UNIVERSIDADE FEDERAL DE MATO GROSSO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

RELATÓRIO DE ESTÁGIO SUPERVISIONADO DESENVOLVIMENTO DO SISTEMA DE EMISSÃO SIMULTÂNEA DE FATURAS DE ÁGUA E ESGOTO UTILIZANDO ANDROID

LUIZ HENRIQUE PACHECO SGUAREZI

CUIABÁ – MT
2015

UNIVERSIDADE FEDERAL DE MATO GROSSO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

RELATÓRIO DE ESTÁGIO SUPERVISIONADO DESENVOLVIMENTO DO SISTEMA DE EMISSÃO SIMULTÂNEA DE FATURAS DE ÀGUA E ESGOTO UTILIZANDO ANDROID

LUIZ HENRIQUE PACHECO SGUAREZI

Relatório apresentado ao Instituto de Computação da Universidade Federal de Mato Grosso, para obtenção do título de Bacharel em Sistemas de Informação.

CUIABÁ – MT
2015

UNIVERSIDADE FEDERAL DE MATO GROSSO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

LUIZ HENRIQUE PACHECO SGUAREZI

Relatório de Estágio Supervisionado apresentado à Coordenação do Curso de Sistemas de Informação como uma das exigências para obtenção do título de Bacharel em Sistemas de Informação da Universidade Federal de Mato Grosso

Aprovado por:

Prof. MSc. Thiago Meirelles Ventura
Instituto de Computação

Prof. MSc. Daniel Avila Vecchiato
Instituto de Computação
(ORIENTADOR)

Prof. MSc. Nilton Hideki Takagi
Instituto de Computação
(Coordenador de Estágios)

Esp. João Novaes de Campos Filho
(SUPERVISOR)

DEDICATÓRIA

Dedico este trabalho primeiramente a Deus que iluminou o meu caminho durante esta caminhada.

A todos os professores do curso, que foram tão importantes na minha vida acadêmica e no desenvolvimento desta monografia.

À minha família pelo apoio, por sua capacidade de acreditar em mim e investir em mim.

AGRADECIMENTOS

A Deus por ter me dado saúde e força para superar as dificuldades.

A minha família que sempre me apoiou nos estudos e nas escolhas tomadas.

Ao meu orientador Prof. MSc. Daniel Avila Vecchiato, pelo suporte, pelas suas correções e incentivos.

Aos meus colegas pelo companheirismo e disponibilidade para me auxiliar em vários momentos.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

ABNT	Associação Brasileira de Normas Técnicas
ADT	*Android Development Tools* – Ferramentas de Desenvolvimento Android
AOA	*Android Open Accessory*
API	*Application Programming Interface* - Interface de Programação de Aplicativos
ARM	*Advanced Risc Machine* - Nome da empresa criada para licenciar a fabricação de processadores nessa tecnologia.
ARM	*Advanced RISC Machine*
CPU	*Central Processing Unit* – Unidade de Processamento Central
GPS	*Global Positioning System* - Sistema de Posicionamento Global
GPU	*Graphics Processing Unit* - Unidade de Processamento Gráfico.
IDE	*Integrated Development Enviroment* - Ambiente de Desenvolvimento Integrado
IPC	*Inter-Process Communication* – Comunicação Entre Processos
JDK	Java Development Kit – Kit de Desenvolvimento Java
LTE	*Long Term Evolution* - Evolução a Longo Prazo, tecnologia de telefonia móvel também conhecida como 4G.
MVC	*Model-View-Controller* - Modelo-Visão-Controle
NFC	*Near Field Communication* - Comunicação de Campo Próximo em tradução livre.
SDK	*Software Development Kit* - Kit de Desenvolvimento de Software
SIP	*Session Initiation Protocol* - Protocolo de Iniciação de Sessão
SQL	*Structured Query Language* - Linguagem de Consulta Estruturada
UI	*User Interface* - Interface de Usuário
UID	*User* ID - ID de usuário
USB	*Universal Serial Bus*, é um tipo de conexão que permite a conexão de

periféricos sem a necessidade de desligar o computador.

UTMS *Universal Mobile Telecommunications Service* - Sistema de Telecomunicações Móveis Universal

WiFi Abreviação de "*Wireless Fidelity*" - Tecnologia de comunicação que não faz uso de cabos.

RESUMO

Este trabalho descreve as atividades realizadas no período de estágio supervisionado realizado na empresa Nortec Consultoria Engenharia e Saneamento Ltda pelo discente Luiz Henrique Pacheco Sguarezi, supervisionado por João Novaes de Campos Filho e orientado por Daniel Avila Vecchiato. O principal objetivo do estágio foi o desenvolvimento de um aplicativo para dispositivos móveis capaz de coletar dados de consumo de água e esgoto, além de fazer emissão de faturas simultâneas. O desenvolvimento foi motivado pela necessidade de aprimoramento no processo de coleta de dados e emissão de faturas. O processo utilizado anteriormente era realizado por meio de coleta manual com fichas de leitura e a entrega das faturas era realizada posteriormente, o que não acompanha as novas tecnologias no mercado tornando a empresa menos competitiva. Este aplicativo, denominado Vidhya Mobile, foi desenvolvido utilizando a linguagem de programação Java, utilizando também o SDK Android para o desenvolvimento do padrão Android, a biblioteca ZSDK_API para conexão com a impressora móvel Zebra RW 420 e o ambiente de desenvolvimento Eclipse. O aplicativo foi desenvolvido, testado e implantado no município de Sinop – MT e atendeu todas as necessidades propostas em seu escopo, a utilização de smartphones provou-se uma solução viável para o problema, possibilitando a emissão simultânea de faturas, pelo seu baixo custo e possiblidade de comunicação com a impressora móvel.

1. INTRODUÇÃO

Este relatório de estágio supervisionado descreve e especifica os conceitos e práticas utilizadas durante o período de estágio que ocorreu entre 27 de outubro de 2014 a 12 de dezembro de 2014. O estágio foi realizado na empresa Nortec Consultoria Engenharia e Saneamento Ltda para o desenvolvimento de um sistema de emissão de faturas simultâneas para dispositivos móveis com tecnologia Android, que é alimentado por um sistema de gerenciamento já existente por meio de arquivos.

Apenas um desenvolvedor ficou a cargo dos processos de análise e levantamento de requisitos e do desenvolvimento de modelos. Foram desenvolvidos diagramas baseados em UML como o de entidade relacionamento para auxiliar o desenvolvimento da base de dados, o modelo de caso de uso, diagrama de classes e diagrama de sequência para organizar o fluxo de processos. Todo o processo de análise e desenvolvimento foi supervisionado pelo analista sênior da empresa. Foram utilizados padrões de desenvolvimento em cascata, bibliotecas de comunicação com a impressora móvel e versões de teste foram geradas semanalmente durante o processo de implementação.

