UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

EVANDRO WESTPHALEN CARLOS GOMES

GLOBAL INTERFACE PROCESS USANDO ANDROID ATRAVÉS DE WEBSERVICES

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2011

EVANDRO WESTPHALEN CARLOS GOMES

GLOBAL INTERFACE PROCESS USANDO ANDROID ATRAVÉS DE WEBSERVICES

Monografia de especialização apresentada ao Departamento Acadêmico de Informática – DAINF – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de "Especialista em Tecnologia Java".

Orientador: Prof. Paulo Maurício de Lucchi Bordin.

CURITIBA

2011

AGRADECIMENTOS

Utilizo este espaço para agradecer a Deus que permitiu forças para alcançar este momento.

Os familiares que em nenhum momento negaram apoio tenha sido ele qualquer que fosse participando ativamente desta conquista.

Os amigos, que, de uma forma ou de outra contribuíram durante essa caminhada.

Ao orientador Paulo Bordin que esteve a disposição durante todo o desenvolvimento deste trabalho orientando e validando cada etapa.

À instituição, oferecendo espaço e apoio dos professores sempre prestativos diante das necessidades, esclarecendo dúvidas e ajudando quando solicitado.

Por fim, a empresa OHM Technology que permitiu utilizar os seus produtos como exemplo e recebeu de braços abertos as propostas de melhoria bem como a solução oferecida pelo produto final do trabalho.

RESUMO

O GIP (*Global Interface Process*) é um sistema desenvolvido pela OHM Technology e tem como principal função prover segurança ao controle de acesso de uma empresa. Desenvolvido utilizando a linguagem Java e em parceria com a placa controladora *globalAccess*, permite a seus clientes controlar o acesso a ambientes e verificar variáveis do ambiente como temperatura, umidade, entre outras. Desenvolvida pela Google e oferecida desde 2007, a plataforma Android é uma solução completa e de código aberto exclusivamente para dispositivos móveis, com isso é possível desenvolver aplicações para diferentes dispositivos baseando-se apenas nesta solução sem a necessidade de grandes adaptações. Este trabalho tem como objetivo compreender o uso, arquitetura e funcionamento da plataforma Android e desenvolver uma aplicação que permitisse acessar o sistema GIP por um dispositivo móvel utilizando uma rede sem fio. Como resultado, obteve-se uma aplicação que pode auxiliar um funcionário a validar a instalação e funcionamento das placas de controle de acesso.

Palavras-chave: OHM Technology. Controle de acesso. Linguagem Java. Plataforma Android.

ABSTRACT

The GIP (Global Interface Process) is a system developed by OHM Technology and their main function is to provide security to the access control of a company. The GIP was developed using Java language and along with the globalAccess controller board, it allows clients to control the place access and check the environment variables such as temperature, humidity and others. Developed by Google and offered since 2007, the Android platform is a complete open source solution exclusively for mobile devices, this way it is possible to develop applications for different devices based solely on this solution without adaptations. This study has as objective to understand the use, architecture and operation of the Android platform and build an application that allows accessing to the GIP system through a mobile device using a wireless network. As a result, this work obtained an application that can help an employee to validate the installation and operation of access control boards.

Keywords: OHM Technology. Access Control. Java language. Android platform.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A empresa OHM Technology, localizada em Curitiba no estado do Paraná, atuante na área de segurança ao controle de acesso, oferece diversas soluções para seus clientes e dentre elas o sistema GIP (*Global Interface Process*) em conjunto com a placa controladora *globalAccess*. Esta solução permite aos clientes controlarem o acesso aos ambientes da empresa, bem como monitorar variáveis de ambiente como, por exemplo, temperatura, umidade e sensores de presença. Para uma maior facilidade de verificação do funcionamento das controladoras, é necessária uma aplicação móvel que permita esta verificação de maneira facilitada. Atualmente são necessários dois funcionários para validar a instalação e funcionamento de uma controladora *globalAccess*, um funcionário junto à placa testando suas entradas e saídas e outro acessando um computador junto ao sistema GIP (*Global Interface Process*) validando em tela os valores gerados.

A plataforma móvel Android é oferecida desde 2007, através da parceria OHA (*Open Handset Alliance*) entre a empresa de buscas Google e outras empresas da área de tecnologia. A plataforma móvel Google Android é inserida no mercado com o intuito de acelerar a inovação em dispositivos moveis e oferecer aos consumidores uma experiência mais rica, barata e melhor (GARGENTA, 2011). A plataforma Android, sendo uma solução de código aberto, oferece uma solução completa aos desenvolvedores de aplicativos e permite que uma mesma aplicação possa ser utilizada em diferentes dispositivos da plataforma.

Este trabalho estuda o funcionamento e a estrutura da plataforma Android para dispositivos móveis, e efetua a análise e desenvolvimento de uma aplicação baseada na plataforma visando integrar a solução OHM Technology com a solução Google Android. Com isso, pretende-se facilitar e melhorar a experiência do responsável pela verificação da placa controladora *globalAccess* através do sistema GIP (*Global Interface Process*) utilizando um dispositivo móvel.

1.1 PROBLEMA

Sistemas de segurança responsáveis por fazer um controle de acesso são cada vez mais necessários em empresas e instituições, sejam elas de qualquer área. Seus responsáveis se vêem obrigados a buscar uma solução que lhes permitam maior segurança juntamente com eficiência sobre quem entra, sai ou possa tomar qualquer outro tipo de ação dentro da organização.

Produtos como estes estão cada vez mais visíveis no mercado, o GIP (*Global Interface Process*) por sua vez, é responsável pela integração de vários sistemas de controle de acesso diferentes a uma só interface, além de trabalhar com seu hardware de forma direta, a placa controladora *globalAccess*. Esta placa tem como função controlar, através de suas saídas a relé, um interruptor acionado eletricamente, o acesso a locais controlados como, por exemplo, portas, catracas e cofres, bem como monitorar condições do ambiente, como, por exemplo, temperatura, umidade e estados de sensores.

Apesar dos sistemas oferecidos possuírem acesso remoto em sua intranet e muitas vezes até mesmo pela internet, através do browser, nenhum destes sistemas integrados ao GIP oferece uma maneira de interação através de um dispositivo móvel para ser utilizado pelo usuário quando este estiver em campo, atendendo algum chamado ou apenas realizando manutenção de rotina. Atualmente são necessárias duas pessoas em locais diferentes para aferir o funcionamento de uma placa de controle de acesso. A primeira tem como função de realizar alguns comandos junto à placa em si e a segunda acessando o sistema efetuando os comandos.

Devido a dificuldade de aferir o funcionamento das placas de controle de acesso que se encontram longe de um ponto de acesso, faz-se necessário dois funcionários para tal se comunicando, quando possível, via radio, tornando o processo mais complexo e suscetível a erros.

1.1.1 Proposta

Através da compreensão da plataforma *Android* para dispositivos móveis implementar uma interface com o sistema de controle de acesso GIP (*Global Interface*

Process) que permita aos usuários, responsáveis pela instalação, configuração e manutenção das placas *globalAccess,* efetuar a leitura das entradas, alterar o estado das saídas, bem como acessar um pequeno relatório de transações.

1.1.2 Objetivo Geral

Analisar e implementar um aplicativo baseado na plataforma Android para dispositivos móveis que, através de uma interface com o sistema GIP (*Global Interface Process*), controle e diagnostique a placa de controle de acesso *globalAccess*.

1.1.3 Objetivos Específicos

- Compreender a plataforma Android para dispositivos móveis e o processo de desenvolvimento de aplicativos;
- Elaborar e aplicar uma análise sobre o problema, para então planejar o sistema a ser desenvolvido;
- Desenvolver um aplicativo para solução do problema que controle as saídas, leia as entradas e permita a geração de um pequeno relatório de transações.

1.2 ORGANIZAÇÃO DO TRABALHO

O Capítulo 2 descreve o uso, funcionamento e importância de sistemas de controle de acesso.

O Capítulo 3 detalha uma revisão bibliográfica sobre a plataforma Android e sua utilização.

O Capítulo 4 apresenta a solução encontrada para o problema, bem como o projeto, especificação, implementação e validação da aplicação.

O Capitulo 5 mostra uma análise dos resultados obtidos com a execução deste trabalho e implementação da aplicação como solução.

Por fim, o último Capítulo apresenta as conclusões finais sobre este trabalho e relata possíveis trabalhos futuros que podem ser desenvolvidos a partir deste trabalho.

2 SISTEMAS DE SEGURANÇA AO CONTROLE DE ACESSO

A segurança de uma empresa é um quesito de grande importância. Cada vez mais as empresas estão preocupadas com o controle de acesso dos funcionários dentro do ambiente da empresa, uma vez que nem todos os funcionários podem ter acesso a todos os locais da empresa. Um exemplo que pode ser citado é com relação à sala de servidores da empresa, onde se encontram os computadores com todos os dados da empresa.

Sistemas de controle de acesso tem como função aferir todo o acesso aos ambientes que controla, verificando usuários e permissões destes, permitindo ou bloqueando o acesso. Algumas soluções, além do controle de acesso, possuem também automação de ambientes verificando o estado destes ambientes e controlando-os conforme necessário.

Através das diferentes configurações das controladoras é possível controlar catracas, torniquetes, portas, portões e cancelas de acordo com as necessidades do cliente. Em alguns casos, os sistemas de controle de acesso são facilmente adaptáveis. Através de um controle de acesso, é possível, além do controle de ambiente, a geração de relatórios, manutenção de recursos humanos e controle de terceiros. Conforme o nível de acesso do usuário é possível acessar diferentes áreas do sistema e do ambiente.

O sistema GIP (*Global Interface Process*) juntamente com a placa controladora *globalAccess* desenvolvidos pela empresa OHM Technology, localizada em Curitiba no estado do Paraná, oferece uma solução bastante completa para segurança ao controle de acesso e automação de ambientes. Os dois produtos serão descritos a seguir.

