Forensic I.T.: uma Ferramenta Forense de Automação da Extração de Dados em Dispositivos Móveis da Plataforma *Android*.

Ítalo Marcius Virgínio da Silva
Instituto Federal da Bahia
Salvador - Bahia - Brasil
italo.marcius@outlook.com

Orientadora: Flávia M. S. Nascimento
Instituto Federal da Bahia
Salvador - Bahia - Brasil
flaviamsn@ifba.edu.br

RESUMO

Este trabalho apresenta a Forensic I.T. - *Forensic Investigation Tool* - uma ferramenta de auxílio a computação forense, que tem por objetivo automatizar o processo de extração de dados de um dispositivo móvel baseado na plataforma Android, com interferência mínima no ambiente. Além disto, esta ferramenta disponibiliza um filtro que pode ser bastante útil na fase de análise dos dados obtidos após a extração. Opcionalmente, os dados extraídos podem ser exportados para um arquivo XML. A Forensic I.T. deverá ser utilizada principalmente por peritos criminais, para dar apoio a uma investigação forense baseada em dispositivos móveis.

Palavras-chave

Android, computação forense, dispositivos móveis

1. INTRODUÇÃO

Nas últimas décadas, a tecnologia evoluiu consideravelmente, trazendo grandes avanços que hoje fazem parte do cotidiano das pessoas. A cada dia surge uma nova ferramenta que promete facilitar e agilizar as atividades diárias. A popularização de tecnologias e dispositivos, impulsionada pela facilidade de acesso aos mesmos, acaba por gerar uma dependência destas ferramentas.

Hoje, dispositivos como *tablets* e *smartphones* estão completamente imersos na vida das pessoas. Uma grande motivação para que isto acontecesse foi o incremento notável na capacidade computacional destes dispositivos, o que vem fazendo com que eles facilmente assumam as tarefas que antes eram desempenhadas exclusivamente em computadores *desktop* e *notebooks*.

Os *smartphones*, por exemplo, são considerados como peças fundamentais por um número crescente de pessoas, seja como uma ferramenta de comunicação, que permite fazer ligações e trocar mensagens de texto e áudio, e até mesmo vídeo, ou ainda como uma ferramenta de entretenimento, gerência de compromissos pessoais e de trabalho, dietas, medicamentos, desempenho esportivo, dentre outras.

Juntamente com a popularização dos dispositivos móveis, o Android - sistema operacional da Google Inc® - surgiu como uma proposta promissora para gerenciar *smartphones* e *tablets* e desponta como um dos *softwares* mais utilizados no mundo [13].

Assim, *smartphones/tablets* e Android perfazem um cenário onde milhares de informações (*e-mail*, mensagens de texto, fotos, vídeos, chamadas, *logins* e senhas) transitam a todo momento entre os usuários. Atualmente, quando um usuário perde um destes dispositivos ou é roubado, a menor preocupação é com o equipamento em si. A grande questão está relacionada a indisponibilidade de uma ferramenta que habitualmente controlava as atividades cotidianas e evidentemente com todos os dados e informações pessoais que são armazenadas nestes dispositivos e que podem ser acessadas por outrem.

Visando o grande número de usuários de dispositivos móveis, os números de ataques a essa plataforma crescem de forma progressiva. Em 2014, a Karpersky divulgou alguns dados estatísticos em seu boletim de segurança [16], de acordo com o qual havia:

- 4.643.582 pacotes de instalação mal-intencionados;
- 295.539 novos programas móveis maliciosos;
- 12.100 *trojans* bancários móveis.

Considerando o período de Novembro de 2013 a Outubro de 2014, o Kaspersky Lab evitou 1.363.549 ataques. No mesmo período nos anos de 2012-2013, foram 335.000 ataques. Isso quer dizer que o número de ataques a dispositivos Android foi quatro vezes maior em comparação com os 12 meses anteriores.

Além disso, 19% dos usuários de Android encontrou uma ameaça em dispositivo móvel pelo menos uma vez durante o ano, o que representa quase um em cada cinco usuários. Outros 53% do ataques a dispositivos Android utilizaram Trojans móveis projetados para roubar dinheiro do usuário (*trojans* SMS e *trojans* bancários) [16].

Neste cenário, a computação forense surge então como uma forma de subsidiar em processos criminais envolvendo meios digitais. Em temos de tecnologia móvel, o perito forense será responsável por analisar os dispositivos, que de certa forma estão potencialmente ligados a um crime, extraindo os dados que forem considerados relevantes e gerando um relatório pericial, que poderá ser utilizado como evidências em um possível julgamento.

.

O presente trabalho apresenta uma ferramenta de auxílio a computação forense, chamada Forensic I.T. - *Forensic Investigation Tool*. Esta é responsável pela automação do processo de extração de dados em um dispositivo móvel, que seja baseado na plataforma Android, realizando a extração automatizada de dados bit a bit, apresentando um filtro em tela para ajudar a buscar dados extraídos dos dispositivos e possibilitando exportar os dados para um arquivo no formato XML. O objetivo da Forensic I.T. é servir como ferramenta de apoio a peritos criminais durante o processo de investigação forense.

2. ESTRUTURA DO TRABALHO

Este documento está dividido como segue. A Seção 3 apresenta a motivação e a justificativa para o desenvolvimento da ferramenta. A Seção 4 apresenta um sumário dos trabalhos relacionados a esta pesquisa. A Seção 5 apresenta os conceitos relevantes sobre computação forense para ajudar na contextualização do trabalho e para melhor entendimento da proposta. Já a Seção 6 discorre sobre o Android, sua história, sua arquitetura, seus sistema de dados e suas ferramentas. A Seção 7 apresenta a ferramenta proposta, contextualizando o uso da mesma na perícia computacional e detalhando seus requisitos de funcionamento. A Seção 8 apresenta o resultado dos testes executados e da avaliação da ferramenta, que para uma maior adequação foi efetuada com apoio da Perícia Forense da Superintendência da Polícia Federal Regional Bahia. Por fim, a conclusão e possíveis desdobramentos são apresentados na Seção 9.

3. MOTIVAÇÃO

Da mesma forma que a tecnologia pode ser utilizada de forma benéfica, como ferramenta de apoio, ela pode ser utilizada para atividades ilegais, se aproveitando de falhas na segurança dos dispositivos e na dificuldade em rastrear as informações que ali trafegam. Os *smartphones* podem ser utilizados para fraudar mensagens em *e-mails*, trocar de informações ilegais, divulgação de conteúdo de pornografia infantil, distribuir imagens e informações privadas, sem o consentimento do proprietário, bem como para ter acesso a informações confidenciais, como senhas por exemplo.

De fato, a popularização dos *smartphones* e da plataforma Android provocou um aumento nos crimes cibernéticos relacionados a estas tecnologias. Neste sentido, a perícia criminal ganha papel de destaque, já que é um método de prova que analisa um conjunto de elementos relacionados a uma infração, permitindo coletar, analisar e gerar provas da existência do crime. A partir da premissa "crimes sempre deixam vestígios", a computação forense, por sua vez, tem como objetivo identificar, coletar, analisar, processar e gerar provas da existência e autoria de atividades ilícitas ligadas a área da informática, gerando evidências digitais de um crime [24].