Dadas as atividades da empresa com o gerenciamento de departamentos de água e esgoto de municípios do interior de Mato Grosso, viu-se a necessidade de modernização do método de coleta de dados de consumo e emissão de faturas desses departamentos, que estavam utilizando processos manuais, como fichas de leitura. Por exemplo, alguns municípios já realizavam impressão simultânea, mas utilizavam equipamentos e sistemas defasados para realizar esta atividade.

Inicialmente, foram propostos duas frentes de desenvolvimento, uma com coletores de dados com tecnologia Windows Mobile e outra com Android. Após reuniões e levantamentos de custo beneficio, percebeu-se que a tecnologia Android oferecia melhores opções por ter plataforma *Open Source* e os dispositivos de coleta seriam *Smartphones* com preços mais acessíveis.

Alguns dos aspectos analisados para a escolha da plataforma foram:

- Custos com licenças para softwares de desenvolvimento;
- Comparação de preço, durabilidade e usabilidade dos dispositivos;

- Disponibilidade de dispositivos no mercado;
- Facilidade de atualização do projeto para novas versões da plataforma.

O escopo do projeto foi definido com base nas necessidades da empresa e as limitações de *hardware* e *software*. Representantes dos departamentos também foram ouvidos para que necessidades pontuais fossem avaliadas, uma vez que alguns municípios exigem particularidades legais em relação à emissão de faturas, por exemplo, o município de Sinop - MT que exige um modelo específico de tabela de qualidade de agua com os principais parâmetros monitorados mensalmente conforme a Portaria nº 2.914/2011.

Os requisitos funcionais foram escolhidos com base no escopo e com auxilio de entrevista com os gerentes de cada departamento. Também foi utilizado o método de etnografia com visitas aos departamentos para coleta de informações, observação de processos e procedimentos relativos à coleta de dados.

O restante deste documento esta organizado da seguinte forma.

- Capítulo 2: trata da revisão de literatura da teoria utilizada durante o estágio.
- Capítulo 3: são descritos os materiais, técnicas e métodos que especificam as ferramentas utilizadas no decorrer dos trabalhos.
- Capítulo 4: os resultados são descritos com imagens, tabelas e gráficos. Por fim,
- Capítulo 5: descreve as dificuldades encontradas durante a realização do estágio supervisionado.
- Capítulo 6: são feitas as considerações finais sobre os resultados obtidos no trabalho.

2. REVISÃO DE LITERATURA

Este Capítulo visa abordar os aspectos mais relevantes à área do estágio, relacionando os fundamentos teóricos e práticos, sob os quais o presente relatório se relaciona.

2.1. **Desenvolvimento para dispositivos Android**

Dispositivos móveis como *smartphones* e *tablets* estão cada vez mais presentes no cotidiano pessoal e profissional e seus sistemas operacionais se tornaram mais importantes. O Android é um sistema operacional direcionado à dispositivos móveis sendo capaz de operar com baixo poder de processamento, bateria e controla componentes de hardware como GPS, câmera, sensores de iluminação e movimento, WiFi, Bluetooth, UMTS (telefonia 3G) e LTE (telefonia 4G) (ANDROID DEVELOPERS, 2012). O Android é capaz de abstrair as funcionalidades do hardware e possibilitar seu uso em um ambiente definido para as aplicações. Aplicações Android são escritas em Java e são processadas por uma máquina virtual, denominada Dalvik, que executa seu próprio código binário.

A plataforma foi inicialmente desenvolvida pela empresa Android Inc. que foi comprada pela Google que lançou o Android Open Source Project (AOSP) em 2007. Uma aliança de 78 empresas formam a Open Handset Alliance (OHA) que é responsável pelo desenvolvimento de distribuição do Android. As principais empresas desta aliança são: Nvidia, HTC, Dell, Intel, Motorola, Qualcomm, Samsung, LG, T-Mobile e Google (OPEN HANDSET ALIANCE, 2014).

O desenvolvimento do Android tem uma rápida rotina de atualizações e lançamentos de novas versões e o mesmo acontece com a evolução dos dispositivos móveis. O desenvolvimento de aplicações deve estar atento a estas atualizações de *software* e *hardware* fazendo uso da documentação do SDK, de artigos e de conteúdo na Internet.

2.2. Modelo de desenvolvimento em cascata

O modelo cascata, também é chamado de ciclo de vida clássico ou tradicional, sugere uma abordagem sequencial e sistemática para o desenvolvimento de software.

Ou seja, as fases do projeto são bem definidas, iniciando com o levantamento de requisitos ou necessidades do cliente, passando para a fase de planejamento onde são definidas estimativas, cronograma e acompanhamento. Após isso, é realizada a modelagem e a análise e projeto, seguindo da construção onde o projeto é codificado e testado, por fim é realizada a implantação, suporte e feedback do software concluído (DEVMEDIA, 2014).

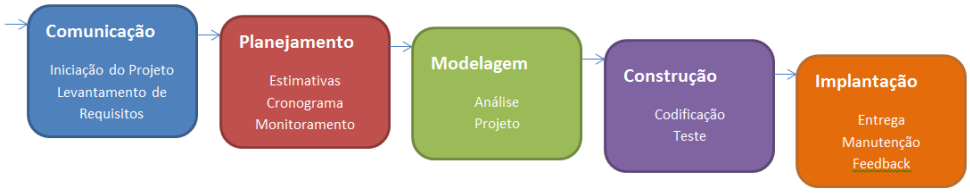

Figura 1 - O Modelo em cascata estático. O andamento do processo flui de cima para baixo, como uma cascata (PRESSMAN, 2011).

A Figura 1 demonstra as etapas do modelo cascata. O modelo cascata é utilizado principalmente quando os requisitos de um determinado problema são bem compreendidos. Uma forma de utilizar o modelo cascata é quando precisamos fazer adaptações ou aperfeiçoamentos em um sistema já existente. Também podemos utilizar o modelo cascata quando um software necessita de uma nova funcionalidade e os requisitos estão bem definidos e são estáveis. Por exemplo, quando temos um sistema já pronto e precisamos fazer uma adaptação porque alguma lei governamental foi alterada ou criada (DEVMEDIA, 2014).

Mesmo sendo um modelo antigo ainda é muito utilizado na indústria, mas este processo recebe muitas críticas que gerou questionamentos sobre a sua eficácia. Os principais problemas encontrados no modelo cascata são:

- Os projetos raramente seguem o fluxo sequencial que o modelo propõe;
- A interação é sempre necessária e está presente, criando problemas na aplicação do modelo;
- Em princípio, é difícil para o cliente especificar os requisitos explicitamente, o que acarreta a incerteza natural do início de qualquer projeto;
- Demora em obter resultados, pois uma versão funcional não estará disponível até o final do desenvolvimento. Qualquer erro ou mal entendido, se não for

detectado até que o software seja revisado, pode ser acarretar grandes problemas (DEVMEDIA, 2014).

Apesar desses problemas, o modelo Cascata tem um lugar bem definido e importante nos trabalhos de engenharia de software. Ele fornece um padrão do qual se encaixam métodos para a análise, projeto, codificação e manutenção.