2.1 SOFTWARE GLOBAL INTERFACE PROCESS

O sistema GIP (*Global Interface Process*), desenvolvido totalmente em linguagem de programação Java, tem como principal função a integração de sistemas de recursos humanos e sistemas de controle de acesso. Dentre suas características, pode-se citar a fácil integração com diferentes sistemas, além do controle de placas de acesso unificado utilizando uma solução totalmente baseada em web, ou seja, onde o usuário final pode acessar o sistema utilizando qualquer computador de qualquer lugar de sua intranet ou até mesmo via internet necessitando apenas informar seu usuário e senha. A Figura 1 exibe a captura de tela do

sistema acessado via browser, nela pode-se verificar sua interface, baseada em ícones, menus e janelas, e o funcionamento da tela gráfica do sistema, que permite o operador saber como está o ambiente no momento, através da leitura dos estados de entrada e saída da placa *globalAccess*.

Figura 1 - Captura de tela do sistema GIP.
Fonte: Autoria própria.

Através do GIP é possível criar diferentes grupos de usuários os quais podem acessar qualquer parte do sistema com diferentes níveis de permissão, sendo eles: leitura, gravação e exclusão. O controle de recursos humanos permite que sejam adicionados os funcionários da empresa podendo ser permanentes ou temporários, ficando a cargo do sistema a manutenção dos mesmos permitindo ou bloqueando o acesso desses funcionários nas datas pré-determinadas. Em conjunto com o controle de recursos humanos o controle de acesso trabalha para que os envolvidos possam, junto com um cartão de acesso, se movimentar dentro da instituição.

O controle de acesso feito pelo GIP se dá em parceria com a placa *globalAccess*, que com suas entradas digitais ou analógicas e saídas digitais, permite o total controle sobre os ambientes liberando ou bloqueando o acesso a eles utilizando-se da ligação com portas, portões e catracas à suas saídas a relé. O sistema permite também a automação de ambientes

ligando às saídas da placa os dispositivos desejados, como luzes, ventiladores, ares-condicionados, ou qualquer que seja o item e necessidade de automação.

2.2 PLACA CONTROLADORA *GLOBALACCESS*

A controladora de acesso *globalAccess* pode ter seu funcionamento tanto em modo *offline*, baseado nas configurações e permissões de acesso previamente armazenadas em sua memória interna, ou conectado ao sistema GIP, enviando e recebendo informações em tempo real. Pode controlar o acesso de até 4 ambientes, com leitor de cartão na entrada e botão de saída, ou 2 ambientes, com leitores de cartão na entrada e na saída. Possui 8 saídas a relé ou transistor e 8 entradas universais configuráveis para monitoramento de sensores de porta, perímetro, temperatura, umidade e outros.

Sua configuração é totalmente flexível, permitindo ao instalador escolher para cada ambiente controlado, as entradas de leitor de cartão, entradas de campo e saídas controladas que melhor irão atender as necessidades do local. A Figura 2 exibe a captura de tela da janela de configuração a placa *globalAccess* via sistema GIP, onde podem ser vistas as saídas a relé a direta, entradas de ambiente ao topo e entradas de leitores na parte inferior.

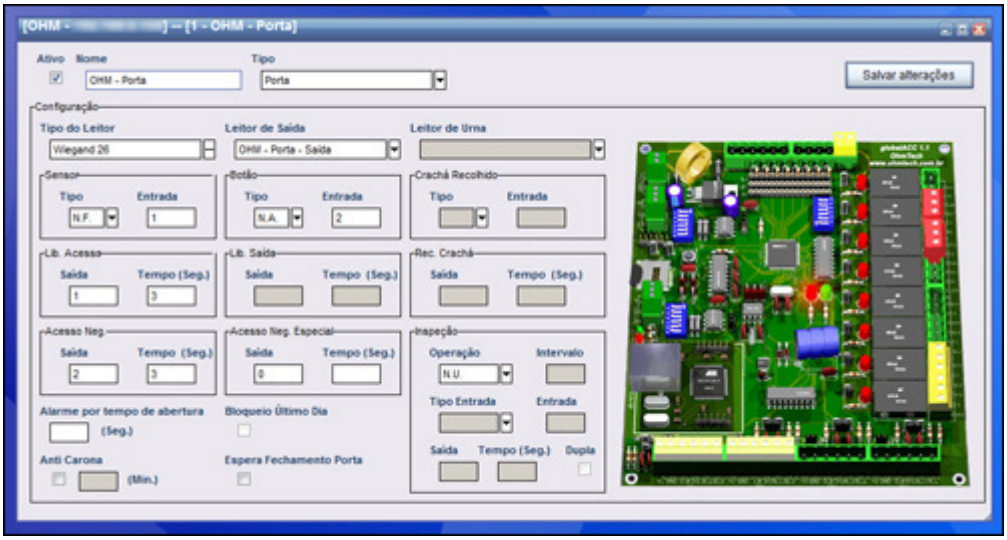

Figura 2 - Captura de tela de configuração da placa *globalAccess*.
Fonte: Autoria própria.

Cada controladora *globalAccess* suporta até 524.288 usuários cadastrados e possui memória para até 11 milhões de eventos em fila e aceita os padrões de leitores de cartão

Wiegand26, *Wiegand32*, *WiegandCustom*. Sendo *Wiegand* uma interface definida para leitores de cartão a qual baseia-se na utilização de três pares de fios para alimentação, dados e sinalização de leds.

Todo processamento local é realizado por um microcontrolador ATMEGA128 da fabricante ATMEL. Possui 64K Bytes de memória NVRAM, *Real Time Clock* com precisão de 19ppm o que significa uma margem de erro de 10 minutos ao ano que pode ser corrigido através de atualização pelo sistema GIP, 2 *gigabytes* de memória flash através de um cartão de memória e comunicação TCP/IP em barramento 10/100Mbps.

Através de um conjunto de chaves é possível configurar o endereço lógico da controladora, formatar sua memória e alterar o tipo de comunicação, para RS232 ou RS485, dos canais seriais da mesma.

Dentre as funcionalidades desenvolvidas para esse equipamento estão a possibilidade do software da controladora ser atualizado remotamente através do sistema GIP permitindo a adição de novas funcionalidades e correção de erros sem que a mesma necessite ser enviada para o fabricante. Além do envio de comandos via canal auxiliar de comunicação para outros equipamentos e lógicas de inspeção para o ambiente controlado, a *globalAccess* faz a contagem das liberações de acesso e conforme sua configuração que pode ser aleatória, sequencial ou remota, faz o bloqueio de passagem do usuário e ativa um sinal de aviso para que sejam tomas as providências em relação ao usuário bloqueado.

3 SISTEMA OPERACIONAL ANDROID

Observando o crescimento intenso na área de telefonia móvel, em 2005 a empresa de buscas Google adquire a empresa Android Inc focando os objetivos da mesma no desenvolvimento para plataformas móveis. Em novembro de 2007, juntamente com o anúncio do novo produto da Apple, o Iphone, o Google revela uma parceria com mais 33 outras empresas, a OHA (*Open Handset Alliance*), sendo elas operadoras de celular, fabricantes de aparelhos, semicondutores, software e comércio, buscando acelerar a inovação em dispositivos móveis e oferecer uma melhor experiência aos usuários finais (GARGENTA, 2011). Ao mesmo tempo em que é feito o anúncio da parceria, o primeiro pacote de desenvolvimento para a plataforma é liberado em sua versão beta tornando público o sistema operacional para plataformas móveis Google Android. O logotipo com o mascote do Android é exibido na Figura 3.

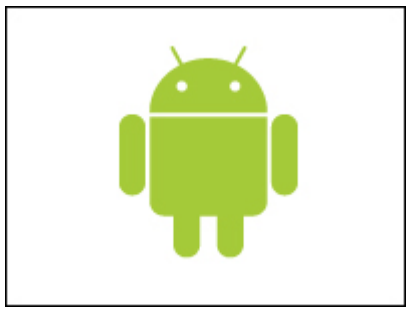

Figura 3 - Logotipo e mascote do sistema Android.
Fonte: Android Media.

Visando revolucionar o mercado de dispositivos móveis, pela primeira vez uma plataforma aberta separa o hardware do software que trabalha sobre ele (GARGENTA, 2011). Permitindo assim diferentes dispositivos utilizarem a mesma aplicação sem necessidade de que a mesma tenha que ser adaptada a cada situação oferecendo uma solução completa e bem estruturada para os desenvolvedores de aplicativos para celulares. Isso ocorre, pois com a criação de uma única plataforma para vários modelos de celulares há uma situação de ganho-ganho, onde os desenvolvedores podem produzir conteúdo que rodam em diferentes aparelhos e as empresas de celulares que podem oferecer uma gama maior de aplicativos para seus clientes.

A plataforma Android foi desenvolvida especificamente para soluções móveis, atendendo as necessidades dos dispositivos e respeitando suas limitações. É uma plataforma

de código aberto significando que qualquer pessoa que desejar pode visualizar e alterar sua estrutura como bem entender sem necessidade de alguma licença de uso ou direitos autorais (GARGENTA, 2011). Como desenvolvedor de software isso permite o melhor entendimento do sistema para futuras aplicações. Como desenvolvedor de hardware, é possível adaptar de maneira mais precisa ao seu produto, fazendo com que o mesmo utilize o máximo que seu hardware pode oferecer.

A plataforma Android permite também a integração dos serviços oferecidos pela própria Google, como Google *Maps*, Google *Reader*, Google *Docs* e outros. Além do desenvolvimento de aplicativos facilitado, o Android conta hoje com a loja de aplicativos online, a Android *Market*, anunciada em agosto de 2008. Nela é possível os desenvolvedores disponibilizarem seus aplicativos de forma paga ou gratuita aos usuários finais que podem se beneficiar desses produtos. Para as soluções pagas os usuários tem um período de até 15 minutos para avaliar e desistir da aquisição.