A computação forense pode ser considerada de certa forma uma área recente da perícia criminal, tendo em vista que a primeira ferramenta que se tem registro foi criada em 1984. Porém, a computação forense voltada para dispositivos móveis é uma área nova, e portanto, com bastante conteúdo a ser explorado [10].

Os peritos computacionais com foco em dispositivos móveis ainda enfrentam muitos problemas, dentre eles a volatilidade das informações nos *smartphones*. Maiores detalhes sobre a memória e armazenamento de dados são apresentados na Seção 6.

Enquanto os estudos acerca da computação forense crescem de forma desacelerada - por conta de algumas situações já apresentadas, como a evolução contínua das tecnologias -, os ataques cibernéticos a este tipo de plataforma crescem gradativamente. Cada vez mais usuários são vítimas de algum tipo de ação criminosa com apoio de algum artifício da computação. Além disso, existe o fato dos aparelhos móveis ainda apresentarem uma baixa segurança dos dados, se comparado com a quantidade de dispositivos e ferramentas disponíveis para um computador *desktop*, por exemplo.

A popularização dos *smartphones*, a menor gama de pesquisas na área de computação forense voltada para dispositivos Android (quando comparado com outras áreas da computação forense) e o crescente número de ataques a esses usuários foram os três fatores determinantes que motivaram o desenvolvimento deste trabalho.

Assim, o objetivo aqui pode ser detalhado como:

- realizar a extração automatizada de dados de dispositivos móveis baseados em Android, bit a bit;
- fornecer um método para checar eventuais perda de dados durante o processo de extração de dados periciais - usando o algoritmo MD5;
- realizar uma cópia de segurança dos dados extraídos;
- fornecer um filtro em tela para uma pesquisa rápida sobre os dados extraídos;
- exportar os dados extraídos para um arquivo no formato XML.

4. TRABALHOS RELACIONADOS

Esta seção apresenta algumas ferramentas que auxiliam no processo de extração de dados de um dispositivo móvel.

CelleBrite - UFED

Universal Forensic Extraction Device (UFED), é um produto da Cellebrite criado com foco em forense digital e indústria investigativa. O UFED é um dispositivo portátil que contém software para *desktop* opcional, cabo de dados, adaptadores e outros periféricos. Este produto não necessita qualquer conexão com um software para *desktop*. O dispositivo foi projetado e estará disponível especificamente para as forças policiais e os militares, as agências de inteligência e segurança corporativa [19].

O UFED pode extrair, decodificar e analisar os seguintes dados:

- contatos da agenda telefônica;
- todos os tipos de conteúdo multimídia;
- mensagens SMS e MMS;
- registros de chamadas;
- dados do IMEI e SIM.

A ferramenta também permite interagir com sistemas de arquivos de diferentes sistemas operacionais, como o Android, iOS, Symbian e Windows Mobile.

O UFED permite ao usuário realizar tanto a extração física quanto a extração lógica. A extração lógica

desta ferramenta é dividida em duas, a "lógica"e "arquivos de sistema". Ambas são capazes de obter dados como SMS, contatos, *logs* - *formatação* de chamadas e arquivos de multimídia, porém apenas a extração de sistemas de arquivos é capaz de obter dados ocultos. Em nenhuma das duas opções é possível se obter dados excluídos [3].

FTK Imager

Forensic Toolkit Imager (FTK Imager), é uma ferramenta forense gratuita desenvolvida pela AccessData. Ele é capaz de gerar imagens de um disco específico, como um HD ou um SD card, assim como é capaz de gerar imagem de uma partição do disco. O *FTK Imager* disponibiliza opção de salvar a imagem do disco em um arquivo ou em segmentos, e ambos podem ser reconstruídos posteriormente.

Esta ferramenta disponibiliza as seguintes opções de formato de arquivo para gerar a imagem:

- Raw: Formato de copia bit a bit do disco original. Este formato é gerado, em geral, pelo comando `dd` e não está associado a nenhum pacote forense específico;
- SMART: Formato de imagem criado para a ferramenta Smart da ASR Data;
- EnCase: Formato de imagem criado para a ferramenta EnCase da Guidance Software;
- Advanced Forensics Format (AFF): formato de armazenamento de disco que inclui um metadado forense[1].

A ferramenta é capaz de gerar evidências dos dados extraídos, exportar arquivos de evidências, criar imagens forenses e converter imagens existentes.

EnCase

EnCase é uma ferramenta compartilhada dentro de um conjunto de produtos comerciais da Guidance Software, voltados à investigação digital. O EnCase se aplica nas seguintes fases da análise forense: fase de aquisição, fase de análise e fase de apresentação.

Esta ferramenta executa uma copia bit a bit de um dispositivo e trabalha com arquivos em um formato próprio, o *EnCase Evidence File Format*. Esta ferramenta ainda possibilita ao usuário criar scripts para automatizar algumas tarefas.

A EnCase tem compatibilidade com as seguintes plataformas mobile:

- Apple iOS;
- RIM BlackBerry;
- Google Android;
- Windows Mobile;
- SIM *cards*;
- iTunes e *backup files*.

[1]Um arquivo de metadado forense pode conter importantes informações sobre o arquivo periciado, podendo ser útil em um processo pericial, tais como: informações referentes ao arquivo em si e informações excluídas ou suprimidas [11]

A ferramenta realiza, durante a extração, uma contínua verificação para garantir que uma maior quantidade de dados serão adquiridos. Os examinadores podem ainda definir a quantidade de dados a ser extraída durante o processo de aquisição, visando dessa forma um processo mais ágil.

Oxygen Forensic Extractor

Oxygen Forensic Extractor é uma ferramenta comercial desenvolvida pela Oxygen Software. Esta ferramenta se enquadra na fase de aquisição de dados, fornecendo uma extração lógica em dispositivos móveis. A ferramenta é aplicável em telefones, *smartphones* e PDAs e possibilita extrair algumas informações do dispositivo periciado, como contatos, eventos de calendário, *logins* e senhas associados ao dispositivo, SMS, *logs* de eventos e arquivos de mídia e seus respectivos metadados (mencionados na Tabela 1 como "Arquivos em geral").

A Oxygen tem compatibilidade com diversos dispositivos das seguintes plataformas mobile:

- Apple iOS;
- Symbiam;
- Blackberry;
- Google Android;
- Windows Mobile.

A ferramenta ainda disponibiliza um menu de fácil entendimento para o usuário e facilita a análise dos dados com relatório dos dados extraídos.

De forma a sumarizar os dados das ferramentas apresentadas e evidenciar as contribuição da ferramenta proposta, apresentamos a Tabela 1.

5. COMPUTAÇÃO FORENSE

A ciência forense pode ser entendida como o conjunto de técnicas e conhecimentos científicos que podem ser utilizados para solucionar crimes [5].