2.3. Arquitetura Android

A arquitetura Android é subdivida em cinco camadas, ilustradas na Figura 2, sendo divididas em:

- *Applications*: responsáveis por criar e gerenciar os componentes e recursos essenciais;
- *Application Framework*: fornece todas as funcionalidades necessárias para a construção de aplicativos, através das bibliotecas nativas;
- *Android Runtime*: composto pela máquina virtual chamada Dalvik VM;
- *Libraries*: ficam acima do kernel e são escritas em C ou C++ e são utilizadas por diversos componentes do sistema;
- *Linux Kernel*: base do Android que é uma versão modificada do kernel Linux 2.6, que provê vários serviços essenciais, como segurança, rede e gerenciamento de memória e processos e uma camada de abstração de hardware.

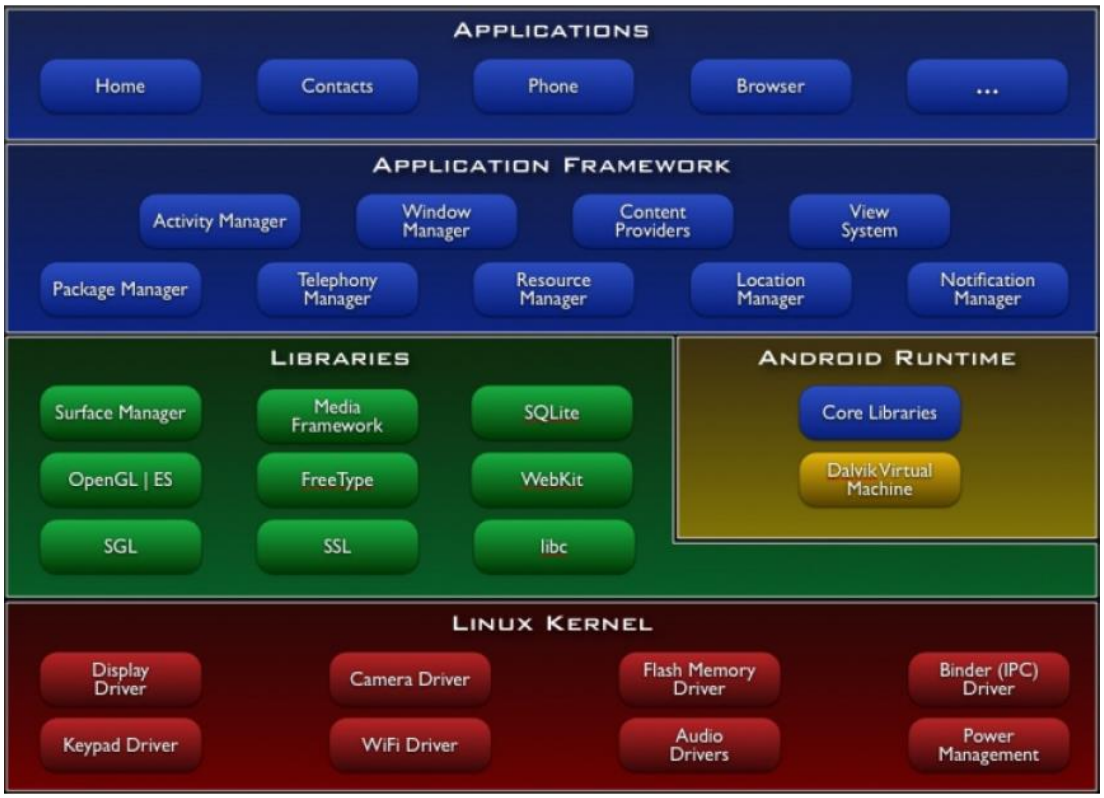

Figura 2 – Arquitetura do Sistema Android. Componentes verdes são escritos em C/C++, os azuis são escritos em Java e rodam na Dalvik VM (ANDROID OPEN SOURCE PROJECT, 2012).

O kernel usado é o Linux 2.6, modificado para atender as necessidades especiais que os dispositivos móveis em gerenciamento de energia, gerenciamento de memória e em ambiente de execução. Alguns componentes Linux foram aproveitados como *bluez* para o suporte Bluetooth e *wpa_suplicant* para a encriptação WiFi (PROGRAMANDO O ANDROID, 2012).

O Android foi projetado para ser executado em dispositivos com pouca memória e baixo poder de processamentos. As bibliotecas de CPU e GPU tem o código fonte otimizados para esses dispositivos. Bibliotecas básicas como *libc* ou *libm* foram desenvolvidas para o baixo consumo de memória (PROGRAMANDO O ANDROID, 2012).

No Android, aplicações escritas em Java são executadas em sua própria máquina virtual, que por sua vez é executada em seu próprio processo no Linux, isolando-a de outras aplicações e facilitando o controle de recursos (ANDROID DEVELOPERS, 2012).

O Android Runtime consiste na máquina virtual Dalvik e principais bibliotecas Java. A máquina virtual Dalvik é um interpretador de código binário que foi transformado a partir de código Java. O Dalvik é compilado em seu próprio código fonte enquanto as principais bibliotecas são escritas em Java, que são interpretadas pelo Dalvik.

Na camada acima, escrita em Java, fica a *framework* de aplicações, que fornece todas as funcionalidades necessárias para a construção de aplicativos, por meio das bibliotecas nativas. Aplicações Android podem possuir quatro tipos de componentes: *activities*, *services*, *content providers* e *broadcast receivers*. Além destas peças fundamentais em uma aplicação, existem os recursos, que são compostos por layouts, *strings*, estilos, imagens e o arquivo de manifesto, que declara os componentes da aplicação e os recursos do dispositivo que ela irá utilizar (ANDROID DEVELOPERS, 2012).

2.4. Persistência de dados

O Android provê várias opções de persistência de dados para aplicações. A solução varia de acordo com a necessidade específica da aplicação, podendo ser acessível somente a uma aplicação específica ou acessível a outras aplicações e usuários (ANDROID DEVELOPERS, 2014).

Exemplificando algumas dessas opções de armazenamento temos.

- Preferências Compartilhadas: armazenam dados primitivos em pares de valores chave;
- Armazenamento Interno: armazena dados na memória do dispositivo;
- Armazenamento Externo: armazena dados públicos na memória externa compartilhada;
- Base de Dados SQLite: armazena dados estruturados em uma base de dados privada e Conexão Network que armazena dados na rede através de seu próprio servidor (ANDROID DEVELOPERS, 2014).

2.4.1. Base de Dados SQLite

O Android provê total suporte para bases de dados SQLite. As bases criadas são acessíveis por nome para qualquer classe da aplicação, mas não são acessíveis fora

da aplicação na qual foi criada. O Android SDK inclui o gerenciador de banco de dados sqlite3 que permite operações com tabelas, executar rotinas SQL e controlar funções na base de dados (ANDROID DEVELOPERS, 2014).