3.1 VERSÕES

A plataforma Android, como consequência de um software em constante atualização, se encontra na versão 3.0 com o apelido de *Honeycomb*. Suas versões estão em constantes mudanças, algumas delas menores apenas por pequenas correções em seu uso e outras maiores como em melhoria de desempenho e seu nível de API (*Application Programming Interface*). Cada versão juntamente com seu nível de API permite ao desenvolvedor saber quais funcionalidades terá a disposição e os dispositivos que poderão receber sua aplicação. O Quadro 1, exibido na página seguinte, relaciona as diferentes versões do Android, seus níveis de API e apelidos.

Versão	Nível de API	Apelido
Android 1.0	1	
Android 1.1	2	
Android 1.5	3	*Cupcake*
Android 1.6	4	*Donut*
Android 2.0	5	*Eclair*
Android 2.01	6	*Eclair*
Android 2.1	7	*Eclair*
Android 2.2	8	*Froyo*
Android 2.3	9	*Gingerbread*
Android 2.3.3	10	*Gingerbread*
Android 3.0	11	*Honeycomb*

Quadro 1 - Versões do sistema Android.
Fonte: Gargenta, 2011.

A partir das versões o desenvolvedor pode escolher qual usar e saber o público que poderá estar alcançando. A Gráfico 1 demonstra a distribuição da utilização das diferentes versões da plataforma Android pelos usuários em janeiro de 2011 (GARGENTA, 2011).

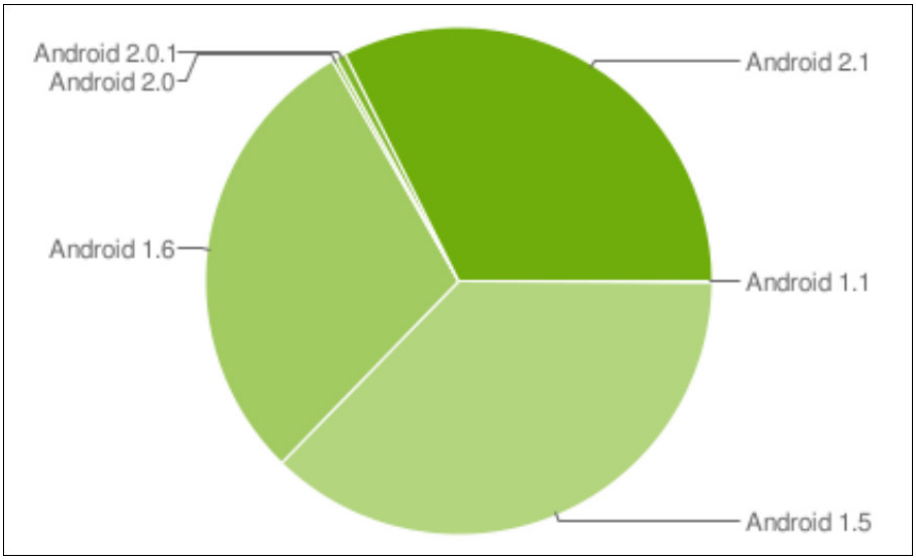

Gráfico 1 - Distribuição de uso da plataforma Android em janeiro de 2011.
Fonte: Gargenta, 2011.

3.2 DESENVOLVIMENTO DE APLICATIVOS MÓVEIS

Para desenvolvedores já habituados à programação utilizando a linguagem Java, o processo de criação de um aplicativo utilizando a plataforma Android se dá de forma até mesmo simplificada, bastando entender alguns conceitos de como o mesmo funciona. Rodando sobre Linux utilizando Java é possível, através do conceito de orientação a objetos referente ao reuso de código, transportar uma aplicação totalmente desktop para uma aplicação móvel com pequenos ajustes desde que respeitadas as limitações de hardware. A plataforma Android tem potencial para derrubar barreiras em direção ao sucesso em desenvolvimento e vendas de aplicativos de telefonia móvel devido sua portabilidade e facilidade de uso e implementação dos produtos para o mesmo (ROGERS et al., 2009).

3.2.1 Plataforma

A plataforma é desenvolvida com base no sistema operacional Linux, se constitui por um conjunto de ferramentas que permitem atuar em todas as fases do projeto para a criação de uma aplicação específica. Mesmo tendo sido baseado no sistema Linux, Pereira e Silva (2009) não consideram o sistema operacional Android um Linux completo por não possuir alguns componentes e padrões característicos das plataformas Linux, como, por exemplo, suporte a algumas bibliotecas e componentes de interface encontrados nos sistemas correlatos. A plataforma Android é uma solução completa para tecnologia móvel, oferecendo um pacote com diversos programas para celular, já com um sistema operacional, aplicativos e interface para o usuário.

Foi construída de modo a oferecer o maior aproveitamento dos dispositivos móveis pelos desenvolvedores durante a criação de uma aplicação. Por ser verdadeiramente aberto, permite o desenvolvedor acessar a qualquer funcionalidade do núcleo do telefone, como chamadas, mensagens, câmera e outras para que sejam adaptadas e aprimoradas visando sempre a evolução do sistema e melhor experiência de uso pelo usuário final (PEREIRA; SILVA, 2009).

Por ser uma solução de código aberto, à medida que novas tecnologias surgem, estas podem ser incorporadas rapidamente. Isso ocorre, pois as empresas responsáveis podem

trabalhar sobre a plataforma, permitindo assim, que a comunidade possa adaptar facilmente essas tecnologias a seus dispositivos.

3.2.1.1 Arquitetura

A arquitetura é construída em camadas, onde cada uma tem suas características específicas. A arquitetura da plataforma é dividida em 5 partes, sendo elas, da mais interna para mais externa, núcleo, bibliotecas, ambiente de execução, framework de aplicação e aplicativos. A Figura 4 exibe como é construída essa arquitetura e a composição de cada camada.

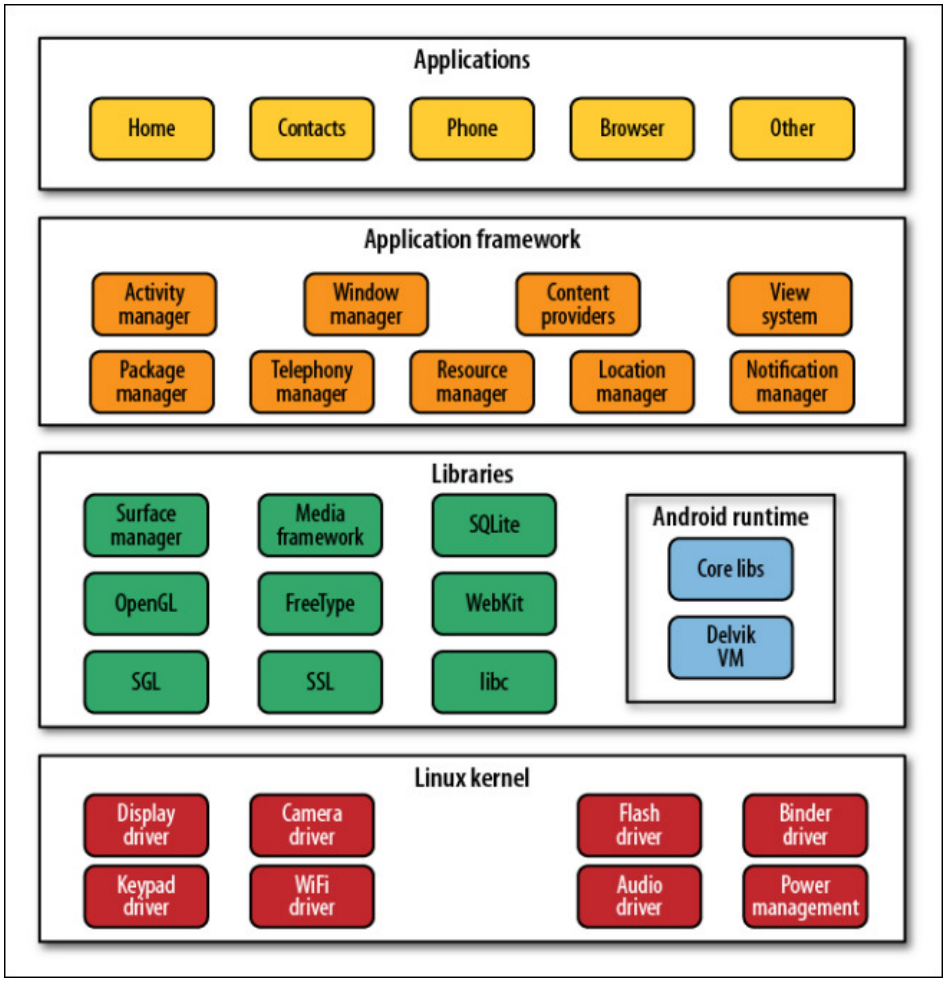

Figura 4 - Arquitetura do sistema Android.
Fonte: Gargenta, 2011.

A primeira camada (*Linux kernel*) contempla o núcleo do sistema baseado em Linux que foi escolhido para esse papel dentro da plataforma Android por diversos motivos, entre eles ser um bom sistema operacional, oferecer portabilidade e segurança além de possuir seu código de forma aberta. Permite que seja feita a divisão entre o hardware e o software do dispositivo. É responsável por diversas e importantes funções do sistema como gerenciamento de energia, memória, processos e todos os *drivers* que venham ser necessários para controle como do display, da câmera, do módulo de conexão sem fio, entre outros (PEREIRA; SILVA, 2009).