O ramo da ciência forense responsável por solucionar casos de crimes cibernéticos, ou seja casos que tenham alguma relação com aspectos da computação, é a computação forense. Existem alguns outros termos para definir essa área de atuação, como: Forense Computacional, Perícia Computacional, Informática Forense, Perícia Digital e Forense Digital. Neste documento utilizaremos o termo "computação forense".

A definição de computação forense pode ser encontrada em algumas fontes [5] [6]. Aqui, destacamos:

> "Uma área da ciência da computação que se desenvolve gradualmente para atender à demanda oriunda da área da Criminalística, ou como também, uma parte da criminalística que se apropria de fundamentos da ciência da computação."[18]

Uma outra forma de visualizar a perícia forense computacional é como o processo das técnicas e metodologias criadas na computação forense, aplicado com apoio de ferramentas apropriadas para obter dados e artefatos e tendo por objetivo qualificá-los como vestígio, evidência ou prova no âmbito judicial [6].

Tabela 1: Sumário das Ferramentas de Investigação Forense

	Forensic I.T.	Cellebrite	FTK Imager	Encase	Oxygen
Plataforma	Linux, Windows	Windows	Windows	Windows	Windows
Licença	Proprietário	Proprietário	Proprietário	Proprietário	Proprietário
Fase da Perícia	Aquisição e Análise	Aquisição	Aquisição	Aquisição e Apresentação	Aquisição
Validação de Integridade MD5	Sim	Sim	Sim	Sim	Sim
Cópia de Segurança	Sim	Não	Não	Não	Não
Formato de Relatório	XML	XML, CSV, HTML, PDF, XLS	XML	XML	XML, CSV, HTML, PDF, XLS
Análise	Base de Dados SQL	Arquivos apagados	Disco e Partições	Disco e Partições	Arquivos em geral

Do ponto de vista de organização de processos, a Computação Forense pode ser dividida em quatro grandes etapas: Preservação, Extração, Análise, Apresentação.

A seguir veremos uma breve explanação sobre cada uma das etapas citadas para melhor entendimento do processo pericial como um todo e de que forma ele se aplica no âmbito das tecnologias móveis.

5.1 Preservação

A fase de Preservação consiste em garantir que os dados, armazenados no dispositivo a ser analisado pelo perito criminal, jamais sofram qualquer modificação [5] [6]. Especificamente, esta é uma fase que requer muito cuidado, tendo em vista que qualquer operação, por mais simples que pareça, pode alterar os dados de um dispositivo móvel. Analisando um *smartphone*, por exemplo, o dispositivo em questão deve, preferencialmente, estar com a bateria carregada com carga máxima, visando evitar que o mesmo descarregue por completo e desligue durante a etapa de extração de dados. Além disso, o mesmo deve ser ligado restritamente em um local onde não exista qualquer tipo de sinal da rede de telefonia, visando evitar que o dispositivo receba novas ligações, mensagens ou emails e modifique os dados mais antigos da memória [6].

A preservação dos dados em dispositivos móveis é um pouco mais complicada, já que na maioria das vezes não é possível remover o disco rígido e efetuar a clonagem para outra mídia, como pode ser efetuado em um computador *desktop*. Dessa forma, na maioria dos casos o processo de extração é realizado utilizando o próprio aparelho, requisitando uma maior atenção do perito e exigindo a mínima interação com o dispositivo.

5.2 Extração

A fase de extração é o ponto de atuação principal da proposta apresentada neste trabalho. Durante esta fase os dados preservados serão extraídos para uma futura análise. Tendo em vista os dispositivos móveis, mais especificamente os *smartphones*, podemos ter diversas informações relevantes a serem extraídas, tais como: *SMS e MMS, e-mails, logins* e senhas, registros de chamadas recebidas e realizadas, páginas da *web* visitadas, fotos, áudios, vídeos, contatos telefônicos, entre outros dados. Cabe ao perito no momento da etapa de extração, em posse de seu conhecimento e experiência, julgar um dado como relevante ou não para o caso.

O processo de extração de dados pode ser realizado de forma física ou lógica, dependendo do dispositivo analisado. A extração lógica é mais simples de ser executada, ela se configura pelo acesso ao sistema de arquivos, acessando dessa forma apenas registros que não foram apagados, com exceção de alguns arquivos, como por exemplo os dados armazenados na SQLite que é a base da dados utilizada na arquitetura do Android, que podem conter alguns registros previamente excluídos. O modo de extração lógica, normalmente, só requer a modo de depuração USB ativado no dispositivo e geralmente não necessita de acesso *root*, que é o acesso aos arquivos do dispositivo Android com permissão de administrador. O acesso *root* será detalhado na Seção 7.

A extração física por outro lado, acessa o meio de armazenamento físico diretamente, desprezando o sistema de arquivos. A maior vantagem dessa técnica é o acesso a uma grande quantidade de informações excluídas, isso se deve ao fato de que o sistema de arquivos muitas vezes só marca os dados como excluídos ou obsoletos, mas na verdade não apaga definitivamente, a menos que seja necessário.

Ainda é possível classificar a extração de dados em *smartphones* em "manual"ou "automática". A definição destes dois processos pode ser descrita como [6]:

Extração manual

A extração manual se configura quando todo o conteúdo do aparelho é percorrido manualmente pelo perito, por meio da navegação através dos próprios menus do aparelho. Assim, os dados devem ser anotados para posterior análise e confecção do laudo pericial. É uma operação bastante mecânica e simples, se comparado com o método de extração automática, porém devem ser tomados todos os cuidados de preservação já citados anteriormente, para que não seja apagado nenhum dado contido em qualquer fonte de memória do dispositivo.

Extração automática

A extração automática se configura quando o perito se utiliza de *softwares* e/ou *kits* específicos para extrair os dados. O processo consiste basicamente em realizar a cópia bit a bit dos dados do aparelho, uma clonagem dos dados a serem analisados, sem a necessidade de navegação manual no dispositivo em processo de analise.

Tendo em vista o processo de extração, os aparelhos periciados dispõem de algumas opções para conectar o dispositivo móvel com um computador, como por exemplo, os cabos de dados ou as conexões sem fio (in-

fravermelho, *wi-fi*, *bluetooth*, *NFC* [2]). No caso da perícia computacional, a conexão com fio é mais indicada por ser mais segura e eficaz [6].

5.3 Análise

A fase de análise é o momento do perito criminal examinar os dados extraídos na fase anterior a fim de encontrar evidências que tenha alguma relação com o delito que está sendo investigado [5] [6].

No caso dos aparelhos *smartphone*, a fase de análise, na maioria dos casos, é nada mais do que descrever os dados presentes no aparelho periciado. Nessa fase pode ser necessário utilizar algumas técnicas específicas para a análise, como por exemplo técnicas de quebra de senhas e/ou criptografias.

5.4 Apresentação

Na etapa de apresentação é onde será elaborado o laudo de exame do aparelho periciado, além de registrar especificações relacionadas ao material em questão. No caso dos aparelhos celulares, algumas informações se fazem necessárias, como: número IMEI[3] , dados da bateria, marca e modelo, operadora e número do cartão SIM.