O SQLite é uma ferramenta de banco de dados SQL embutida ao sistema, ou seja diferente de outros bancos de dados, pois ele não possui um servidor de processos. O SQLite lê e escreve diretamente em arquivos de disco comuns, o banco de dados completo com várias tabelas, índices, gatilhos e *views* esta contido em um único arquivo de disco. O formato do arquivo de banco de dados é multiplatafoma podendo ser copiado entre sistemas de 32 bits ou 64 bits de ou entre arquiteturas *big-endian* e *little-endian*.

SQLite é uma biblioteca compacta. Com todos os recursos ativados, o tamanho da biblioteca pode ser inferior a 500KB, dependendo das configurações da plataforma-alvo e de otimização do compilador. Se os recursos opcionais forem desativados, o tamanho da biblioteca SQLite pode ser reduzida para menos de 300KB. O SQLite também pode ser configurado para ser executado em um espaço mínimo de pilha (4KB) e heap (100KB), tornando SQLite uma escolha popular de banco de dados em memória restrita para *gadgets* como celulares, PDAs e MP3 *players* (SQLITE, 2014).

2.5. Comunicação

O Android fornece diversas APIs que possibilitam uma aplicação se conectar e interagir com outros dispositivos através de Bluetooth, NFC, Wi-Fi P2P, USB, e SIP, além de conexões de rede padrão.

2.5.1. Bluetooth

A plataforma Android inclui suporte para a rede Bluetooth, que permite que um dispositivo sem fio trocar dados com outro dispositivo através desta rede. As APIs Bluetooth são interfaces nativas do Android que auxiliam o desenvolvimento de aplicações que necessitam de comunicação sem fio. Essas APIs permitem que um dispositivo Bluetooth se conecte a outro dispositivo com a mesma tecnologia, permitindo comunicação ponto-a-ponto e multiponto através de recursos sem fio.

Um aplicativo pode utilizar as APIs Bluetooth para verificar se há outros dispositivos Bluetooth próximos, consultar o dispositivos Bluetooth emparelhados, conectar-se a outros dispositivos por meio de descoberta de serviço, transferir dados de e para outros dispositivos e gerenciar conexões.

2.5.2. USB

A arquitetura Android utiliza dois modos de suporte a dispositivos e acessórios USB, o modo acessório e o modo host. No modo acessório o *hardware* USB externo se comporta como um host USB, isto dá dispositivos Android que não têm capacidades de conexão a capacidade de interagir com o *hardware* USB.

Acessórios USB Android devem ser projetados para trabalhar com dispositivos Android e devem aderir ao protocolo AOA (*Android Open Accessory*) possibilitando a comunicação. Exemplos de dispositivos incluem câmeras digitais, teclados, mouses e controladores de jogos. A Figura 3 mostra as diferenças entre os dois modos.

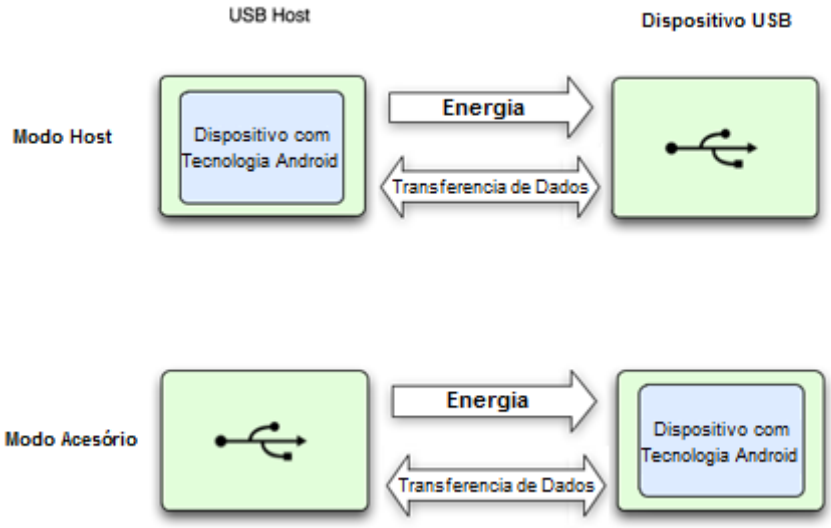

Figura 3 - Host USB e modo acessório [Android Developers 2012].

2.6. Segurança de aplicação

A plataforma Android foi construída de modo que sua segurança não ficasse tão dependente dos desenvolvedores. Sua arquitetura de segurança permite que controles sejam aplicados de forma transparente para o desenvolvedor. Isso é alcançado por meio do confinamento de aplicações (*Sandbox*), pelo esquema de permissões tanto do sistema de arquivos como de chamadas de API e pelo mecanismo de IPC (*Inter-*

Process Communication), que também aplica tais permissões para conceder acesso ou não entre os diferentes componentes (ANDROID OPEN SOURCE PROJECT, 2012).

O desenvolvimento da plataforma é baseado em torno de algumas características, conforme visto a seguir:

- Dispositivos de *hardware* - assim como o próprio Linux, diversas configurações de *hardware* são suportadas pelo Android, entre elas: *smartphones*, *tablets*, *settop-boxes* e *e-readers*. Cada um desses dispositivos possui controles de segurança implementados em *hardware* como por exemplo o ARM (*Advanced RISC Machine*) TrustZone e o ARM XN (*execute never*) e o sistema é capaz de se utilizar de tais capacidades;
- Sistema operacional - baseado no kernel do Linux, o núcleo do Android provê a interface para a utilização do dispositivo. O acesso a todos os recursos é mediado pelo sistema operacional e fica restrito aos controles de segurança implementados pelo sistema operacional;
- Ambiente de execução de aplicações (Android *Application Runtime*) - A maioria dos aplicativos para Android são desenvolvidos em Java e rodam na máquina virtual Dalvik, embora seja possível criar aplicações que executam código nativo na plataforma do dispositivo. Ainda assim, essas aplicações, juntamente com os outros serviços providos pela plataforma e pelas bibliotecas, que rodam código nativo, executam em um ambiente confinado denominado *sandbox* de aplicação. Cada aplicativo é executado com seu próprio UID (*User* ID, ou ID de usuário) e em seu próprio processo restringindo o acesso de outras aplicações ao processo.

A segurança da arquitetura baseia-se fortemente nos mecanismos de segurança aplicados pelo kernel do Linux e na disponibilização de uma comunicação interprocesso segura. Todo código de aplicação, incluindo as que executam código nativo, ficam restritas pelo *sandbox* de aplicação, que é implemetado por meio de um modelo de isolamento de processos baseado em usuários, que é aplicado pelo kernel.

3. MATERIAS, TÉCNICAS E MÉTODOS

No início do período de estágio foram realizadas reuniões para definir um melhor gerenciamento e acompanhamento do projeto, foram criadas tarefas e prazos a serem cumpridos. Tais tarefas foram selecionadas para durarem no mínimo 10 semanas totalizando 300 horas de acordo com as regras do estagio supervisionado. A Tabela 1 mostra o resultado das divisões das tarefas e do tempo previsto para sua realização.

Tabela 1 - Carga horaria de atividades do estagio supervisionado.