Em seguida a camada de bibliotecas (*Libraries*) oferece todo o suporte que o sistema operacional pode necessitar. Estas bibliotecas podem prover às aplicações mais complexas, por exemplo, o desenvolvimento gráfico, acesso a banco de dados, internet. Desenvolvidas em C/C++ são utilizadas por diversos recursos do sistema e podem ser acessadas através da camada superior pelos desenvolvedores. Entre as bibliotecas oferecidas encontram-se algumas de mais importância para o desenvolvimento de aplicações, como a *Webkit*, um renderizador de páginas da internet, a *SQLLite*, uma ferramenta de acesso ao banco de dados da aplicação, a *SGL* e a *3D libraries* que oferecem aceleração e criação de gráficos em 2D e 3D.

A camada de ambiente de execução (*Android runtime*) é responsável pelas aplicações em execução possuindo uma instância da máquina virtual *Dalvik* para cada. Projetada para executar várias máquinas virtuais em paralelo, foi desenvolvida para obter melhor desempenho e integração com hardware e sistemas com menor potência e memória. Para o sistema operacional, limitado fisicamente, oferece otimização de consumo de memória, bateria e processador. Suas bibliotecas oferecem grande parte das funcionalidades encontradas na linguagem Java, que depois de escritas nessa linguagem são compiladas para serem executadas na máquina virtual *Dalvik* permitindo assim que a aplicação seja executada em qualquer dispositivo Android.

Os recursos necessitados pelos aplicativos encontram-se na camada do *framework*, estes recursos são responsáveis pelo funcionamento das aplicações e oferecem acesso a diversos pontos do sistema, como a outras aplicações e gerenciadores, como o do GPS e conexão sem fio. Todas as funcionalidades oferecidas pelo sistema podem ser acessadas pela aplicação através das APIs oferecidas por essa camada.

Na camada de aplicações (Applications), superior a todas as outras, permanecem os aplicativos escritos em Java, sejam eles nativos do sistema, inseridos pelo fabricante ou instalados pelo usuário. Como, por exemplo, pode-se citar os clientes de email, navegadores,

calendários e jogos, além de outros que os fabricantes e a comunidade podem oferecer em seus sites ou no próprio Android *Market*.

3.2.2 Definições e Permissões

Cada aplicação desenvolvida para Android deve, obrigatoriamente, possuir um arquivo em seu diretório raiz que dispõe de todas as suas definições e permissões. Este arquivo descreve as classes da aplicação, os dados que ela irá utilizar e os recursos do dispositivo que ela irá necessitar. Nele também estão dados sobre nome, descrição e versão da aplicação.

O arquivo "AndroidManifest.xml", responsável por conter essas definições e permissões, possui uma estrutura em XML (*Extensible Markup Language*) única onde cabe ao desenvolvedor apenas preenchê-la. Todos os recursos que serão utilizados pela aplicação devem estar contidos no arquivo, como por exemplo, caso seja necessário para a aplicação acessar a internet a permissão de nome "*android.permission.INTERNET"* deve estar presente. São diversas permissões para diferentes situações e usos, para isso, os desenvolvedores podem verificar as permissões que serão utilizadas junto à documentação da plataforma Android.

3.2.3 Conceito

Todas as aplicações escritas em linguagem Java para Android depois compiladas geram um arquivo com extensão ".apk". Estes podem ser utilizados pelos desenvolvedores para distribuir os aplicativos para que os usuários possam instalá-los em seus dispositivos. Depois de instaladas estas aplicações podem possuir diferentes e importantes componentes que o sistema por instanciar sempre que seja preciso, sendo eles *activities, services, broadcast receivers* e *content providers*.

Activity é o componente mais utilizado, na maioria dos casos representados por uma tela na aplicação. É constituído de varias telas que fazem a interface com os usuários e

respondem aos comandos do mesmo. Cada *activity* pode assumir diversos estados onde estes irão definir seu ciclo de vida, a Figura 5 representa um ciclo de vida completo de uma *activity*.

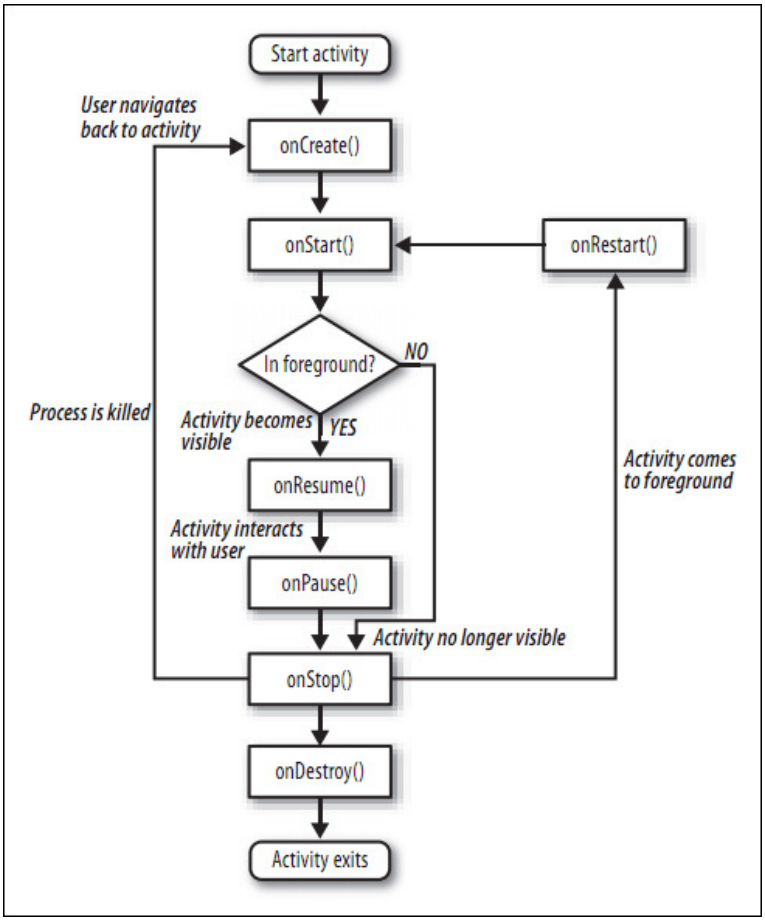

Figura 5 - Ciclo de vida de uma *activity* no sistema Android.
Fonte: Rogers et al, 2009.

Os principais estados de uma *activity* são (PEREIRA; SILVA, 2009):

- *OnCreate*(): Quando a aplicação é iniciada;
- *OnStart*(): Após a aplicação ser iniciada, é chamado este método quando ela se torna visível para o usuário;
- *OnResume*(): Método chamado assim que a aplicação é iniciada;
- *OnPause*(): Antes da aplicação entrar em modo de espera, o método é chamado para que seja possível tomar as devidas providencias e armazenar as informações que podem ainda não ter sido persistidas;
- *OnStop*(): Quando a aplicação está para ser parada por não estar mais sendo utilizada ou ter perdido o foco, este método é acionado;

- *OnDestroy*(): Pode ser acionado no fim da activity, ou pela necessidade do sistema para liberação de recursos;
- *OnRestart*(): Após a aplicação ser interrompida, quando a mesma volta a se tornar ativa, este método é acionado.

Outro componente essencial para o sistema operacional é o *service* que é um conjunto de códigos sem interface alguma, sem interação direta do usuário e rodam em segundo plano que, diferente das *activities* não possuem um ciclo de vida curto e definido. O *service* não é fechado quando o usuário percorre entre as aplicações, só é fechado definitivamente através de um comando direto para isso. O usuário pode fechar um *service* se conectando no mesmo e efetuando a ordem ou através do próprio sistema operacional forçando seu fechamento.

Já os *broadcast receivers* são componentes responsáveis pela reação de uma aplicação ao receber um *intent* de um evento. Um evento pode ser gerado na mudança de estado do sistema, numa chamada recebida, na detecção de redes sem fio, num alarme do relógio, entre outros. Estes eventos devem ser tratados pela aplicação que não precisa estar necessariamente ativa no momento para ser acionada.

O componente *content provider* pode ser utilizado na necessidade de compartilhar informações entre aplicações. Permite que uma aplicação disponibilize um diretório comum para que outras aplicações possam acessar seus dados bem como os alterar dessa forma compartilhar dados importantes para o funcionamento das duas ou mais aplicações envolvidas.

A maioria dos componentes podem ser ativadas por mensagens assíncronas que, além do conteúdo dessas mensagens, contém os dados necessários para a aplicação, chamadas de *intents* essas mensagens podem dizer qual ou quais são as aplicações destinos e qual a ação que a aplicação deve tomar. Estas mensagens podem ser explícitas quando enviadas diretamente a uma aplicação destino ou implícitas quando enviadas aos filtros do sistema para que as aplicações registradas para aquela *intent* a receba.

3.2.4 Armazenamento de dados

Em sistemas desenvolvidos para desktop e servidores, na maioria dos casos, o acesso aos arquivos podem ser efetuados por qualquer aplicação, bastando no máximo informar as

credenciais de acesso como usuário e senha. Já os sistemas desenvolvidos utilizando a plataforma Android possuem acesso apenas aos dados que lhe são relacionados, nenhuma aplicação possui acesso aos dados das outras aplicações, na necessidade de disponibilizar dados entre as aplicações deve ser utilizando *Content Provider* discutido no tópico 3.2.2.

O sistema de armazenamento de dados da plataforma Android oferece quatro maneiras de armazenar e recuperar dados no dispositivo, sendo elas, via preferências, arquivos, bases de dados e rede.

O primeiro modo que pode ser utilizado é o via preferências, que permite armazenar pares de valores de tipos primitivos, geralmente utilizado para manter as preferências do usuário de maneira simplificada. Para utilizar as preferências pode ser vinculado um nome as mesmas ou não, caso não seja vinculado um nome essa preferência será perdida no fim da aplicação sendo assim apenas um dado temporário.