No laudo pericial, deve ser relatado os dados que foram obtidos durante as fases anteriores, visando as evidências do crime investigado. Além disso, no laudo devem ser descritos todos os métodos utilizados para a preservação, obtenção e análise dos dados.

O perito deve ter consciência que o laudo pericial é um documento técnico-científico, que deve além de tudo apresentar clareza nas informações.

6. ANDROID

6.1 História

O sistema operacional Android, surgiu em 2003, foi desenvolvido por Andy Rubin, Rich Miner, Nick Sears e Chris White que fundaram a Android Inc®. Na ocasião, Rubin definiu o Android como [8]:

> "Dispositivos móveis mais inteligentes e que estejam mais cientes das preferências e da localização do seu dono".

O Android oferecia um sistema operacional móvel *open source*, baseado no *Kernel Linux*. O sistema contava com uma interface simples, funcional e também integrada a vários instrumentos. A ideia era oferecer um sistema gratuito para todas as pessoas que quisessem ter acesso a ele e também ser simples aos desenvolvedores [8].

A Google adquire a empresa Android Inc® em 2005 implantando então a *Google Mobile Division*. Após dois anos de trabalho, no final de 2007, Andy Rubin faz o seguinte anúncio do plano da companhia sobre o Android, no blog oficial da Google [21]:

> Android é a primeira plataforma verdadeiramente aberta e abrangente para dispositivos móveis. Isto inclui um sistema operacional, interface de usuário e aplicações – tudo que é necessário, em termos de software, para executar em um telefone celular, porém sem os obstáculos que tem prejudicado a inovação em computação móvel. Nós temos desenvolvido o Android em cooperação com o *Open Handset Alliance*, que consiste em mais de 30 tecnologias e líderes do mercado de telefonia móvel incluindo Motorola, Qualcomm, HTC e T-Mobile. Através da profunda parceria com operadoras, fabricantes de dispositivos, desenvolvedores e outros, nós esperamos permitir um ecossistema aberto para o mundo móvel através da criação de uma plataforma de software aberto padrão. Nós idealizamos que o resultado final será o melhor e mais rápido passo para a inovação que trará aos clientes móveis aplicações e possibilidades inimagináveis[4].

Uma semana após o anúncio, a Google divulgou uma prévia do *SDK (Software Development Kit)* o que permitiu que a empresa criasse o primeiro Desafio de Desenvolvimento Android (*Android Developer Challenge*) para premiar o aplicativo Android que fosse considerado o mais inovador. Esse evento acabou por atrair e estimular os desenvolvedores a programar diversos aplicativos para a plataforma Android.

Após três anos da aquisição e com todo esse *marketing* o Android foi lançado comercialmente no mercado, o lançamento se deu oficialmente em 22 de outubro de 2008, executando em um *HTC Dream*[5].

Atualmente, o Android é o sistema operacional móvel mais utilizado do mundo, e, em 2013, possuía a maior porcentagem das vendas mundiais de sistemas operacionais móveis, de acordo com pesquisa publicada pelo IDC [1].

6.2 Arquitetura

Quando pensamos na arquitetura do sistema operacional Android, vale lembrar que ele é baseado no *kernel* do *linux* e herdou algumas características do mesmo. Assim sendo, podemos analisar a sua arquitetura dividindo em cinco camadas: camada de aplicações, camada de framework das aplicações, camada de bibliotecas, camada de *runtime* e camada de *kernel linux*, como está sendo apresentado na Figura 1.

[2] *Near Field Communication* (Comunicação por Campo de Proximidade), mais conhecido como NFC, é uma tecnologia que permite a transferência de informações sem fio entre dispositivos que estejam próximos um do outro. Os dispositivos que utilizam essa tecnologia podem ser telefones celulares, tablets, crachá, entre outros e eles podem realizar ações como pagamento de comprar em mercado, transferência de imagens e compra de ingresso para cinema

[3] International Mobile Equipment Identity (Identificação Internacional de Equipamento Móvel), mais conhecido por IMEI, ele é um número de identificação global de cada aparelho e é único para cada telefone.
O número IMEI é formado por quatro grupos de números "000000-00-000000-0"e esse conjunto de números é o que identifica o celular ou smartphone. Os números IMEI ficam armazenados em um banco de dados de Registro de Identidade de Equipamentos (EIR). Para localizar o IMEI do aparelho basta digitar "*#06#"[9]

[4] Livre tradução. O texto original pode ser encontrado em [21]

[5] O *HTC Dream*, também conhecido como *T-Mobile G1*, é um *smartphone* desenvolvido pela fabricante *HTC*. É o primeiro telefone com o sistema operacional Android e tinha como concorrentes de mercado marcas como iPhone, Blackberry e Symbian.

