UNIVERSIDADE FEDERAL DO PAMPA

TAIRA DANIELA RODRIGUEZ ALMEIDA

S.C.A.M
UMA SOLUÇÃO PARA CONTROLE DE ACESSO A DISPOSITIVOS MÓVEIS

Bagé
2015

TAIRA DANIELA RODRIGUEZ ALMEIDA

S.C.A.M
UMA SOLUÇÃO PARA CONTROLE DE ACESSO A DISPOSITIVOS MÓVEIS

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Orientador: Érico Marcelo Hoff do Amaral

Bagé
2015

TAIRA DANIELA RODRIGUEZ ALMEIDA

S.C.A.M
UMA SOLUÇÃO PARA CONTROLE DE ACESSO A DISPOSITIVOS MÓVEIS

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido em: 31 de Janeiro de 2015.

Banca examinadora:

_____

Prof. MSc. Érico Marcelo Hoff do Amaral
Orientador
UNIPAMPA

_____

Prof. Dr. Milton Roberto Heinen
UNIPAMPA

_____

Prof. MSc. Fábio Luís Livi Ramos
UNIPAMPA

Dedico este trabalho a minha família, em especial a meu pai e minha irmã Ana por estarem sempre do meu lado incondicionalmente.

Em memória do querido Bolha, Rafael Quilião de Oliveira, como quem tive o prazer de compartilhar momentos inesquecíveis. Dedico esta conquista a ti, obrigado por fazer parte da minha historia, saudades eternas.

AGRADECIMENTOS

A meu pai pelo apoio, carinho, dedicação durante todos estes anos e pelo grande esforço financeiro que fez para eu poder terminar esta grande etapa da minha vida, sem ele nada disso seria possível.

A minha família, minha mãe, grande incentivadora, obrigada pelo apoio e carinho de sempre. A meus irmãos, que me incentivaram e apoiaram incondicionalmente em especial a minha irmã Ana, trilhamos esse caminho juntas, e juntas chegamos ao nosso objetivo, obrigada pelo apoio e carinho durante todo esse tempo, obrigada por estar sempre ao meu lado.

Ao meu orientador, Professor Érico Amaral, o meu sincero agradecimento por toda a disponibilidade, apoio, orientação, dedicação, e por todos os ensinamentos ao longo deste trabalho e durante todo o curso.

Agradeço a todos os professores do curso que me engrandeceram como pessoa e como profissional.

Ao meus colegas, que se tornaram grandes amigos, obrigado pela ajuda, motivação e pelos inúmeros momentos inesquecíveis que compartilhamos.

Finalizo agradecendo a todos os meus amigos, pela amizade, ânimo, entusiasmo, estímulo, ajuda e, especialmente, por estarem sempre presentes, em todos os momentos da minha vida.

A todos, meus sinceros agradecimentos

RESUMO

Acompanhando o avanço das tecnologias computacionais, os dispositivos móveis tem evoluído convertendo-se na plataforma de comunicação e computação preferida das pessoas. Com o aumento na demanda por mobilidade, dispositivos portáteis com mais recursos estão sendo lançados no mercado. Com isso, observa-se um aumento substancial na quantidade de aplicações disponíveis que proporcionam ao usuário maior facilidade para realizar diversas atividades. Devido às capacidades computacionais e ao uso intrinsecamente pessoal, estes dispositivos estão carregados de informações confidenciais. No entanto, a crescente popularidade e facilidade de utilização destas tecnologias contribuem para uma maior incidência de problemas de segurança. Um dos principais mecanismos para proteção das informações adotados por estes equipamentos são as soluções de autenticação. Contudo, os métodos mais utilizados são baseados em estratégias tradicionais de acesso. No presente trabalho, propõem-se um modelo de autenticação que tem como objetivo conciliar segurança e usabilidade, reduzindo o número de vezes que o usuário precisa interagir ativamente com o mecanismo de autenticação para acessar o sistema.

Palavras-chave: Dispositivos móveis. Segurança. Autenticação. Android.

ABSTRACT

Along with the advance of computational technologies, mobile devices have evolved becoming the preferred computing platform and people of communication. With the increased demand for mobility, more powerful devices are being released to the market. Based on that we can observe a substantial increase in the number of available applications that helps the user to realize several tasks easier. Due to the capabilities that the current systems provide and intrinsically personal use, these devices are loaded with confidential information. However, the growing popularity and ease of use of these technologies contribute to a higher incidence of security problems. One of the main mechanisms for information protection adopted by these devices, are authentication solutions, yet most widely used methods are based on traditional access strategies. In this paper, we propose a model for authentication which has the purpose of reconciling security and usability, reducing the number of times the user needs to actively interact with the authentication mechanism to access the system.

Keywords: Mobile device. Security. Authentication. Android.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ADT – *Android Development Tools*

API – *Application Programming Interface*

CIA – *Confidentiality, Integrity and Availability*

CSS – *Context Aware Computing*

CPU – Unidade central de processamento

DVM – Maquina Virtual Dalvik

GID – *Group ID*

GPS – *Global Positioning System*

GPU – *Graphics Processing Unit*

HTTP - *Hypertext Transfer Protocol*

IDC – *International Data Corporation*

IDE – *Integrated Development Environment*

ISSO – Organização Internacional de Normalização

JVM – *Java Virtual Machine*

MD5 - *Message-Digest algorithm 5*

MMS – *Multimedia Messaging Service*

OAH – *Open Handset Alliance*

PDA – *Personal digital assistant*

PIN – Número de identificação pessoal

AS – Sistema de autenticação

SDK – *Software Development Kit*

SO – Sistema operacional

SQL – *Structured Query Language*

SMS – *Short Message Service*

UID – Identificador de utilizador

UML – *Unified Modeling Language*

VM – *Virtual Machine*

XML – *Extensible Markup Language*

SUMÁRIO

1. INTRODUÇÃO

O avanço da tecnologia implicou em profundas mudanças na vida dos seres humanos. A disseminação dos dispositivos computacionais portáteis, aliados aos avanços da comunicação sem fio, está possibilitando o acesso à informação em qualquer lugar e a qualquer momento. Este cenário, denominado de computação móvel, esta mudando a vida cotidiana das pessoas. É clara a dependência da tecnologia para realizar as atividades mais simples.

Em particular, os telefones inteligentes multifuncionais fornecem a capacidade de executar uma ampla gama de ações que vão além da comunicação. Hoje em dia, existem soluções tecnológicas para qualquer tarefa, desde transferência de dinheiro, lazer, até educação. Com a popularidade destes dispositivos, o nível de funcionalidade está se expandindo de forma significativa, e assim, observa-se um aumento contínuo na variedade de aplicativos disponíveis (FISCHER *et. al.*, 2013; KHAN, 2012).

Além disso, a capacidade de armazenamento dos telefones móveis aumentou consideravelmente. Estes dispositivos contêm informação sensível sobre os seus proprietários, incluindo códigos de acesso diversos, comunicação pessoal, registros de chamadas, contatos, fotos, vídeos e localização geográfica (ASHER *et. al.*, 2011).

É possível observar que a utilização destes dispositivos vem crescendo substancialmente a cada ano, tanto para o uso pessoal, como também no campo corporativo. Segundo dados da IDC (*International Data Corporation*)[1], acredita-se que 1,0 bilhões de unidades de *smartphones* tenham sido comercializadas em todo o mundo em 2013, com previsão de aumento de 19,3% para 2014.

Ante este enorme crescimento, os usuários devem se preocupar com a exposição de informações confidenciais, uma vez que seus dispositivos móveis armazenam grandes quantidades de dados pessoais e privados. Contudo, este crescimento acelerado e a capacidade de instalar aplicações de terceiros dá lugar a vários problemas de segurança. Geralmente os usuários não são conscientes dos riscos potenciais inerentes a utilização destes dispositivos.

Dada a crescente importância dos *smartphones*, é relevante avaliar os riscos associados à privacidade e segurança destes dispositivos. Segundo Chin (2013),

[1] International Data Corporation (IDC), Analyze the future, 2012. Disponível em:

conhecer os riscos, vulnerabilidades e ameaças, permite implementar mecanismos adequados para controlar ou limitar as consequências negativas de utilizar o aparelho.

Atualmente, o mecanismo de segurança para proteger as informações mais utilizadas é a autenticação. Neste sentido, dispositivos móveis e sistemas de *desktop* compartilham o mesmo princípio subjacente de abordagem de autenticação (MARQUES, 2013). No entanto, segundo Falaki *et. al.* (2010), a utilização dos dispositivos móveis é significativamente diferente, a maioria das interações são muito curtas, assim os usuários tendem a desativar os mecanismos de segurança em favor da conveniência.

Em ambientes móveis, a usabilidade é um requisito importante a ser considerado. Segundo Cranor e Garfinkel (2005), o equilíbrio entre segurança e usabilidade é a base para identificação de um comportamento de segurança adequado. Entretanto, conforme Hong *et. al.* (2007), segurança e usabilidade são raramente integradas de forma satisfatória no projeto e desenvolvimento de sistemas móveis.

Neste contexto, surge o problema de pesquisa. É necessário a criação e implementação de mecanismos que permitam a segurança da informação e que atendam a requisitos de usabilidade, com a finalidade de garantir acesso apenas a usuários autorizados.

1.1 Motivação

Os riscos de segurança e privacidade inerentes ao uso de dispositivos móveis são um desafio permanente. Com o contínuo avanço das tecnologias móveis, têm-se dispositivos mais potentes com altas capacidades de processamento e armazenamento, sendo amplamente utilizados em diversas áreas.

O uso cada vez mais disseminado de *smartphones* continuará interferindo na vida humana. Muitas atividades que antes só eram possível executar em computadores tradicionais, atualmente estão disponíveis nos dispositivos móveis, além disso, muitas destas atividades manipulam informações sensíveis. Por outro lado, essa migração vem acompanhada de grandes desafios. A exposição indevida destas informações pode acarretar em serias consequências.

Para evitar a exposição de informações sensíveis, se faz necessária a criação de métodos de autenticação que evitem o acesso a informações e recursos a pessoas não autorizadas. Desbloquear os aparelhos com este tipo de mecanismo preserva a privacidade dos usuários, no entanto, os métodos utilizados atualmente são um legado dos *desktops (*MARQUES, 2013).

Apesar da popularidade destes dispositivos, muitas vezes, todo seu potencial não pode ser aproveitado, devido às preocupações do usuário com segurança e privacidade (CHIN *et. al.*, 2012). Neste cenário, destaca-se a necessidade de mecanismo de segurança que viabilizem o uso de serviços mais críticos, assim, torna-se relevante a criação de novos métodos autenticação que permitam melhorar a experiência do usuário com relação à segurança.

Este trabalho busca uma solução ao problema de autenticação em ambientes móveis, sem a necessidade de *hardware* adicional e que ofereça segurança e usabilidade desejada pelos usuários.

1.2 Problema de Pesquisa

Os mecanismos de autenticação mais utilizados em dispositivos móveis são baseados em senhas. Porém, muitas vezes, digitar as senhas torna-se uma tarefa indesejada para usuário. Com isso, surge a necessidade de implementar novos mecanismos que permitam realizar a autenticação de forma eficiente.

Desta forma, surge o problema de pesquisa: é possível implementar uma solução para autenticação em dispositivos móveis, aprimorando aspectos de interação do usuário como o sistema e que garanta um nível adequado de segurança?

1.3 Objetivos

O trabalho tem como objetivo geral desenvolver um modelo de controle de acesso as informações armazenadas em dispositivos móveis. Para este fim, têm-se os seguintes objetivos específicos:

- Realizar um referencial teórico sobre conceitos importantes para desenvolvimento do trabalho;

- Realizar um estudo sobre as principais aplicações, disponíveis no mercado e soluções propostos por pesquisadores da área para o controle de acesso a dispositivos móveis;
- Estudar as técnicas e linguagens de programação para dispositivos móveis;
- Realizar a análise e desenvolvimento de uma modelo, para o controle de acesso as informações em dispositivos móveis**;**
- Desenvolver a análise do projeto deste software.
- Implementar um protótipo do modelo com base na análise realizada**;**
- Testar e avaliar os resultados obtidos.

1.4 Organização do Trabalho

O presente trabalho esta organizado em 8 capítulos. No capítulo 2, é realizado o embasamento teórico, expondo os conceitos importantes para desenvolvimento do trabalho. O capítulo 3 apresenta uma caracterização da pesquisa, bem como, a descrição do procedimento metodológico adotado para realização do trabalho. Já o capítulo 4, apresenta o modelo proposto para autenticação. O capítulo 5 contém uma descrição da modelagem do sistema. O capítulo 6 apresenta o processo de desenvolvimento do protótipo da solução. No capítulo 7 são apresentado os testes realizados, assim como a análise dos resultados obtidos. Finalmente, no capítulo 8 são expostas as considerações finais do trabalho.

2. REFERENCIAL TEÓRICO

Este capítulo apresenta a revisão bibliográfica construída para o desenvolvimento do presente trabalho. Esboça o conceito geral de computação móvel e realiza uma descrição das principais características presente no sistema operacional Android. Em seguida, são realizados estudos sobre conceitos relacionados à segurança em sistemas computacionais. No final do capítulo, são apresentados os trabalhos com abordagens semelhantes estudados relevantes para este projeto.

2.1 Computação Móvel

O ambiente de computação tradicional exige que os usuários mantenham-se conectados a uma infraestrutura fixa. Esta característica limita a utilização de computadores tradicionais e faz com que seja difícil a utilização dos mesmos quando o usuário esta em movimento.

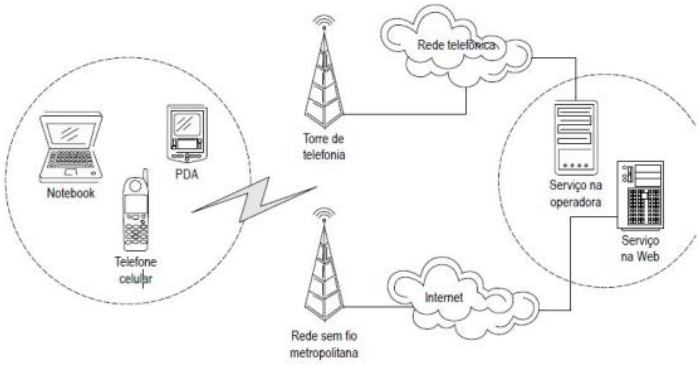
Para superar esta limitação surge à computação móvel. Ela baseia-se na necessidade de acessar informações a qualquer hora em qualquer momento ou, mais geralmente, a computação a qualquer hora e em qualquer lugar. Assim o computador torna-se um dispositivo sempre presente, independentemente da localização do usuário (JOHNSON, 2007).

Entende-se por computação móvel, quaisquer sistemas cujas capacidades podem ser utilizadas quando estão em movimento, e geralmente proporcionam serviços que não se encontram em sistemas tradicionais. Assim, o principal objetivo é oferecer aos usuários um ambiente computacional similar ao proporcionado por computadores tradicionais, porém fornecendo mobilidade, permitindo que usuários tenham acesso a serviços e informações independentes de onde estão localizados (MATEUS e LOUREIRO, 1998). Segundo Figueiredo e Nakamura (2003), toda a comunicação entre os dispositivos móveis deve ser realizada através de tecnologias de redes sem fio. Assim, a comunicação sem fio é um suporte para a computação móvel, que por sua vez, explora diferentes tecnologias.

Segundo Mateus e Loureiro (1998), a computação móvel amplia o conceito de computação distribuída. Para os autores, a arquitetura básica de um ambiente de

computação móvel consiste na tradicional infraestrutura de comunicação fixa com computadores estáticos ligada a uma parte móvel representada por uma área ou célula, onde existe a comunicação sem fio. A Figura 1 ilustra a arquitetura básica de um ambiente de computação móvel.

Figura 1 - Arquitetura básica de um ambiente de computação móvel.



Fonte: Johnson, 2007.

A Figura 1 exemplifica um cenário de computação móvel, em que as redes híbridas de telefonia celular e as redes sem fio são a infraestrutura necessária para o acesso aos serviços móveis.

2.1.2 Dispositivos Móveis

Segundo Myers *et. al.* (2004), os dispositivos móveis se tornaram populares nas mais diversas áreas de atividades devido a sua simplicidade, funcionalidades e portabilidade, tornando-se um componente indispensável na vida moderna. Diversos aparelhos se classificam como dispositivos móveis. Alguns exemplos que são amplamente utilizados são: *Personal Digital Assistants* (PDA's), *smartphones*[2] e *tablets*.

Nos últimos anos, o mercado destes dispositivos tem sido afetado pelo surgimento de *smartphones*. Assim, os dispositivos móveis tem evoluído de serem simples agendas eletrônicas PDA's para se tornar sofisticado e compactos computadores pessoais.

Smartphones são telefones celulares que oferecem recursos de computação avançados e opções de conectividade (SOIKKELI, KARIKOSKI, e HAMMAINEM,

[2] Telefones inteligentes.

2011). Esses recursos permitem que o usuário realize uma ampla gama de atividades que antes só era possível realizar em computadores tradicionais. Assim, os *smartphones* tornam-se uma parte cada vez mais integrada da vida cotidiana dos usuários. Entre as características mais importantes, podemos citar: (GONZALES e GONZALES, 2013).

- Unidade central de processamento (CPU): os *smartphones* contam com potentes CPUs que permite executar uma variedade de aplicações. Hoje em dia, existem celulares com processadores *multicore*[3] com capacidade de executar múltiplos processos simultaneamente (*multiprocessing*). Além disso, contam com *Graphics Processing Unit* (GPUs) dedicado ao processamento de imagens e funções multimídia. Todos esses recursos de *hardware* disponíveis são gerenciados pelo sistema operacional.
- Sistema Operacional (S.O): fornece uma interface entre o usuário final e o hardware. Tem como objetivo gerenciar os recursos de hardware e software para dispositivos. Entre os sistemas operacionais mais populares se encontram *Android, iOS* e *Windows Phone*.
- Aplicações: capacidade de executar uma infinidade de aplicações, além disso, os sistemas operacionais permitem programar e instalar novas aplicações.
- Conectividade*:* redes 3G e 4G, assim como, acesso *Wi-Fi* e *bluetooth*.
- Sensores: Estes dispositivos vêm equipados com uma ampla gama de sensores, entre eles: câmeras, acelerômetro, *Global Positioning System* (GPS) e sensores de proximidade.

O uso de *smartphones* é fundamentalmente diferente ao de computadores tradicionais. Os novos recursos e serviços oferecidos por estes dispositivos moldam os hábitos dos usuários em relação ao seu uso. Examinar as características de sessão de uso, tais como duração da sessão e o número de sessões por período de tempo, ou tempo de interação, nos fornecem uma visão geral do comportamento do usuário.

Segundo pesquisa realizada por Soikkeli *et. al.* (2011), ao examinar essas características, observaram que o uso de *smartphones* é altamente diversificado por

[3] Processadores com várias CPUs.

usuários. Desta maneira, aplicações e serviços precisam ter levar em consideração as características de uso de utilizadores individuais. Assim faz-se necessário que as aplicações sejam capazes de aprender e adaptar-se comportamento do usuário. Os autores também ressaltam que o contexto do utilizador final afeta as características das sessões de uso.

Ao contrário dos *desktops* e *laptops*, onde os usuários tendem a usar por um período longo e contínuo de tempo, os telefones móveis são acessados periodicamente, em resposta a um determinado evento, por exemplo, notificação de e-mails recebidos. Falaki *et. al.* (2010) realizaram uma pesquisa a fim de caracterizar o uso de *smartphones* com relação às interações do usuário e ao uso de aplicativos. Segundo os autores, as interações com os telefones variam, mais em geral são muito curtas, porém acontecem com muita frequência, e na maioria das interações uma única aplicação é utilizada.

Tais aparelhos oferecem aos seus usuários diversas funcionalidades, consequentemente precisam armazenar uma grande quantidade de dados confidenciais. Também são utilizados como interface com outros serviços, como *e-mail*, redes sociais, *mobile banking*[4]*,* assim dados mais sensíveis podem ser armazenados (CHONG *et. al.*, 2009). Estes serviços oferecem mecanismos de autenticação, porém, as senhas são armazenadas no dispositivo, logo o usuário não é obrigado a digitar novamente suas credenciais cada vez que ele usa o serviço. Portanto, somente o proprietário do dispositivo deve ser capaz de ler, modificar ou apagar esses dados.

2.1.3 Sistemas operacionais para dispositivos móveis

Segundo Silberschatz *et. al.* (2004), sistema operacional é um conjunto de programas com a função de gerenciar os recursos de *hardware* e *software*, além de fornecer uma interface ao usuário final.

No âmbito de dispositivos móveis, podem-se citar três grandes sistemas operacionais, são eles: iOS, Windows Phone e Android.

O iOS[5] é o sistema operacional móvel da Apple Inc.[6], presente no iPhone, iPod Touch, iPad e Apple *TV*. Não pode ser executado em hardware de terceiros,

[4] Ferramentas que disponibilizam alguns serviços bancários através de dispositivos móveis.
[5] Informações disponíveis em: https://www.apple.com/es/ios/

porém, contém muitas funcionalidades presentes somente neste sistema operacional.

O sistema operacional Android foi lançado em 2008 pela Open Handset Alliance[7] (OHA). A plataforma é licenciável, assim, qualquer fabricante que atenda aos pré-requisitos estabelecidos pela Google[8] pode usá-lo em seus aparelhos.

O Windows Phone[9] é uma versão móvel do Windows. Desenvolvido pela Microsoft, a plataforma foi lançada em 2010 como sucessora do Windows Mobile. Assim como o Android, o Windows Phone também é licenciável e está presente em diferentes modelos e marcas. Porém, a Microsoft é mais exigente em termos de *hardware*.

Neste trabalho optou-se pela utilização do sistema operacional Android visto que é o sistema operacional móvel mais utilizado no mundo, de acordo cm dados da Net Applications[10].

2.2 Sistema Operacional Android

O sistema operacional Android é uma pilha de *software*s *open source* para dispositivos móveis que inclui sistema operacional, *middleware*[11] e aplicações para fornecer as funcionalidades básicas dos dispositivos. Oferece suporte completo à programação, os recursos oferecidos incluem ferramentas de desenvolvimento (compiladores, emuladores, etc.), bibliotecas, *Applications Programming Interface (*APIs) e *Frameworks* (LECHETA, 2010).

Segundo o Instituto de pesquisas Gartner[12] a plataforma do Google crescerá 26% em comparação com o ano de 2013 e chegará a mais de 1.1 bilhões de aparelhos em 2014. Em 2015, os números devem aumentar para mais de 1,2 bilhões de aparelhos.

---

[6] Empresa multinacional norte-americana que tem como objetivo projetar e comercializar produtos eletrônicos.
[7] Grupo formado por mais de 30 empresas lideres de mercado de telefonia juntamente com a Google
[8] É uma empresa multinacional de serviços online e software.
[9] Informações disponíveis em http://www.windowsphone.com/pt-BR/
[10] Disponível em: http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=9&qpcustomb=1&qpct=4&qpsp=175&qpnp=12&qptimeframe=M
[11] Um middleware é a designação genérica utilizada para referir aos sistemas de software que se executam entre as aplicações e os sistemas operacionais.
[12] Disponível em: <http://www.gartner.com/newsroom/id/2645115

2.2.1 Versões do android

Diversas atualizações são realizadas no sistema operativo, cada atualização no Android refere-se à solução de algum *bug* ou a adição de alguma funcionalidade nova. O sistema operacional Android possui um esquema de versionamento baseado em dois fatores:

1. *Platform Version*: é o nome comercial da versão do Android.

2. *API Level*: é um número inteiro que é incrementado a cada nova versão do Android, utilizado apenas internamente pelos desenvolvedores.

A seguir é apresentada uma tabela com algumas versões e API correspondentes do sistema operacional lançadas até o momento. Também é apresentado o numero relativo de dispositivos que executam uma determinada versão da plataforma Android até o momento.