Atividades Executadas	Horas Previstas	Horas Utilizadas
• Identificação e estudo do problema, identificando as necessidades da empresa com o produto;	12	18
• Analise de requisitos funcionais e não funcionais do sistema;	18	22
• Modelagem da Arquitetura e da estrutura do aplicativo englobando padrões de programação e armazenamento e transmissão de dados;	30	22
• Metodologia de desenvolvimento;	12	18
• Escolha de aplicativos e plataformas de desenvolvimento o Escolha e limitação de versão da plataforma (Android);	18	12
• Desenvolvimento da base de dados;	30	36
• Desenvolvimento de layout do aplicativo e desenvolvimento de código fonte;	60	72
• Desenvolvimento de Modulo de comunicação com Impressora Móvel;	30	42
• Desenvolvimento de interface de carga e descarga de dados no aplicativo;	30	30
• Validação de segurança dos dados e testes de desenvolvimento;	60	48

• Acompanhamento de implantação e testes com usuário;	30	30
Total:	**330**	**350**

3.1. Metodologia de Desenvolvimento

Para o desenvolvimento deste projeto apenas um desenvolvedor ficou responsável por todas as etapas do projeto, esta decisão foi tomada devido a impossibilidade de paralelização do trabalho e da urgência para o desenvolvimento da ferramenta. Isso resultou também na dificuldade de contratação e treinamento de novos funcionários e a necessidade da realização do estágio supervisionado.

Algumas reuniões com os clientes, no caso os utilizadores finais do sistema, foram necessárias para definição do escopo. Estas reuniões foram supervisionadas pelo responsável do setor de tecnologia e informação da empresa que auxiliou a escolha do processo de desenvolvimento.

O processo de desenvolvimento foi utilizado o modelo em cascata, pois os requisitos são bem definidos devido a existência de um sistema principal já pronto e operando. Este projeto, portanto, se caracteriza como uma evolução de um sistema já existente otimizando o processo de coleta de dados de consumo e emissão de faturas através de dispositivos móveis com tecnologia Android.

3.2. Modelagem da Arquitetura

O framework de UI (*User Interface*, ou Interface de Usuário) do Android é organizado em torno do padrão MVC (*Model-View-Controller*, ou Modelo-Visão-Controle) que fornece ferramentas para construção de um controlador que trata as interações com o usuário.

Para o armazenamento dos dados do aplicativo foi utilizado o SQLite que é nativo para sistemas Android que oferece vantagens para a programação em dispositivos móveis, como diversos recursos relacionados a tolerância a falhas e acesso concorrente de dados. Por exemplo, muitos sistemas de bancos de dados utilizam tipagem estática, mas o SQLite não armazena informações de tipo referentes ao banco de dados. Em vez disso, ele delega essa responsabilidade a

linguagens de alto nível, que mapeiam estruturas do banco de dados em tipos de alto nível.

Para o projeto em questão, a transmissão de dados se dá com arquivos de texto gerados a partir de outro sistema de gerenciamento da própria empresa. O sistema principal gera arquivos de texto formatados que são interpretados e carregados na base de dados do aplicativo móvel. Também ocorre a transmissão de dados entre o dispositivo e uma impressora móvel por meio de Bluetooth. A Figura 4 demonstra a comunicação entre esses sistemas e dispositivos.

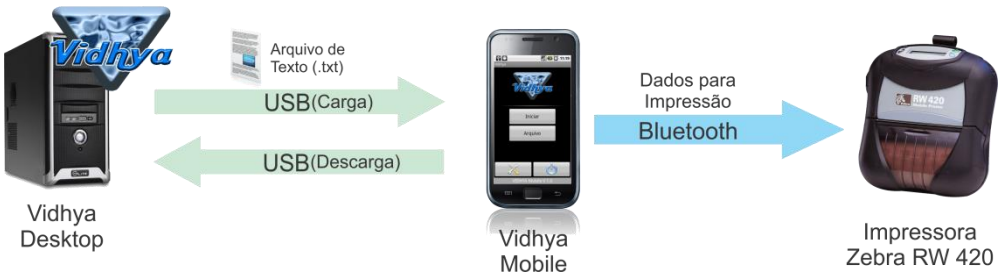

Figura 4 - Comunicação entre sistemas e dispositivos.

3.3. Ferramentas de desenvolvimento

Para o desenvolvimento de um aplicativo Android são necessários o SDK Android, o JDK (Java *Development* Kit, ou Kit de Desenvolvimento Java) e o IDE (*Integrated Development Enviroment*, ou Ambiente de Desenvolvimento Integrado) que são instalados separadamente.

3.3.1. Eclipse

A IDE escolhida para o desenvolvimento foi o Eclipse, que é uma plataforma para desenvolvimento de diversas linguagens e também pode ser personalizada para determinado SDK. Com o *plug-in* ADT (*Android Development Toolkit*) são adicionadas funcionalidades específicas no desenvolvimento para Android.

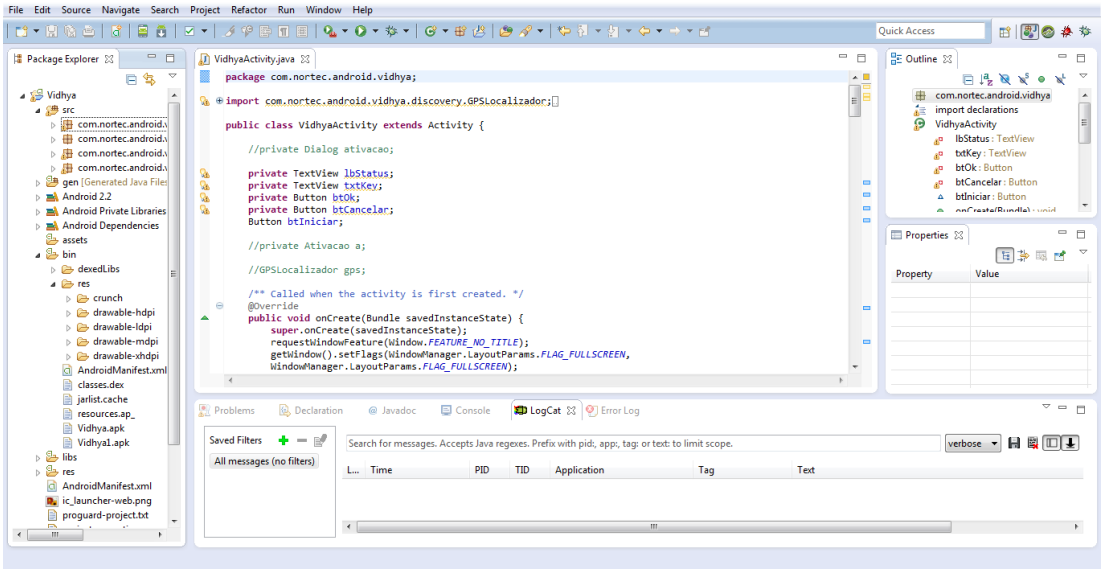

Figura 5 - Ambiente de desenvolvimento Eclipse.