A segunda maneira possível de armazenar dados no Android é através da utilização de arquivos, que pode ser diretamente no dispositivo ou num cartão de memória removível. Os arquivos devem ser acessados pela aplicação através do nome e caminho completo do mesmo. Outra aplicação, mesmo que possua os nomes e caminhos dos arquivos da aplicação, não terá acesso aos mesmos, pois o acesso sempre é feito de maneira local pelas aplicações.

É possível também utilizar uma pequena ferramenta, simples, porém eficiente, que funciona como um sistema gerenciador de banco de dados, o SQLite. Esta ferramenta permite criar tabelas, inserir, manipular e consultar dados nessas tabelas que ficam armazenadas em arquivo em disco onde são acessados apenas pelo SQLite. Esta maneira de armazenar dado se mostra muito eficiente e, segundo Pereira e Silva (2009), sua utilização é de vital importância devida sua simplicidade de uso em relação aos recursos oferecidos.

Apesar de não muito utilizada é possível armazenar dados através da rede, que quando disponível, permite armazenar e recuperar dados em locais de rede se baseando em protocolos de internet.

3.2.5 Acesso à Web Services

Em muitos casos, há a necessidade de dois sistemas de natureza diferentes comunicarem-se, porém não foram desenvolvidos de forma que permitam a comunicação facilitada ou até mesmo por suas plataformas não permitirem. Para tal, *web services* podem

ser utilizados para resolver o problema, segundo Rabelo (2011), da dificuldade de interação entre sistemas. O acesso à *web services* tem como função, através de esquemas simplificados em XML, permitir que duas aplicações desenvolvidas em linguagens diferentes possam se comunicar por meio de serviços.

O sistema Android não oferece uma solução nativa para consumo de *web services*, porém, através do protocolo SOAP (*Simple Object Access Protocol*) e utilizando a biblioteca KSOAP2 é possível efetuar acessos a *web services* em dispositivos móveis. Para tal acesso, além da biblioteca que permitirá a aplicação consumir um *web service*, é também necessária que seja adicionada ao projeto a permissão de acesso a internet pelo dispositivo.

Depois de adicionadas as bibliotecas e a permissão ao projeto, o processo de consumo de um *web service* se dá de forma simplificada através de três classes da biblioteca, *SoapObject* e *SoapSerializationEnvelope*, responsáveis pelas informações, parâmetros e envelopamento da requisição, e a classe *HttpTransportSE* a qual efetivamente irá transportar essas informações ao servidor e retornar a resposta em relação a requisição.

O retorno dos valores pelo servidor utilizando tipos primitivos é feito através da classe *SoapPrimitive*, e da classe *SoapObject* para objetos complexos, cabendo a aplicação saber o tipo retornado e o tratar da melhor maneira dentro de seu interesse (USHISIMA, 2011).

3.2.6 Interface com o Usuário

A interface com o usuário se baseia nos objetos *views* e *viewgrpoups* oferecidos pela plataforma Android. Estes objetos formam a base da interface no sistema, organizando hierarquicamente, onde se representada em forma de árvore, os *viewgroups* seriam os ramos dessa árvore e as *views* suas folhas. A Figura 6 mostra a hierarquia da interface com o usuário representada em forma de árvore.

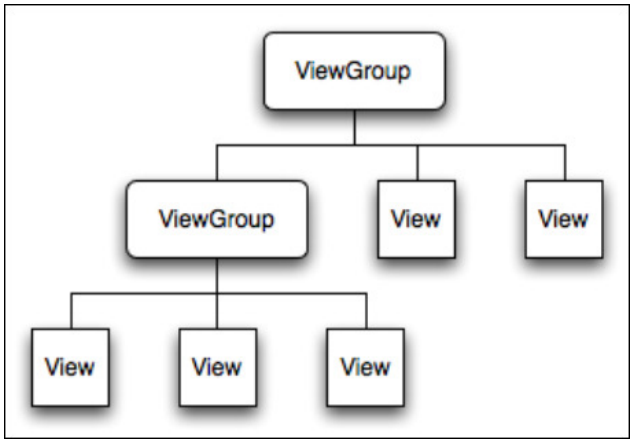

Figura 6 - Hierarquia da interface com o usuário no sistema Android.
Fonte: Android Developers.

Uma *view* é a representação do que o usuário verá durante a execução de uma aplicação, nada mais é que um retângulo, sendo ele na horizontal ou vertical, vazio que pode receber diversos componentes com diferentes tipos interação cada. Já *viewgroup* é uma *view* que irá conter e gerenciar uma ou mais *views* e até mesmo outros *viewgroups*. Dos itens que podem ser adicionados às *views*, os principais são:

- *TextView*: Caixa de exibição de textos, utilizada para exibir alguma informação ao usuário, como um titulo ou nome de um campo;
- *EditText*: Caixa de entrada de texto, utilizada para capturar alguma informação do usuário podendo ser formatada;
- *Button*: Botão de ação que pode ser utilizado para oferecer alguma ação ao usuário, como salvar ou excluir dados;
- *RadioButton*: Botão de dois estados onde o usuário pode selecionar apenas uma opção dentre as oferecidas, no preenchimento do sexo por exemplo onde pode ser selecionado apenas masculino ou feminino;
- *Checkbox*: Marcador de dois estados onde o usuário pode selecionar nenhum ou mais de uma opção das exibidas, num cadastro de interesses é comumente usado;
- *TogleButton*: Botão de dois estados, ativo e inativo, útil ao exibir o estado de um item qualquer da aplicação onde este possua apenas a possibilidade de estar ativado ou desativado;
- *ImageButton*: Botão de ação que pode ter uma imagem vinculada, utilizado para mais personalização da aplicação;
- *ImageView*: Imagem simples, para a exibição de ícones ou até mesmo fotos que a aplicação possa conter.

Os itens, depois de adicionados a uma *view*, possuem diversas propriedades que podem ser configuradas conforme a necessidade e objetivo para o mesmo em relação à interface. A propriedade ID pode ser definida como uma referência, de preferência única, para aquele item. Através das propriedades, por exemplo, é possível definir a posição e tamanho do item, cor do texto e de fundo, tamanho do texto, entre outros.

Estes itens podem ser organizados na tela para o usuário seguindo um dos padrões pré-definidos pelo sistema, que podem ser *FrameLayout*, modelo mais simples onde se pode adicionar apenas um item, *LinearLayout*, organiza os itens um sobre os outros como se fosse uma coluna, *RelativeLayout*, posiciona os itens em relação a eles mesmos ou seu item superior, *TableLayout*, como se fosse uma tabela exibe os itens em celular e colunas, e *AbsoluteLayout*, já obsoleto onde era possível definir exatamente a posição de cada item exibido. Todo o processo de criação da interface e disposição dos itens pode ser feito através de uma ferramenta de criação qualquer que, por fim, gera o resultado em XML ou diretamente em XML o que permite ao desenvolvedor maior liberdade na criação e edição do *layout* da aplicação.

3.3 DISTRIBUINDO UMA APLICAÇÃO

Criar aplicações Android para uso próprio pode ser interessante, porém, em algum momento o desenvolvedor pode desejar distribuir sua solução com outros usuários que possuam também dispositivos com a plataforma Android. Depois de finalizada uma aplicação que será distribuída, esta deve ser empacotada gerando um arquivo com extensão .APK que é composta pelos seguintes componentes:

- Arquivos executáveis para a máquina virtual: Estes representam todo o código da aplicação depois de compilado para ser executado pela máquina virtual do Android, a *Dalvik*.
- Recursos: São todos os itens que a aplicação necessita que não seja código, como arquivos de imagem, áudio, vídeo e outros.
- Bibliotecas: Eventualmente durante a implementação da aplicação podem ser utilizadas algumas bibliotecas que serão adicionas ao pacote no momento da criação do arquivo final da aplicação.

Com o arquivo da aplicação criado é possível testá-la diretamente num dispositivo Android. Apesar de nesse ponto já ser possível antes iniciar sua distribuição se torna interessante assinar digitalmente a aplicação, identificando o desenvolvedor como criador e responsável pela mesma. Para assinar uma aplicação é necessário criar uma chave que será usada sempre que for assinada uma nova aplicação, essa chave contém dados como nome do desenvolvedor, empresa desenvolvedora, nome fantasia, cidade e estado.

Após preparada e testada localmente a aplicação, é possível distribuí-la para outros usuários, dentre as diversas opções de distribuição, é possível enviar o arquivo final diretamente aos usuários fazendo todo o processo de maneira manual, assim como se pode utilizar alguma loja de aplicativos como a Android *Market*.

Para distribuir via a loja de aplicativos do Google, basta o desenvolvedor se cadastrar, adquirir uma licença de desenvolvedor através do pagamento de uma taxa única de inscrição e publicar suas aplicações, que podem ser gratuitas ou pagas. Em caso de aplicações pagas, fica reservada uma comissão de 30% para a loja em relação ao valor estipulado para o aplicativo. A utilização de lojas de aplicativos ao invés de distribuí-los manualmente se mostra útil quando há o desejo de atingir um grande número de usuários.

4 CRIAÇÃO DE UMA APLICAÇÃO DE CONTROLE E DIAGNÓSTICO PARA UM SISTEMA CONTROLE DE ACESSO

Como visto anteriormente no Capítulo 2, sabe-se que um controle de acesso é um recurso de grande importância dentro de uma empresa. Porém no caso do sistema de controle de acesso GIP, pode-se encontrar na manutenção e diagnóstico das placas *globalAccess* alguma dificuldade de gerenciamento, visto que atualmente são necessárias duas pessoas em locais diferentes para aferir seu funcionamento. A primeira tem como função de realizar alguns testes de ligações junto à placa em si e a segunda acessando o sistema verificando os comandos e ligações. Por essa razão, faz-se necessário dois funcionários se comunicando muitas vezes à distancia, quando possível, via rádio, tornando o processo mais complexo e suscetível a erros.