Tabela 1 - Versões Android

	API Level	Platform Version	Distribuição
Android 2.2	8	Froyo	0.4%
Android 2.3.3 – 2.3.7	10	Gingerbread	7.8%
Android 4.0.3 - 4.04	15	Ice Cream Sandwich	6.7%
Android 4.1.x	16	Jelly Bean	19.2%
Android 4.2.x	17		20.3%
Android 4.3	18		6.5%
Android 4.4	19	KitKat	39.1%

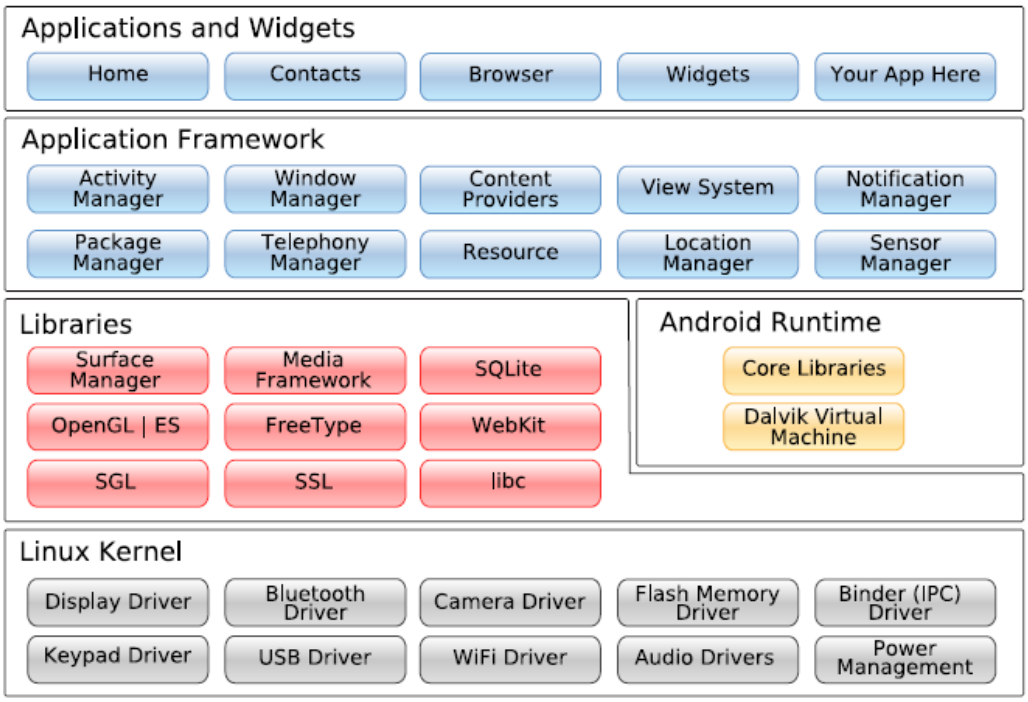
Fonte: https://developer.android.com/intl/es/about/dashboards/index.html

A equipe de desenvolvimento Android tenta manter a compatibilidade entre as versões, o que possibilita que um aplicativo desenvolvido na API *Level* 15, por exemplo, seja compilado na API *Level* 18. Porém nem sempre isso é possível, pode acontecer que uma funcionalidade antiga deixe de existir em versões posteriores.

2.2.1 Arquitetura Android

A arquitetura do sistema operacional Android é dividida em camadas: *Applications*, *Application Framework*, *Libraries*, *Android Runtime* e o *kernel*[13] do *Linux,* onde cada camada utiliza os serviços providos pela camada inferior, começando de baixo para cima, conforme a Figura 2 (BURNETTE, 2009).

Figura 2 - Arquitetura do Sistema Android



Fonte: Burnette, 2009.

A seguir é descrito em detalhes a constituição desta arquitetura:

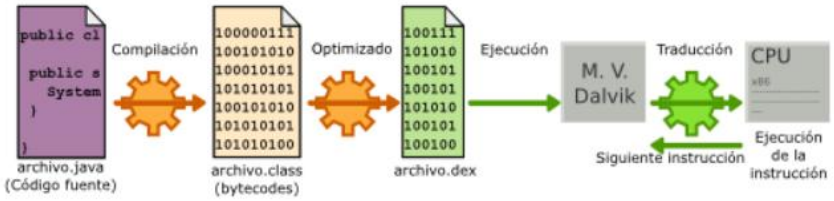
- *Application:* é a camada superior da arquitetura onde se encontram um conjunto de aplicações essenciais para o provimento das funções básicas do dispositivo, escritas na linguagem Java (PEREIRA e DA SILVA, 2012). Como exemplos destas aplicações podem citar: serviço de SMS/MMS, email, calendário, navegador web e agenda. Nesta camada também rodam as aplicações de terceiros.
- *Application Framework*: Nesta camada encontram-se todos os recursos e APIs utilizados pelos aplicativos. Os desenvolvedores de aplicações têm

[13] Componente central do sistema operacional.

acesso total ao *framework* de aplicações do Android. Foi criado para abstrair a complexidade e permitir a reutilização de procedimentos aumentando a produtividade na hora do desenvolvimento.

- *Libraries:* Consiste em um conjunto de bibliotecas nativas escritas em C/C++ utilizadas por vários componentes do sistema. As bibliotecas fornecem várias funções e serviços, que são utilizadas ao longo do desenvolvimento. As funcionalidades oferecidas pelas bibliotecas são acessadas através do *framework* de aplicação.
- *Android Runtime:* O ambiente de execução do Android é composto por um conjunto de bibliotecas e Maquina Virtual Dalvik (DMV), criada especialmente para o sistema Android, foi otimizada e adaptada às peculiaridades dos dispositivos móveis. A DMV interpreta e traduz o código Java para uma linguagem compreensível pelo S.O. Não trabalha diretamente com *bytecode* Java, mas transforma em um código mais eficiente que o original, projetado para pequenos processadores, roda aplicações no formato Dalvix *executable* (.dex). As aplicações rodam em um ambiente virtual Dalvik. Assim o ambiente de execução é sempre o mesmo independente do dispositivo físico, dando portabilidade às aplicações (GUTIERREZ, 2012).

Figura 3 - Processo de transformação e execução do código do Android.



Fonte: Gutierrez, 2012.

A Figura 3 apresenta o processo de transformação e execução de uma aplicação. Toda e qualquer aplicação em Android roda dentro de seu próprio processo, isto é, no contexto da sua instância de máquina virtual.

Cada aplicativo roda em sua própria instância da Dalvik VM, e cada instância é gerenciada pelo seu próprio processo no Linux, ou seja, os aplicativos não compartilham a mesma instância da Dalvik VM. Isto garante maior segurança, pois um aplicativo não poderá interferir na execução de outro aplicativo.

- *Linux Kernel*: Utiliza o *Kernel Linux* como camada de abstração entre o hardware e a pilha de *software*. É responsável pelo gerenciamento de processos, de memória, de rede e pela segurança do sistema. A versão utilizada é a 2.6 do *Linux*, porém foram feitas algumas modificações para obter melhor desempenho em dispositivos móveis (BURNETTE, 2009).

  O sistema Android se beneficia de certas características do sistema operacional Linux, como:

  - ✓ Suporte Multitarefa: possibilita executar várias tarefas ao mesmo tempo;
  - ✓ Suporte Multiusuário: isto permite ao Android disponibilizar um usuário para cada aplicação instalada, garantindo maior segurança e privacidade dos dados dos aplicativos, o que impede que um aplicativo instalado prejudique os dados de outro ou acesse sua base de dados e seus arquivos.

Assim, observa-se que a plataforma Android é uma pilha de *software*, onde cada camada agrupa vários programas com funções específicas, dando suporte às camadas adjacentes.

2.2.2 Estrutura das Aplicações Android

A principal característica das aplicações Android é que qualquer aplicação pode utilizar componentes de outras sempre que obtenha as permissões adequadas. Sempre que se inicia uma aplicação, o sistema operacional atribui um processo Linux, no entanto é possível atribuir um processo diferente a cada componente.

As aplicações são compostas por componentes que são responsáveis pelo provimento das diversas funcionalidades suportadas pelas APIs do Android. São eles: *Activity, Broadcast Receivers, Services e Content Providers*. Lecheta (2010) descreve cada componente:

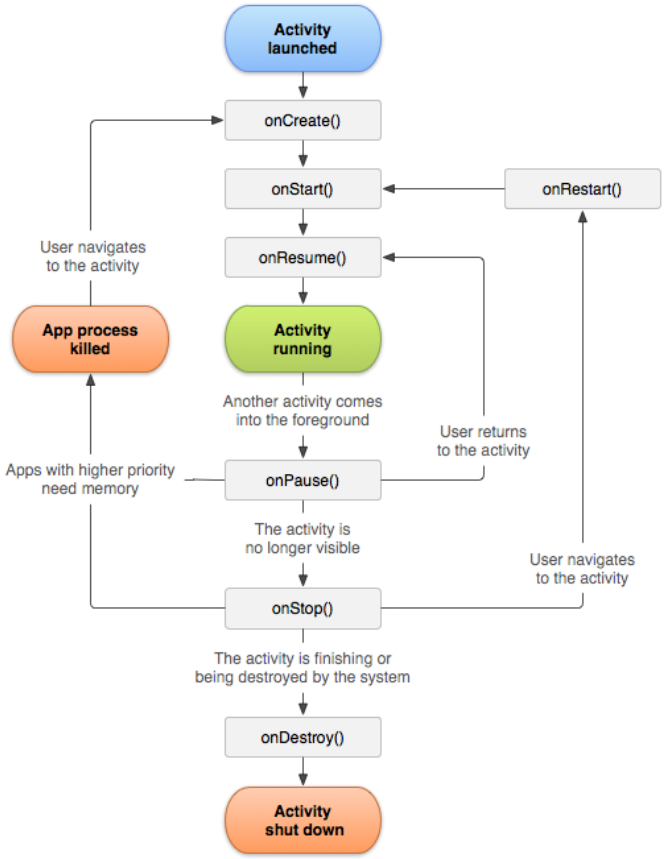
Activities

Activities são classes Java que controlam os eventos de uma tela do aplicativo. Todas as interfaces às quais o utilizador tem acesso numa aplicação são implementadas por *Activities*.

Uma aplicação pode conter uma ou mais *Activity,* cada uma representa uma ação particular e bem definida que o utilizador pode realizar, não existe uma relação de hierarquia entre atividades. É possível que haja comunicação entre estes componentes, ou seja, uma *Activity* pode chamar outra *Activity* e elas podem trocar informações entre si. Esta comunicação é feita através de da classe *Intent*. Uma *Intent* é basicamente uma mensagem enviada a partir de um aplicativo para o núcleo Android, solicitando a execução de alguma ação.

As *Activities* possuem um ciclo de vida bem definido que é gerenciado pelo sistema operacional, porém o desenvolvedor deve levar em consideração cada estado possível para garantir o correto funcionamento dos aplicativos. (LECHETA, 2010). A execução das *Activities* funciona como uma pilha, onde a *Activity* atual (que está interagindo com o usuário) está sempre no topo da pilha.

Figura 4 - Ciclo vida de uma atividade em Android.



Fonte: http://developer.android.com/intl/es/reference/android/app/Activity.html

Os retângulos representam os métodos, que serão descritos as seguir, e os ovais representam os principais estados de uma *activity.*

- OnCreate: É chamado apenas uma vez quando a activity é criada, deve referenciar a tela que será apresentada ao usuário.
- OnStart: É chamado quando a activity está ficando visível e já tem uma tela definida.
- OnResume: É chamado quando a *activity* foi parada temporariamente e está retornando à execução.
- OnPause: É chamado quando a *activity* está sendo tirada do topo da execução. Geralmente é utilizado para salvar o estado da aplicação.
- OnStop: É chamado quando a *activity* não está mais visível e está em segundo plano.
- OnDestroy:Executa os últimos processamentos antes da *activity* ser literalmente encerrada. (LECHETA, 2010).

Service

Um serviço é destinado à execução de uma tarefa, mas não tem uma interface associada, ou seja, a execução é realizada em *background*. É um componente que roda em segundo plano realizando operações que levam muito tempo de execução ou realizando trabalho para processos remotos.

Os *Services* são prioridade no Android, perdem apenas para a Activity em execução, ou seja, se em um determinado momento o Android precisar liberar espaço na memória do dispositivo, os *Services* serão os últimos componentes a serem eliminados.

Um *Service* pode ser iniciado através de qualquer componente do Android, ou seja, tanto uma *Activity* quanto um *Broadcast Receiver* ou um próprio *Service* pode iniciá-lo.

Broadcast Receivers

É um componente que responde *broadcasts* disparados a todo o sistema. O Android envia uma mensagem para todo o sistema quando determinados eventos acontecem, e esta mensagem pode ser respondida por quem tiver interesse em recebê-las. Esta mensagem é chamada de *broadcast* e o componente responsável por respondê-la é chamado de *Broadcast Receive*. Através dos *Broadcast Receivers* podemos disparar determinadas ações quando um determinado evento acontece.

Muitos *broadcasts* são originados pelo sistema, por exemplo, um broadcast anunciando que a bateria está baixa, assim às aplicações podem adotar comportamentos adequados, as aplicações também podem iniciar *broadcast*.

Content Providers

É um recurso que possibilita o compartilhamento de informações entre aplicativos de forma segura e, através dele, podemos permitir outros aplicativos a consultar, inserir, atualizar e deletar (se o *content provider* permitir) as informações compartilhadas. Por exemplo, o sistema Android fornece um *content provider* que gerencia as informações de contato do usuário. Assim, qualquer aplicação com as

permissões apropriadas podem consultar parte do *content provider* para ler e escrever informação sobre uma pessoa em particular.

As aplicações podem estar compostas por um ou mais componentes, sendo que cada tipo de componente tem um objetivo específico, funções distintas, e um ciclo de vida particular (PEREIRA e DA SILVA, 2012).

2.2.3 Armazenamento de dados

O Android fornece diversas opções para persistir dados em uma aplicação. A opção de armazenamento deve ser escolhida de acordo com as necessidades especificas da aplicação, por exemplo, se os dados devem ser privados para uma aplicação ou acessíveis a outras aplicações e quanto espaço os dados exigem. A Tabela 2 mostra as opções para persistência de dados.

Tabela 2 - Opções para persistência de dados Android

Tipo	Características
Internal Storage	Armazenar dados privados na memória do dispositivo. Por padrão, os arquivos salvos na memória interna são privados para a sua aplicação e outras aplicações não podem acessá-los.
External Storage	Armazenamento externo: Armazenar dados públicos sobre o armazenamento externo compartilhado. Pode ser uma mídia de armazenamento removível (como um cartão SD) ou um armazenamento (não removível) interno. Os arquivos salvos para o armazenamento externo podem ser lidos por todos e podem ser modificado pelo usuário.
Network Connection	É possível usar a rede para armazenar e recuperar dados através de serviços baseados na web
Shared Preferences	Armazenar dados primitivos privados em pares chave-valor.
SQLite Databases	Armazenamento de dados estruturados em um banco de dados particular. Android oferece suporte completo para SQLite . Qualquer banco de dados será

	acessível pelo nome através de qualquer classe no aplicativo, mas não fora do aplicativo.

Fonte: Próprio Autor, 2014.

SQLite

O SQLite é um banco de dados Open Source que suporta recursos de banco de dados relacionais padrão, como sintaxe SQL. É um banco auto-contido, compacto, e sem necessidade de configuração ou instalação, requer apenas um pouco de memória em tempo de execução. Isto o torna a escolha natural para um ambiente em que devemos prezar por desempenho, disponibilidade de memória e praticidade de uso.

Para atender a necessidade de persistir dados o Android oferece suporte nativo ao banco de dados SQLite. Contém as classes de gerenciamento de banco de dados SQLite que um aplicativo poderia usar para gerenciar o seu próprio banco de dados particular (ANDROID DEVELOPER).

Usando um banco de dados SQLite no Android não requer qualquer configuração ou administração de banco de dados. Uma característica muito importante é que no Android cada aplicativo possui o seu banco de dados exclusivo apenas o aplicativo que o criou poderá acessá-lo. Esta segurança é possível devido ao fato de cada aplicativo possuir um usuário exclusivo no Android, e cada usuário tem acesso apenas ao seu banco de dados.

2.3 Segurança em Dispositivos Móveis

O crescente desenvolvimento das tecnologias móveis inseridas em nosso cotidiano faz emergir uma série de questões relativas à segurança e privacidade dos usuários e suas informações pessoais.

Em contraste com computadores *desktop*, dispositivos móveis são projetados para serem móveis e assim, não se restringe a um local específico. As preocupações dos usuários com questões de segurança estão impedindo-os de tirar o máximo proveito da tecnologia à sua disposição. Segundo Clarke e Furnell (2005), funções de segurança são o segundo fator, logo atrás do consumo de energia, que o

usuário considera na hora de comprar um dispositivo. Para Asher *et. al.* (2011), muitos usuários estão relutantes em realizar tarefas sensíveis em seus dispositivos.

A capacidade de armazenamento de *smartphones* tem aumentado significativamente permitindo aos usuários armazenar uma maior quantidade de dados, incluindo fotos, vídeos, mensagens SMS e *e-mails*. Muitos tipos de informações mantidas em *smartphones* podem ser considerados sensíveis, no sentido de que teria efeitos prejudiciais para o proprietário, se fossem expostas. (MUSLUKHOV *et. al.*, 2013).

Além disso, estes dispositivos são mais vulneráveis a ameaça de roubo e perda acidental que pode levar à exposição de dados e assim a violação de privacidade. Pode-se argumentar que muitos desses roubos são a fim de obter o dispositivo e não as informações nele contidas, porém, as ameaças de exposição e perda de informação não podem ser ignoradas (CLARKE e FURNELL, 2005). Ainda, estes dispositivos se conectam a uma ampla variedade de redes para troca de informações, aumentando as possibilidades de ataques à segurança dos usuários.

Sem um mecanismo de autenticação ativo, cada pessoa com acesso físico ao dispositivo poderia usá-lo. Isso inclui acesso aos dados armazenados e a serviços que não necessitam de autenticação adicional. A falta de segurança no nível da aplicação significa que o acesso as informações privadas devem basear-se exclusivamente a processos de autenticação em nível do dispositivo para fornecer proteção contra acesso não autorizado.

Outro risco relacionado com o conteúdo em dispositivo móvel é que os arquivos, muitas vezes, são armazenados em mídia removível (como cartões SD), em vez de apenas na memória do dispositivo. Isto implica que as informações ficam desprotegidas pelo método de autenticação aplicado ao dispositivo.

2.3.1 Segurança Android

A segurança da arquitetura baseia-se nos mecanismos de segurança aplicados pelo *kernel* Linux*,* que providencia ao Android um conjunto de aspectos chave no seu modelo de segurança, são eles:

- Um modelo de permissões baseado no usuário;
- Isolamento de processos;

2.3.1.1 *Sandboxing*

Sandboxing é uma abordagem para o gerenciamento de aplicações que limita os ambientes em que determinado código pode executar. O objetivo do *sandboxing* é melhorar a segurança, isolando uma aplicação para evitar que *malwares*, intrusos, recursos do sistema ou outras aplicações se comuniquem com o aplicativo protegido. Android utiliza essa abordagem, onde cada aplicativo é executado em sua própria "*sandbox*".

Android é executado no kernel do Linux, que foi construído com suporte multiusuário. No entanto, um dispositivo Android não é projetado para ser compartilhado entre vários usuários. Ao contrário de um computador de mesa, onde as aplicações são executadas como o mesmo identificador de utilizador (*user ID* - UID), aplicativos do Android são isolados um do outro, rodam em processos separados sob UIDs distintos, cada um com permissões diferentes.

A adaptação feita pelo Android trata cada aplicativo como um usuário, atribui um UID exclusivo para o aplicativo em tempo de instalação, e caso seja necessário um ID de grupo (*group* ID – GID). Cada aplicação instalada possui um diretório próprio no sistema de arquivos, onde todos os arquivos associados serão armazenados. Apenas o UID do aplicativo possui total acesso a esse diretório, o grupo ao qual pertence e os outros não possuem qualquer permissão de acesso. Este controle impossibilita o acesso por parte de aplicativos a recursos protegidos do sistema ou de outros aplicativos.

Android utiliza este mecanismo para isolar aplicações e recursos do sistema. Cada aplicativo é executado em seu próprio processo separado de Dalvik VM, fornecendo prevenção de vários tipos de divulgação de informações, ou seja, uma aplicação não pode acessar as informações sensíveis localizado no sistema, assim como as informações armazenadas no espaço particular de outra aplicação. Acesso não autorizado a recursos de hardware como GPS, câmera ou rede de comunicação pode ser evitado utilizando este mecanismo.

2.3.1.2 Modelo de Permissões

Android usa um modelo de permissão declarativa para a segurança em nível de aplicativo, para impor certas restrições de acesso das aplicações a determinados

recursos no sistema e funcionalidades de outras aplicações. Em algumas situações, é desejável acessar a recursos do sistema e trocar informações entre as aplicações, como também, acessar funcionalidades umas das outras.

Quando uma aplicação necessita acessar a determinados recursos protegidos do sistema é necessário que sejam requisitadas as permissões adequadas. Para acessar a esses recursos, as requisições de permissões devem ser declaradas explicitamente no manifesto da aplicação (*AndroidManifest.xml*). O utilizador que instala a aplicação será responsável pela concessão dessas permissões em tempo de execução. O usuário deve conceder todas as permissões solicitadas, a fim de instalar o aplicativo. Como resultado, o aplicativo pode acessar irrestritamente os respectivos recursos durante sua execução.

AndroidManifest.xml, é um arquivo XML (*Extensible Markup Language*) de configuração e existente na raiz de todas as aplicações, que descreve nome, versão, componentes da aplicação, direitos de acesso, arquivos de biblioteca referenciados pelo aplicativo e é utilizado também para definir o nível de API (API *level*) requerido pela aplicação.

2.3.2 Conceito de Segurança da Informação

A informação é um ativo para as organizações e indivíduos. A divulgação, modificação imprópria ou indisponibilidade poderá incorrer em prejuízo ou lucros perdidos para a organização ou a indivíduo, portanto a segurança da informação faz-se cada vez mais necessária (TODOROV, 2007).

Na literatura existem várias definições para segurança e, na maioria, é caracterizada a necessidade de se manter no sistema um conjunto de propriedades: confidencialidade, integridade e disponibilidade. Os três princípios ajudam a alcançar a segurança da informação ideal e são referidos como a tríade CIA (WHITMAN e MATTORD, 2011). De acordo ao contexto em que são aplicadas, as interpretações destes princípios podem variar.

Embora estes conceitos sejam aplicados a computação tradicional, também são validos para computação móvel, uma vez que estes sistemas são vulneráveis aos mesmos risco e muitas vezes têm esses riscos aumentados devido a certas características da computação móvel.

A confidencialidade surge da necessidade de garantir que a informação seja acessível apenas às pessoas autorizadas. A integridade refere-se à proteção das informações de acidental ou intencional modificação, que pode afetar a validade dos dados. Assim, a integridade garante à confiabilidade dos dados ou recursos. Por fim, a disponibilidade especifica a capacidade de utilizar a informação ou recurso quando desejado, assegurando que as pessoas autorizadas tenham acesso às informações (BISHOP, 2004; WHITMAN e MATTORD, 2011).

É importante destacar que os fatores apresentados baseiam-se no uso de um recurso. O processo que analisa as requisições e as valida a fim de conceder acesso denomina-se controle de acesso, sendo o principal fator na proteção de recursos. É através dele que é possível conceder acesso ao recurso solicitado. Em geral, compreende três processos (TODOROV, 2007): autenticação, autorização e auditoria, que serão expostos a seguir.

Conforme Bishop (2004), o processo de autenticação consiste em obter a informação de autenticação de uma entidade, a análise dos dados, e determinar se ele está associado a essa entidade. A autorização é o processo que determina se um usuário já identificado e autenticado tem permissão para acessar aos recursos e informações (TODOROV, 2007). Autorização determina o nível de acesso e direito de um usuário a determinados recursos e/ou serviços, assim, a autenticação e fundamental para a autorização. A função da auditoria é registrar todas as ações dos usuários diante de um recurso possibilitando determinar os acessos (devidos e indevidos) ao recurso e registrar as ações dos usuários autenticados.

2.3.3 Autenticação do Usuário

Uma parte integrante da segurança de um sistema computacional é a autenticação, que procura confirmar a identidade de um usuário com a finalidade de conceder acesso aos seus recursos e informações. Todorov (2007) menciona que a autenticação normalmente se refere à determinação e validação da identidade do usuário. Segundo o autor, o processo de autenticação é constituído por duas etapas: a identificação, o usuário atual tem que comunicar a identidade que ele pretende usar. A etapa de autenticação, por sua vez, valida a identificação feita na etapa anterior, assim o usuário precisa fornecer a prova para a identidade selecionada. Se

a prova for válida, a identidade é aceita. O passo de identificação pode ser omitido, se a prova é única no sistema.

A autenticação é geralmente o primeiro estágio de interação dos usuários com a segurança do sistema, é o ponto de entrada, uma vez autenticado, o usuário pode ter acesso aos recursos e informações.

Segundo Whitman e Mattord (2011), a autenticação pode ser conseguida através da utilização de uma ou mais das três abordagens fundamentais, são elas: i) o que a entidade sabe (autenticação baseada em conhecimento; ii) o que a entidade tem (baseada na propriedade); iii) o que a entidade é (baseada na característica).