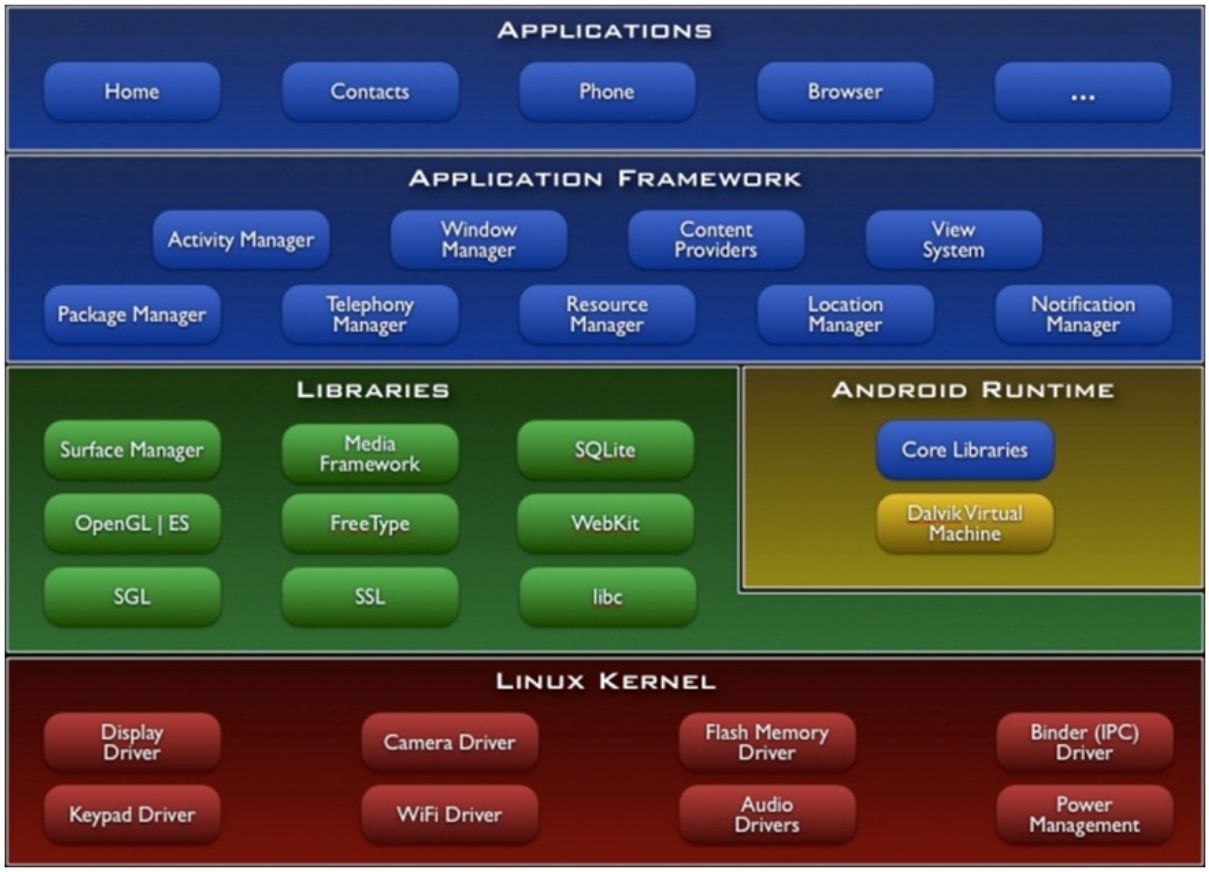

Figura 1: Representação da Arquitetura do Android

A camada de aplicações é onde estão localizadas as aplicações que serão executadas no sistema operacional. Esta é a camada que facilita a interação entre o usuário e o dispositivo móvel, podendo encontrar os diversos aplicativos disponíveis (SMS, calculadora, *e-mail* , mapas, etc).

A camada de *framework* das aplicações é a camada onde as aplicações são gerenciadas. Os desenvolvedores de aplicativos para Android tem total acesso aos *frameworks* utilizando como um conjunto de ferramentas com o qual será possível modelar aplicações mais complexas.

A camada de bibliotecas é um conjunto de instruções que direcionam o dispositivo a lidar com diferentes tipos de dados, incluindo as biblioteca C / $C++$ usadas por diversos componentes do sistema. Também podem ser encontrados nessa camada funções de acesso a banco de dados do Android (SQLite).

A camada de *runtime* é responsável por instanciar uma máquina virtual que é criada para cada aplicação que está em execução, executando cada aplicação em seu próprio processo. O ponto positivo dessa camada é que nenhuma aplicação é dependente de outra e, dessa forma, caso uma aplicação venha a parar, não afeta quaisquer outras aplicações executando no dispositivo naquele momento, simplificando assim o gerenciamento de memória do dispositivo, pois a máquina virtual é desenvolvida de forma tal a permitir que múltiplas instâncias executem ao mesmo tempo.

E por último, a camada de *kernel linux* é o núcleo do sistema e onde está localizado o sistema operacional da plataforma, derivado do *kernel* do *linux* (versão 2.6) e herdando muitas características (gerenciamento de processos, protocolos de redes, *threads*, entre outras). Nessa camada é possível encontrar programas de gerenciamento de memória, configurações de segurança e diversos *drivers* de *hardware*.

6.3 NAND flash

Os dispositivos Android, necessitam de vários tipos de memória para funcionar. Como qualquer computador, os dois tipos principais de memória requeridos são voláteis, nesse caso a memória RAM, e memória não volátil, a memória NAND flash [13].

A memória NAND flash é uma memória não volátil e, portanto, os dados são preservados mesmo quando o dispositivo está sem energia ou é reiniciado. O NAND flash é utilizado para armazenar não só arquivos de sistema, mas também partes significativas de dados do usuário.

Este tipo de memória tem características muito diferentes em relação a mídias magnéticas encontrados em discos rígidos modernos. Algumas propriedades fazem a NAND flash a memória ideal para o armazenamento em dispositivos móveis, ao mesmo tempo que apresenta algumas oportunidades para os analistas forenses.

Uma importante característica é que a memória NAND flash não tem partes físicas que se movem mecanicamente, como os pratos giratórios e braços encontrados em discos rígidos magnéticos mais tradicionais. Essa característica melhora a durabilidade e reduz o tamanho e o consumo de

energia do dispositivo.

A memória também possui baixo custo para a fabricação, este ponto faz com que seja muito popular entre os fabricantes. Um efeito colateral do processo de fabricação e tecnologia em geral é que a NAND flash tem *bad blocks* embarcados diretamente do fabricante. O fabricante geralmente testa a memória como parte do processo de fabricação e marca *bad blocks* em uma estrutura específica na memória, que vem descrita na sua documentação. Alguns programas, que interagem diretamente com a NAND flash, podem ler os marcadores de *bad block* e muitas vezes implementar uma tabela de blocos defeituosos que pode acompanhar a execução logicamente e remover tais blocos da operação.

O algoritmo utilizado para gerir de forma eficaz o NAND flash em Android é o código de nivelamento de desgaste que é executado na escrita de dados em toda memória NAND flash para evitar o excesso de utilização de uma única área, liberando esses blocos mais rapidamente, apresentando um melhor desempenho em processamento.

Os dispositivos Android foram projetados para integrar os componentes da memória NAND flash diretamente, e dessa forma uma camada de gerenciamento de software era necessária. A camada selecionada para gerir o NAND flash foi o sistema de dispositivos *Memory Technology* (MTD), que foi desenvolvido para atender as necessidade do NAND flash e dispositivos semelhantes, devido às suas características únicas.

Antes de utilizar o MTD, o Linux dava suporte principalmente a dispositivos com sistema em caracteres e dispositivos com sistema em bloco. No Android, o MTD fornece não só a *block interface* para o NAND flash mas também outras funções críticas.

6.4 YAFFS2

O Yet Another Flash File System 2, ou YAFFS2, é um dos principais sistemas de arquivos do sistema operacional Android; é um sistema *open source* desenvolvido para dispositivos NAND flash e agrega diversas características importantes que atendem as necessidades dos dispositivos móveis, tais como:

- Tem um sistema de arquivos estruturado em *log*, que protege os dados mesmo que sofra uma interrupção inesperada de energia;
- Capaz de lidar com *bad blocks* (setores do meio de armazenamento com defeito);
- Rápido processamento e ocupa pouca memória RAM.

YAFFS2 adota um sistema de memória em blocos, através do subsistema de MTD, e cada bloco contém um determinado número de páginas.

Todas as estruturas de dados armazenadas em YAFFS2 são referenciadas como objetos e podem ser arquivos, diretórios e/ou *links*. Uma dos importantes partições em YAFFS2 é a "data/data" onde pode ser encontradas databases que armazenam dados como *MMS*, *SMS*, contatos, calendário, registros de chamadas, configurações do sistema, senhas e *logins*, *e-mails*, entre outros.