A IDE foi utilizada porque facilita o desenvolvimento para dispositivos Android, integrando e dinamizando o processo de modelagem e programação, como mostra a Figura 5. O uso da IDE é de licença pública e a integração é possível através da licença *Android Software Development Kit*.

3.3.2. Astah Community

O **Astah Community** é uma ferramenta de modelagem UML de fácil acesso e de simples utilização, é desenvolvido na plataforma Java, o que garante sua portabilidade para qualquer plataforma que possui uma máquina virtual Java.

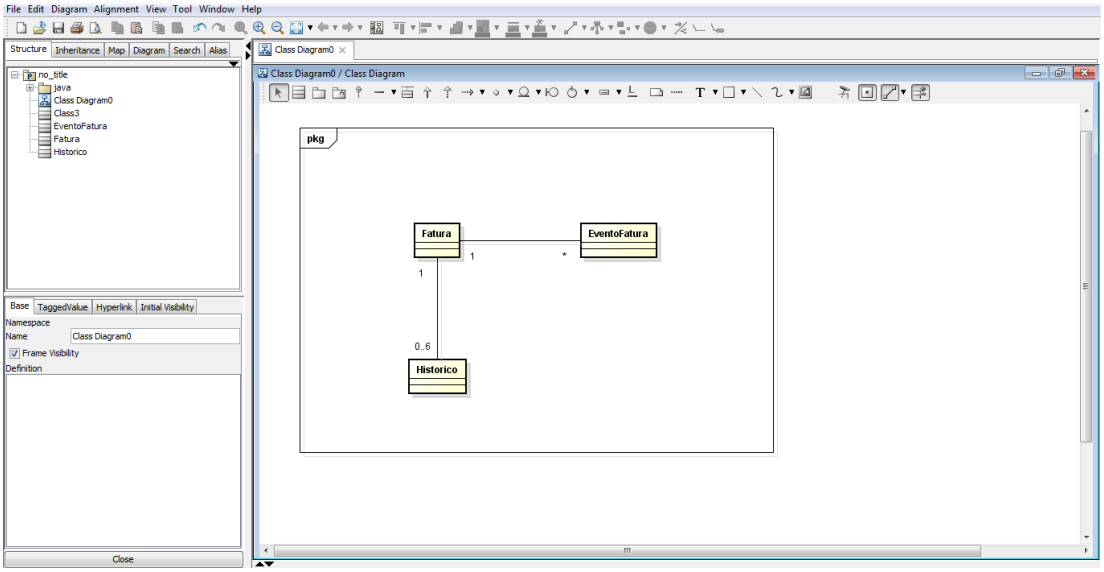

Figura 6 - Ferramente de modelagem Astah Community.

Foi utilizado o Astah Community por ser uma ferramenta gratuita que disponibiliza recursos úteis no processo de documentação e modelagem do projeto.

3.3.3. Emulador Genymotion

O Genymotion (GENYMOBILE, 2013) é um emulador que vem com imagens pré-carregadas de versões do Android e por isso tem um desempenho semelhante a de um dispositivo móvel físico. O Genymotion é uma alternativa ao emulador incluso do ADT Bunde, sua vantagem é rodar em cima de uma imagem Intel x86 enquanto o emulador do ADT Bundle executado em arquitetura ARM, o que o torna menos eficiente.

A escolha deste modelo de emulador foi devido a sua interface simples e de fácil configuração, ainda com o ganho em desempenho pelo rápido tempo de resposta. O fabricante disponibiliza uma versão gratuita do produto com algumas limitações, mas a ferramenta atendeu as necessidades do projeto.

3.4. Desenvolvimento da base de dados

A base de dados foi elaborada a partir da documentação existente para o padrão de emissão de faturas do sistema base que gerencia e gera os dados de carga para o sistema móvel. Partindo deste ponto foi elaborado um Modelo de

Entidade Relacionamento, demonstrado na Figura 6, e um dicionário de dados para criação do banco de dados e SQLite.

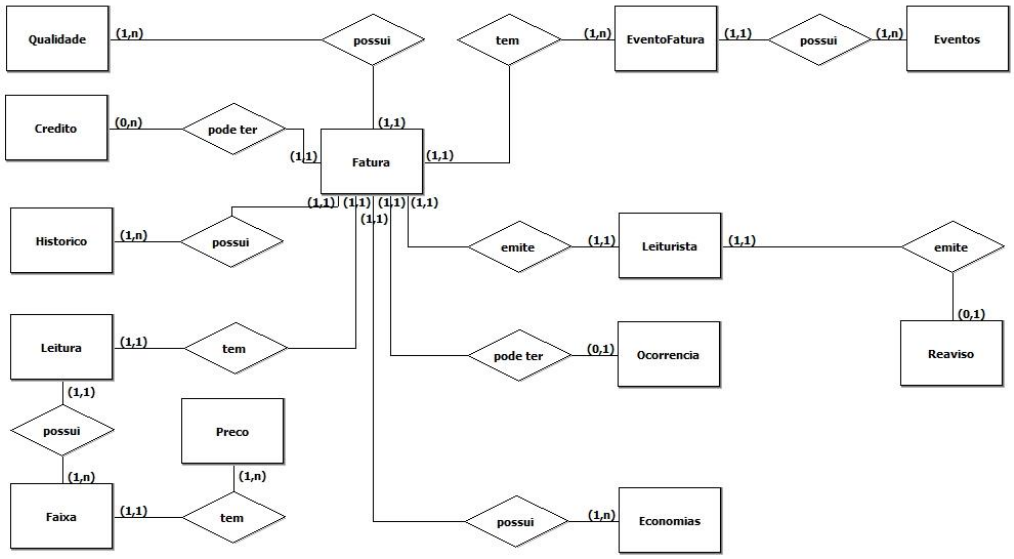

Figura 7 - Modelo Entidade Relacionamento do sistema Vidhya Mobile.

3.5. Impressora Zebra RW 420

A impressora Zebra RW 420, ilustrada na Figura 6, é de uma série de impressoras de alto desempenho para impressão de recibos e faturas em papel térmico. O design modular permite aos usuários escolher entre opções sem fio, leitores de cartão, comunicação em rede e integração com acessórios.

Figura 7 - Impressora móvel Zebra RW 420

Esta impressora foi escolhida devido a possibilidade de sincronização através de tecnologia Bluetooth compatível com a utilizada em *smartphones*, também por sua robustez para o uso em campo, devido a sua carcaça rígida e uma ótima autonomia de bateria podendo durar até 12 horas de utilização intensa.

As impressoras Zebra tem o seu próprio padrão de comunicação o que faz necessário o uso da biblioteca ZSDK_API que possibilita a comunicação entre o dispositivo Android e a impressora. A biblioteca é disponibilizada gratuitamente pelo fabricante, sendo necessário somente um cadastro junto ao site.

4. RESULTADOS

Neste Capítulo serão apresentados os resultados do estágio supervisionado por meio de telas do sistema com descrições de funcionalidades e comentários.