Os tipos de autenticação estão descritos a seguir:

- Autenticação baseada na característica (o que é): são chamados biométricos. A autenticação biométrica verifica a identidade baseando-se tanto em características físicas únicas (face, palma da mão, íris, etc.) como em características comportamentais (voz, escrita à mão, assinatura, dinâmica da digitação, etc.). Biometrias físicas são traços biológicos que são inatos ou desenvolvidos naturalmente e as biometrias comportamentais são características aprendidas ou desenvolvidas ao longo da utilização constante, e que podem variar fortemente ao longo do tempo (KARNAN, AKILA, e KRISHNARAJ, 2011).
- Autenticação baseada no conhecimento (o que se sabe): verificar um usuário com base em algo que ele sabe, por exemplo, senhas, informações pessoais ou qualquer que seja considerada desconhecida por outros. Essa é a forma de autenticação mais utilizada por sistemas em geral, tendo como base a definição de uma senha.
- Autenticação baseada na propriedade (o que se tem): Caracterizada por um objeto físico que o usuário possui, por exemplo, um *token*[14].

Existem diversos métodos que verificam a identidade de um usuário antes de lhe conceder o acesso aos recursos, no entanto, estas tecnologias oferecem diferentes níveis de segurança (KARNAN, AKILA, e KRISHNARAJ, 2011). Segundo Marques (2013), atualmente, os principais métodos de autenticação para *smartphones* dependem de *Personal Identification Number* (PINs), senhas, e, mais recentemente, códigos gráficos. A autenticação baseada em senha tem a vantagem

[14] Dispositivo eletrônico gerador de senhas.

de ser simples de implementar e de usar poucos recursos, no entanto, por ser uma abordagem baseada no conhecimento, podem ser aprendidas, portanto, são vulneráveis a uso indevido (SARAVANAN *et. al.*,2014).

Estudos apontam que as senhas ainda impõem uma sobrecarga cognitiva ao utilizador, resultando em adoção de práticas inseguras, como escolher senhas fracas ou reutilizar senhas (ADAMS e SASSE, 1999). Em um estudo realizado por Clarke *et. al.* (2002), 41% de seus entrevistados expressaram preocupações com relação aos PINs e senhas alfanuméricas, apoiando a necessidade de técnicas de autenticação alternativas.

Apesar de ser uma das formas de autenticação mais difundida da atualidade, a autenticação de usuários baseada em senhas apresenta diversas vulnerabilidades em termos de segurança. Um dos problemas de segurança mais usuais no processo de autenticação baseados em senhas e padrões gráficos em dispositivos móveis são os ataques de observação direta chamados de *shoulder surfing*. Como dispositivos móveis são frequentemente usados em locais públicos, a entrada pode facilmente ser observada por um atacante e a senha do usuário é exposta (DE LUCA, 2014). Aviv *et. al.* (2010) destacam a importância das ameaças por ataques por manchas, a maioria dos *samrtphones* de hoje possui como principal forma de interação a telas sensíveis ao toque, como isso, resíduos oleosos dos dedos dos usuários permanecem no visor do dispositivo e podem ser utilizadas para deduzir dados introduzidos pelo usuário.

O uso de *token* como mecanismo adicional de autenticação têm-se tornado cada vez mais comum no mercado. No entanto, para Furnell *et. al.* (2008), *tokens* não são uma solução prática no contexto móvel, já que o usuário precisa ter em mãos o objeto de autenticação, aumentando o risco de o objeto ser perdido ou esquecido.

Devido a estes problemas, técnicas de autenticação baseadas em características biométricas físicas e comportamentais vêm sendo utilizadas para garantir a autenticidade dos usuários. Com o aumento da capacidade computacional dos *smartphones* e dos acessórios acoplados a ele, como é o caso das câmeras digitais, tornou-se possível a implementação de métodos biométricos para estes dispositivos. Estes métodos tendem a se concentrar nas características físicas dos indivíduos, no entanto, muitas destas técnicas dependem de melhorias de hardware, por exemplo, leitores de impressões digitais ou *software* para autenticação eficiente.

Além disso, existem também preocupações dos utilizadores relacionadas com o armazenamento de dados biométricos (PONS e POLAK, 2008).

Biometria comportamental, por outro lado, é comumente usada para autenticação contínua. Como o termo comportamental indica, essas abordagens baseiam-se em pistas comportamentais dos usuários e a autenticação pode acontecer de forma implícita. Atualmente, pesquisadores estão investigando uma série de técnicas com foco em padrões de comportamento dos indivíduos, por exemplo, padrão de digitação no teclado (CLARKE e FURNELL, 2006).

Furnell *et. al.* (2008), consideram que as soluções biométricas juntamente como abordagens baseadas no conhecimento são as soluções mais prováveis para dispositivos móveis. De acordo com Babu e Venkataram (2009), existe uma tendência recente em considerar o comportamento dos usuários em sistemas computacionais, a fim de caracterizar os padrões de um usuário específico. Os autores definem um comportamento como as ações e reações de um usuário específico enquanto opera um sistema.

Segundo Peddemors (2009), os seres humanos são criaturas de hábitos, além disso, o uso destes dispositivos é caracterizado pelo contexto e varia de pessoa para pessoa, assim pode-se usar o comportamento repetitivo dependente do contexto para determinar um padrão de utilização. Uma questão importante em relação aos métodos de autenticação, é que existe uma inclinação a assumir que todas as aplicações, serviços e informações acessíveis nos dispositivos possuem igual valor (FURNELL *et. al.,* 2008). No entanto, cada serviço e/ou informação necessita níveis de segurança diferentes. Por exemplo, a proteção necessária exigida por uma mensagem de texto é diferente da proteção necessária para acessar uma conta bancária. Assim, pode ser atribuídas maior proteção as operações mais críticas (FURNELL *et. al.,* 2008).

2.3.4 Usabilidade

A usabilidade dos sistemas de segurança tornou-se uma questão importante na pesquisa sobre a eficiência e a aceitação do usuário.

Segundo Organização Internacional de Normalização (ISO) 9241-11[15], define-se usabilidade como:

> "À medida que um sistema, produto ou serviço pode ser utilizado por um determinado tipo de utilizador na concretização de determinados objetivos, com eficácia, eficiência e satisfação num determinado contexto de utilização".

Esta definição concentra-se nos objetivos dos usuários (eficácia), a velocidade com que os objetivos são alcançados (eficiência) e a satisfação dos usuários com o sistema dentro de um contexto especificado. Portanto, a definição de usabilidade é contextual, um sistema considerado utilizável em um contexto, pode não ser em outro (KAINDA *et. al.,* 2010).

As metas de segurança e usabilidade são conflitantes, no entanto, acredita-se que existe uma compensação entre estes conceitos (KAINDA *et. al.*, 2010). Segundo Cranor e Garfinkel (2005), o equilíbrio entre segurança e usabilidade é a base para se ter o comportamento de segurança adequado.

Do ponto de vista do usuário, os mecanismos de autenticação são muitas vezes vistos como obstáculo, assim, os usuários podem desativar mecanismos irritantes (MAYRHOFER, 2007). Os usuários muitas vezes não estão conscientes ou subestimam os riscos, que os mecanismos de autenticação podem combater.

Segundo Renaud (2005), a usabilidade de um mecanismo de autenticação pode fazer ou quebrar a segurança do sistema. Portanto, é necessário analisar a segurança, mas também a facilidade de utilização do mecanismo de autenticação. Para estudar este último é necessário concentrar-se no usuário, ou seja, identificar principais usuários e seus principais casos de uso.

2.5 Trabalhos Correlatos

Para o desenvolvimento desta pesquisa foram estudados alguns trabalhos correlatos, a fim de auxiliar na construção do conhecimento sobre o tema. A seguir são descritos três propostas, relevantes, relacionadas a presente pesquisa.

Hayash *et. al.* (2013), propõem uma solução CASA (*Context Aware Scalable Authentication*), o modelo utiliza múltiplos fatores passivos para ajustar o nível de

[15] Disponível em : http://www.iso.org/iso/home.html

autenticação. É utilizada uma função probabilística que seleciona o fator ativo em função de um conjunto de fatores passivos, entre eles, a localização do usuário. No protótipo implementado pelos autores, foram utilizados como fator ativo: PIN, senha, e nenhum fator, e como fator passivo a localização do usuário. Assim com base na localização selecionavam que fator ativo deveria ser ativado.

Os autores realizaram uma pesquisa empírica para investigar o potencial da utilização da localização como um fator para autenticação. O aplicativo utilizado na investigação registrava a localização dos usuários a cada três minutos, independentemente se os participantes estavam interagindo com seus telefones. Para controlar o uso do telefone, o aplicativo registrava os processos ativos a cada 30 segundos. Para cada participante, foi calculado o número de ativações de telefone em cada um dos locais onde os participantes passaram a maior parte de seu tempo. Os resultados indicaram que as pessoas gastaram 57,8% do seu tempo em dois locais, e estes dois lugares eram responsáveis por 60,8% do total de eventos de ativação do telefone. Estes resultados apresentam evidências que a localização afeta a utilização dos *smartphone*, sugerindo que localização pode ser um fator a ser considerado.

Shi *et. al.* (2010), propõem um mecanismo de autenticação implícita através da aprendizagem do comportamento do usuário. O mecanismo apresenta uma técnica para determinar o cálculo de uma pontuação de autenticação baseada nas atividades recentes do usuário. Dado um modelo de usuário e alguns comportamentos observados recentemente, se calcula a probabilidade de que o dispositivo está nas mãos do usuário legítimo. Para calcular tal pontuação, eventos habituais são identificados, a pontuação aumenta sempre que um evento habitual é observado, a pontuação diminui quando eventos negativos são detectados. Quando a pontuação chega a um limite inferior, o usuário deve autenticar explicitamente.

Pesquisadores de segurança do Instituto de Tecnologia da Geórgia desenvolveram um novo sistema é chamado LatentGesture. O sistema baseia-se no "toque de assinatura", que refere-se à maneira única de tocar e interagir com o dispositivo. O aplicativo desenvolvido é capaz de aprender os movimentos do usuário e travar o aparelho caso outra pessoa tente usá-lo. O sistema foi testado em um estudo técnico de laboratório usando telefones Android, e obtiveram resultados precisos, 98% dos telefones e 97% dos *tablets* apresentaram os resultados esperados (SARAVANAN *et. al.,* 2014).

Com uma base teórica fundamentada, o próximo capítulo apresenta o procedimento metodológico adotado neste trabalho.

3. METODOLOGIA

Neste capítulo descreve-se o procedimento metodológico adotado para atingir os objetivos desta pesquisa. Primeiramente aborda-se a caracterização da pesquisa, em seguida, é descrito o procedimento metodológico adotado.

3.1 Caracterização da Pesquisa

Vergara (2006), propõe dois critérios de classificação para o tipo de pesquisa, caracterizados quanto aos fins e quanto aos meios. Quanto aos fins, a pesquisa pode ser classificada como: exploratória, descritiva, metodológica aplicada e intervencionista. Quanto aos meios, a pesquisa pode ser classificada como: documental, bibliográfica, de laboratório, participante e estudo de caso.

Assim a presente pesquisa foi definida como exploratória, tendo como objetivo proporcionar maior entendimento sobre o tema em estudo. Quanto aos meios, classifica-se como bibliográfica, uma vez que foram utilizados materiais já elaborados, constituídos principalmente de livros e artigos científicos para levantar informações sobre os principais assuntos relacionados à pesquisa.

Segundo Gil (2010), as pesquisas podem ser classificadas conforme sua finalidade, em básica, baseando-se na aquisição de novos conhecimentos e pesquisa aplicada, voltada para a aplicação de conhecimentos já existentes para a aquisição de novos conhecimentos e resolução de problemas encontrados no ambiente dos pesquisadores.

A pesquisa será realizada de forma aplicada já que tem como objetivo propor uma solução para um problema específico.

3.2 Procedimento Metodológico

A metodologia empregada para o desenvolvimento do trabalho consiste inicialmente de uma pesquisa bibliográfica referente a conceitos relevante para o desenvolvimento do trabalho, com o objetivo formar um referencial teórico consistente e visualizar o estado da arte. Essa etapa envolveu estudos sobre segurança da informação, dispositivos móveis e plataforma Android. Em seguida, foi realizada uma pesquisa sobre as principais técnicas utilizadas para autenticação do

usuário em dispositivos móveis identificando as vantagens e desvantagens de cada uma. O resultado desta pesquisa é apresentado no capítulo 2.

Com base no conhecimento adquirido, foi realizada a descrição de um modelo de autenticação para dispositivos móveis. Para isso, foi realizado um estudo de trabalhos correlatos que fundamentaram o modelo proposto. A especificação do modelo foi desenvolvida utilizando diagramas *Unified Modeling Language* (UML). Para modelagem e geração dos diagramas foi utilizado o *software* Astah Community[16].

Com a especificação do modelo concluída foi iniciada a etapa de desenvolvimento do protótipo da solução proposta. Para do desenvolvimento do protótipo foi utilizada a linguagem de programação Java, foi utilizado o ambiente de desenvolvimento integrado (IDE) Android Studio[17], juntamente com o emulador Android Genymotion[18]. Genymotion é um emulador do Android disponível para Windows, Linux e Mac OS que oferece dispositivos virtuais de vários *smartphones* e *tablets* Android para testar aplicativos em um *desktop*. Este ambiente foi configurados sob a plataforma Windows 7.

A seguir foram realizados alguns testes empíricos, em seguida foram avaliados os resultados obtidos nos testes e como base nestes resultados foi realizada uma discussão sobre os princípios de segurança da aplicação desenvolvida.

[16] Disponível em: http://astah.net/
[17] Disponível em http://developer.android.com/intl/es/sdk/index.html
[18] Disponível em https://www.genymotion.com

4. MODELO DE AUTENTICAÇÃO

Com base na proposta de Hayash *et. al.* (2013), que consiste em selecionar o fator de autenticação em função de outros fatores, no modelo proposto, utilizamos a localização como um fator decisivo para selecionar o método de autenticação que será ativado. Assim, o nível de segurança oferecido depende da localização do usuário.

Na solução proposta por Shi *et. al.* (2010), a latência de detecção depende da combinação dos diferentes recursos utilizados para o cálculo da pontuação. Além disso, os autores utilizam o comportamento do usuário para o cálculo da pontuação para negar ou aceitar a autenticação. Experimentos realizados pelos autores mostraram que as características obtidas a partir do histórico do dispositivo podem ser eficazes para distinguir os utilizadores. O modelo proposto se baseia na ideia de autenticação implícita proposta pelos autores, porém utiliza o comportamento do usuário para criar perfis de uso, ou seja, em um determinado contexto, se a autenticação implícita estiver ativa, uma serie de aplicações estarão disponíveis sem necessidade de interação direta com o método. O perfil consiste em características que são construídas usando a história de ações do usuário em seu dispositivo ao longo do tempo.

O modelo de controle de acesso proposto está relacionado com a biometria comportamental e uma abordagem baseada no conhecimento. Permite ao usuário realizar autenticação explícita ou implícita, o método implícito refere-se a uma abordagem que utiliza as observações do comportamento do usuário para autenticação, ou seja, o usuário não precisa participar ativamente no processo autenticação.

A autenticação implícita protege informações e recursos do telefone com base em seu contexto atual, assim, com o comportamento do usuário podemos definir quais os dados e serviços são acessíveis em um contexto específico. A ideia principal do modelo é adaptar a tela ao contexto em que o usuário se encontra. Desta maneira a aplicação precisa levar em consideração as características de uso de utilizadores individuais, sendo necessário que a aplicação seja capaz de aprender e adaptar-se ao comportamento do usuário.

A necessidade de autenticação frequente impõe uma sobrecarga indevida o utilizador. Conforme Clarke e Furnell (2006), um sistema que pode realizar a

autenticação continuamente do usuário sem interromper a interação normal é uma solução desejada por usuários de dispositivos móveis. Assim, um sistema de autenticação implícita fornece uma maneira de proteger informações sensíveis sem incomodar o utilizador.

Para conseguir o objetivo, utilizam-se as informações de contexto disponíveis nos dispositivos móveis, tempo e localização. Quando usamos o dispositivo móvel realizamos ações repetitivas que dependem de certas condições do contexto, assim, com estas informações pode-se determinar um padrão de comportamento cotidiano do usuário. Com base no padrão identificado define-se que informações e/ou serviços estarão disponíveis em um contexto específico.

4.1 Descrição do Modelo

A solução proposta consiste em uma modelo de controle de acesso a dispositivos móveis que tem como base o contexto do usuário. Várias informações diferentes, como a localização do usuário, tempo entre interações e autenticação anteriormente realizada são cruzadas para identificar o método de autenticação será ativado.

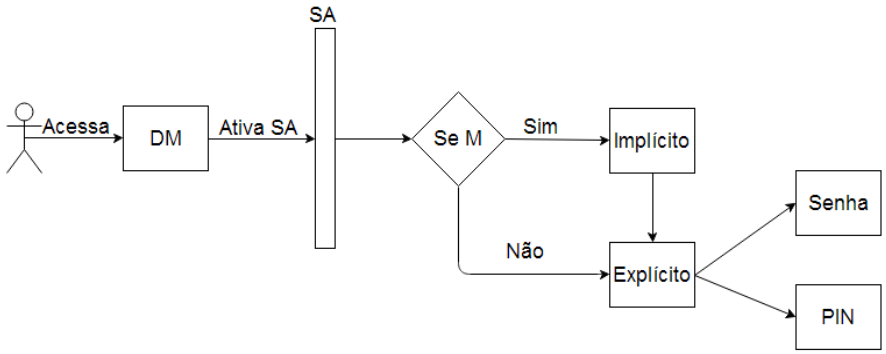
O controle de acesso adapta-se ao contexto onde o usuário se encontra, com o tempo a ferramenta aprende quais aplicativos são usados em qual contexto e fornece acesso direto a esses aplicativos.

A aplicação possui três funcionalidades básicas:

- Autenticação explícita do usuário;
- Autenticação implícita do usuário;
- Determinar padrão de comportamento;

A Figura 5 apresenta de maneira simplificada o modelo de autenticação proposto.

Figura 5 - Modelo de autenticação proposta.



Fonte: Próprio Autor, 2014.

Quando o usuário acessa o dispositivo móvel, o sistema de autenticação (SA), previamente instalado, é ativado, assim a identidade do usuário é comprovada pelo processo de autenticação. A autenticação pode ser:

- Autenticação explícita: Este método consiste na abordagem baseada no conhecimento (PINs e senhas). A abordagem utilizada para autenticar explicitamente variar de acordo com a localização do usuário.
- Autenticação implícita: comprova-se a identidade do usuário levando em consideração o padrão de comportamento do usuário. Isto é, em um determinado contexto o usuário terá acesso aos recursos identificados como o padrão de comportamento para este contexto. Se o usuário desejar acessar outro recurso autenticação explícita será ativada.

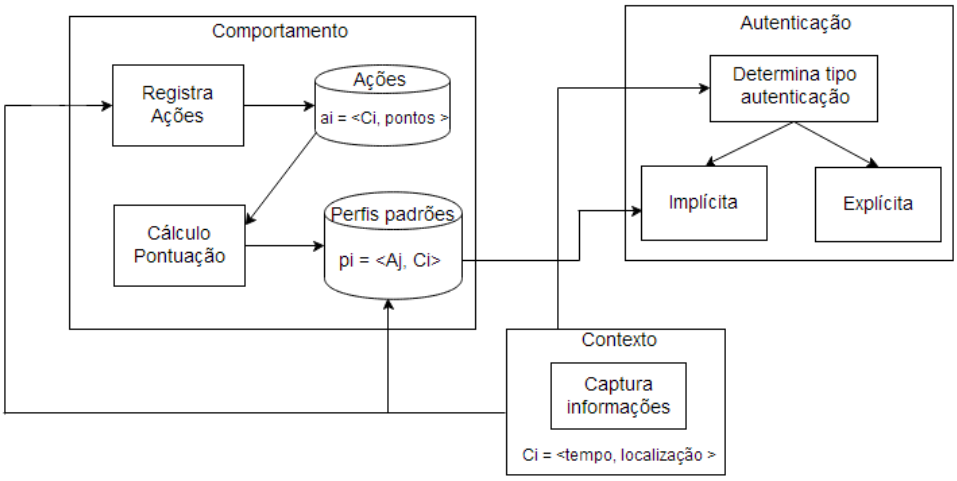
A componente de condição M considera as informações de contexto de localização, informações sobre tempo de intervalo entre uma sessão e outra e tipo de autenticação anterior para determinar qual método de autenticação será ativado. A autenticação implícita só será ativada se certas condições são satisfeitas. O objetivo é proporcionar ao usuário uma solução que oferece usabilidade e assim garantir a segurança.

4.2 Arquitetura do Modelo

Nesta seção apresentamos a arquitetura da solução proposta, descrevendo todos os componentes e os elementos de cada componente, assim como, o fluxo de informações entre os componentes.

A arquitetura do sistema foi dividida em três componentes principais, são eles: comportamento, contexto e autenticação. A Figura 6 ilustra os três componentes principais e a relação entre eles.

Figura 6 - Arquitetura da autenticação proposta.



Fonte: Próprio Autor, 2014.

O objetivo do componente de comportamento é obter um padrão de comportamento do usuário. O modelo é obtido a partir do comportamento passado do usuário. Quando ocorrer a autenticação explícita, se assume que o utilizador é de fato o verdadeiro proprietário do dispositivo. Assim, quando ocorre a autenticação explícita independente do método utilizado, as ações dos usuários sobre o dispositivo são monitoradas a fim de determinar um padrão de comportamento.

O comportamento do usuário pode ser descrito através do relacionamento entre contexto e atividade. Define-se o comportamento do usuário como:

$$pi = < A_i , C_i >$$

onde pi representa o padrão de comportamento do usuário no contexto C_i e A_i representa o conjunto de atividades realizadas em um contexto específico.

O conjunto de atividades pode ser definido por:

$$A_i = < a_1, a_2, a_3 ..., a_n >$$

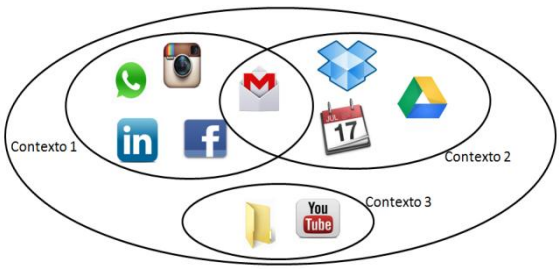
onde (a1...an) representa cada atividade realizadas pelo usuário. Atividade refere-se a qualquer ação realizada pelo usuário sobre o dispositivo, por exemplo, acessar email, visualizar fotos.

A localização e o tempo desempenham um papel importante na modelagem comportamental de indivíduos em um ambiente móvel, já que cada evento tem uma correlação entre estes dois atributos.

Definido um contexto específico, cada vez que o usuário realiza uma nova atividade a autenticação explícita será ativada e será atribuída uma pontuação para essa atividade. Depois de registrada varias vezes, a pontuação atinge um nível aceitável (previamente definido), com isso, assume-se que o usuário apresenta uma tendência a realizar essa ação nesse contexto específico, assim, essa atividade passará a formar parte do conjunto de atividade padrão do usuário. Uma nova atividade registrada será considerada um comportamento normal. Se essa ação não for registrada por certo período de tempo a pontuação irá decair, quando atingir o limite a atividade será excluída do conjunto de atividade padrão do usuário.

Quando ocorrer a autenticação implícita, o usuário terá acesso somente aos recursos especificados pelo padrão de comportamento identificado. A Figura 7 ilustra esse conceito. Quando ocorre a autenticação explícita o usuário tem acesso a todos os recursos do dispositivo.

Figura 7 - Modelo de acesso a recursos dependendo do contexto.



Fonte: Próprio Autor, 2014.

O usuário terá a opção de registrar os aplicativos que quer ter acesso direto num contexto específico. No entanto, a solução oferece mecanismo para que o

sistema possa adaptar-se dinamicamente ao comportamento do usuário através interações com o sistema.

Para identificar o contexto do usuário utilizamos recursos que são comumente encontrados em dispositivos móveis, como *smartphones.*

O contexto pode ser definido por:

Ci = <tempo, localização>

Tempo refere-se a um período, com base no horário será calculado o período do dia, por exemplo: manhã, tarde e noite. Um exemplo de contexto pode ser c= <tarde, trabalho>. A função do componente contexto é capturar as informações de contexto para passar essas informações para os outros componentes.

O componente de autenticação considera as informações enviadas pelo componente de contexto e as informações sobre tempo de intervalo entre uma sessão e outra para determinar qual método de autenticação será ativado.