Com a finalidade de resolver esse problema, a proposta deste trabalho é desenvolver uma solução para a necessidade de obter dados do sistema sobre uma controladora a partir de um dispositivo móvel através de uma rede sem fio. Para isso, foi criada uma aplicação capaz de se conectar ao servidor do sistema GIP através de um *web service*, capturar os dados necessários e retorná-los de maneira simples para o usuário, neste caso, o funcionário que estará aferindo o funcionamento da controladora.

O processo de desenvolvimento irá consistir na especificação, análise e projeto da solução, bem como na implementação e validação da aplicação. A descrição do desenvolvimento desse aplicativo será apresentada nas seções seguintes. Primeiramente será apresentada a descrição da análise e projeto do sistema, mostrando diagramas de classes e modelo de banco de dados. Em seguida serão mostrados detalhes da implementação da interface do aplicativo. Por fim serão descritas as abordagens utilizadas para validar o aplicativo.

4.1 ESPECIFICAÇÃO, ANÁLISE E PROJETO

O produto criado é uma aplicação para dispositivos móveis que utilizam a plataforma Android. Deverá, com o menor número de telas e da maneira mais simples possível, permitir

a conexão do usuário para com o sistema GIP provendo o acesso às controladoras do sistema e os dados referentes a essas controladoras.

Buscando suprir as necessidades da aplicação, o projeto foi dividido em duas partes, uma para a aplicação em si e outra sendo apenas uma interface para o *web service* a ser realizado pelo servidor.

Para a especificação da solução para a aplicação foram utilizados dois diagramas, de classes para modelagem orientada a objetos e de entidade relacionamento para criação do banco de dados. Foram elencadas 5 classes de valores para a aplicação além do servidor de serviço, sendo elas, Servidor, Controladora, Transação, Entrada e Saída. A Figura 7 exibe o diagrama de classes da aplicação.

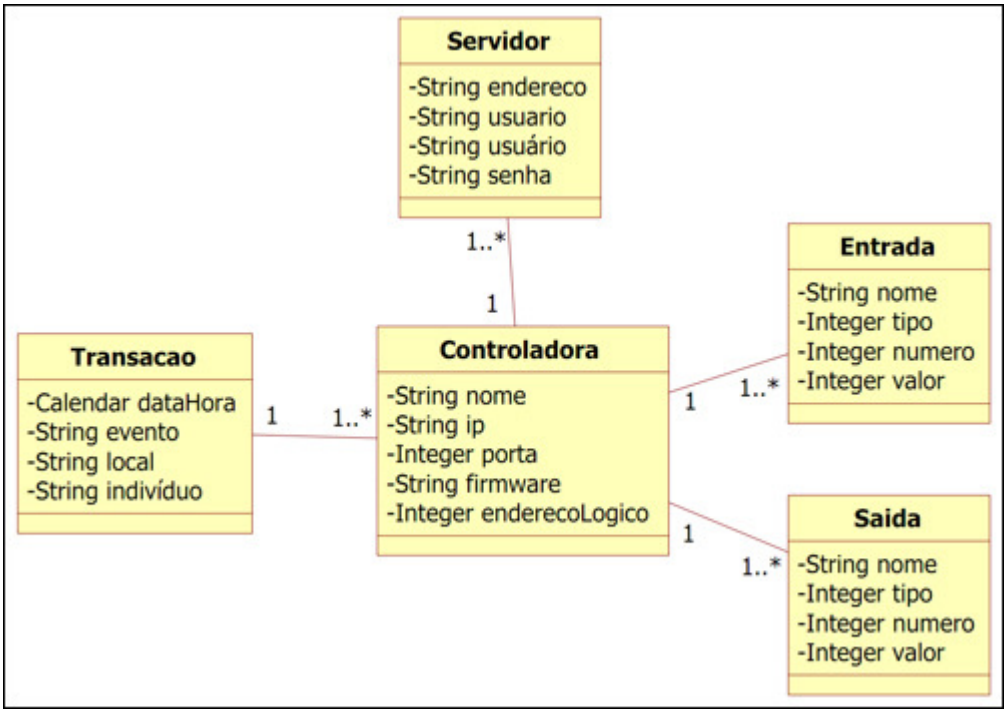

Figura 7 - Diagrama de classes da aplicação.
Fonte: Autoria própria.

Além das classes de dados do diagrama, foi preparada uma classe de controle responsável pela conexão da aplicação com o *web service*. Esta classe é responsável por acessar e capturar, através da interface do serviço, os dados relacionados ao servidor, controladoras, entradas, saídas e transações. A Figura 8 exibe a classe de controle WebService, seus métodos e retornos.

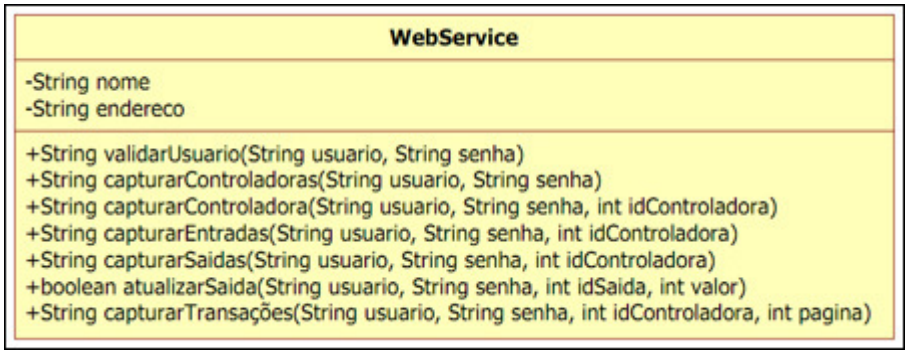

Figura 8 - Classe de controle WebService.
Fonte: Autoria própria.

Para a classe de interface do serviço, além das mesmas classes de valores de objetos e diagrama de entidade-relacionamento, foram definidos 7 métodos suficientes para suprir toda as requisições da aplicação. A Figura 9 exibe os métodos definidos e oferecidos pelo serviço implantados no servidor GIP, que foram, validarUsuario, capturarControladoras, capturarControladora, capturarEntradas, capturarSaidas, atualizarSaida e capturarTransações.

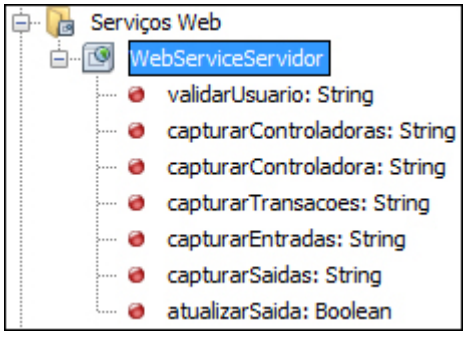

Figura 9 - Interface do Web Service.
Fonte: Autoria própria.

O método validarUsuario tem como função apenas a validação inicial do usuário para conexão com o servidor, em caso de sucesso a aplicação pode prosseguir capturando os demais dados, ou então alertar o usuário de que os dados inseridos estão incorretos. O método capturarControladoras retorna uma *string* com alguns dados de todas as controladoras cadastradas no servidor separando-as por um caractere chave. Escolhida uma controladora, através do método capturarControladora todos os dados relacionados na controladora em específico são devolvidos. Para cada controladora é possível tomar 3 ações: verificar os estados de suas entradas, manipular suas saídas ou acessar suas últimas transações. O método capturarEntradas retorna o estado de todas as entradas da controladora. O método capturarSaídas retorna o estado de todas as saídas da controladora e permite, através do método atualizarSaida, alterar o estado de uma saída especifica. Por fim, o método

capturarTransações retorna uma página específica sempre com 10 transações escolhidas pela aplicação.

Foram utilizadas 8 telas e para cada tela será definida uma *activities* responsável por validar e popular os dados da tela, bem como utilizar-se da classe de controle para fazer conexão com o *web service* e capturar os dados referentes à tela. As telas e *activities* definidas foram “Apresentação”, “Início”, “Servidor”, “Sobre”, “Controladora”, “Entradas”, “Saídas” e “Transações”.

Após a realização do diagrama de classes da aplicação, foi possível desenvolver o diagrama de entidade-relacionamento para o banco de dados, exibido na Figura 10.