A Figura 8 demonstra a tela inicial do aplicativo gerenciando o acesso as funções do sistema, podendo iniciar o processo de coleta de dados a partir do botão **Iniciar**, acessar o controle de carga e descarga de arquivos através do botão **Arquivo**, além de gerenciar as configurações ou sair do sistema.

Figura 9 - Tela inicial.

Antes de carregar a tela inicial, o sistema verifica as configurações de impressão simultânea que estão armazenadas na base de dados e caso estejam ativadas será ativado um requerimento de ativação do Bluetooth, demonstrado na Figura 10.

Figura 11 - Solicitação para ativação do Bluetooth.

Ao selecionar a opção de configurações na tela inicial o sistema exibirá a tela demonstrada na Figura 12. As configurações exibidas são exclusivamente relacionadas ao processo de impressão simultânea das faturas, as configurações de faturamento gerais e de cada consumidor são configuradas no sistema principal. É possível ativar ou desativar a impressão simultânea e cadastrar o endereço Bluetooth da impressora móvel através do botão de busca.

Figura 13 - Tela de configuração de impressão simultânea.

O cadastro do endereço da impressora móvel só é possível através de um mecanismo de busca de dispositivos próximos, não sendo possível a digitação do mesmo. A Figura 14 demostra o processo de busca desenvolvido para listar o endereço MAC e o nome cadastrado do dispositivo, após a escolha é necessário salvar as configurações.

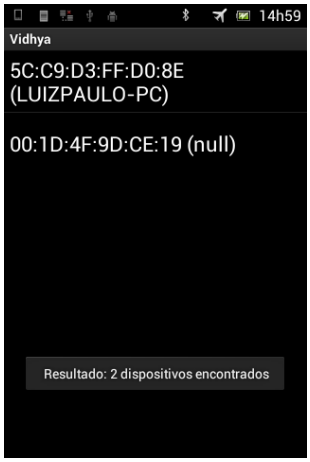

Figura 15 - Busca de dispositivos Bluetooth próximos.

A Figura 16 mostra a tela de carga e descarga de arquivos de texto na base de dados do sistema, para o processo de carga dos dados o sistema busca os arquivos em uma pasta especifica da memória do celular e gera uma lista com os arquivos presentes, o processo de descarga gera um arquivo com as faturas já processadas e o salva em uma pasta específica.

Figura 17 - Carga e descarga de arquivos.

A Figura 18 mostra a lista de arquivos para carga do dispositivo. Após a seleção ao clicar no botão Carga o sistema exibe uma tela de espera enquanto processa e carrega o banco de dados com os dados do arquivo.

Figura 19 - Seleção e carga de arquivo.

Um fluxo alternativo no processo de carga pode ocorrer quando já existem dados carregados no banco de dados provenientes de uma carga anterior. O processo de carga apaga os dados e carrega novos dados para processamento. Para evitar erros

o sistema gera uma mensagem de alerta sobre a possibilidade de perca de dados não descarregados, como demostrado na Figura 20.

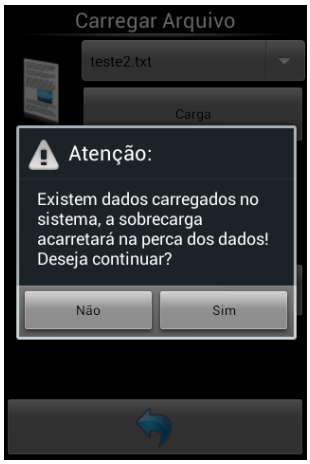

Figura 21 - Mensagem de alerta de perca de dados.

Iniciando o processo de coleta de dados a Figura 22 exibe a tela de navegação entre unidades consumidoras que são ordenados de acordo com uma sequência de faturamento, previamente cadastrada no sistema principal. O usuário do sistema segue esta sequência verificando os dados exibidos na tela, podendo avançar, retroceder, buscar e faturar (coletar consumo e emitir fatura) uma unidade consumidora.

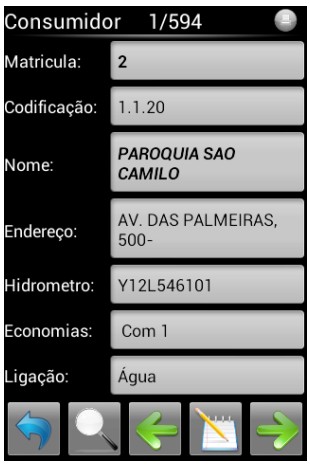

Figura 23 - Tela de navegação de consumidores.

A Figura 24 exibe a tela de busca do sistema, para localizar uma unidade consumidora é necessário digitar a matrícula, o nome ou a codificação da unidade consumidora. Caso a busca obtenha sucesso será exibida uma lista com as

combinações encontradas e basta selecionar a linha correspondente ao consumidor para exibir os dados na tela de navegação.

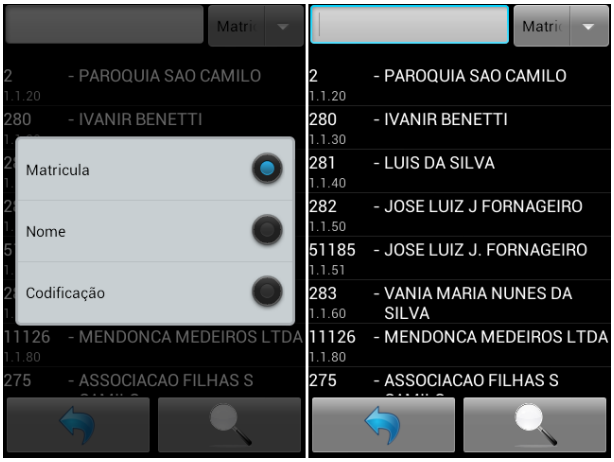

Figura 25 - Busca de unidade consumidora.

A Figura 26 exibe a tela de coleta de dados e emissão de fatura, o sistema exibe uma confirmação do nome e número de hidrômetro e informações sobre o histórico de faturamento, também é possível inserir ocorrências de faturamento caso existam.

Figura 27 - Faturamento de unidade consumidora.

Para inserir uma leitura de hidrômetro foi criado um teclado específico para aplicação, o campo **Leitura** da tela de faturamento só pode ser alterado por meio deste teclado devido a falta de legibilidade dos teclados padrões do Android. O teclado foi desenvolvido para ser mais legível e fácil de operar evitando erros de

digitação, que são comuns neste tipo de trabalho. A Figura 28 demonstra o teclado utilizado no sistema.

Figura 29 - Tela de inserção de leitura.

Também na tela de faturamento a Figura 30 é exibido o procedimento para inserir ocorrências de faturamento caso necessário, primeiramente deve-se selecionar um *checkbox* ativando a seleção da ocorrência. As ocorrências de faturamento são cadastradas no sistema principal e carregadas no banco de dados do dispositivo móvel.