6.5 SDK e sua ferramenta ADB

Normalmente, os SDKs (*Software Development Kits*) são disponibilizados por projetos *open source* para que desenvolvedores de software do mundo todo tenham uma melhor integração com os seus respectivos *softwares*. No kit tem praticamente todas as ferramentas que são necessárias para programar e emular aplicativos, no caso deste trabalho, para a plataforma Android. Tendo em vista o trabalho proposto neste documento, uma das ferramentas do SDK, o adb shell, será importante por interligar o computador e o dispositivo periciado.

`Adb shell` (*Android Debug Bridge Shell*) é uma ferramenta do kit de desenvolvimento que funciona como um terminal de comandos, que possibilita acessar e controlar através de um computador uma instância de um dispositivo emulado, ou um dispositivo físico conectado.

Sob a ótica da perícia forense, o adb é fundamental para a etapa de extração, acessando dados que até então eram inacessíveis na memória do dispositivo e realizando a cópia segura, com o mínimo de perda possível de dados.

O SDK do Android não só proporciona uma visão mais profunda sobre a plataforma Android, mas também fornece ferramentas para investigar um dispositivo, tanto do ponto de vista forense, quanto do ponto de vista da segurança. Desta forma, o perito tem a capacidade de interagir com um dispositivo Android conectado via USB, por exemplo, desde que o recurso de depuração USB esteja ativado.

O Android SDK, em conjunto com o adb, é capaz de realizar uma gama de ações no dispositivo periciado, tais como visualizar, alterar e extrair dados da memória, instalar e executar aplicativos, entre outros.

7. PROPOSTA

Se considerarmos os últimos 20 anos, é possível perceber que o grau de dependência tecnológica para realização das atividades cotidianas vem aumentando. Uma das evidências é que hoje os *smartphones* são considerados indispensáveis nas mais diversas tarefas do dia a dia. Podemos mencionar diversos fatores que aumentam esta dependência, tais como a facilidade de acesso a tecnologia, a capacidade de conectividade, a inclusão digital, a simplificação e agilização de tarefas complexas e triviais. O *smartphone* acaba por se tornar uma peça de fundamental importância no cotidiano do homem.

Com a popularização dos *smartphones*, o sistema operacional Android surge e logo se destaca no mercado como sistema operacional líder entre os dispositivos móveis, por conta do seu baixo custo e da sua flexibilidade para lidar com diversos dispositivos disponíveis no mercado, se destacando quando comparado aos principais concorrentes: o Iphone e o Blackberry.

Contudo, toda tecnologia que se torna foco da sociedade e que se apresenta como uma ferramenta agregadora de benefícios, ganha uma notória popularidade e por consequência acaba se tornando foco também de atividades criminais. Dessa forma, com o intuito de abranger o maior número de possíveis vítimas, diversos *smartphones* passam a receber uma grande quantidade de ataques cibernéticos diariamente.

Como mencionado anteriormente, o SDK do Android possui diversas ferramentas para desenvolvedores, inclusive para investigar um dispositivo, tanto do ponto de vista forense quanto do ponto de vista da segurança, por exemplo o `adb`, editor gráfico, bibliotecas, AVD (*Android Virtual Device*) e o UI Automator Test Framework [2]. Por conta disto o SDK pode se tornar uma ferramenta perigosa quando utilizada com foco em atos ilícitos.

Diante dos excessivos ataques criminais sofridos pela so-

ciedade, a computação forense atua como uma vertente no sentido de solucionar os variados casos que envolvem diferentes tipos de tecnologia. Um aspecto que pode contar negativamente para os peritos forense é que por conta da tecnologia estar em constante evolução, surgem atualizações e, consequentemente, os métodos utilizados pela perícia podem ficar defasados com uma frequência maior que o desejado.

Esta seção apresenta a ferramenta de auxílio à perícia computacional em dispositivos móveis baseados na plataforma Android, Forensic I.T. - *Forensic Investigation Tool*. Esta ferramenta fornece um método de extração automatizada de dados bit a bit da memória do dispositivo, tendo como foco a menor perda de dados durante o processo de extração de dados periciais, dispondo os dados para o processo de análise, por facilitar o acesso aos dados obtidos através de filtros e/ou de um arquivo XML exportado, além de dispor um método para verificar a integridade do arquivo.

7.1 A ferramenta: Forensic I.T.

Forensic Investigation Tool, é uma ferramenta que possibilita ao perito - usuário no contexto da computação forense, realizar a extração dos dados de um dispositivo móvel baseado no sistema operacional Android de forma lógica e automatizada e com o mínimo de interação manual possível com o aparelho. O fluxo principal desta ferramenta é apresentado na Figura 2.

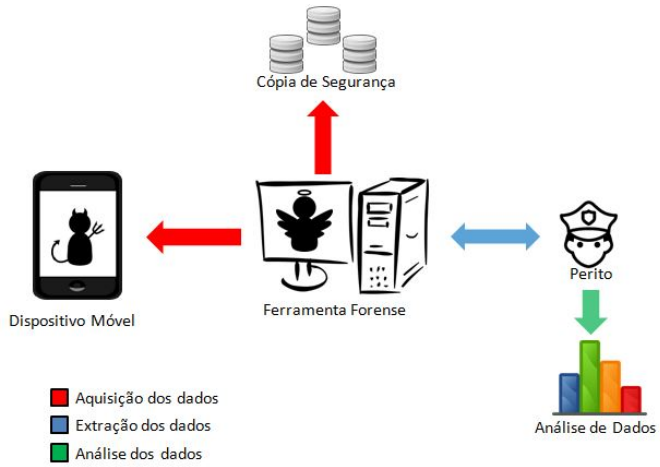

Figura 2: Fluxo de Operações da Forensic I.T.

A ideia é que o aparelho analisado seja conectado a um computador via cabo USB e a partir daí o aplicativo faça todo o processo de extração e disponibilize os dados de maneira simples, preparando-os para a etapa seguinte de análise das informações extraídas.

A extração de dados terá como foco os arquivo da base de dados do Android e é realizada através do comando `dd` que faz a clonagem bit a bit da memória, criando uma cópia idêntica no cartão de memória removível (*SD Card*), sem maiores efeitos no arquivo original. Vale ressaltar que a ferramenta foi desenvolvida tendo como base as informações da plataforma Android 2.3 (*Gingerbread*) e é indicada a aplicação para a mesma por questões de compatibilidade. Apesar disto, pode ser utilizada, até o momento, para a versão 5.0 (Lollipop) e anteriores, uma vez que a arquitetura do Android no que diz respeito ao armazenamento das bibliotecas, principalmente a SQLite, foi mantida.

7.2 Comando de clonagem de dados

A Forensic I.T. foi implementada com base no comando de clonagem de dados `dd`, através do `adb shell`. Para facilitar o isolamento com a aplicação e prover maior flexibilidade, os comandos que resultarão no processo de copia bit a bit da base de dados são gravado em um arquivo *shell script*. O conteúdo deste arquivo pode ser visto no Algoritmo 1.