A estrutura para escolher qual método será ativado leva em considerações alguns fatores, são eles:

- Intervalo de tempo (A): As interações com o dispositivo ocorrem com muita frequência, assim, será considerado o intervalo de tempo entre as interações. Se o tempo de intervalo entre as sessões é menor ou igual a um t definido então A=1, caso contrário A=0.
- Localização (B): se localização é a mesma onde aconteceu a ultima autenticação a variável C=1, de outra forma C=0.

Com isso, o tipo de autenticação (explícita ou implícita) depende dos fatores descritos anteriormente, assim

Tipo de Autenticação = <Intervalo de Tempo, Localização>

A Tabela 3 especifica quando ocorre à autenticação explícita (1) e implícita (0). Para isso foi utilizado uma combinação binária dos 2 fatores.

Tabela 3- Tipos de autenticação.

A	B	Tipo de Autenticação (S)
1	1	0
1	0	0
0	1	1
0	0	1

Fonte: Próprio Autor, 2014.

Para garantir segurança ao processo de autenticação implícita, o usuário só terá acesso aos recursos utilizados habitualmente

O método de autenticação explícita oferece flexibilidade por permitir diferentes níveis de segurança, que depende da abordagem utilizada. No protótipo que será implementado será utilizado senha alfanuméricas (maior segurança) e PIN (menor segurança). Quando for detectada uma localização conhecida será ativado o PIN, caso contrário, será ativada a senha alfanumérica. Assim, pode-se melhorar a segurança em locais menos visitados, melhorando a usabilidade em locais visitados frequentemente, por exemplo, casa e local de trabalho.

5. IMPLEMENTAÇÃO DO MODELO

Neste capítulo, serão apresentados detalhes da implementação do modelo proposto. Em primeiro lugar foi realizado uma análise dos requisitos do sistema. Em seguida é apresentada a fase de modelagem do sistema utilizando diagramas UML.

5.1 Definição de Requisitos

Segundo Sommerville (2011), os requisitos de um sistema são uma descrição do que o sistema deve fazer os serviços que o sistema oferece e suas restrições. Para auxiliar o desenvolvimento da aplicação, foram definidos os requisitos que devem ser atendidos por ela. Nesta seção são apresentados os requisitos funcionais e não funcionais identificados.

5.1.1 Requisitos Funcionais

O objetivo principal do sistema é permitir a autenticação do usuário, que pode de forma explícita ou implícita. Desta forma, o sistema deve fornecer as seguintes funcionalidades:

- Localizar dispositivo: A aplicação deve ser capaz de capturar informações de localização.
- Registrar senha: O usuário deve ser capaz de registrar a senha para o método de autenticação explícita.
- Registrar horário: deve ser capaz de registrar o tempo que ocorrer às atividades a fim de determinar o período do dia (por exemplo, manha, tarde ou noite).
- Determinar comportamento padrão do usuário: A aplicação deve ser capaz de monitorar todas as atividades realizadas pelo usuário no dispositivo a fim de determinar o comportamento padrão.
- Armazenamento de dados: a aplicação deve ser capaz de armazenar os dados persistentes.

- Realizar autenticação: deve ser capaz de autenticar explícita ou implicitamente o usuário a fim de conceder acesso aos recursos.

5.1.2 Requisitos não Funcionais

Os requisitos não funcionais estão relacionados às propriedades do sistema (SOMMERVILE, 2011). Foram identificados alguns requisitos não funcionais do sistema, apesar de não serem essenciais para o funcionamento do sistema, são considerados por questões de qualidade do sistema.

- Minimizar uso da bateria: dispositivos móveis são alimentados por bateria. A autenticação implícita acontece em segundo plano, este processo vai utilizar uma parte da capacidade da bateria, assim o sistema deve consumir o mínimo de recursos.
- Minimizar dados armazenados: Os dados coletados devem ser armazenados em algum lugar no dispositivo temporariamente ou permanentemente. Devido à reduzida capacidade de armazenamento destes dispositivos, e importante reduzir e otimizar os dados armazenados.
- Desempenho: Deve oferecer um desempenho satisfatório.
- Facilidade de uso: deve ser intuitivo, de fácil utilização.

5.2 Modelagem do Sistema

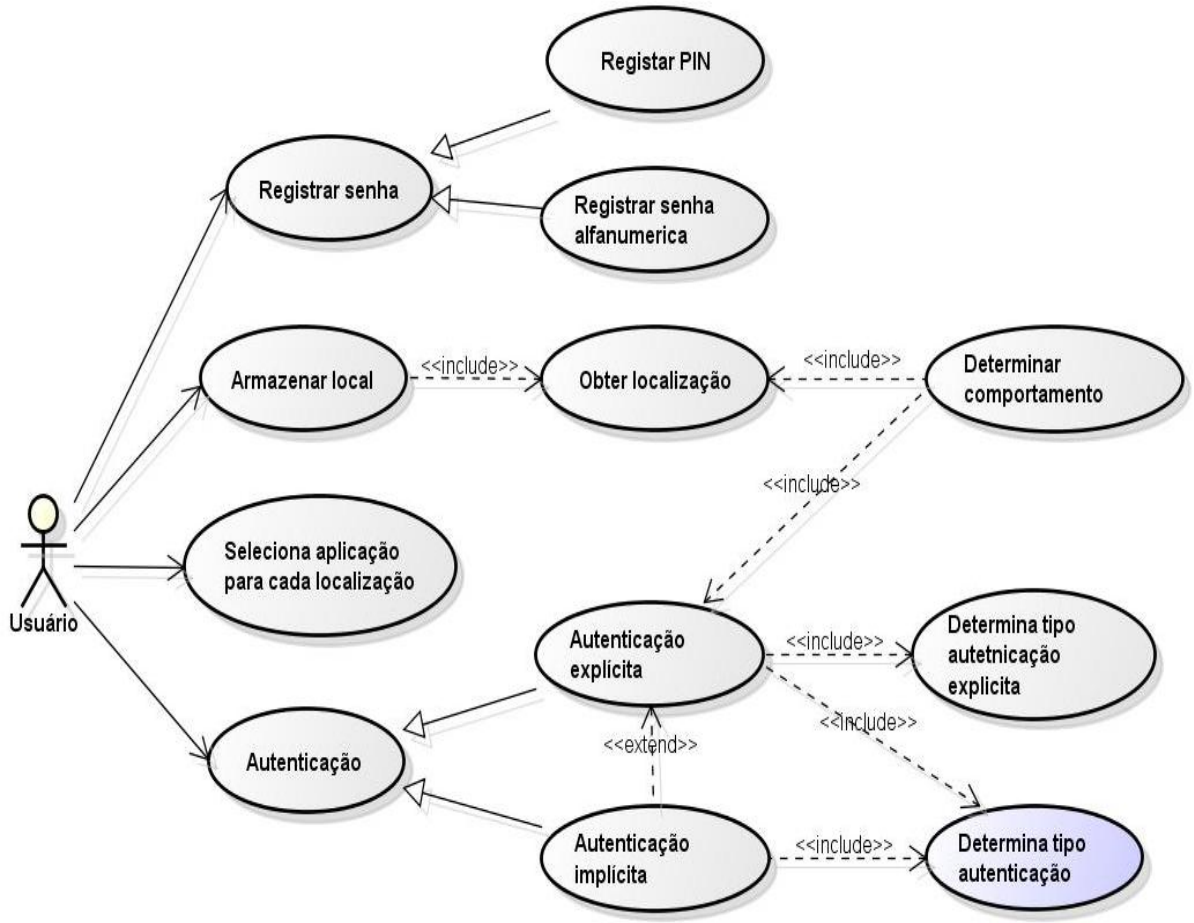
A modelagem do sistema é um processo que consiste no desenvolvimento de modelos abstratos, onde cada um representa uma perspectiva diferente do sistema (SOMMERVILLE, 2011). Assim, a modelagem desenvolvida foi baseada em notação UML, para isso foi utilizada a ferramenta *Astah Community*, a qual foi responsável geração dos diagramas.

5.2.1 Casos de Uso

Nesta seção são descritos os casos de uso do protótipo que contempla as principais funcionalidades da aplicação, assim este diagrama apresenta uma visão

simples de uma interação de um ator com o sistema (SOMMERVILLE, 2011). Os casos de uso são ilustrados na Figura 8.

Figura 8 - Casos de Uso.



Fonte: Próprio Autor, 2014.

A seguir, são apresentadas tabelas com a descrição dos casos de uso.

O caso de uso registrar senha especializa o comportamento de outros dois casos de uso: registrar PIN e registrar senha alfanumérica. Assim, o usuário poderá selecionar o método de autenticação e registra a senha que será usada quando este estiver ativo.

Tabela 4 - Descrição do caso de uso Registrar senha

Nome do caso de uso	Registra senha
Ator principal	Usuário
Descrição	Usuário registra a senha
Pré- condições	
Pós- condições	Senha armazenada

Fonte: Próprio Autor, 2014.

O caso se uso Armazenar local possibilita ao usuário armazenar os locais onde a autenticação implícita será ativada. Para isso, é necessária a inclusão do caso de uso obterá a localização.

Tabela 5 - Descrição do caso de uso Armazenar local

Nome do caso de uso	Armazenar local
Ator principal	Usuário
Descrição	Seleciona os locais onde a autenticação implícita será ativada
Pré- condições	Usuário autenticado no sistema
Pós- condições	Local armazenado

Fonte: Próprio Autor, 2014.

Para cada localização armazenada, o usuário será capaz de selecionar as aplicações que terá acesso sem necessidade de autenticação explícita.

Tabela 6 - Descrição do caso de uso Selecionar aplicação para cada localização

Nome do caso de uso	Seleciona aplicação para cada localização
Ator principal	Usuário
Descrição	Usuário seleciona as aplicações para que sejam disponibilizadas em cada localização
Pré- condições	Usuário autenticado no sistema Local armazenado
Pós- condições	Aplicações armazenadas no perfil do usuário

Fonte: Próprio Autor, 2014.

A determinação da localização do dispositivo desempenha um papel importante na abordagem proposta, já que a localização é um fator fundamental para definir o contexto e o comportamento do usuário. Obtém-se a localização e o tempo a fim de determina o contexto ci =<tempo, localização>.

Tabela 7 - Descrição do caso de uso Obter localização

Nome do caso de uso	Obter localização
Ator principal	Usuário
Descrição	Sistema determina a localização do dispositivo
Pré- condições	
Pós- condições	Localização obtida ci =<tempo, localização>

Fonte: Próprio Autor, 2014.

As ações realizadas sobre o dispositivo móvel sou monitoradas pelo sistema com o objetivo de obter um comportamento padrão para cada localização.

Tabela 8 - Descrição do caso Determinar comportamento

Nome do caso de uso	Determinar comportamento
Ator principal	Usuário
Descrição	Sistema monitora as ações do usuário para determinar o comportamento padrão para cada localização e período de tempo.
Pré- condições	Local armazenado Usuário autenticado
Pós- condições	Atividade armazenada no banco de dados ai=<Ci, pontuação> Atividade armazenada no perfil do usuário

Fonte: Próprio Autor, 2014.

O caso de uso Autenticação especializa o comportamento de outros dois casos de uso, são eles autenticação explícita e autenticação implícita.
Com relação à autenticação explícita, será solicitado ao usuário que digite a senha ou PIN previamente registrado.

Tabela 9 - Descrição do caso Autenticação explícita

Nome do caso de uso	Autenticação explícita
Ator principal	Usuário
Descrição	Usuário digita senha ou PIN, sistema verifica e autentica
Pré- condições	Senha ou PIN armazenada
Pós- condições	Usuário tem acesso ao dispositivo

Fonte: Próprio Autor, 2014.

O usuário acessa ao dispositivo sem a necessidade de digitação de senha ou PIN, porém o acesso é restringido a certos recursos definidos no comportamento do usuário.

Tabela 10 - Descrição do caso Autenticação implícita

Nome do caso de uso	Autenticação implícita
Ator principal	Usuário
Descrição	Usuário acessa ao dispositivo
Pré- condições	Padrões de comportamento armazenado
Pós- condições	Usuário tem acesso aos recursos disponíveis no padrão de comportamento

Fonte: Próprio Autor, 2014.

O objetivo deste caso de uso é determinar qual o método de autenticação será ativado levando em considerações alguns fatores considerados relevantes.

Tabela 11 - Descrição do caso de uso Determinar tipo autenticação

Nome do caso de uso	Determinar tipo autenticação
Ator principal	Usuário
Descrição	Tipo de Autenticação=<Ultima Autenticação, Intervalo de Tempo, Localização>. O sistema faz uma combinação binária dos 3 fatores e determina o tipo de autenticação que será ativado
Pré- condições	Senhas registradas Padrão de comportamento armazenado
Pós- condições	Método de autenticação ativado

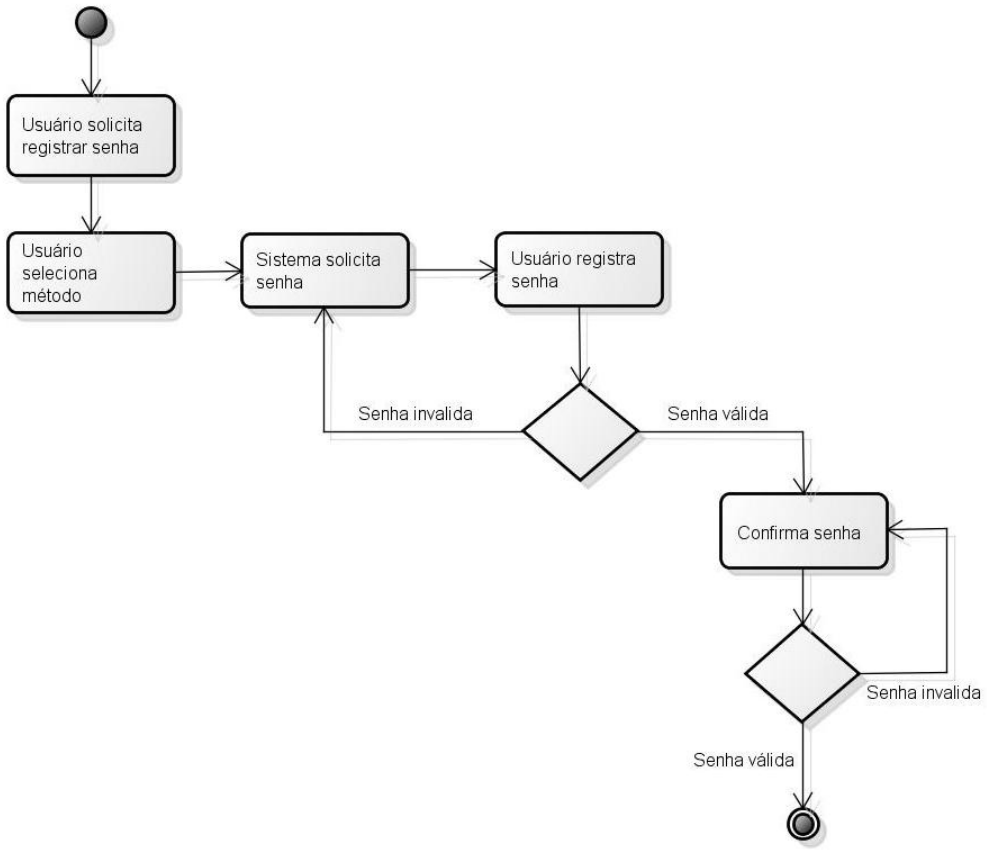
Fonte: Próprio Autor, 2014.

Uma vez descritos os principais casos de uso do sistema, pode-se descrever o fluxo de atividade de cada um.

5.2.2 Diagrama de Atividades

O diagrama de atividades define os fluxos de eventos que são realizados desde o início de um caso de uso até seu encerramento. A seguir são apresentados os diagramas de atividade de cada caso de uso apresentado anteriormente.

Figura 9 - Diagrama de atividade Registrar senha



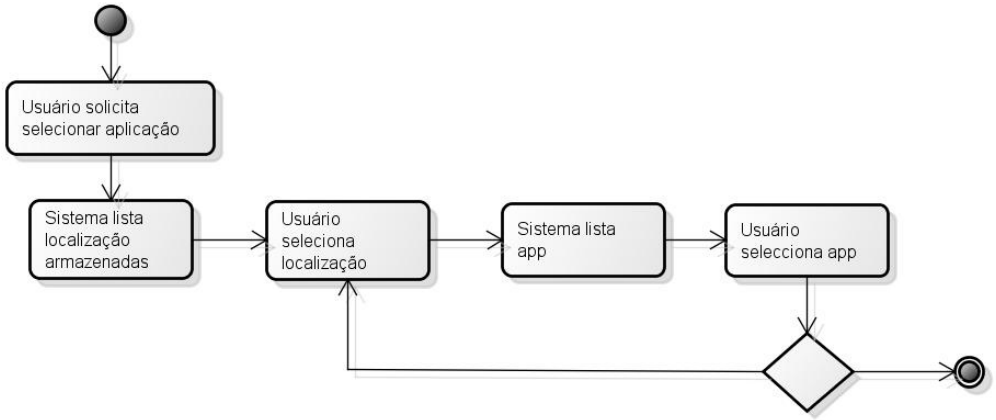
Fonte: Próprio Autor, 2014.

A Figura 9 apresenta o fluxo de atividades necessário para registrar uma senha, válido tanto para PIN como para senhas alfanuméricas. O fluxo de eventos principais consiste em 5 atividades.

1. Usuário solicita registrar senha;
2. Usuário seleciona método;
3. Sistema solicita senha;

4. Usuário registra a senha, se a senha for valida ocorre o passo 5, caso contrário executa-se o passo 3;
5. Usuário confirma a senha, se a senha for valida o caso de uso é finalizado, senão retorna ao passo 4.

Figura 10 - Diagrama de atividade Selecionar aplicação para cada localização.

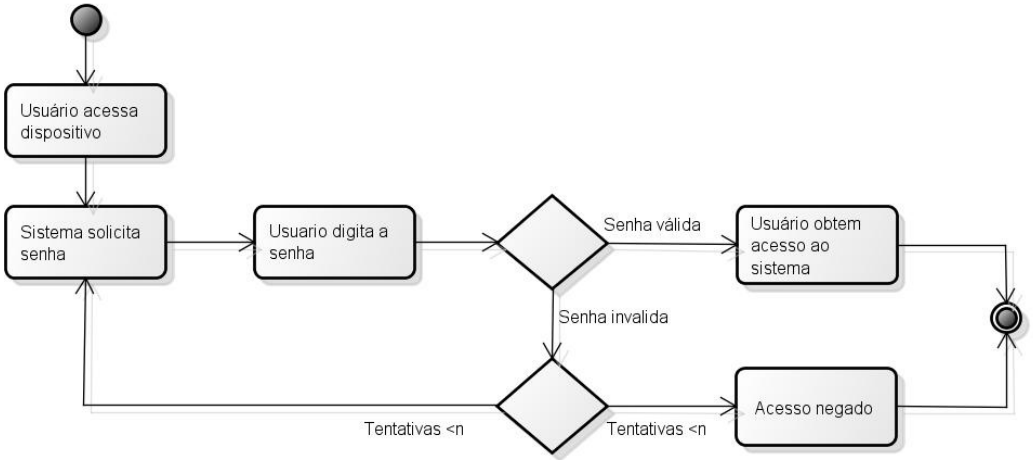


Fonte: Próprio Autor, 2014.

O fluxo de eventos principal de selecionar uma aplicação consiste em 5 eventos, como mostra a Figura 10.

1. O caso de uso é iniciado quando o usuário solicita selecionar aplicação;
2. O sistema apresenta ao usuário uma lista de localizações armazenadas;
3. Usuário seleciona a localização de interesse;
4. O sistema apresenta ao usuário a lista de aplicações disponíveis do dispositivo;
5. Por fim, o usuário seleciona todas as aplicações que deseja, se o usuário desejar selecionar outra localização executa-se a passo 3.

Figura 11 - Diagrama de atividade de Autenticação explícita.

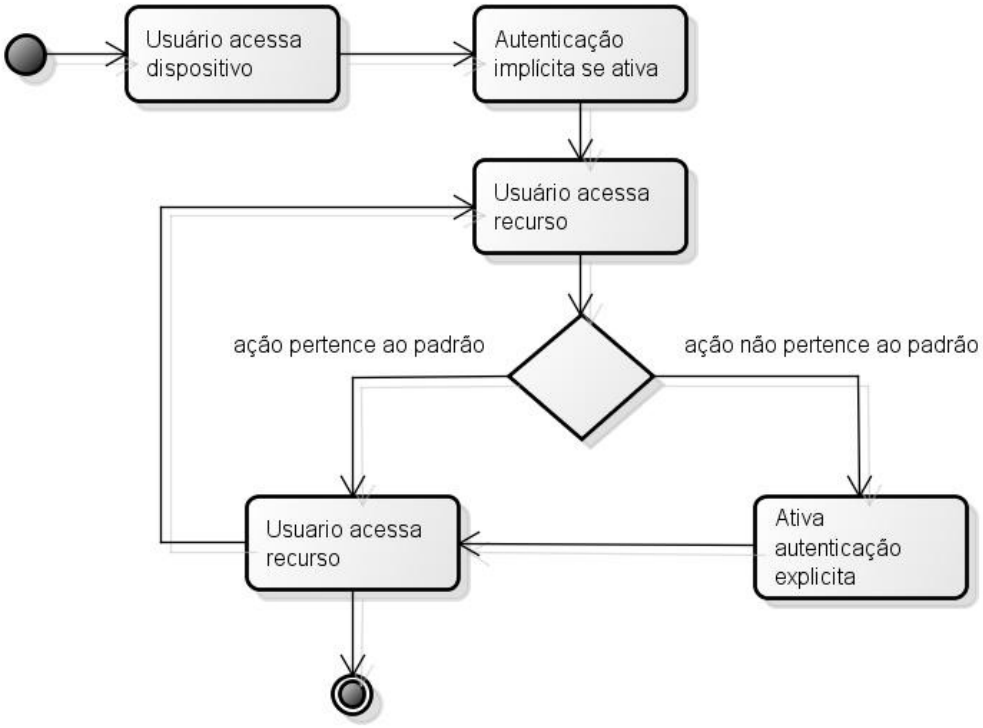


Fonte: Próprio Autor, 2014.

Como pode ser observado na Figura 11, o fluxo principal deste caso de uso é composto por 4 eventos, são eles:

1. É iniciado quando o usuário acessa ao dispositivo;
2. O sistema solicita a senha ou PIN, dependendo do tipo de autenticação que estiver ativa;
3. Usuário digita a senha;
4. Se a senha for correta o sistema fornece acesso ao sistema. Caso contrário, verifica o número de tentativas, se for menor ao numero permitido, retorna ao passo 2, senão finaliza negando o acesso.

Figura 12 - Diagrama de atividade autenticação implícita.

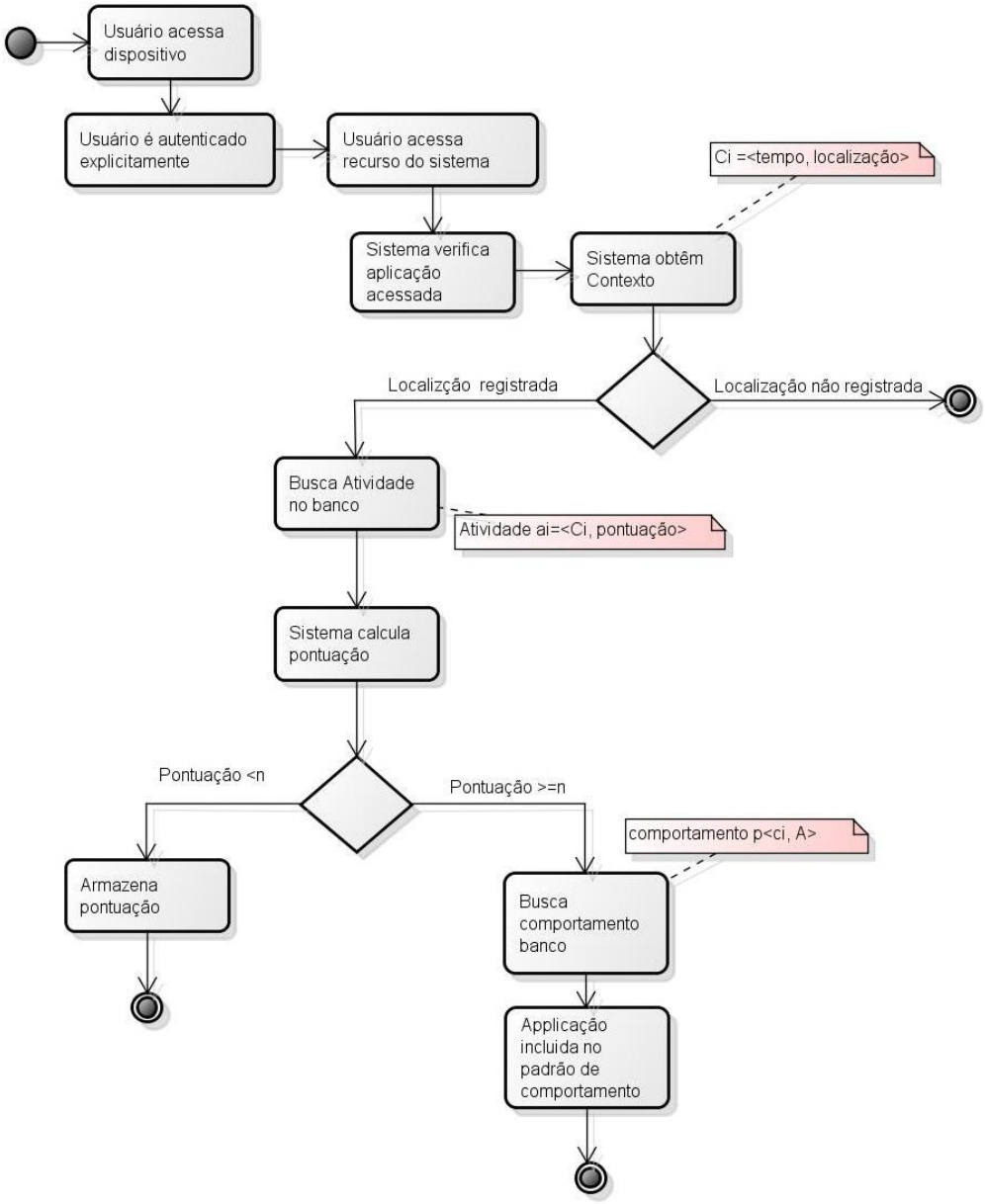


Fonte: Próprio Autor, 2014.