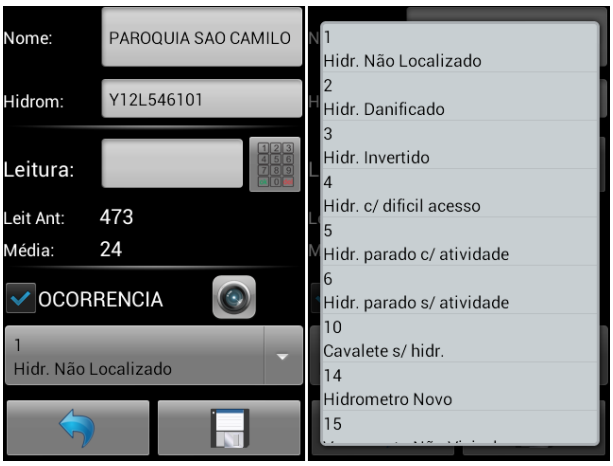

Figura 31 - Procedimento de inserção de ocorrência de faturamento.

Um fluxo alternativo a este processo é caso o sistema detecte alguma inconsistência na leitura, para evitar falhas foi desenvolvido uma verificação de acordo com parâmetros pré-configurados em relação à média e volume de consumo

em metros cúbicos. Caso seja detectado um possível erro o sistema exibe mensagens de alerta. A Figura 32 mostra algumas situações deste tipo.

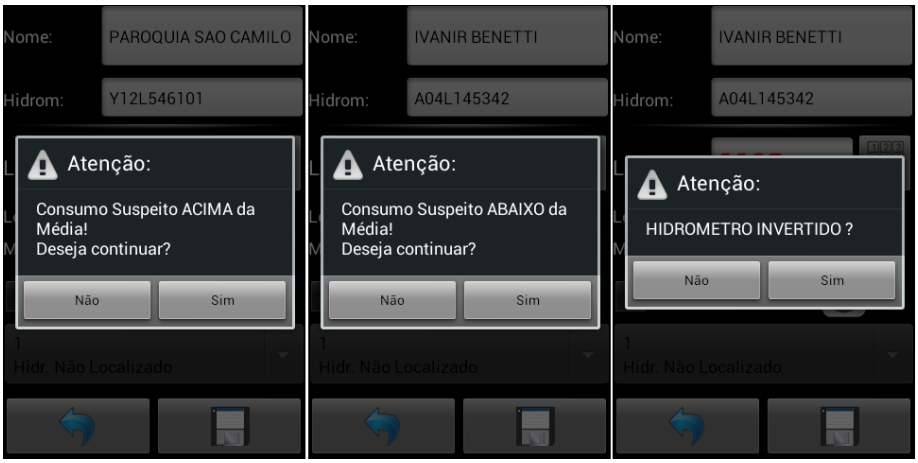

Figura 33 - Mensagens de alerta.

Por fim, para finalizar o processo de coleta de dados o usuário do sistema deve confirmar os dados digitados e salvar a leitura no caso de coleta sem impressão simultânea ou imprimir a fatura e entregá-la ao consumidor e o sistema exibe a tela de navegação de consumidor com os dados do próximo consumidor. A Figura 34 exibe um modelo de fatura impresso pelo sistema.

Figura 35 - Fatura de água e esgoto.

5. DIFICULDADES ENCONTRADAS

A sincronização e transmissão de dados para impressora móvel a partir de um *smartphone* foi o maior desafio encontrado no projeto. Este tipo de impressora tem uma linguagem específica de configuração e impressão tomando algum tempo em pesquisa e aprendizado, além de inúmeros testes para ajustar os dados aos campos de impressão. A fabricante das impressoras disponibiliza um grande volume de material, como tutoriais e projetos exemplo para auxiliar este processo.

Outra dificuldade foi desenvolver um *layout* de exibição que se adequasse ao ambiente de trabalho que o sistema seria utilizado. As coletas de dados são feitas em campo, em ambientes com muita luz ou até mesmo sob a luz do sol. A solução encontrada foi desenvolver um padrão de máximo contraste entre a tela e as informações, utilizando fontes como maior tamanho possível e orientando a utilização dos *smartphones* com o máximo de brilho e contraste.

6. CONCLUSÕES

A modernização do processo de coleta de dados e emissão de faturas simultâneas é indispensável para o acompanhamento do mercado de gestão de saneamento básico atual. O Vidhya Mobile conseguiu atender a todas as expectativas que foram propostas, utilizando dispositivos baseados em Android e ferramentas de desenvolvimento gratuitas ou de licença livre. Com isso, a ferramenta se tornou viável e moderna.

Novas tecnologias estão sendo implementadas e colocadas no mercado a todo o momento fazendo com que os processos sejam revistos e modernizados. No caso do Vidhya Mobile, uma possível evolução é a sincronização de dados online, utilizando pacote de dados, tecnologia já presente nos *smartphones*. Esta sincronização possibilita interação em tempo real entre o dispositivo móvel e o sistema principal, as leituras de consumo realizadas são automaticamente enviadas para o banco de dados principal fazendo com que o gerente de faturamento tenha acesso a estes dados, em posse destas informações o gerente pode tomar decisões imediatas com relação a problemas no faturamento.

7. REFERÊNCIAS BIBLIOGRÁFICAS

ANDROID, developers. The developer's guide, Disponível em <http://developer.android.com/guide/index.html >. Acesso em: 12 nov. 2014.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS; 1989. *Referências Bibliográficas*, NBR 6023. Rio de Janeiro.

ASTAH, COMMUNITY. Disponível em <http://astah.net/ >. Acesso em: 25 nov. 2014.

DALVIK, VM. 2014. Dalvik Virtual Machine. Disponível em <http://www.dalvikvm.com/ >. Acesso em: 10 nov. 2014.

DEVMEDIA; 2014. Introdução ao Modelo Cascata, Disponível em <http://www.devmedia.com.br/introducao-ao-modelo-cascata/29843 >. Acesso em: 25 out. 2014.

ECLIPSE, Getting Started. Disponível em <https://eclipse.org/ >. Acesso em: 05 nov. 2014.

GENYMOBILE, SAS.; 2014. Genymotion, Disponível em <http://www.genymotion.com >. Acesso em: 20 nov. 2014.

GOOGLE, INC.; Android Open Source Project. Disponível em <http://source.android.com/ >. Acesso em: 20 nov. 2014.

MEDNIEKS, ZIGURD; 2012. Programando o Android. Primeira Edição. São Paulo – SP : Novatec Editora Ltda.

NUDEKMAN, GREG; 2013. Padroes de Projeto para o Android. 1ª Edição. São Paulo, SP Brasil: Novatec Editora LTDA.

Open Handset Aliance (2014). Android Overview. Disponível em: <http://www.openhandsetalliance.com/android_overview.html> Acesso em: 03 nov. 2014.

PRESSMAN, R. Engenharia de Software: Uma abordagem Profissional. 2011. 7º edição. Editora Bookman.

SQLITE. About SQLite. 2014. Disponível em <http://www.sqlite.org/ >. Acesso em: 20 nov. 2014.