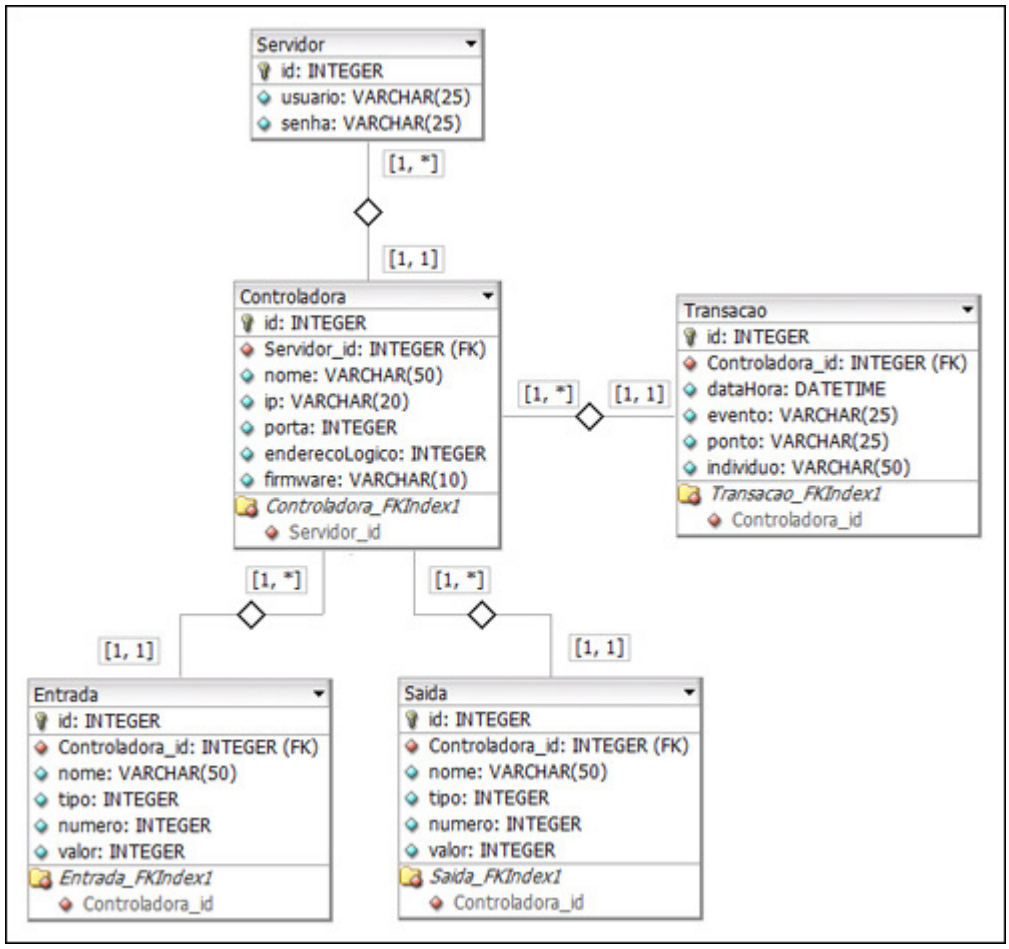

Figura 10 - Diagrama de entidade-relacionamento da aplicação.
Fonte: Autoria própria.

5 IMPLEMENTAÇÃO

A aplicação necessita de acesso a internet para captura dos dados via *web service*, para tal, foi adicionada ao projeto a permissão "*android.permission.INTERNET"*. Todas as telas exibidas possuem interface própria, definidas em arquivos XML independentes. Cada *activity* possui o mesmo nome que a tela e é responsável por atualizar os dados cada vez que a tela é aberta ou a cada solicitação de atualização pelo usuário.

As telas da aplicação foram construídas de forma a manter um padrão entre elas, sendo os principais componentes do tipo *TextView*, *EditText*, *Checkbox*, *Button*, *ImagemButton* e *ToogleButton.* Todas as classes *activities* de cada tela implementaram a interface *OnClickListener* que permitiu a estas recebem os comandos do usuário pela interface a ele exibida. Cada classe possui uma instância da classe WebService, que é a classe de controle para conexão com o *web sevice* implantado no servidor. A partir da classe de controle são capturados os dados referentes ao servidor e exibidos em tela.

A primeira tela a ser exibida ao usuário é a tela "Apresentação", que é apenas uma imagem de alguns segundos enquanto a aplicação inicia, para a exibição dessa imagem foi utilizado um componente do tipo *ImageView*. Para tal foi definido um contador de 2 segundos. A tela de apresentação é exibida na Figura 11 nos modos vertical e horizontal.

Figura 11 - Tela de apresentação.
Fonte: Autoria própria.

A segunda tela, “Início”, permite ao utilizador da aplicação conectar-se a um servidor, para isso deve selecionar ou preencher o endereço do servidor a ser acessado juntamente com usuário e senha. Pode também excluir um dos servidores cadastrados ou acessar a tela que apresentação uma pequena descrição sobre o aplicativo. Esta tela utiliza os dados armazenados em banco de conexões anteriores para exibir os servidores e o método validarUsuario oferecido pelo classe WebService para efetuar a conexão. A Figura 12 exibe a tela de início da aplicação.

Figura 12 - Tela de início.
Fonte: Autoria própria.

A tela “Sobre”, exibida na Figura 13, traz uma pequena descrição sobre o desenvolvimento da aplicação e seu objetivo. Nesse ponto o utilizador pode sair da aplicação ou voltar à tela “Início”.

Figura 13 - Tela sobre.
Fonte: Autoria própria.

Uma vez conectado com sucesso no servidor pela tela “Início”, é exibida uma tela com todas as controladoras do sistema para que usuário possa acessá-las individualmente. Na tela “Servidor” é possível voltar à tela “Início”, atualizar a lista de controladoras ou acessar uma controladora específica como pode ser visto na Figura 14. Esta tela, a partir do método capturarControladoras, utiliza uma tabela do tipo *TableLayout* para exibir as controladoras e componentes do tipo *ImageBotton* prontos para receber o comando do usuário e acessar a tela da controlado específica.

Figura 14 - Tela do servidor.
Fonte: Autoria própria.

Selecionada uma controladora, é possível verificar seus dados como endereço de IP, endereço lógico e firmware capturados através do método capturarControladora exibidos em *TextViews*. Pode-se também acessar as telas “Entradas”, “Saídas” ou “Transações” através dos botões em tela. O usuário pode escolher entre voltar à tela “Servidor”, atualizar os dados da controladora ou acessar uma das telas com mais dados referentes à controladora. A Figura 15 exibe a tela “Controladora”.

Figura 15 - Tela da controladora.
Fonte: Autoria própria.

Utilizando o método capturarEntradas, a tela “Entradas” permite o usuário verificar na tabela exibida o estado de todas as entradas da controladora, assim seria possível, por exemplo, validar os sensores de um ambiente controlado. A Figura 16 exibe as 8 entradas e seus estados de uma controladora.

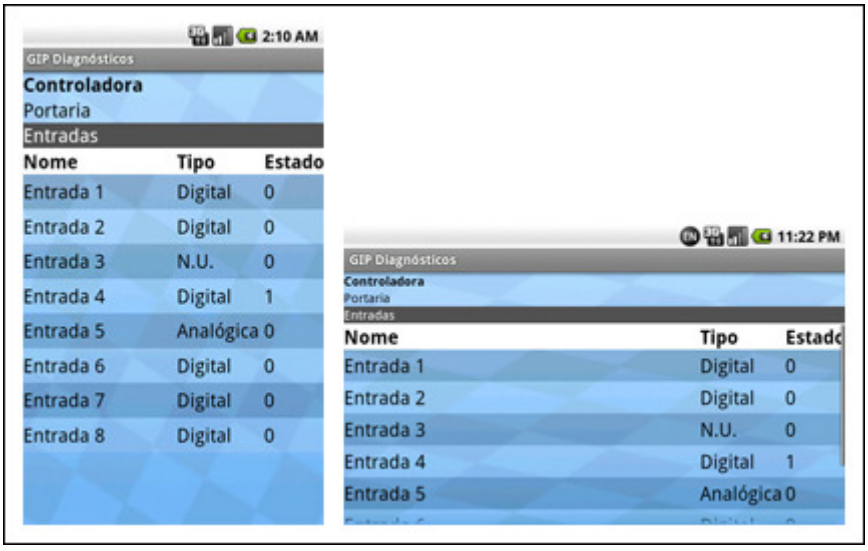

Figura 16 - Tela de entradas.
Fonte: Autoria própria.

Acessada através da tela "Controladora" e buscando os dados através do método capturarSaídas, a tela "Saídas" exibe o estado de todas as saídas das controladoras e permite alterá-las alternando entre valores de ligada e desligada com o método atualizarSaída. A cada ação do botão do tipo *ToogleButton* é enviado um comando ao servidor com o estado atualizado da saída. A Figura 17 exibe as saídas da controladora onde se encontram algumas saídas ligadas e outras desligadas pelo usuário.

Figura 17 - Tela de saídas.
Fonte: Autoria própria.

Por fim, a tela "Transações", através do método capturarTransações, exibe as 10 últimas transações da controladora, podendo o usuário atualizar a tela a qualquer momento

para que possua sempre os últimos dados reais da controladora. O usuário pode também navegar pelas transações mais antigas da controladora podendo voltar a qualquer momento à primeira página de transações, para tal, a aplicação passa o número da página desejada ao método da classe WebService. A Figura 18 exibe a tela “Transações” onde se pode perceber os ícones de navegação ao topo e as transações na parte inferior, sendo necessárias barras de rolagem para visualização total dos dados, para as barras de rolagem foram usados dois componentes *ScrollView* e *HorizontalScrollView*.

Figura 18 - Tela de transações.
Fonte: Autoria própria.

O fluxo das telas permite o utilizador acessar a qualquer tela da aplicação rapidamente utilizando poucos comandos. Assim torna o sistema de fácil uso e intuitivo. A Figura 19 exibe o fluxo que pode ser percorrido pelo utilizador entre as telas da aplicação, todas as telas acessadas podem voltar à tela anterior com exceção da tela de apresentação que é exibida apenas uma vez quando a aplicação é iniciada.

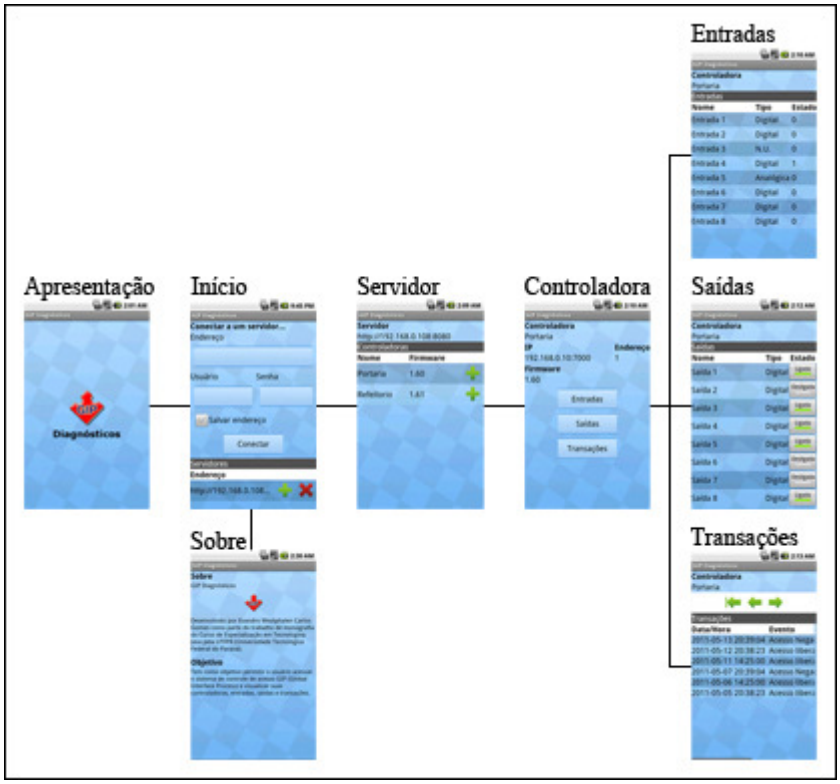

Figura 19 - Fluxo das telas.
Fonte: Autoria própria.