Como exposto na Figura 12, para realizar a autenticação implícita são necessário 4 eventos principais apresentados a seguir:

1. O usuário acessa ao dispositivo;
2. O sistema ativa o método que monitora as ações sobre o dispositivo;
3. Logo, o usuário tem acesso aos recursos disponíveis no contexto específico;
4. Se não for constatado nenhum comportamento anormal, o usuário continua com acesso, caso contrário, a autenticação explícita é ativada.

Figura 13 - Diagrama de atividade determinar comportamento.



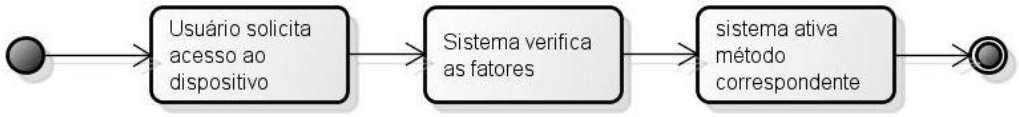
Fonte: Próprio Autor, 2014.

O fluxo principal deste caso de uso contém 10 eventos:

1. O caso de uso é iniciado quando o usuário acessa ao dispositivo;
2. Logo, é necessário que a autenticação explícita seja realizada;
3. Usuário tem acesso a todos os recursos do sistema;
4. Sistema identifica a aplicação acessada;
5. Obtém ci= <tempo, localização>;

6. Se o contexto não estiver armazenado no sistema o caso de uso é encerrado, caso contrário, se realiza a busca da atividade no banco;
7. O sistema calcula a pontuação para essa atividade;
8. A pontuação é comparada com um valor definido, se não for atingido esse valor, a pontuação é armazenada e o caso de uso é finalizado;
9. Se a pontuação atingir esse valor, se realiza a busca do comportamento p_i = <c_i, A_i>,armazenada para esse contexto;
10. Por fim, a aplicação é incluída no conjunto de aplicações de p_i.

Figura 14 - Diagrama de atividade Selecionar tipo Autenticação.

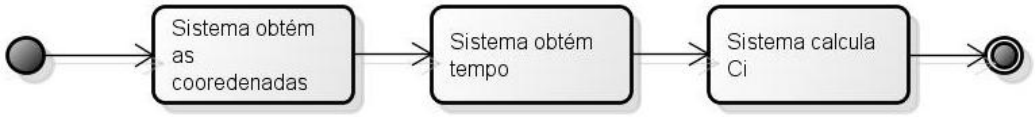


Fonte: Próprio Autor, 2014.

Este caso de uso apresenta somente 3 eventos principais, apresentados na Figura 14, são eles:

1. Inicialmente o usuário solicita acesso ao dispositivo;
2. Sistema verifica os fatores definidos;
3. Sistema ativa o método de acordo com os fatores verificados.

Figura 15 - Diagrama de atividade de Obter localização.



Fonte: Próprio Autor, 2014.

O objetivo deste caso de uso é obter o contexto descrito pela localização e pelo tempo, assim tem-se 3 eventos principais, ilustrados na Figura 15:

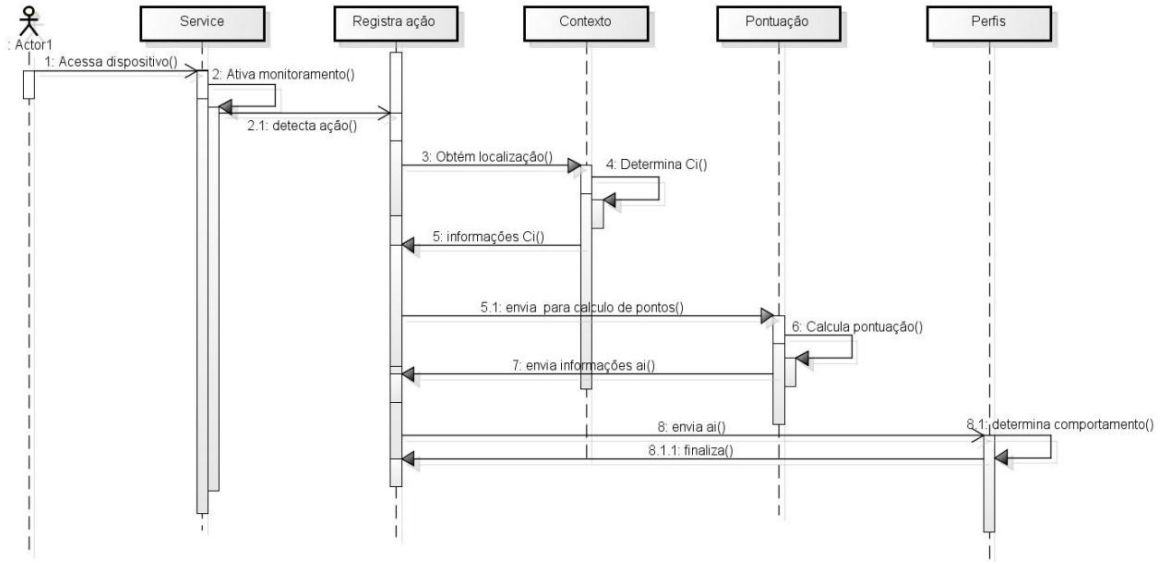
1. O sistema obtém as coordenadas geográficas;
2. O sistema também registra o horário que ocorre a etapa 1;
3. Com base nestas informações é calculado o Ci= <tempo, localização>.

Com base na análise realizada para cada caso de uso, podemos definir as interações entre os objetos para executar os casos de uso.

5.2.3 Diagrama de Sequência

O diagrama de sequência é um diagrama de interação que representa como os objetos do sistema interagem entre si para que um ator atinja seu objetivo na execução de um caso de uso (BEZERRA, 2006). A seguir, são apresentados os diagramas de sequência dos principais casos de uso.

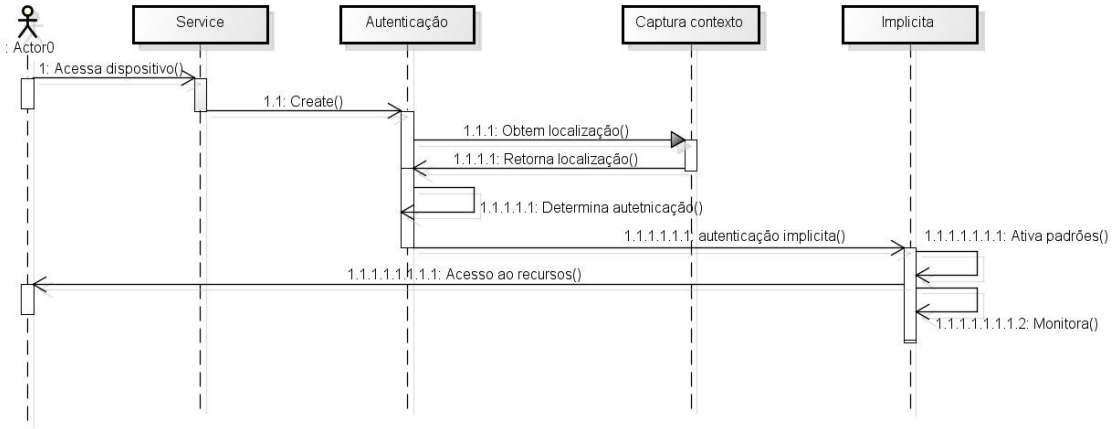
Figura 16 - Diagrama de sequência Determina comportamento.



Fonte: Próprio Autor, 2014.

A Figura 16 representa a sequencia de interações entre objetos para determinar o comportamento do usuário. Quando o usuário tem acesso ao dispositivo, o serviço, que é ativado para monitorar as ações sobre o dispositivo, cria uma instancia de um objeto que fica responsável por uma determinada atividade. A partir deste objeto são instanciados os restantes para concluir a tarefa.

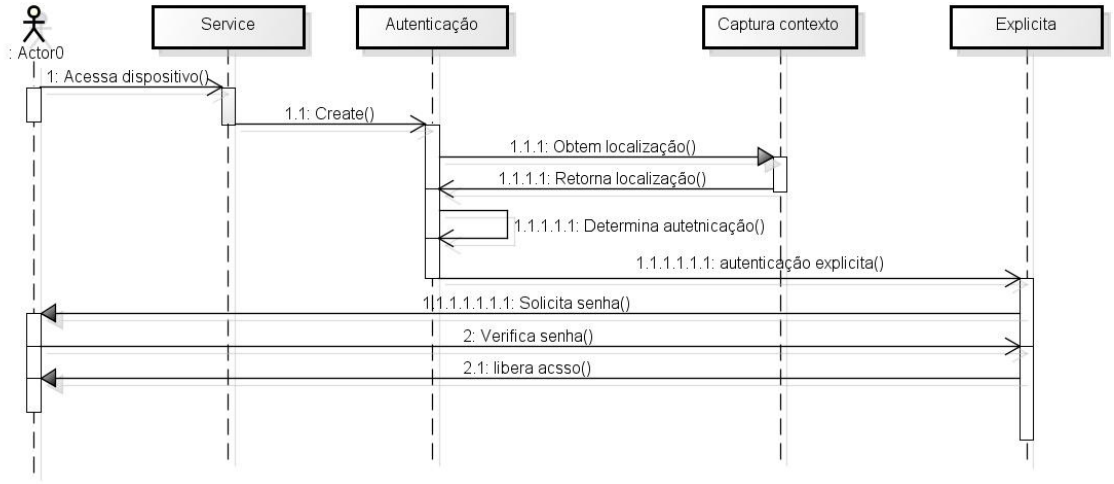
Figura 17 - Diagrama de sequência Autenticação implícita.



Fonte: Próprio Autor, 2014.

Na Figura 17, observa-se que quando o usuário tem acesso ao dispositivo a fim de obter uma autenticação implícita, a classe Service, responsável por monitor o dispositivo, cria um objeto da classe Autenticação para determinar o tipo de autenticação. Para isso, é necessário obter a localização, em seguida, o padrão de comportamento correspondente é ativado, assim, o usuário tem acesso aos recursos. Logo é ativado o método que monitora as ações do usuário, responsável por identificar qualquer ação não habitual.

Figura 18 - Diagrama de sequência Autenticação explícita.



Fonte: Próprio Autor, 2014.

Pode-se observar na Figura 18 que ocorre às mesmas instâncias observadas no diagrama anterior. No entanto, será realizada uma autenticação explícita, para

isso, apos determinar o tipo de autenticação, será instanciado um objeto da classe Explícita o qual será responsável pela interação com o usuário.

Após concluir o projeto do software, a próxima etapa do trabalho consiste no desenvolvimento de um protótipo para validação do modelo proposto neste trabalho.

6. DESENVOLVIMENTO DO PROTÓTIPO

Uma vez expostos os objetivos e características principais do modelo, será apresentado detalhadamente o processo de desenvolvimento do protótipo referente à proposta. O objetivo prosseguido nas linhas seguintes é mostrar o desenvolvimento do código relevante, assim como fornecer uma explicação geral do funcionamento de aplicações Android.

O objetivo do protótipo desenvolvido é validar a solução proposta, para isso, foram adicionadas somente as funcionalidades básicas da ferramenta. São elas:

- Realizar autenticação explicita e implícita;
- Armazenar perfis;
- Configurar as senhas;
- Selecionar aplicações para cada perfil;

6.1 Ambiente de Desenvolvimento

Para a implementação do protótipo poderiam ser utilizadas varias linguagens de programação, entre elas C/C++, .NET e Java, contudo foi utilizada a linguagem Java, visto que esta é caracterizada como a linguagem padrão de programação para Android.

Existem diversos ambientes de desenvolvimento integrado (IDE) disponíveis para desenvolvimento Android, entre os mais utilizados estão Android Studio e Eclipse[19]. Neste trabalho optou-se pela utilização do Android Studio, por ser o IDE oficial para o desenvolvimento de aplicativos Android. Foi utilizado o emulador Genymotion, o principal motivo para a escolha deste emulador é a rapidez, além disso, apresenta uma forma e amigável para simulação de hardware como GPS. O ambiente de programação foi configurado sob a plataforma Windows 7.

O primeiro passo para configurar o ambiente de desenvolvimento foi instalar o Java JDK 7[20] (*Java Development Kit*) que possui todo o ambiente necessário para desenvolver e executar aplicativos em Java, após a instalação foi configurada a variáveis de ambiente. Em seguida foi realizada a instalação do Android Studio.

[19] Eclipse é um IDE para desenvolvimento Java, porém suporta várias outras linguagens a partir de *plugins*. Disponível em: https://www.eclipse.org/

[20] Disponível em http://www.oracle.com/technetwork/pt/java/javase/downloads/jdk7-downloads-1880260.html

Para utilizar o emulador Genymotion é necessário ter instalado Oracle VirtualBox[21]. Como o ambiente de desenvolvimento foi configurado sob plataforma Windows, não foi necessário baixar o VirtulaBox separadamente uma vez que ele está incluído no Genymotion. Para fazer o download do emulado é necessário criar um conta gratuita. Após instalar o genymotion é possível criar vários dispositivos virtuais. Para adicionar um novo dispositivo virtual, é necessário fazer *login* com nome de usuário e a senha da conta Genymotion. A fim de integrar a ferramenta de desenvolvimento Android Studio com o emulador foi necessário instalar um *plugin* para o Android Studio.

Concluída a configuração do ambiente de desenvolvimento foi iniciado o desenvolvimento do protótipo.

6.2 Base de dados

Visto a necessidade de armazenamento de uma grande quantidade de informações, definido pelo modelo, o protótipo necessita de uma solução de banco de dados para atender tal funcionalidade. Foi utilizado SQLite visto que Android oferece suporte nativo ao banco de dados SQLite.

A forma de criar, atualizar e conectar-se a um banco de dados é utilizando uma classe chamada SQLiteOpenHelper. Assim, é possível criar uma classe que derive da classe SQLiteOpenHelper e que atenda as necessidades concretas da aplicação.

A classe SQLiteOpenHelper tem um construtor, geralmente não é necessário subscrever, e dois métodos abstratos onCreate() e onUpgrade(), que devem ser definidos para criar o banco de dados e atualizar a sua estrutura respectivamente. O método onCreate() é executado automaticamente quando é necessário a criação do banco, ou seja, quando ainda não existe. Neste método são definidas a criação das tabelas e a inserção dos dados iniciais. O método onUpgrade() é executado automaticamente quando é necessário atualizar a estrutura da base.

Para fazer uso do banco de dados é necessário instanciar um objeto da classe que deriva da SQLiteOpenHelper, isso pode ter vários efeitos:

[21] Disponível em https://www.virtualbox.org/

1. Se o banco de dados já existe e a versão coincide com a versão solicitada, é realizada a conexão com banco.
2. Se o banco de dados existe, mas a versão não coincide, é executado automaticamente o método onUpgrade() para converter o banco a mesma versão e logo é realizada a conexão.
3. Se o banco não existe, é executado o método onCreate() para criar o banco e logo é realizada a conexão.

Por padrão, todos os bancos de dados SQLite criados por aplicações Android utilizando este método são armazenadas na memória do dispositivo em um ficheiro com o mesmo nome do banco de dados situados na seguinte rota

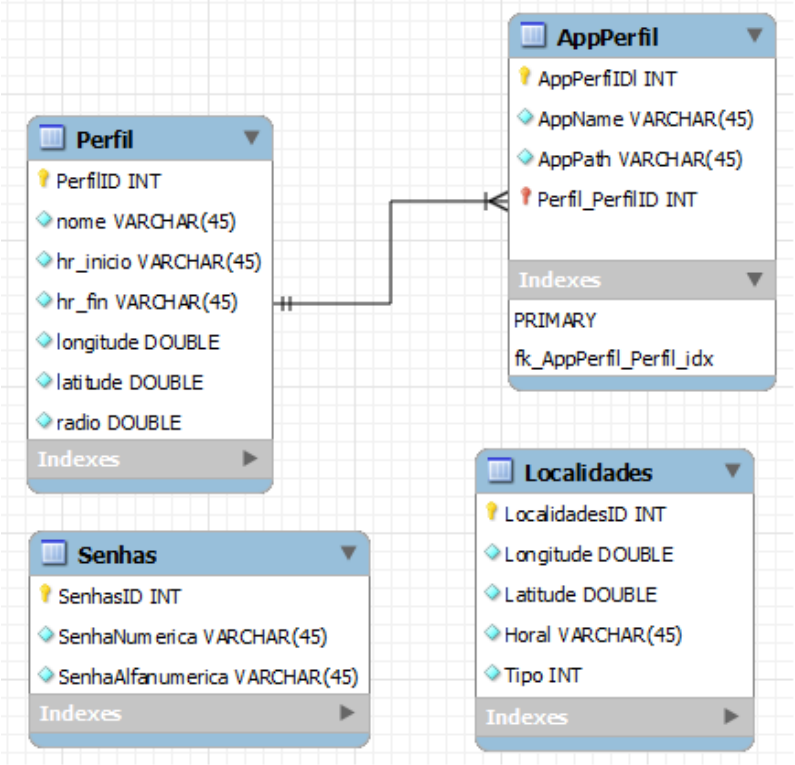
/data/data/pacote_java_da_aplicação/databases/nome_banco_dados

onde “pacote_java_da_aplicação é o nome da aplicação e ome_banco_dados” é o nome da base de dados atribuída na hora da criação.

O protótipo S.C.A.M necessita de um banco de dados para armazenar as informações sobre os perfis definidos pelos usuários, além de armazenar as informações necessárias para realizar o controle da aplicação. Para isso foram definidas quatro tabelas.

Com a definição das tabelas e seus atributos foi construído o diagrama entidade relacionamento desta aplicação. A Figura 19 ilustra o diagrama ER.

Figura 19 - Diagrama ER da aplicação



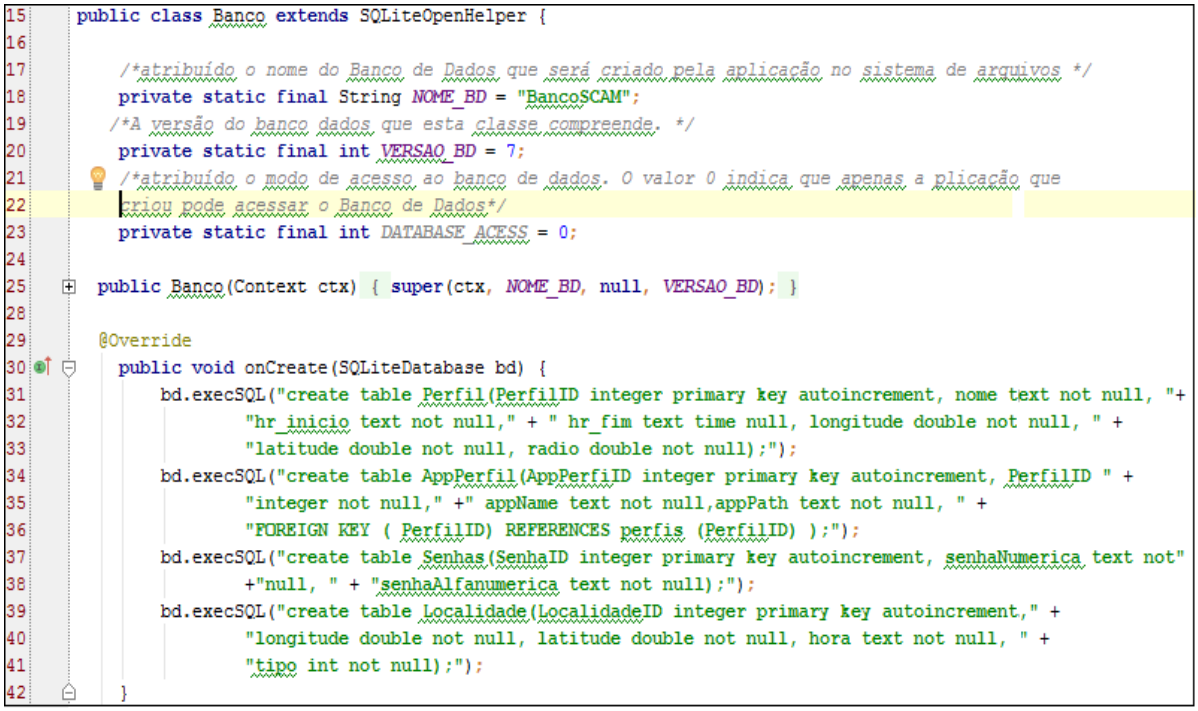
Fonte: Próprio Autor, 2014.

O diagrama representa a estrutura do banco de dados da aplicação. Contém quatro tabelas:

- Tabela Perfil: a tabela Perfis foi modelada para armazenar as informações relativas aos perfis definidos pelo usuário. Para definir os perfis são necessários os dados de localização representados pela coluna latitude e longitude e os horários para definir o período do dia. Além disso é armazenado o nome dado pelo usuário é o raio utilizado para controle da aplicação.
- Tabela AppPerfil: armazena os aplicativos disponíveis para cada perfil. Para isso armazena-se o nome do aplicativo e o *path* do pacote necessário para iniciar a aplicação. Contém um campo IDPerfil para relacionar com a tabela Perfil.
- Tabela Localidade: esta tabela armazena as informações necessárias para controle da aplicação. Esta tabela armazena informações de localidade, horários que ocorrem as autenticações e o tipo autenticação.
- Tabela Senha: responsável por armazenar as senhas registradas pelo usuário.

Para criar o banco de dados, inicialmente deve-se criar uma classe, a qual será o banco de dados do protótipo, e extender da classe SQLiteOpenHelper, conforme a Figura 20.

Figura 20 - Classe responsável pela criação do banco de dados da aplicação



```
15  public class Banco extends SQLiteOpenHelper {
16
17      /*atribuído o nome do Banco de Dados que será criado pela aplicação no sistema de arquivos */
18      private static final String NOME_BD = "BancoSCAM";
19      /*A versão do banco dados que esta classe compreende. */
20      private static final int VERSAO_BD = 7;
21      /*atribuído o modo de acesso ao banco de dados. O valor 0 indica que apenas a plicação que
22      criou pode acessar o Banco de Dados*/
23      private static final int DATABASE_ACESS = 0;
24
25      public Banco(Context ctx) { super(ctx, NOME_BD, null, VERSAO_BD); }
28
29      @Override
30      public void onCreate(SQLiteDatabase bd) {
31          bd.execSQL("create table Perfil(PerfilID integer primary key autoincrement, nome text not null, "+
32                  "hr_inicio text not null," + " hr_fim text time null, longitude double not null, " +
33                  "latitude double not null, radio double not null);");
34          bd.execSQL("create table AppPerfil(AppPerfilID integer primary key autoincrement, PerfilID " +
35                  "integer not null," +" appName text not null,appPath text not null, " +
36                  "FOREIGN KEY ( PerfilID) REFERENCES perfis (PerfilID) );");
37          bd.execSQL("create table Senhas(SenhaID integer primary key autoincrement, senhaNumerica text not"
38                  +"null, " + "senhaAlfanumerica text not null);");
39          bd.execSQL("create table Localidade(LocalidadeID integer primary key autoincrement," +
40                  "longitude double not null, latitude double not null, hora text not null, " +
41                  "tipo int not null);");
42      }
```

Fonte: Próprio Autor, 2014.