Algoritmo 1: Shell Script: Comando "dd"

```
cd C:/AndroidDev/adt-bundle/sdk/platform-tools
./adb devices
./adb shell "su -c
'dd if=/data/data/
com.android.browser/databases/browser.db
of=/sdcard/browser.db'"
exit
exit
```

O comando `dd` é um comando oriundo de sistemas operacionais Unix [4], cujo objetivo principal é converter e copiar arquivos. Ele copia todo o conteúdo de uma partição de origem para uma partição de destino, ainda tendo a possibilidade de converter o formato do arquivo durante a copia.

Tendo em vista a ferramenta desenvolvida, o comando recebe como parâmetros o diretório de origem da base de dados (`if`) e o diretório de destino do clone (`of`), no caso o SD card. Podendo ainda definir o `bs` (*blocksize*, ou tamanho do bloco), que define o número de bytes gravados ao mesmo tempo [17]. Para cada uma das operações realizadas, a saída será gravada em *log* para eventuais consultas.

7.3 Biblioteca SQLite

Durante um exame forense um dos focos da perícia em um dispositivo Android será principalmente as bibliotecas e, em particular, os arquivos de dados SQLite [13].

O SQLite é uma biblioteca popular de banco de dados SQL, que dispensa um SGDB, e isso se deve a alguns fatores, entre eles, o fato de ser *open source*, ao contrário de outros sistemas de gerenciamento de banco de dados relacionais tradicionais, como Oracle, MySQL, e o Microsoft SQL Server. Além disso, o banco de dados SQLite está totalmente contido num único arquivo *cross-plataform*[6]. [13]

SQLite utiliza a extensão `.db` para seus arquivos de bases de dados e eles são geralmente armazenados no seguinte diretório `/data/data/<packageName>/databases`, por exemplo `data/data/com.android.browser/databases/`. Entretanto, não há restrição na criação de objetos em outro caminho. Neste diretório podemos encontrar bases de dados contendo diversos tipo de informações, como histórico de navegadores de internet, *e-mails* ou contatos, que podem ser relevantes durante um processo pericial. Assim sendo, os arquivos de bancos de dados SQLite são uma rica fonte de dados forense. [22]

A Forensic I.T. tem como foco a extração de dados das bases de dados Android e as bases de dados disponibilizadas na ferramenta são:

[6]Também conhecido como multiplataforma, é um programa ou sistema que pode ser executado em mais de uma plataforma.

- browser.db: armazena informações referentes ao navegador da internet, como usuário, senha e *URLs* acessadas, dados digitados no formulário do navegador e histórico de sites. Estes arquivos podem ser encontrados na pasta `/data/data/com.android.browser /databases/`;

- osp.db: armazena informações referentes as credenciais do usuário no sistema da marca do dispositivo (Samsung), tais como nome de usuário e senha encriptada na pasta `/data/data/com.osp.app.signin /databases/`;

- mmssms.db: armazena as informações referentes a *MMS* e *SMS* enviados e recebidos na pasta `/data/data/com.android.providers.telephony/databases/`;

- contacts2.db: armazena informações referentes a contatos salvos no dispositivo na pasta `/data/data/com.android.providers.contacts/databases/`.

Além das bases de dados do Android, a Forensic I.T. disponibiliza também duas bases de dados do aplicativo Whatsapp[7]. Tendo em vista a popularidade deste aplicativo para a comunicação entre os usuários, principalmente no Brasil, onde o Whatsapp é o quarto maior consumidor de internet móvel [12], este deve ser um dos focos de atuação da perícia forense. As bases do Whatsapp disponibilizadas são:

- msgstore.db: contem as informações de todas as mensagens enviadas e recebidas pelo aplicativo;

- wa.db: armazena informações da lista de contatos do usuário.

Ambos arquivos estão localizados na pasta `/data/data/com.whatsaapp /databases/`.

De fato, algumas dentre estas bases de dados podem ser obtidas sem a necessidade do comando `dd` e consequentemente do usuário `root`[8]. A alternativa é usar o comando `backup` disponibilizado pelo `adb`, a partir do qual é gerado um arquivo cópia referente as bases do dispositivo inteiro.

Entretanto, apesar de o comando `backup` ser suficiente para as bases do Android, algumas bases de dados de interesse da perícia não são incluídas quando este comando é executado, como por exemplo as base de dados do aplicativo Whatsapp. Por isso, a opção que fizemos foi de usar o comando `dd` para a extração dos dados de qualquer base.

7.4 Aplicação na Perícia Forense

A extração automática em um dispositivo Android poderia ser realizada pelo perito forense executando os comandos de acesso ao dispositivo e cópia dos dados um a um no terminal de comando, com auxílio do SDK do Android. Este processo poderia demorar horas dependendo da quantidade de arquivos a serem processados [17].

Mesmo sendo um método de extração automática, acaba por ser um tanto quanto cansativo por conta do esforço repetitivo para executar cada um dos comandos. Diante deste cenário, a Forensic I.T. automatiza esse esforço do perito em executar diversas linhas de comando no terminal até que se consiga extrair os dados, acessando o dispositivo.

Ao invés disto, o trabalho manual do perito fica diminuído, uma vez que a ferramenta procede com uma clonagem dos dados e posterior extração. Isto sem mencionar o fato de que através da Forensic I.T. este processo fica transparente ao usuário (em termos de comandos). Ao final, os dados são disponibilizados de forma simples e objetiva, facilitando o processo de extração e agilizando o processo pericial como um todo.

É importante ressaltar aqui que o processo executado pela ferramenta desenvolvida está de acordo com as premissas da computação forense na medida em que busca preservar a integridade dos dados. Um dos cuidados fundamentais a serem seguidos pelo perito para o bom desempenho do processo pericial é "Não deve ser realizada a pericia na mídia original". Assim, o processo de extração de dados através da clonagem dos dados contidos na da mídia original é importante e requer cautela para que durante o processo o dado original não seja comprometido, assim como descreve Vargas [24]:

> "Em Forense Digital, a excelência não é uma opção, é uma necessidade operacional."

7.5 Hash MD5

As ferramentas forenses precisam utilizar algum método para garantir a integridade dos dados copiados, e assegurar que durante a cópia e/ou extração nem mesmo um bit tenha sido alterado na aplicação. Uma das formas de garantir isto é através do cálculo de uma função *Hash*.

O algoritmo *hash* gera uma valor (chave), que se refere ao resultado da função *hash*, que poderá ser comparado futuramente. Em todas as vezes que o novo valor gerado for diferente do valor original, isso significa que os dados sofreram algum tipo de alteração [23]. Os tipos mais comuns de *hash* utilizados por ferramentas forense são Sha-1, Sha-256 e o MD5 [13]. A Forensic I.T., usa o algoritmo MD5.

O *hash* MD5 é um algoritmo desenvolvido pela RSA Data Security Inc. Ele é unidirecional, ou seja, não pode ser transformada novamente na informação que lhe deu origem. O método de verificação é feito pela comparação de dois valores de *hash*, comparando os valores de *hash* gerado para um mesmo arquivo em momentos diferentes. O processo de verificação pode também ser utilizado para garantir que dois arquivos são idênticos ou a integridade de um arquivo. O *hash* MD5 é um algoritmo de 128 bits que normalmente é representado por uma sequência de 32 caracteres hexadecimais [20].