5.1 VALIDAÇÃO

Para validar o total funcionamento da aplicação, foram abordadas diferentes maneiras de validação. A aplicação foi validada inicialmente durante seu desenvolvimento através do próprio emulador da plataforma. Utilizando a ferramenta de *debug* que o próprio emulador oferece junto ao ambiente de desenvolvimento foi possível corrigir pequenas falhas e verificar os pontos que poderiam apresentar alguma inconsistência. A Figura 20 exibe o emulador da plataforma Android executando a aplicação em desenvolvimento.

Figura 20 - Emulador da plataforma Android em execução.
Fonte: Autoria própria.

Além do emulador, foi utilizado um celular da marca Sony Ericsson modelo Xperia Mini Pró com a versão 2.1 do sistema operacional. Os mesmos passos executados no emulador foram executados no aparelho, com o intuito encontrar algum mau funcionamento da aplicação, que se mostrou bastante consistente.

Para validar a interface do *web service* a ser implantado no sistema GIP, foi criada uma pequena aplicação em Java ligada a um banco de dados com o mesmo tipo de informação que seria utilizada pelo sistema de controle de acesso. Essa aplicação teria como função apenas implementar os métodos a serem oferecidos e retornar os dados encontrados em banco para a aplicação Android simulando o sistema GIP. O simulador possui também uma pequena janela que é atualizada automaticamente cada vez que um dado referente às saídas da controladora é alterado, podendo assim ser visualizado em tempo real quando a aplicação Android envia um comando de alteração do estado e o servidor o atende. A Figura 21 exibe a aplicação simulando o servidor GIP com os valores das saídas da controladora.

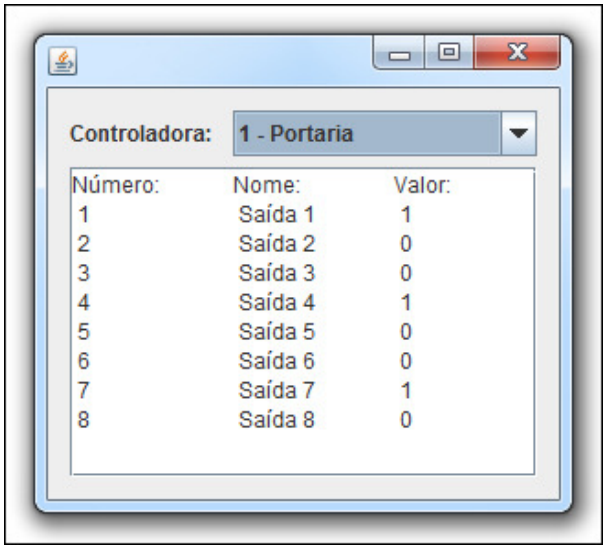

Figura 21 - Aplicação simulando servidor GIP.
Fonte: Autoria própria.

Após os testes via emulador e celular conectando-se a aplicação simuladora em Java, foi verificada a aplicação trabalhando diretamente com a interface já implantada no servidor GIP. Nessa etapa foi possível verificar todos os dados de cada controladora em tempo real e também alterar os estados de suas saídas, o que se mostrou muito útil já no primeiro momento.

6 RESULTADOS

Após estudar o funcionamento da plataforma Android, foi possível entender a estrutura e o processo de desenvolvimento para a arquitetura, obtendo uma melhor abordagem no desenvolvimento da aplicação. Foi analisada sua estrutura, arquitetura e interface.

Analisado o funcionamento de *web services*, observou-se que a plataforma não oferece de maneira nativa o acesso a esses serviços, sendo necessária a utilização de uma classe externa a solução Android para suprir essa necessidade, porém esta classe externa mostrou-se ser de fácil uso e adaptação.

Foi possível projetar, implementar e validar uma aplicação baseada na plataforma Android atendendo os requisitos propostos, de oferecer uma aplicação Android para acesso, controle de diagnóstico das placas *globalAccess* junto ao sistema de controle de acesso GIP. Para tal foram utilizados diagramas de classe e modelos de entidade-relacionamento os quais permitiram um melhor entendimento da solução, tanto para a aplicação quanto para o desenvolvimento da interface para o *web service* a ser realizado.

Como produto final obteve-se uma aplicação em Android que se conecta de maneira satisfatória ao servidor GIP, que por sua vez recebeu a implementação do *web service* proposto.

7 CONSIDERAÇÕES FINAIS

Através da solução oferecida pelo sistema GIP e a partir da dificuldade de aferir o funcionamento das placas *globalAccess* de controle de acesso que, em alguns casos, podem se encontrar distantes de uma estação de trabalho, podem ser necessários dois funcionários para concluir a validação da instalação, que se comunicam via radio, tornando o processo complexo e suscetível a erros.

O objetivo foi, através da compreensão da plataforma, desenvolver uma aplicação completa capaz de suprir as necessidades do usuário final, nesse caso, o responsável pela instalação e verificação das placas de controle de acesso. O estudo da plataforma Android permitiu desenvolver uma solução capaz de comunicar-se com o sistema GIP e capturar dados referentes não só ao sistema, mas também às placas controladoras, como os estados de suas entradas, saídas e transações.

Como resultado, obteve-se uma aplicação completa baseada na plataforma Android que, utilizando uma interface de serviço implementada pelo sistema GIP, permite o funcionário conectar-se ao sistema de maneira remota e assim, aferir o funcionamento das placas *globalAccess* sem a necessidade de um segundo funcionário se comunicando via radio informando os valores do sistema vistos em tela, diminuindo as chances de inconsistências e erros de instalação.

Além dos testes da aplicação com um emulador, foi possível validar, utilizando um dispositivo móvel com Android, o funcionamento da aplicação sendo utilizada por um funcionário responsável pela instalação de uma placa *globalAccess* onde a solução se mostrou eficiente diante do problema observado.

7.1 TRABALHOS FUTUROS

A partir deste trabalho é possível aprimorar a aplicação desenvolvida, a fim de que permita ainda mais ações do usuário para com o sistema de controle de acesso como, por exemplo, cadastros de novas controladoras, indivíduos e permissões de acesso. Pode-se também melhorar e criar métodos de segurança para o armazenamento de dados na aplicação e na comunicação com o servidor, como senhas e criptografias.

Podem-se ser criadas aplicações para outras plataformas que não Android e não necessariamente móveis capazes de se comunicar com o sistema GIP através da mesma interface de *web service* desenvolvida para este trabalho.

É possível estudar também outros meios de comunicação para a aplicação que não *web services*, como por exemplo, *sockets* para comunicação direta com as placas controladoras *globalAccess*.

8 REFERÊNCIAS

ANJOS, J. **Comparativo APIs Android x JavaME**. Disponível em <http://saloon.inf.ufrgs.br/twiki-data/Disciplinas/CMP167/TF08JulioAnjos/artigo-comparativo.pdf> Acesso em 07 abr. 2011.

BURNETTE, E. **Hello, Android**. 3. ed. Raleigh: The Pragmatic Bookshelf, 2010.

CONDER, S.; DARCEY, L. **Android Wireless Application Development**. Boston: Addison-Wesley, 2011.

FELKER, D.; DOBBS, J. **Android Application Development for Dummies**. Hoboken: Wiley Publishing, 2011.

GARGENTA, **M. Learning Android**. Sebastopol: O'Reilly Media, 2011.

GOOGLE. **Android Developers**. Disponível em <http://developer.android.com> Acesso em: 07 abr. 2011.

GOOGLE. **Android Media**. Disponível em <http://www.android.com/media> Acesso em: 15 abr. 2011.

LEAL, N. G. V. Web Services no Android. Disponível em <http://nglauber.blogspot.com/2011/02/web-services-no-android.html> Acesso em: 15 abr. 2011.

MARTINS, R. J. W. A. **Desenvolvimento de Aplicativo para Smartphone com a Plataforma Android**. 2009. Trabalho de Conclusão de Curso. (Graduação em Engenharia de Computação) - Pontifícia Universidade Católica do Rio de Janeiro. Orientador: Bruno Feijó.

OHM Technology. **OHM Technology**. Disponível em <http://www.ohmtech.com.br> Acesso em 02 abr. 2011.

PEREIRA, L. C. O.; SILVA, M. L. **Android para Desenvolvedores**. Rio de Janeiro: Brasport, 2009.

RABELLO, R. R. **Utilizando Web Services no Google Android**. Disponível em: <http://www.cesar.org.br/site/files/file/WM23_Android_WebServices.pdf> Acesso em 08 abr. 2011.

ROGERS, R.; LOMBARDO, J.; MEDNIEKS, Z.; MEIKE, B. **Android Application Development**. Sebastopol: O'Reilly Media, 2009.

STEELE, J.; To, N. **The Android Developer's Cookbook**. Boston: Addison-Wesley, 2011.

USHISIMA, R. **Consumindo Web Service em aplicações Android**. Disponível em: <http://zbra.com.br/2011/03/30/consumindo-web-service-em-aplicacoes-android> Acesso em 08 abr. 2011.