O método onCreate faz a criação das tabelas no banco. O comando execSQL é nativo da classe SQLiteDatabase, permitindo a inserção de tabelas no banco de dados. É no método OnCreate sobrescrito da super classe que ocorre a criação das tabelas.

O construtor da classe recebe quatro parâmetros, o primeiro é o contexto, o segundo é o nome do banco de dados, o terceiro é a versão do banco e o quarto é uma query armazenada em uma String que possui a criação das tabelas do banco. O *construtor* da classe cria o banco com o nome (NAME_BD) e a versão (VERSAO_BD) do banco de dados.

No S.C.A.M tem-se uma segunda classe chamada DB que é a responsável por toda a manipulação do banco de dados. Desta forma é necessário instanciar a classe DB no código da aplicação e utilizar os métodos que a classe oferece. Todos os métodos para inserir, atualizar, deletar ou consultar uma tabela encontram-se na classe DB.

Deve-se também sobrescrever o método onUpgrade(). Este método é chamado automaticamente quando detectado um incremento no número de versão do banco de dados. Nele pode-se efetuar qualquer operação, como dropar ou alterar alguma tabela. O método é executado dentro de uma transação, então se uma exceção for lançada, um rollback é executado.

6.3 Localização

Os dados de geolocalização do usuário possuem uma importância significativa para a abordagem proposta, visto que a localização é um fator para definir o contexto. Existem diversos métodos para a consulta da localização, a precisão de tal localização depende do método utilizado.

Em Android, as fontes fornecedoras de dados de localização são os *Location providers*, ou provedores de localização. A plataforma Android oferece dois grandes *providers*, *GPS Provider* e *o Network Provider.*

O GPS é uma ferramenta de posicionamento disponível na maioria dos smartphones, que utiliza sinais de dados a partir de satélites para calcular a posição do dispositivo. As informações de localização fornecida a partir do GPS incluem latitude, longitude, altitude, e tempo. Porém, o consumo energético pode ser elevado e também, a precisão e tempo de resposta dependem das condições de tempo do local. *Android Network Location Provider* determina a localização do usuário por meio de torre celular e sinais *Wi-Fi*, proporcionando informações sobre a localização tanto em ambientes fechados como ao ar livre, responde mais rápido e consome menos energia.

No presente trabalho, contamos com técnicas de posicionamento baseadas em *Network Provider* para obter a localização do dispositivo. Em adição a esta técnica, também será coletados dados de localização obtidos a partir de GPS para situações onde não há cobertura Wi-Fi.

O Android disponibiliza um pacote Android.location que fornece uma API que permite obter a localização do dispositivo móvel. Neste pacote encontra-se uma classe designada por LocationManager, que autoriza a seleção dos *providers* de localização que se pretender usar. Nesta aplicação serão usados os seguintes: GPS_PROVIDER (corresponde à localização por GPS) e NETWORK_PROVIDER (corresponde localização por *Network Location Provider*). Além disso, foi necessário

o uso de um LocationListener que recebe periodicamente atualizações sobre a posição atual. É possível definir o período temporal e posicional em que se pretende atualizar a geolocalização do utilizador. Neste caso, será considerado um tempo mínimo para obtenção de uma nova localização e distância mínima entre atualizações. O dispositivo só irá obter nova localização caso tenha passado o tempo mínimo ou o utilizador se tenha deslocado mais da distancia mínima definida.

Para poder acessar aos serviços de localização tanto GPS_PROVIDER como NETWORK_PROVIDER são utilizadas as classes presentes no pacote android.location. As classes básicas utilizadas no protótipo são LocationManager para controlar o dispositivo, LocationListener para ouvir as mudanças de localização e Location para guardas as informações de localização.

Para a localização do utilizador, no momento em que ocorre a autenticação é investigado se o Network Location Provider está ativo. Em caso afirmativo, utiliza-se esse *provider*, caso contrário, é investigado se este possui o GPS ativo. No caso afirmativo, é usado este sistema de geolocalização. Se o *Network Location Provider* e GPS se encontrarem ambos inativos é usada a ultima localização armazenada.

Para que a funcionalidades inerentes à localização sejam possíveis é necessário incluir as permissão adequadas no ficheiro "AndroidManifest.xml", Figura 21.

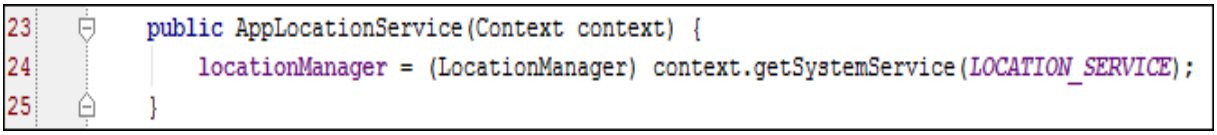
Figura 21 - Permissões de localização presentes no ficheiro AndroidManifest.xml.

```
6       <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
7       <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
8       <uses-permission android:name="android.permission.INTERNET" />
```

Fonte: Próprio Autor, 2014.

O primeiro passo para obter a localização geográfica é obter os mecanismos de localização disponíveis no dispositivo. Para obter essa informação utilizou-se o método getAllProviders() da classe LocationManager. Para isso obtemos uma referencia a *location manager* chamando o método getSystemService (LOCATION_SERVICE), como mostra a Figura 22.

Figura 22 - Referencia a LocationManager



```
23      public AppLocationService(Context context) {
24          locationManager = (LocationManager) context.getSystemService(LOCATION_SERVICE);
25      }
```

Fonte: Próprio Autor, 2014.

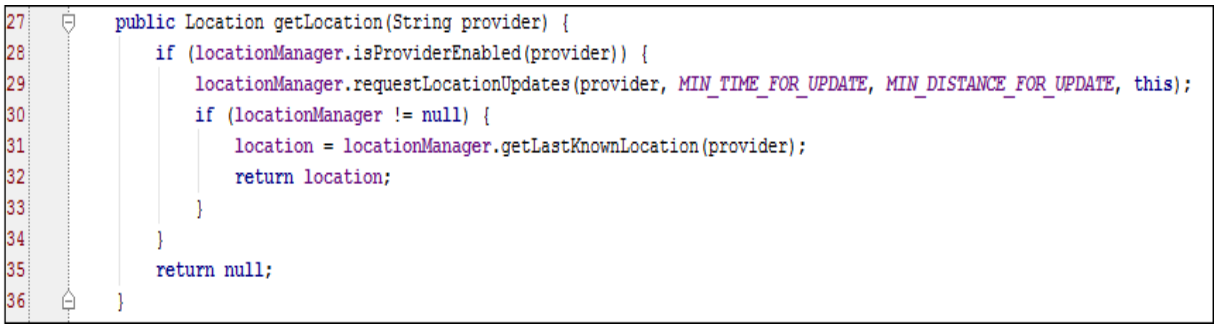
Android proporciona um mecanismo para obter os provedores que cumpram determinados requisitos, para isso é definido um critério de busca através de uma objeto Criteria, ao qual pode-se indicar as características mínimos do provedor que e quer utilizar. No protótipo S.C.A.M os provedores que foram utilizados já estão definidos, assim não é necessário utilizar o objeto Criteria.

Os provedores que são utilizados são:

- LocationManager.NETWORK_PROVIDER. Localização por rede;
- LocationManager.GPS_PROVIDER. Localização por GPS;

No método exposto a seguir, é testado se o provedor está habilitado. Se positivo, é obtida a nova localização do usuário, caso contrário é retornado null, conforme apresentado na Figura 23.

Figura 23 - Método getLocation()



```
27      public Location getLocation(String provider) {
28          if (locationManager.isProviderEnabled(provider)) {
29              locationManager.requestLocationUpdates(provider, MIN_TIME_FOR_UPDATE, MIN_DISTANCE_FOR_UPDATE, this);
30              if (locationManager != null) {
31                  location = locationManager.getLastKnownLocation(provider);
32                  return location;
33              }
34          }
35          return null;
36      }
```

Fonte: Próprio Autor, 2014.

Na Linha 28 da Figura 23 é demonstrado o teste que verifica se o provedor esta habilitado, a string *provider* recebida como parâmetro indica qual o *provedor* deve ser testado. Os valores possíveis para *provider* são LocationManager.GPS_PROVIDER ou LocationManager.NETWORK_PROVIDER. Para obter a nova localização é necessário registar-se ao evento que é lançado cada vez que o provedor recebe novos dados, também é necessário passar alguma indicação sobre a cada quanto tempo ou quanta distância percorrida necessita-se

obter a atualização da posição. Utiliza-se o método requestLocationUpdates (Linha 29 da Figura 23) passando quatro parâmetros:

- Nome do provedor de localização (indicado por *provider);*
- Tempo mínimo entre atualizações, em milissegundos;
- Distancia mínima entre atualizações;
- Instancia do objeto LocationListener;

Na Figura 24 é apresentado o trecho de código onde utilizou-se o método descrito acima para obter a longitude e latitude.

Figura 24 - Obtenção de latitude e longitude

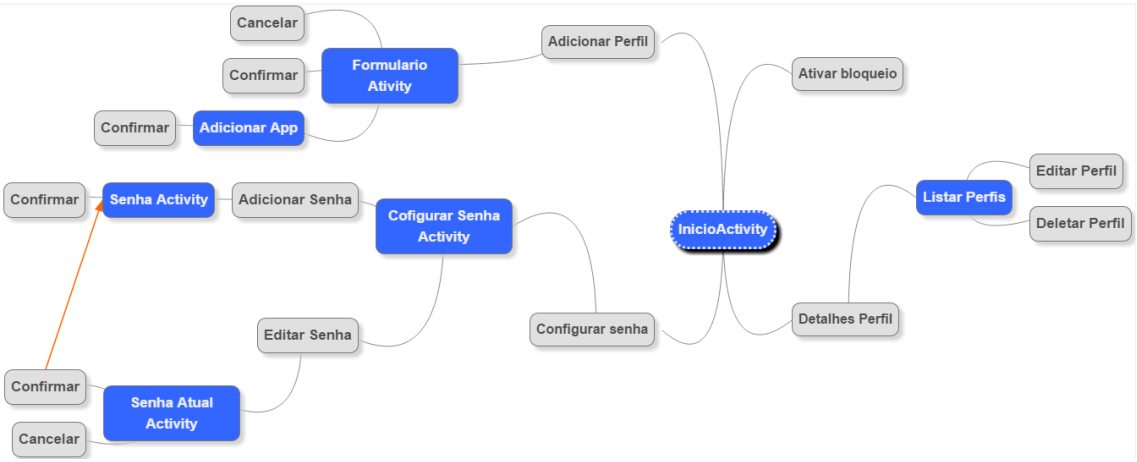
```
219             Location nwLocation = appLocationService.getLocation(LocationManager.NETWORK_PROVIDER);
220             if (nwLocation != null) {
221                 latitude = nwLocation.getLatitude();
222                 longitude = nwLocation.getLongitude();
223             }
224             else {
225                 Location gpsLocation = appLocationService.getLocation(LocationManager.GPS_PROVIDER);
226                 if (gpsLocation != null) {
227                     latitude = gpsLocation.getLatitude();
228                     longitude = gpsLocation.getLongitude();
229                 }
230             }
```

Fonte: Próprio Autor, 2014.

6.4 Funcionalidades

A seguir serão descritas as funcionalidades básicas do protótipo. O mind map apresentado na Figura 25 ilustra as funcionalidades do protótipo que são apresentadas quando o aplicativo é iniciado.

Figura 25 - Mind Map do protótipo



Fonte: Próprio Autor, 2014.

Na Figura 25 os retângulos azuis representam as *activities* que possuem interface como o usuário, e os retângulos cinzas representam as funções presentes em cada interface.

A *activity* denominada InicioActivity é a interface principal do protótipo, através dela temos acesso as quatros funcionalidades principais mencionadas no inicio da seção 6, que serão descritas em detalhes nas próximas linhas. A seguir é apresentada a tela inicial do protótipo.

Figura 26 - Tela inicial do protótipo

Fonte: Próprio Autor, 2014.

6.4.1 Configurar senha

Visto que se trata de um sistema de segurança o uso de senha é uma característica necessária. Conforme descrito no capítulo anterior é necessário configurar duas senhas no sistema. Uma senha numérica, a qual estará ativa sempre que a localização atual do usuário esteja definida em algum perfil armazenado, e uma senha alfanumérica caso contrário.

Quando o usuário acessar a funcionalidade configurar senha a seguinte tela será apresentada:

Figura 27 - Tela configurar senha

Fonte: Próprio Autor, 2014

Terá duas opções:

- Editar senha: caso uma senha já esteja configurada, será necessário ingressar a senha já cadastrada, em seguida será apresentada a tela para ingressar as novas senhas.

Figura 28 - Telas de editar senha e inserir nova senha

Fonte: Próprio Autor, 2014

Para editar a senhas é necessário realizar a atualização da tabela utilizada para armazenar as senhas. A função responsável por atualizar as senhas no banco de dados é mostrada na Figura 29.

Figura 29 - Método que atualiza as senhas no banco

```
106     public void EditarSenha(Senhas senhaNova){
107
108         ContentValues valores = new ContentValues();
109         valores.put("senhaNumerica", senhaNova.getsenhaNumerica());
110         valores.put("senhaAlfanumerica", senhaNova.getsenhaAlfanumerica());
111
112         String[] args = new String[] {String.valueOf(1)};
113         bd.update("Senhas", valores, "SenhaID=?", args);
114     }
```

Fonte: Próprio Autor, 2014

- Configura senha: permite cadastrar a senha que será utilizada para autenticar o usuário. É necessário cadastrar duas senhas, uma senha numérica, utilizada em locais conhecidos e uma senha alfanumérica utilizada em locais desconhecidos. A tela para configurar a senha é apresentada na Figura 28. Caso os campos não coincidam será solicitado novamente ao usuário para ingressar as senhas.

Para configurar uma nova senha é necessário inserir os dados nas tabelas Senhas do banco de dados da aplicação. A Figura 30 expõe o método responsável por essa função.

Figura 30 - Método que insere a senhas no banco

```
116     public void inserirSenha(Senhas senha){
117         ContentValues valores = new ContentValues();
118         valores.put("senhaNumerica", senha.getsenhaNumerica());
119         valores.put("senhaAlfanumerica", senha.getsenhaAlfanumerica());
120
121         bd.insert("Senhas", null, valores);
122     }
```

Fonte: Próprio Autor, 2014

Para evitar o vazamento de informações sensíveis, optou-se pela encriptação destes dados. A adoção de um padrão de criptografia não se demonstrou uma solução viável, visto que esta técnica gerou um retardo significativo na troca de pacotes (comunicação) da aplicação. Sendo assim optou-se por criptografar apenas as senhas. No presente trabalho optou-se por utilizar o algoritmo MD5 (*Message-Digest algorithm* 5) por ser um algoritmo projetado para ser rápido, simples e que atende aos requisitos de segurança necessários para a aplicação.

O MD5 é um algoritmo de *hash* de 128 bits unidirecional, desta forma um *hash* MD5 não pode ser transformado novamente na senha (ou texto) que lhe deu origem. O método de verificação é, então, feito pela comparação das duas *hash* (uma da base de dados, e a outra da tentativa de *login*).

O pacote de segurança java.security fornece determinadas classes úteis para gerar valores de hash. Especialmente a classe java.security.MessageDigest fornece as aplicações a funcionalidade de um algoritmo Message Digest, como MD5 ou SHA.

A Figura 31 apresenta o trecho de código utilizado para criptografar a senha inserida pelo usuário.

Figura 31 - Classe responsável por criptografar dados

```
1  package scam.unipampa.edu.br.scam;
2
3  import java.math.BigInteger;
4  import java.security.MessageDigest;
5  import java.security.NoSuchAlgorithmException;
6
7  public class md5 {
8
9      public String md5(String input) {
10         String md5 = null;
11         if(null == input)
12             return null;
13         try{
14             MessageDigest digest = MessageDigest.getInstance("MD5");
15             digest.update(input.getBytes(), 0, input.length());
16             md5 = new BigInteger(1, digest.digest()).toString(16);
17         }
18         catch (NoSuchAlgorithmException e) {
19             e.printStackTrace();
20         }
21         return md5;
22     }
23 }
```

Fonte: Próprio Autor, 2014

Para obter uma instância de um algoritmo de criptografia *Message Digest*, utiliza-se o método getInstance() da classe MessageDigest (Linha 14 da Figura 31). Utilizamos java.math.BigInteger para converter a mensagem digest em valores hexadecimais de base 16 (Linha 16 da Figura 31).

6.4.2 Inserir Perfil

Esta funcionalidade é responsável por criar os perfis do usuário. É apresentado ao usuário um formulário solicitando umas informações, como ilustrado na Figura 32.

Figura 32 - Formulário para inserir perfil

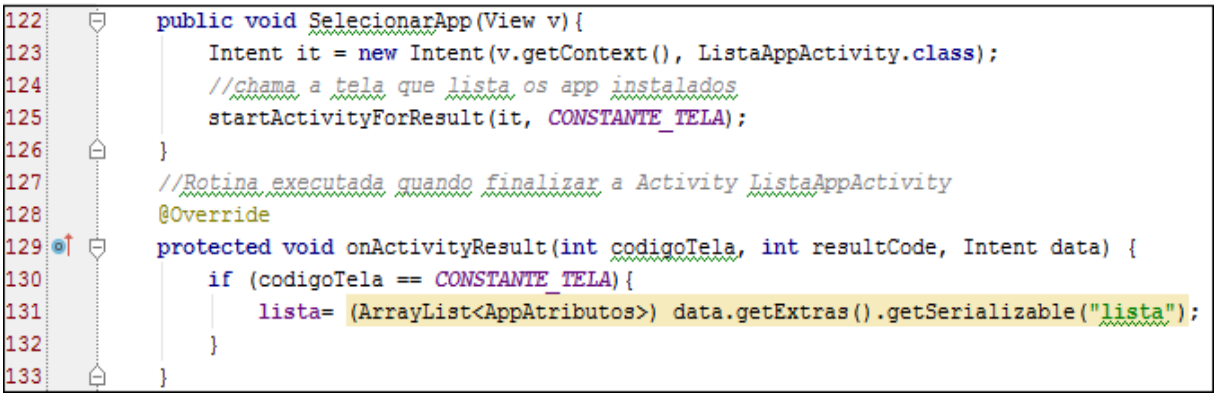
Fonte: Próprio Autor, 2014

Neste ponto é conveniente retomar a concepção apresentada anteriormente (capítulo 4), onde, o contexto é definido como:

Ci = <tempo, localização>

Tempo refere-se a um período, para obter essa informação são solicitados aos usuários os horários de inicio e fim do período. Para cada localização armazenada, o usuário será capaz de selecionar as aplicações que terá acesso quando ocorrer à autenticação implícita. Para isso é apresentada ao usuário uma lista com todas as aplicações do sistema as aplicações instaladas no dispositivo. Quando o usuário clicar no botão selecionar aplicativo o método chamado é mostrado na Figura 33.

Figura 33 - Método selecionarApp



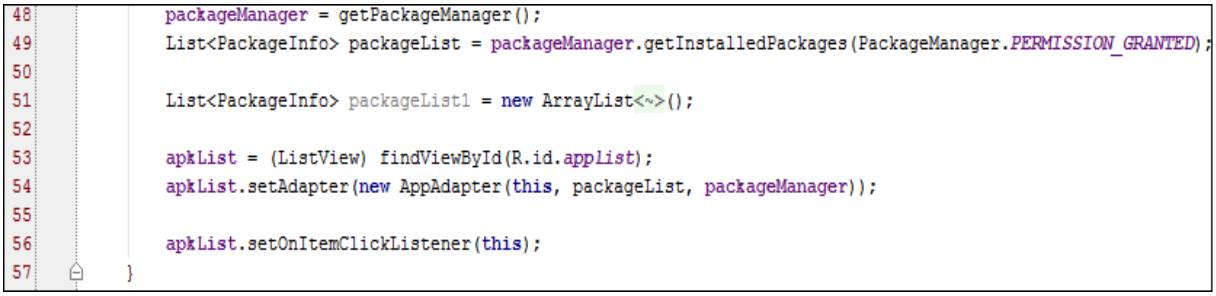
```
122     public void SelecionarApp(View v){
123         Intent it = new Intent(v.getContext(), ListaAppActivity.class);
124         //chama a tela que lista os app instalados
125         startActivityForResult(it, CONSTANTE_TELA);
126     }
127     //Rotina executada quando finalizar a Activity ListaAppActivity
128     @Override
129     protected void onActivityResult(int codigoTela, int resultCode, Intent data) {
130         if (codigoTela == CONSTANTE_TELA){
131             lista= (ArrayList<AppAtributos>) data.getExtras().getSerializable("lista");
132         }
133     }
```

Fonte: Próprio Autor, 2014

Foi utilizado o método startActivityForResult() e onActivityResult() para respectivamente enviar e receber dados de telas chamadas pela a*ctivity* atual.

Na Figura 34 á apresentado o trecho de código responsável por obter essa lista de aplicações.

Figura 34 - Código para obter a lista de aplicativos instalados



```
48          packageManager = getPackageManager();
49          List<PackageInfo> packageList = packageManager.getInstalledPackages(PackageManager.PERMISSION_GRANTED);
50
51          List<PackageInfo> packageList1 = new ArrayList<~>();
52
53          apkList = (ListView) findViewById(R.id.applist);
54          apkList.setAdapter(new AppAdapter(this, packageList, packageManager));
55
56          apkList.setOnItemClickListener(this);
57      }
```

Fonte: Próprio Autor, 2014

A classe packagemanager é utilizada para obter vários tipos de informações relacionadas com os pacotes de aplicativos que estão instalados no dispositivo. Através de getPackageManager se obtém uma instância dessa classe e com o método getInstalledPackeges se obtêm uma lista de todos os pacotes que estão instalados no dispositivo. O método setOnItemClickListener é responsável por monitorar quando um item da lista é clicado. Quando um item da lista é selecionado o método executado é mostrado na Figura 35.

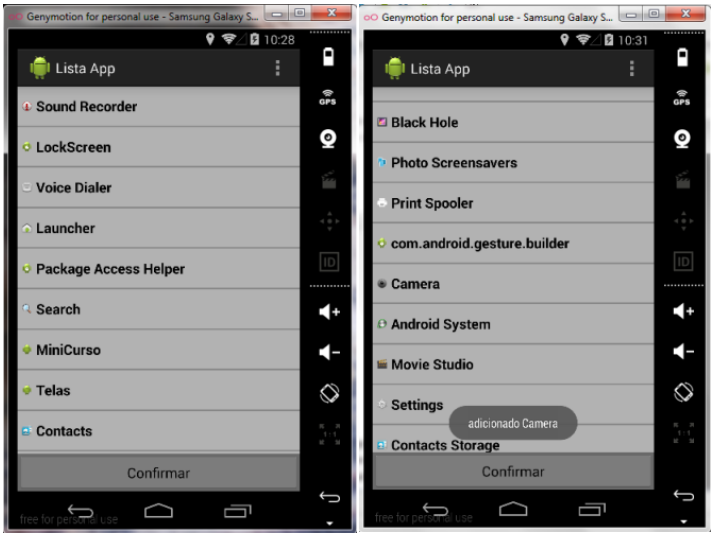
Figura 35 - Método onItemClick

```
66    public void onItemClick(AdapterView<?> parent, View view, int position,long row) {
67        PackageInfo packageInfo = (PackageInfo) parent.getItemAtPosition(position);
68        AppData appData         = (AppData) getApplicationContext();
69        appData.setPackageInfo(packageInfo);
70        nome = (String)getPackageManager().getApplicationLabel(packageInfo.applicationInfo);
71        path = packageInfo.applicationInfo.sourceDir;
72        AppAtributos apps = new AppAtributos(nome, path,0);
73        list.add(apps);
74    }
```

Fonte: Próprio Autor, 2014

As informações sobre o aplicativo são adicionadas a uma lista que logo será passada para a *activity* responsável por armazenar o perfil em questão.

Figura 36 - Lista de aplicativos instalados



Fonte: Próprio Autor, 2014

Quando o usuário clicar em uma aplicação da lista, será informado que a aplicação foi inserida no perfil, caso o usuário não selecione nenhuma aplicação, por padrão serão inseridos todas as aplicações presentes no dispositivo. O botão “confirmar” volta para a tela do formulário.

Para finalizar o cadastro do perfil, o usuário deve clicar em confirmar. O usuário também tem a opção de cancelar a inserção do perfil.

