência das coletas?

Ter um gerenciamento do que é sua responsabilidade é muito bom, fica mais fácil de controlar os prazos e o risco de esquecer algum Condomínio fica bem reduzido. Além disso, toda vez que eu fosse começar a realizar as leituras, precisava passar na Solução para pegar as planilhas de papel, com o aplicativo isso não é mais necessário, ficou muito bom!

6) Você acha relevante o fato do APP possuir três formas de leitura: reconhecimento de voz, reconhecimento de caracteres e teclado?

Acredito que quanto mais alternativas, melhor. Cada um se adapta de sua maneira com as tecnologias disponíveis. Nas leituras que realizei até o momento, utilizei bastante a voz, achei bastante assertivo e rápido, além de poder olhar para o relógio enquanto dito, isso facilitou muito.

APÊNDICE B – ENTREVISTA COM COLABORADOR

1) **Antes da implementação da coleta dos consumos via aplicativo, se perdia muito tempo impostando dados no ERP da empresa?**

Sim, perdíamos em torno de dois dias por mês apenas para copiar as leituras da planilha e impostar no sistema, afinal, entre gás e água temos em torno de 500 condomínios (em torno de 350 de gás e 150 de água).

2) **Havia algum tipo de dificuldade quanto a copia desses dados para digitação no sistema?**

Muitas vezes não entendíamos a letra do leiturista, ficávamos em dúvida e escolhíamos um por dedução, agora com o uso do APP, além de não termos mais que digitar no sistema, não passamos por esse problema.

3) **O fato do gerenciador do APP permitir o download das leituras agrupadas por lote facilita o dia-a-dia na empresa?**

Sem dúvida alguma, esse é o grande ganho da empresa, visto que não só deixamos de ter um re-trabalho como podemos importar para dentro do sistema 350 leituras de uma só vez. O que fazemos após a importação é apenas fazer uma análise para ver se não há um consumo irreal, mas o tempo que se perdia era muito maior.

LISTA DE ANEXOS

ANEXO A – LAYOUT ARQUIVO DE INTEGRAÇÃO COM ERP

INFORACE
INFORMÁTICA LTDA

Layout Integração Sistema Inforace - Rateio Gás

- Arquivo formato: .txt. Cada linha equivale a um registro de produto.
- Preencher o tamanho total dos campos. Numéricos com zeros à esquerda e Caractere espaço em branco à direita.
- Para os campos em branco preencher com zeros os do tipo numérico e com espaços em branco os campos de tipo alfanumérico.

OBS: Caso o campo não tenha valor, enviar com espaços em branco para os campos alfanuméricos e com zeros para os campos numéricos.

Seqüência dos campos:

Posição	Tamanho	Tipo de Dado	Descrição
001-030	30	Alfanumérico	Economia (identificação do apto,sala ou Box)
031-040	10	Numérico	Leitura Anterior
041-050	10	Numérico	Leitura Atual
051-060	10	Numérico	Número do Condomínio
061-070	10	Data	Data (Formato: DD/MM/AAAA)

Rua Nossa Senhora Aparecida, 947 – Bairro Floresta
CEP 95010-520 - Caxias do Sul – RS
Fone/Fax – (54) 3225-1260