6.4.3 Detalhes de perfil

Esta funcionalidade fornece ao usuário uma lista de todos os perfis armazenados. É possível editar ou excluir os perfis cadastrados. Para visualizar os

perfis armazenados é necessário realizar uma consulta ao banco de dados da aplicação, o trecho de código apresentado na Figura 36 é responsável por exibir os perfis na tela.

Figura 37 - Método que exibe os perfis na tela

```
22  public class PerfilActivity extends Activity {
23
24      @Override
25      protected void onCreate(Bundle savedInstanceState) {
26          super.onCreate(savedInstanceState);
27          setContentView(R.layout.activity_perfil);
28
29          ListView listView = (ListView) findViewById(R.id.ListaPerfil);
30          listView.setChoiceMode(listView.CHOICE_MODE_MULTIPLE);
31
32          BD bd = new BD(this);
33          List<Perfil> lista = bd.buscar();
34
35          String[] values = new String[lista.size()];
36          for(int j=0;j<lista.size();j++){
37              values[j]=(String)lista.get(j).toString();
38          }
39          ArrayAdapter<String> adapter = new ArrayAdapter<~>(this,
40                  android.R.layout.simple_list_item_1, android.R.id.text1, values);
41          listView.setAdapter(adapter);
42      }
```

Fonte: Próprio Autor, 2014

Na linha 33, da Figura 37 é chamado o método busca da classe DB, esse método é o responsável por fazer a consulta ao banco de dados e retorna uma lista com todos os perfis armazenados, como mostra a Figura 38.

Figura 38 - Método busca

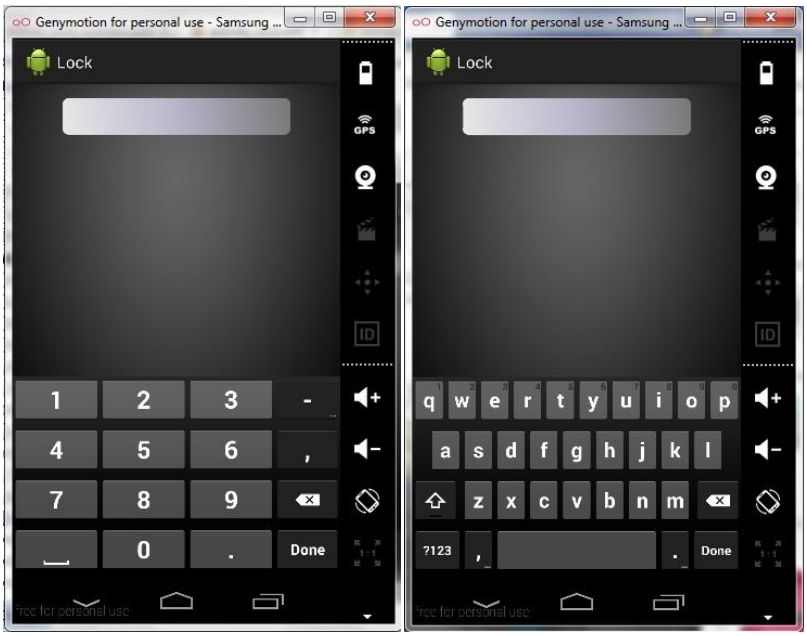
```
211     public List <Perfil> buscar(){
212         List <Perfil> list = new ArrayList <~>();
213         String[] colunas   = new String[]{"PerfilID", "nome", "hr_inicio", "hr_fim","longitude","latitude", "radio"};
214         Cursor cursor = bd.query("Perfil", colunas, null, null, null, null, "nome ASC");
215         if(cursor.getCount() > 0){
216             cursor.moveToFirst();
217
218             do{
219                 Perfil u = new Perfil();
220                 u.setId(cursor.getInt(0));
221                 u.setNome(cursor.getString(1));
222                 u.sethrInicio(cursor.getString(2));
223                 u.sethrFim(cursor.getString(3));
224                 u.setlongitude(cursor.getDouble(4));
225                 u.setLatitude(cursor.getDouble(5));
226                 u.setRadio(cursor.getDouble(6));
227                 list.add(u);
228
229             }while(cursor.moveToNext());
230         }
231         return(list);
232     }
```

Fonte: Próprio Autor, 2014

6.4.4 Ativar bloqueio

Após inserir as informações necessárias, é possível ativar o bloqueio da tela. A tela de bloqueio apresentada ao usuário é mostrada a seguir:

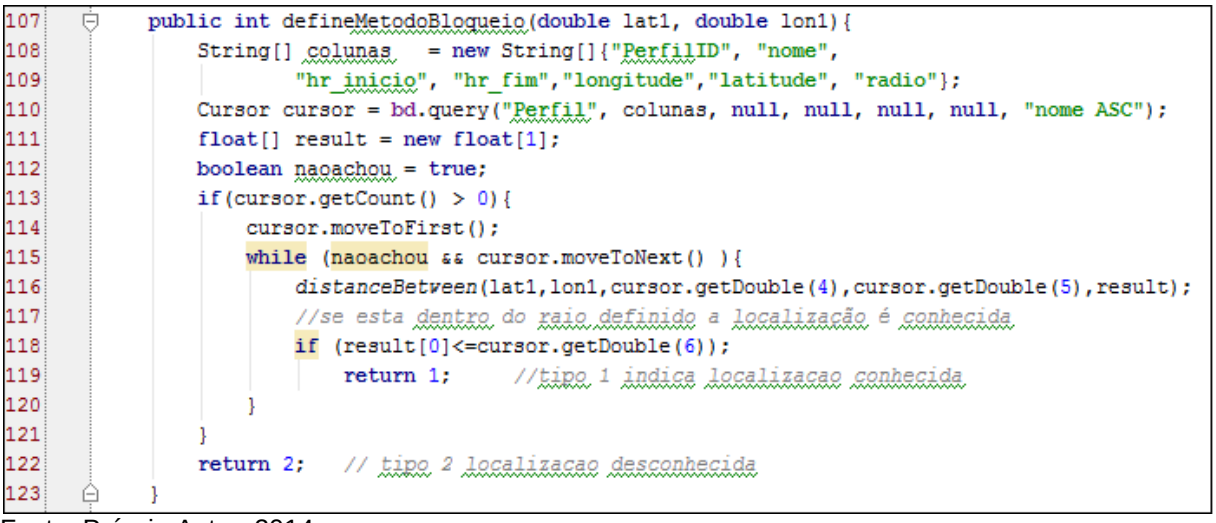
Figura 39 - Telas de bloqueio



Fonte: Próprio Autor, 2014

Como descrito no capítulo 5 é possível realizar a autenticação explicita e implícita. A autenticação explicita pode ser realizada através de senha numérica e alfanumérica. Para definir qual senha será utilizada são realizadas consulta a tabela Perfis, essa tabela contém as informações sobre as localidades conhecidas. Assim quando o dispositivo é ativado é obtida a localização do dispositivo e comparado com as localidades armazenadas na tabela, se a localidade atual estiver presente na tabela Perfis é solicitada a senha numérica, caso contrário, é solicitada a senha alfanumérica. O método que define a tipo de senha solicitada é mostrado a na Figura 40.

Figura 40 - Método defineMetodoBloqueio



```
107     public int defineMetodoBloqueio(double lat1, double lon1){
108         String[] colunas   = new String[]{"PerfilID", "nome",
109                 "hr_inicio", "hr_fim","longitude","latitude", "radio"};
110         Cursor cursor = bd.query("Perfil", colunas, null, null, null, null, "nome ASC");
111         float[] result = new float[1];
112         boolean naoachou = true;
113         if(cursor.getCount() > 0){
114             cursor.moveToFirst();
115             while (naoachou && cursor.moveToNext() ){
116                 distanceBetween(lat1,lon1,cursor.getDouble(4),cursor.getDouble(5),result);
117                 //se esta dentro do raio definido a localização é conhecida
118                 if (result[0]<=cursor.getDouble(6));
119                     return 1;      //tipo 1 indica localizacao conhecida
120             }
121         }
122         return 2;    // tipo 2 localizacao desconhecida
123     }
```

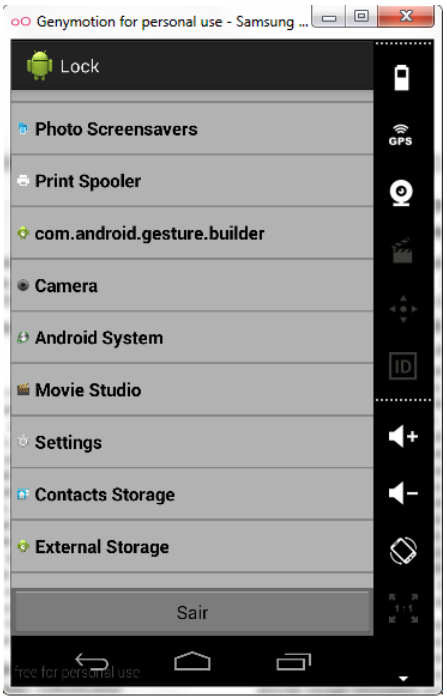
Fonte: Próprio Autor, 2014

Foi utilizado o método distanceBetween() (Linha 116, Figura 40) para obter a distancia entre dois ponto em metros. Se a distancia estiver dentro de um raio definido a localização é considerada conhecida e valor 1 é retornado.

Sempre que ocorrer uma autenticação, algumas informações são armazenadas na tabela Localidade, essas informações são: latitude, longitude, horário e tipo de autenticação (explicita ou implícita). O método que define qual tipo de autenticação será ativado considerações essas informações para a tomada de decisão.

Autenticação implícita fornece acesso direto aos aplicativos selecionados para um determinado perfil, assim quando esta for ativada, será apresentado ao usuário um atalho para esses aplicativos. Isso requer uma consulta ao banco para determinar quais aplicativos estarão disponíveis para esse perfil. A tela apresentada ao usuário é ilustrada na Figura 41.

Figura 41 - Tela autenticação implícita



Fonte: Próprio Autor, 2014

Na Figura 42 é apresentado o trecho do código necessário para iniciar a aplicação quando o usuário clicar em um elemento da lista.

Figura 42 - Código para iniciar uma aplicação

```
127     public void onItemClick(AdapterView<?> parent, View view, int position,
128                             long row) {
129         PackageInfo packageInfo = (PackageInfo) parent
130                 .getItemAtPosition(position);
131         AppData appData = (AppData) getApplicationContext();
132         appData.setPackageInfo(packageInfo);
133
134         String path = packageInfo.packageName;
135
136         Toast.makeText(this, "path " + path, Toast.LENGTH_LONG).show();
137         Intent launcher = getPackageManager().getLaunchIntentForPackage(path);
138
139         startActivity(launcher);
140
141
142     }
```

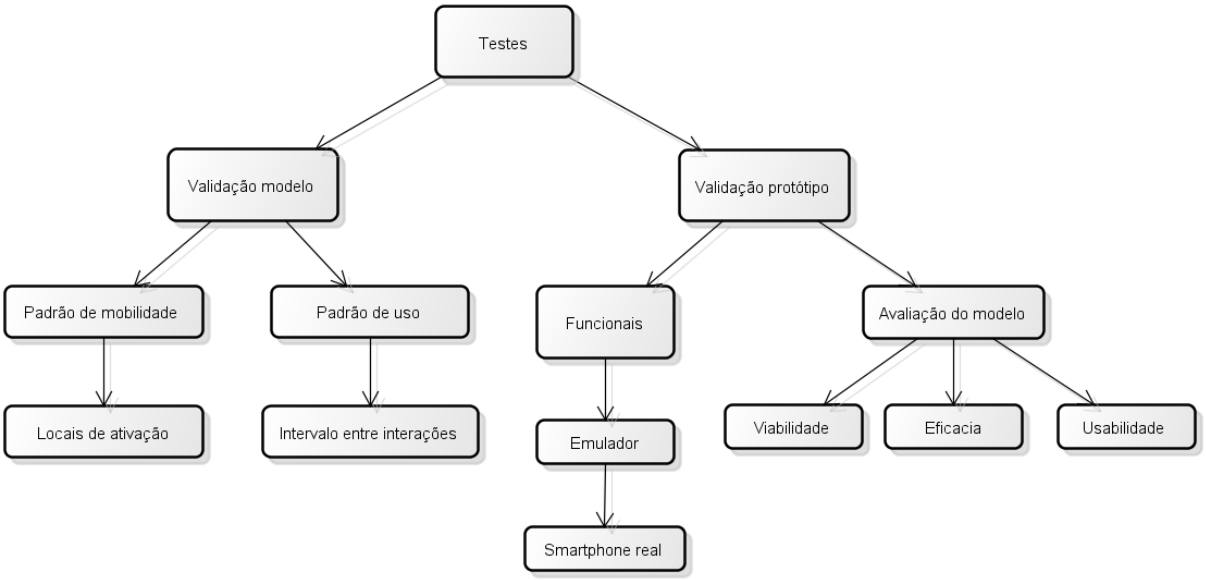
Fonte: Próprio Autor, 2014

Caso o usuário deseje acessar aplicações que não estão na lista deve clicar no botão sair e a tela de bloqueio (Figura 39) será ativada.

7. VALIDAÇÃO DA SOLUÇÃO PROPOSTA

Para validar a solução proposta foram realizados dois testes empíricos com diferentes finalidades. A Figura 43 ilustra os testes realizados assim como a finalidade de cada um.

Figura 43 - Esquema de Testes



Fonte: Próprio Autor, 2014

Os testes foram divididos em duas etapas, primeiramente foram realizados testes a fim de validar o modelo proposto, analisando dois fatores relevantes, o padrão de mobilidade e o padrão de uso dos *smartphones*. Em seguida forma realizados os testes a fim de validar o protótipo implementado, a seguir são descritos os estudos realizados e as finalidades de cada um.

Na primeira avaliação foi investigado o potencial da localização como um fator decisivo para determinar o tipo de autenticação que será solicitado, além de investigar o padrão de uso do *smartphone*. Para isso, o protótipo desenvolvido coleta dados de localização de onde ocorrem as ativações dos telefones. Assim pode-se determinar com que freqüência as pessoas utilizam seus telefones em esses locais.

No segundo estudo foi realizada uma pesquisa de campo com oito participantes durante um período de cinco a sete dias. O protótipo utilizado pelos participantes tinha as funcionalidades básicas de realizar a autenticação explícita e implícita descritas anteriormente. Este estudo teve a finalidade de analisar a

usabilidade do método através de *feedback* dos participante, além de investigar a viabilidade e eficácia. Também foi possível ver na prática o funcionamento da solução proposta.

7.1 Teste de padrão de uso dos *smartphones*

Neste estudo, foram investigados os padrões de mobilidade das pessoas, juntamente com os seus padrões de uso do telefone em relação à localização.

7.1.1 Coleta de dados

O protótipo armazenava a localização sempre que o usuário ativava o telefone, independente do método de autenticação utilizado. Também foi armazenada a hora que ocorriam as autenticações, com isso, foi possível determinar os eventos de ativação do telefone, assim como os intervalos entre as ativações.

Estes dados foram obtidos da tabela Localidades do banco de dados da aplicação. Para ter acesso a esses dados sem fazer root no dispositivo, foi criada uma função que envia os dados para um servidor externo php e são armazenados em um banco de dado mysql através de uma conexão HTTP.

Foi utilizado o servidor Xampp[22] por ser composto por um pacote que inclui os principais servidores de código aberto existentes, incluindo, banco de dados MySQL e Apache com suporte as linguagens PHP.

Utilizou-se o PHPMyAdmin para criar uma base de dados MySQL com tabelas para armazenar as informação enviadas pelo aplicativo. Para cada usuário criou-se uma tabela diferente, já que foi necessário analisar os dados individualmente. Para realizar a conexão com o banco foi implementado um *script* php, mostrado na Figura 43.

[22] Disponível em https://www.apachefriends.org/pt_br/index.html

Figura 44 - Script php para conexão com o banco

```php
<?php

    $conexao = mysql_connect('localhost', 'root', '');
    mysql_select_db('scam', $conexao);

    $lat      = $_POST["lat"];
    $long     = $_POST["long"];
    $hr       = $_POST["hr"];
    $tipo     = $_POST["tipo"];
    $tabela   = $_POST["tabela"];

    $sql = "INSERT INTO".$tabela."(Latitude, Longitude, Hora, Tipo)
            VALUES($lat, $long, $hr, $tipo)";

    $resultado = mysql_query($sql) or die ("Erro:".mysql_error());

    if(!(empty($resultado)))
        echo "Transferindo dados...";
    else
        echo "Ocorreu algum erro";
?>
```

Fonte: Próprio Autor, 2014

Para enviar os dados da aplicação foi necessário criar uma funcionalidade extra no protótipo, a Figura 44 mostra a tela apresentada ao usuário quando é selecionado a funcionalidade Enviar Dados.

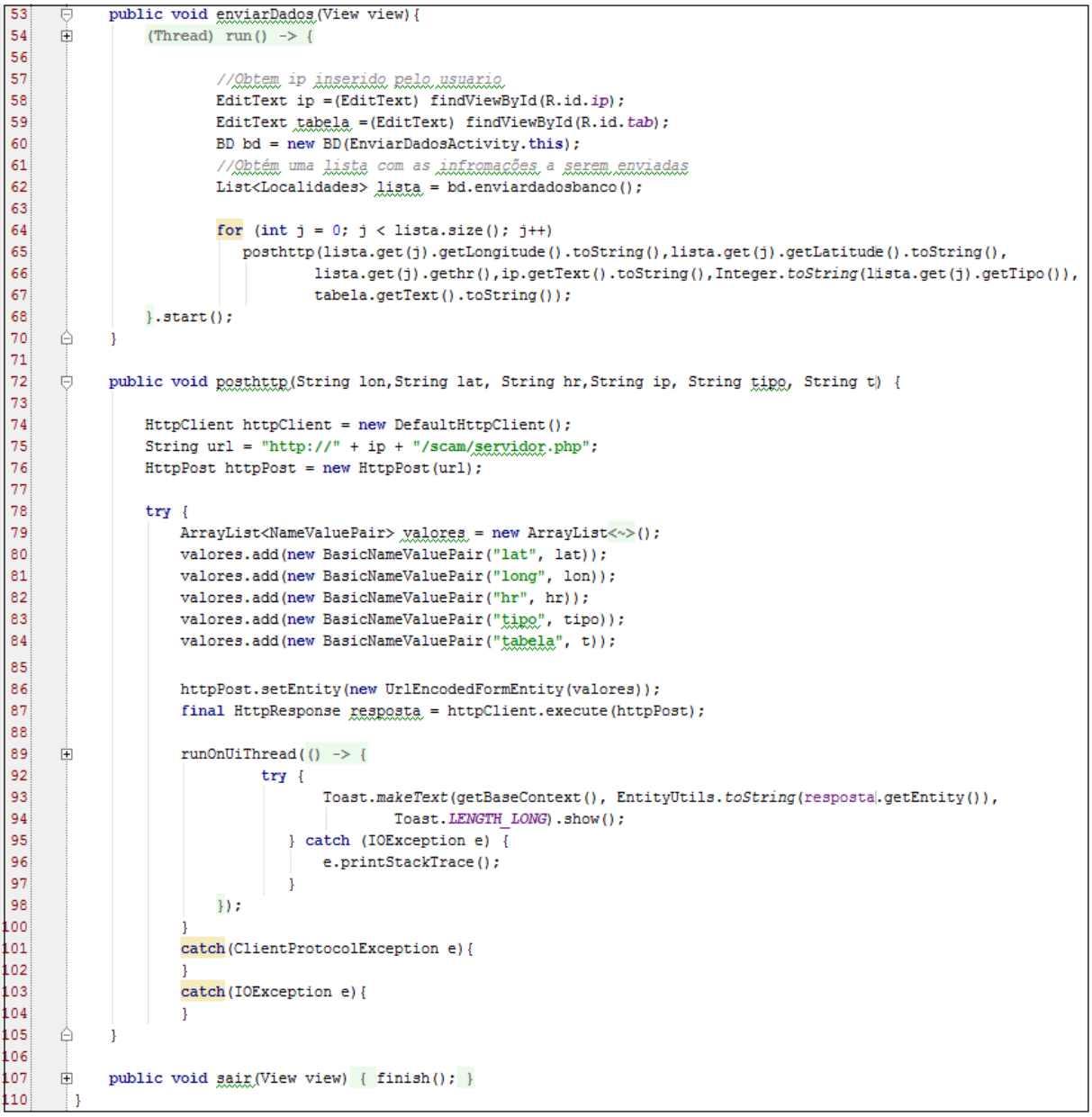
Figura 45 - Tela enviar dados

Fonte: Próprio Autor, 2014

Os dados solicitados são o IP (*Internet Protocol*) onde esta hospedado o servidor web e o nome da tabela responsável por armazenar as informações.

A função que realiza a conexão do protótipo com o servidor é mostrada na Figura 45.

Figura 46 - Função para conexão com o servidor web



```
53      public void enviarDados(View view){
54          (Thread) run() -> {
56
57                  //Obtem ip inserido pelo usuario
58                  EditText ip =(EditText) findViewById(R.id.ip);
59                  EditText tabela =(EditText) findViewById(R.id.tab);
60                  BD bd = new BD(EnviarDadosActivity.this);
61                  //Obtém uma lista com as infromações a serem enviadas
62                  List<Localidades> lista = bd.enviardadosbanco();
63
64                  for (int j = 0; j < lista.size(); j++)
65                     posthttp(lista.get(j).getLongitude().toString(),lista.get(j).getLatitude().toString(),
66                             lista.get(j).gethr(),ip.getText().toString(),Integer.toString(lista.get(j).getTipo()),
67                             tabela.getText().toString());
68          }.start();
70      }
71
72      public void posthttp(String lon,String lat, String hr,String ip, String tipo, String t) {
73
74          HttpClient httpClient = new DefaultHttpClient();
75          String url = "http://" + ip + "/scam/servidor.php";
76          HttpPost httpPost = new HttpPost(url);
77
78          try {
79              ArrayList<NameValuePair> valores = new ArrayList<~>();
80              valores.add(new BasicNameValuePair("lat", lat));
81              valores.add(new BasicNameValuePair("long", lon));
82              valores.add(new BasicNameValuePair("hr", hr));
83              valores.add(new BasicNameValuePair("tipo", tipo));
84              valores.add(new BasicNameValuePair("tabela", t));
85
86              httpPost.setEntity(new UrlEncodedFormEntity(valores));
87              final HttpResponse resposta = httpClient.execute(httpPost);
88
89              runOnUiThread(() -> {
92                      try {
93                             Toast.makeText(getBaseContext(), EntityUtils.toString(resposta.getEntity()),
94                                     Toast.LENGTH_LONG).show();
95                         } catch (IOException e) {
96                             e.printStackTrace();
97                         }
98                  });
100             }
101             catch(ClientProtocolException e){
102             }
103             catch(IOException e){
104             }
105     }
106
107     public void sair(View view) { finish(); }
110 }
```

Fonte: Próprio Autor, 2014

Após a migração dos dados para o banco do servidor web, foi inicializada a análise dos dados. O resultado dessa análise é apresentada na próxima seção. Os dados de localização não são expostos neste trabalho para manter a privacidade dos voluntários.

7.1.2 Análise e resultados

Foram analisados os dados de 5 participantes em um períodos variando de 5 a 7 dias.

Este teste tem objetivo medir três variáveis, são elas:

- Tempo entre interações: foram analisados os horários que ocorreram as ativações do dispositivo para determinar o tempo médio entre as iterações.
- Local que ocorreram as ativações: foram analisados os dados de localização para encontrar um padrão de utilização dependente da localização do usuário.
- Tipo de autenticação: este dado nos permitiu determinar a porcentagem de vezes que a autenticação implícita e explícita foram ativadas.

Foi identificado uma media de 279,6 eventos de ativação por usuário em nosso conjunto de dados. Primeiramente foram analisados os dados de localização para definir os dois locais mais registrados. Assim, foi calculado o número de ativações em cada um desses lugares. A Tabela 12 mostra a distribuição de eventos de ativação de acordo com os locais.

Tabela 12 - Evento de ativação em função da localização

Local	Eventos de ativação (%)
Local 1	33.4
Local 2	31.7
Outros (vários locais)	34.9

Fonte: Próprio Autor, 2014

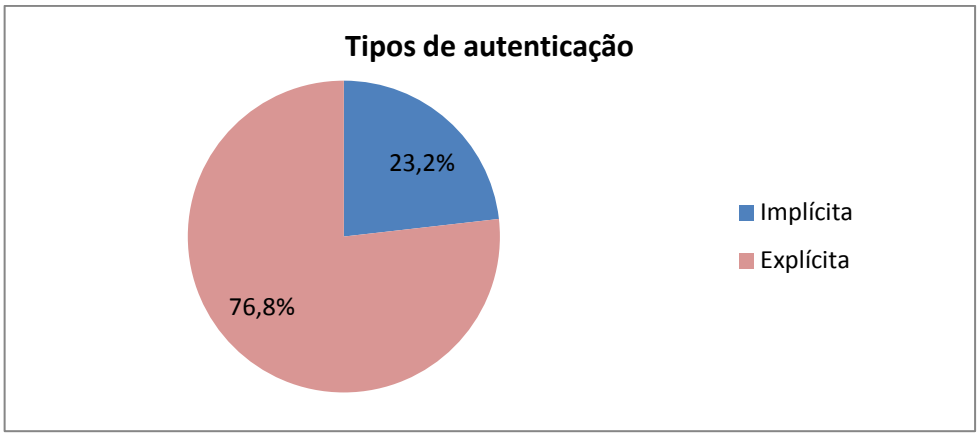
Os resultados mostraram que estes dois locais foram responsáveis por 65,1% do total de eventos de ativação do dispositivo. Este resultado indica que pessoas ativam seus telefones com mais frequência em dois locais.

Foram analisados os intervalos de tempo entre cada interação para cada participante, logo foi calculado a media entre todos os participantes. Como resultado obteve-se que em media os dispositivos são ativados a cada 10.43 minutos. No entanto, pode-se observar que em certos períodos do dia o intervalo de interação é

relativamente menor, assim, muitas vezes digitar senhas tornasse inconveniente. Com isso, a proposta de autenticação implícita pode melhorar a usabilidade dos métodos de autenticação em situações onde o intervalo de interações é curto.

Analisando os tipos de autenticação foi possível identificar que do total de eventos de ativações 23,2 % representam a autenticação implícita como mostra a Figura 47.

Figura 47 - Gráfico tipo de autenticação



Fonte: Próprio Autor, 2014

Com isso observamos que foi reduzido o numero de vezes que foi necessário digitar as senhas. Contudo, esse resultado varia de acordo com o tempo entre interações considerado na aplicação. Quanto maior for esse tempo, menor será o numero de vezes que o usuário necessitará digitar a senha, no entanto, a segurança também será menor

Com os resultados obtidos podemos concluir que a localização do usuário pode ser um fator relevante para determinar a segurança do dispositivo, já que a utilização destes dispositivos, em geral, ocorre com mais freqüência em lugares determinados. Também foi possível observar que a autenticação implícita pode ser uma boa solução para melhorar a usabilidade dos métodos de autenticação, reduzindo o numero de vezes que o usuário precisa digitar as senhas.

Na seção 7.2 é apresentado os testes realizado para avaliar o protótipo implementado.

7.2 Avaliação do protótipo S.C.A.M

Para validar a solução proposta foram realizados dois testes, uma para validação das funcionalidades da aplicação e outro, teste de usabilidade para avaliação da aplicação do ponto de vista do usuário.

O ambiente de teste desta aplicação passou por dois locais distintos. Primeiramente, os testes foram realizados no emulador, um dispositivo móvel virtual executado no computador que simula as configurações reais típicas de hardware e software de um smartphone com Android. Posteriormente, foi utilizado um dispositivo real Samsung S4 Mini. Esta etapa foi responsável por validar a ferramenta, permitindo a correção de erros de codificação. Concluída esta etapa, a ferramenta pode ser disponibilizada para a utilização.

Os testes realizados tiveram os objetivos de investigar a viabilidade e eficácia do protótipo além de estudar a usabilidade o que permitem compreender quais são as preferências do utilizador. Após realizar uma breve descrição do funcionamento do protótipo, este foi disponibilizado para uma amostra de usuários para que pudessem testar por um período de cinco dias. O teste consistiu na utilização da ferramenta individualmente. Logo, foi pedido aos utilizadores que realizassem um pequeno questionário de avaliação.

7.2.1 Obtenção dos dados

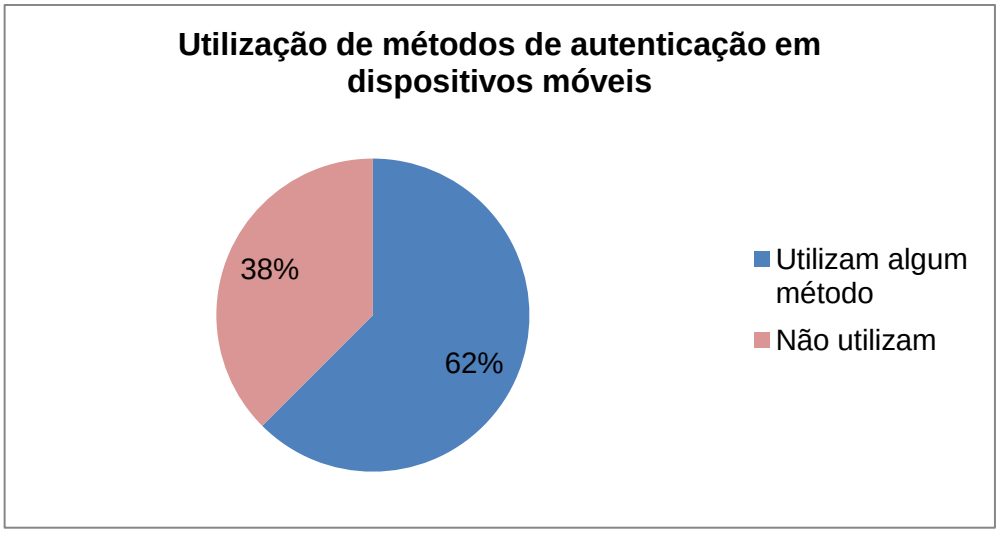
Utilizou-se um instrumento de avaliação na forma de questionário a fim de identificar a viabilidade e usabilidade da ferramenta. O objetivo da aplicação do questionário é identificar o nível de satisfação do usuário. A aplicação do questionário foi realizada de forma online utilizando a ferramenta para criação de formulários da Google. O questionário realizado encontra-se no Apêndice A.

7.2.2 Análise e resultados

Analisando os dados obtidos no questionário, pode-se observar, como ilustra a Figura 48, que 37,5 % dos usuários não utilizam nenhum mecanismo de autenticação para proteger o dispositivo. O principal motivo relatado foi que

consideram que inserir senhas continuamente se torna inconveniente. No entanto 100 % dos participantes disseram ter preocupação com as informações armazenadas em seu dispositivo.

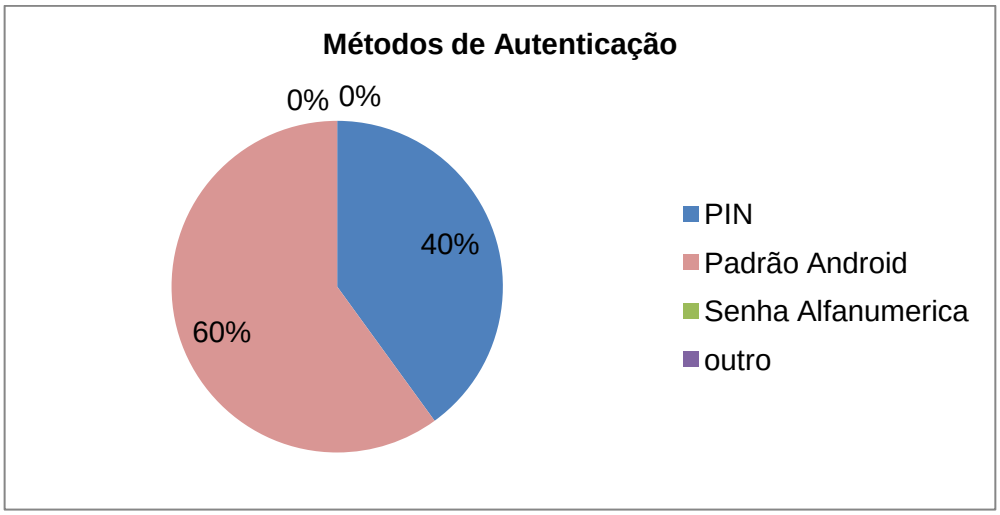
Figura 48 - Gráfico de utilização de métodos de autenticação



Fonte: Próprio Autor, 2014

Conforme mostra a Figura 49, dos participantes que utilizam métodos de autenticação, 60% utilizam o padrão Android e 40% utilizam mecanismos de PIN, o motivo relatado foi por ser mais fácil de memorizar.

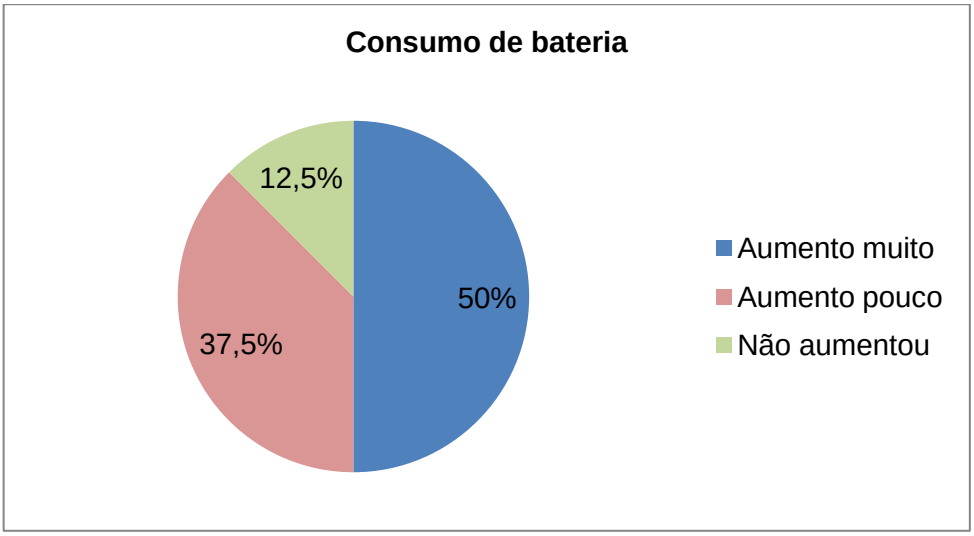
Figura 49 - Métodos de Autenticação



Fonte: Próprio Autor, 2014

Como ilustrado na Figura 50, 87,5% dos participantes relataram que ouve um aumento no consumo de bateria, esse resultado já era esperado devido aos recursos demandados pela aplicação.

Figura 50 - Consumo de bateria



Fonte: Próprio Autor, 2014

Questionados sobre a autenticação implícita, 75% dos participantes consideraram ser uma boa alternativa para melhorar a usabilidade, no entanto, se mostraram preocupados com a possibilidade de uma atacante ter acesso ao dispositivo durante esse período onde a autenticação implícita esta ativada.

A partir da análise das respostas observa-se que o aplicativo atingiu bons níveis de aceitação, assim, pode-se concluir que o modelo S.C.A.M conseguiu cumprir os seus objetivos específicos propostos sendo uma boa solução para ambientes móveis. No entanto, verificou-se que o aplicativo requer certos recursos do dispositivo, o que pode afetar o desempenho do mesmo. Contudo, acredita-se que alguns pontos ainda podem ser melhorados, como o consumo de bateria, para atingir um melhor resultado.

7.3 Análise de Segurança

Nessa seção foram analisados alguns pontos importantes com relação à segurança da solução S.C.A.M.

Para análisar os princípios de segurança da solução, foram criados 3 cenários possíveis. As analises foram realizadas com a premissa que os possíveis atacantes não possuem conhecimento técnico para quebrar as senhas do sistema.

Cenário 1: se o atacante possui acesso as senhas dos usuários, independente da localização ou do tipo de autenticação ativada, a solução apresenta resultado insatisfatório, já que o atacantes terá acesso a todas as informações e recursos do dispositivo.

Cenário 2 : Se o atacante não possui informações sobre as senhas do usuário legitimo e o ataque acontece após um período de tempo suficiente para que a autenticação implícita não seja ativada, a solução apresenta resultado satisfatório tanto em locais conhecidos como em locais desconhecidos.

Cenário 3: Igual ao cenário 2, porém a autenticação implícita pode estar ativada. Neste cenário foi possível identificar potenciais riscos de segurança. A solução não apresenta um mecanismo para detectar atacantes na autenticação implícita. Assim qualquer usuário poderá ter acesso as informações e recursos disponíveis na autenticação implícita.

Atacantes experientes, com conhecimentos técnicos capazes de quebrar o sistema, representam um caso particular, porém não é o foco deste trabalho. Não foram realizados análises de segurança com este tipo de ataques, no entanto, a solução deve apresentar proteção contra este tipo de ataques.

8. CONCLUSÃO

Neste trabalho foi abordado um problema específico encontrado em ambiente de computação móvel. A utilização de dispositivos móveis com ênfase para os *smartphones* é uma tendência crescente. A segurança da informação e privacidade dos usuários se tornam cada vez mais importantes.

Apesar de observarem-se muitos avanços nestes aparelhos, pouco tem mudado na forma como se verifica quem os utiliza. Além disso, acredita-se que não é questão de substituir a abordagem adotada, mais que isso, é fundamental compreender como são utilizados estes dispositivos para que a autenticação seja alcançada de forma mais eficiente.

A autenticação de usuários é a base para o controle de acesso, garantir a identidade dos utilizadores é extremamente importante, já que estes dispositivos operam em ambientes que são mais vulneráveis a perda e exposição dos dados. Com o estudo realizado, identificamos a relevância da usabilidade destes métodos que são muitas vezes inexistentes.

Através da pesquisa realizada, foi identificado que o contexto do utilizador final afeta as características das sessões de uso. Com os resultados dos testes realizados observou-se que as pessoas utilizam os dispositivos com mais frequência em dois lugares, esse resultado indica a importância de considerar a localização nos métodos de autenticação. Também pode-se concluir que ajustar o método de autenticação de acordo com a localização pode melhorar a usabilidade nesses locais mais frequentados.

Com os testes realizados, observou-se que os dispositivos são utilizados com muita frequência, assim conclui-se que mecanismos de segurança que realizem a autenticação de forma transparente ao usuário podem ser uma alternativa para melhorar a privacidade e segurança na utilização destes dispositivos.

A análise de segurança da ferramenta indica que a autenticação implícita pode fornecer riscos, no entanto, demasiada ênfase na segurança pode levar as pessoas a não utilizar nenhum método de proteção.

Finalmente com base nos resultados apresentados, pode-se concluir as metas traçadas para este trabalho foram atingidas. O principal objetivo do modelo proposto era conciliar a segurança e usabilidade, de acordo com os testes de avaliação da ferramenta, pode-se concluir que o modelo proposto atingiu bons níveis

de aceitação por parte dos usuários, conseguindo conciliar a segurança e usabilidade.

REFERÊNCIAS

ADAMS A.; SASSE A. M. **Users are not the enemy.** *Communication of de ACM.* Vol.42, p. 40-46. New York, December 1999.

ASHER, Ben *et. al*. **On the need for different security methods on mobile phones**. In Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI .p.465-473. New York, New York, USA, 2011.

AVIV, A. J. *et. al.* **Smudge attacks on smartphone touch screens**. In Proceedings of the 4th USENIX conference on Offensive technologies. p. 1–7. USA, 2010.

BABU, Sathish B.; VENKATARAM Pallapa. **A dynamic authentication scheme for mobile transactions.** International Jour nal of Network Security. vol. 8, no. 1, p. 59–74, 2009.

BEZERRA, Eduardo. Princípios de Análise e Projeto de Sistemas com UML. Rio de Janeiro: Campus, 2006.

BISHOP, Matt. **Introduction to Computer Security**. Addison-Wesley Professional, 2004.

BURNETTE, E. **Hello, Android: Introducing Google's Mobile Development Platform**. 2. ed. USA: The Pragmatic Bookshelf October, 2009.

CHIN, E *et. al*. **Measuring user confidence in smartphone security and privacy**. In: Proceedings of the Eighth Symposium on Usable Privacy and Security. 2012

CHONG, M.; MARSDEN, G. **Exploring the use of discrete gestures for authentication.** In: Human Computer Interaction - INTERACT 2009, p. 205 – 213. 2009.

CLARKE, N.; FURNELL, S. **Authentication of users on mobile telephones – A survey of attitudes and practices**. Computers & Security. 7.ed, vol.24, p.519–527. 2005.

CLARKE, N. L., *et. al*. **Acceptance of subscriber authentication methods for mobile telephony devices.** Computers & Security. 3. ed, vol.21, 220-228. 2002.

CLARKE, N. L.; FURNELL, S. M. **Authenticating mobile phone users using keystroke analysis**. International Journal of Information Security. 1.ed, vol.6, p.1–14. 2006.

CRANOR, Lorrie F.; GARFINKEL, Simson. **Security and Usability: Designing Secure Systems That People Can Use.** 1. ed. O'Reilly Media, Inc. 2005.

DE LUCA A. *et. al*. **Now You See Me, Now You Don't – Protecting Smartphone Authentication from Shoulder Surfers**. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. p 2937-2946. Toronto, Canada. 2014.

FALAKI, Hosseein *et. al*. **Diversity in smartphone usage**. In Proceedings of the 8th international conference on Mobile systems, applications, and services – MobiSys 10. p. 179-194. New York, New York, USA. 2010.

FIGUEIREDO, Carlos M. S.; NAKAMURA, Eduardo. **Computação Móvel: Novas Oportunidades e Novos Desafios.** T&C Amazônia, Ano 1, no 2, Junho 2003.

FISCHER, D.; MARKSCHEFFEL, B.; SEYFFARTH, T. **Smartphone Security: Overview of Security Software Solutions.** The 8th International Conference for Internet Technology and Secured Transactions. p. 288 – 289. 2013.

FURNELL Steven; CLARKE Nathan; KARATZOUNI Sevasti**. Beyond the pin: Enhancing user authentication for mobile devices.** Computer Fraud and Security, 8 ed. Vol 8, p. 12 -7. 2008.

Gartner Says Worldwide Traditional PC, Tablet, Ultramobile and Mobile Phone Shipments On Pace to Grow 7.6 Percent in 2014. Disponível em: <http://www.gartner.com/newsroom/id/2645115> Acesso em: 15 de maio de 2014.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. São Paulo: Atlas, 2005. p.26.

GONZALES Emiliano; GONZALES Sergio. **Smartphones como Unidad de Sensado y Procesamiento para la Localización de Robots Móviles Utilizando Odometrıa Visual Monocular.** Universidad de Buenos Aires. Departamento de Computación, Buenos Aires, Argentina. 2013.

HAYASHI, E. *et. al*. **CASA: A Framework for Context-Aware Scalable Authentication**. In Proceedings of the Ninth Symposium on Usable Privacy and Security SOUPS'13. No 3. 2013.

IDC: **Worldwide Smartphone 2014–2018 Forecast and Analysis**. Disponível em: <http://www.idc.com/getdoc.jsp?containerId=247140>. Acesso em: 10 de abril de 2014.

JOHNSON, Thienne M. Java para Dispositivos Móveis: Desenvolvendo Aplicações com J2ME. São Paulo: Novatec, 2007.

KAINDA, R.; FLECHAIS, I.; ROSCOE, A.W. **Security and usability: analysis and evaluation**. International conference on Availability, Reliability and Security. p. 275 – 282. 2010.

KARNAN, M.; AKILA, M.; KRISHNARAJ, N. **Biometric personal authentication using keystroke dynamics: A review**. Applied Soft Computing 2 ed. Vol. 11. p. 1565-1573. 2011.

KHAN, Sohail *et. al*. How Secure is your Smartphone: An Analysis of Smartphone Security Mechanisms. International Conference on Cyber Security, Cyber Warfare and Digital Forensic. p 76 –81. 2012.

LECHETA, Ricardo R. **Google Android: Aprenda a criar aplicações para dispositivos móveis com Android SDK**. 2 Ed. São Paulo: Novatec Editora, 2010.

MALEK, B.; MIRI A.; KARMOUCH A. **A framework for context-aware authentication.** In Proceedings of the 4th IET International Conference on Intelligent Environments. p.1 -8. 2008.

MARQUES, Diogo Homem **Building and Evaluating an Inconspicuous Smartphone Autentication Method**. Universidade de Lisboa. 2013.

MATEUS, Geraldo R., LOUREIRO**,** Antônio A. F**. Introdução à Computação Móvel.** 11a Escola de Computação. Rio de Janeiro, 1998

MUSLUKHOV, I. *et. al*. **Know your enemy: the risk of unauthorized access in smartphones by insiders.** In Proceedings of the 15th international conference on Human computer interaction with mobile devices and services (MobileHCI'13). p. 271 – 280. New York, USA, 2013.

MAYRHOFER, Rene. Pervasive Computing Security. 2007.

MYERS, B *et. al*. **Taking handheld devices to the next level. In** Journal Computer IEEE, 12 Ed. Vol. 37, p. 36–43. 2004.

PEDDEMORS, A.; EERTINK, H.; NIEMEGEERS, I. **Predicting Mobility Events on Personal Devices.** Pervasive and Mobile Computing Journal. 4 ed. Vol. 6. p. 401–423. 2009.

PEREIRA, Lucio Camilo Oliva, DA SILVA, Michel Lourenço. **Android Para Desenvolvedores.** 2 Ed. Rio de Janeiro: Brasport, 2009.

PONS, A.P.; POLAK, P. **Understanding user perspectives on biometric technology.** Communications of the ACM, 9 Ed. Vol. 51. p 115-118. New York, USA, 2008

GUTIERREZ, Raul. **Dispositivos móviles y NFC Aplicados al Canjeo de Tickets: BeepVip. Departamento de sistemas Informáticos y Computación.** Valencia, 2012. Disponível em: < https://riunet.upv.es/bitstream/handle/10251/18021/Raul_Broseta_TFM.pdf?sequence=1>

SARAVANAN, Premkumar, *et. al*. **LatentGesture: Active User Authentication through Background Touch Analysis**. In Proceedings of the Second International Symposium of Chinese CHI. p. 110-113. New York, USA, 2014.

SILBERSCHATZ, A; GALVIN, P. B.; GAGNE, G. **Sistemas Operacionais com Java.** 6 Ed. Rio de Janeiro: Elsevier, Editora Campus, 2004.

SHI, E. *et. al*. **Implicit authentication through learning user behavior**. In Proceedings of the 13th international conference on Information security. p. 99-113. 2010.

SOIKKELI, T.; KARIKOSK, I J.; HAMMAINEM, H. **Diversity and end user context in smartphone usage sessions.** In 5t^{h} International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST'11). p. 7-12. 2011.
SOMMERVILLE, Ian. Engenharia de software. 9 Ed. São Paulo: Pearson, 2011.

TODOROV, D. Mechanics of User Identification and Authorization: Fundamentals of Identity Management. Auerbach, Florida, 2007.

VERGARA, Sylvia Constant. **Projetos e relatórios de pesquisa em administração**. São Paulo: Atlas, 2006.

WHITMAN, M. E.; MATTORD, H. J. **Principles of information security**. 4 ed. Boston: Cengage Learning, 2011.

WINDLEY, P. J. **Digital Identity**. 1 ed. O'Reilly Media. 2005

APÊNDICE A – Formulário de Pesquisa

PESQUISA DE CAMPO - Validação do protótipo

A finalidade deste questionário é validar o protótipo referente ao modelo S.C.A.M, este tem como objetivo realizar a autenticação do utilizador para garantir que somente usuários autorizados tenham acesso as informações e recursos do dispositivo móvel.

Questões Gerais

1. Você utiliza algum método de autenticação em seu dispositivo?

( ) Sim

( ) Não

Em caso negativo relate o por que

_____

_____

2. Caso você utilize algum método, qual?

() PIN

() Senha alfanumérica

() Padrão Android

() Outro

3. Qual o motivo por utilizar esse tipo de método?

_____

_____

4. Você em preocupação com a segurança das informações armazenadas em seu dispositivo?

( ) Sim

( ) Não

Questões específicas

5. Você consegui utilizar a ferramenta sem nenhum problema?
( ) Sim
( ) Não

Em caso afirmativo relate o problema

_____

_____

6. Como você considera a precisão da localização geográfica identificada pelo aplicativo?
( ) Satisfatória
( ) Aproximada
( ) Insatisfatória

7. Consumo de bateria aumentou com a utilização do aplicativo
( ) Muito
( ) Pouco
( ) Nada

8. Sentiu diferenças no desempenho do dispositivo?
( ) Muito
( ) Pouco
( ) Nada

9. Você considera que a memorização de duas senhas pode afetar a usabilidade do aplicativo?
( ) Sim
( ) Não

10. Você considera que a autenticação implícita pode melhorar a usabilidade destes métodos
( ) Sim
( ) Não

11. O que você achou da autenticação implícita?

_____

_____

12. Como você avalia a experiência de utilizar um aplicativo um dispositivo móvel?
( ) Muito Bom
( ) Bom

( ) Ruim

13. Sugestões de melhorias:

_____

_____

_____