7.6 Arquitetura

A ferramenta desenvolvida dispõe ao usuário as seguintes funcionalidades:

- Clonagem dados do celular periciado;

- Backup dos dados clonados;

- Extração de dados de qualquer arquivo de base da dados *SQLite*, extraído pela ferramenta ou não;

- Validação de integridade do arquivo, através do cálculo de *hash*;

- Apresentação dos dados extraídos em tela através de uma tabela;

[7]WhatsApp Messenger é um aplicativo multiplataforma, que permite a troca de mensagens entre dispositivos, sem a necessidade de pagar por um *SMS* [14]

[8]Por ser baseado no sistema Linux, boa parte das operações no Android que requerem modo privilegiado, precisam ser feitas usando o usuário `root`.

- Filtro dos dados obtidos na extração (apresentados na tabela);
- Exportação para XML dos dados extraídos;
- Geração de *log* das interações com o dispositivo periciado no formato texto.

Estas interações entre o usuário e o sistema podem ser visualizadas no Diagrama de Caso de Uso da Figura 3.

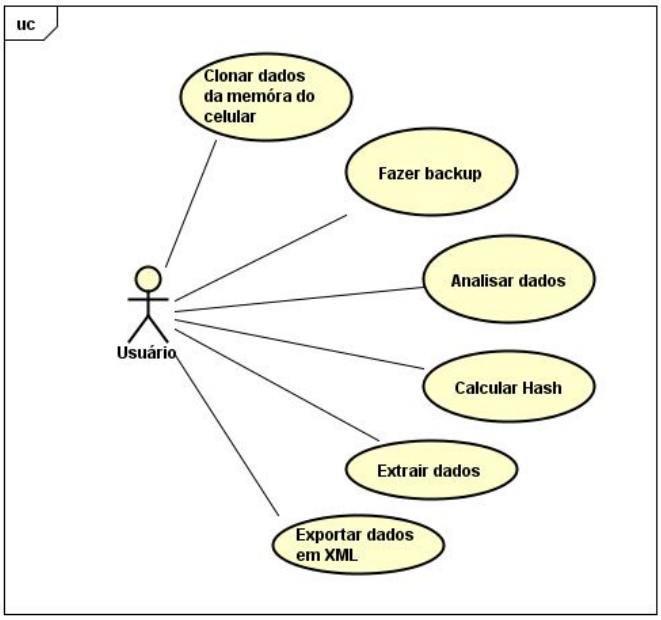

Figura 3: Diagrama de Caso de Uso

As classes da ferramenta proposta nesse trabalho estão dispostas em três camadas: camada FV (*Forensic View*), camada FB (*Forensic Business*) e camada DAO (*Data Access Object*).

A Figura 4 representa o diagrama de classe da ferramenta para melhor entendimento da arquitetura proposta. A seguir será descrito cada uma das camadas, as classes e seus principais aspectos.

7.6.1 Camada de Visão - Forensic View

A camada de visão forense (*Forensic View*) contém os componentes visuais que serão utilizados pelo usuário da ferramenta e será responsável por integrar as demais funcionalidades. A partir desta camada o usuário poderá executar as seguintes ações, que também podem ser visualizadas no diagrama de caso de uso da Figura 3:

- Clonar dados de um dispositivo Android;
- Fazer *backup* dos dados clonados;
- Analisar base de dados SQLite;
- Calcular *hash*;
- Extrair dados de uma base de dados;
- Exportar dados em XML.

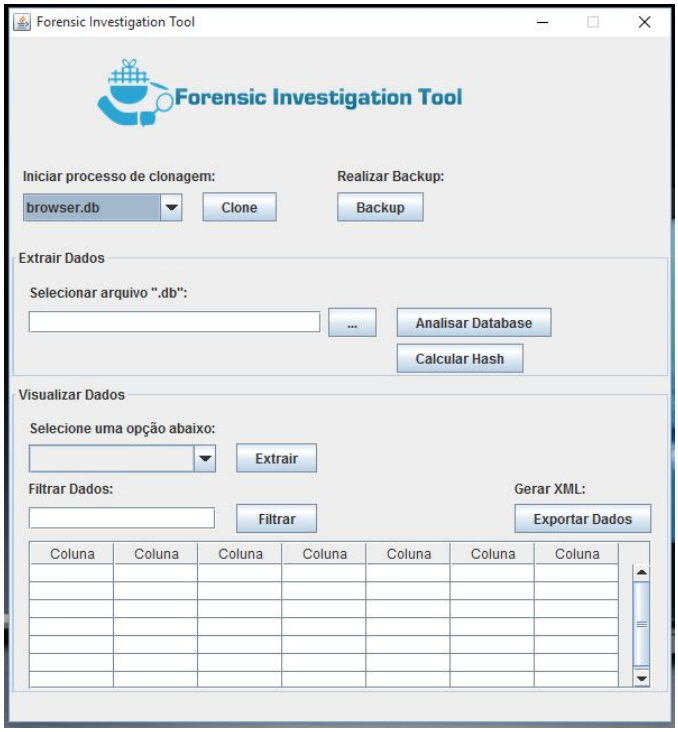

Figura 5: Ferramenta Forensic I.T.

A classe ForensicView, Figura 4, é responsável por dispor o projeto da interface gráfica. A Figura 5 demonstra uma visão da tela da ferramenta para melhor entendimento.

O combobox clone, Figura 5, é composto por seis opções de *shell script* previamente configurados e através dele o usuário da ferramenta seleciona qual das base de dados será extraída do dispositivo periciado. A Figura 6 exibe as opções disponíveis ao usuário.

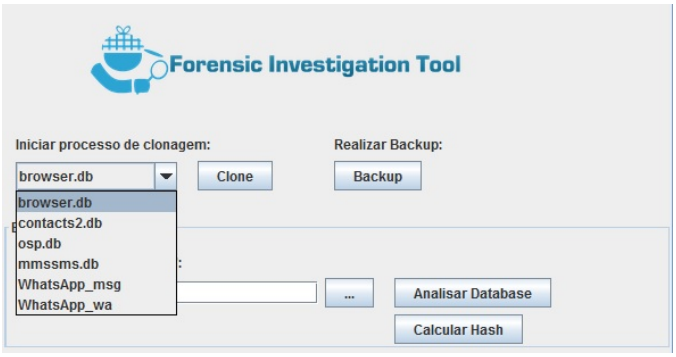

Figura 6: Opções de base de dados

O segundo combobox, localizado ao lado do botão `Extrair`, será preenchido quando o botão Analisar for acionado e exibirá todas as tabelas que compõe a database selecionada para extração, conforme Figura 7.

A tabela é populada quando o usuário seleciona uma das tabelas da base de dados e aciona o botão `Extrair`, como pode ser visto na Figura 8.

Após a tabela ser populada a ferramenta permite ao usuário realizar um filtro nos registros da tabela através do botão

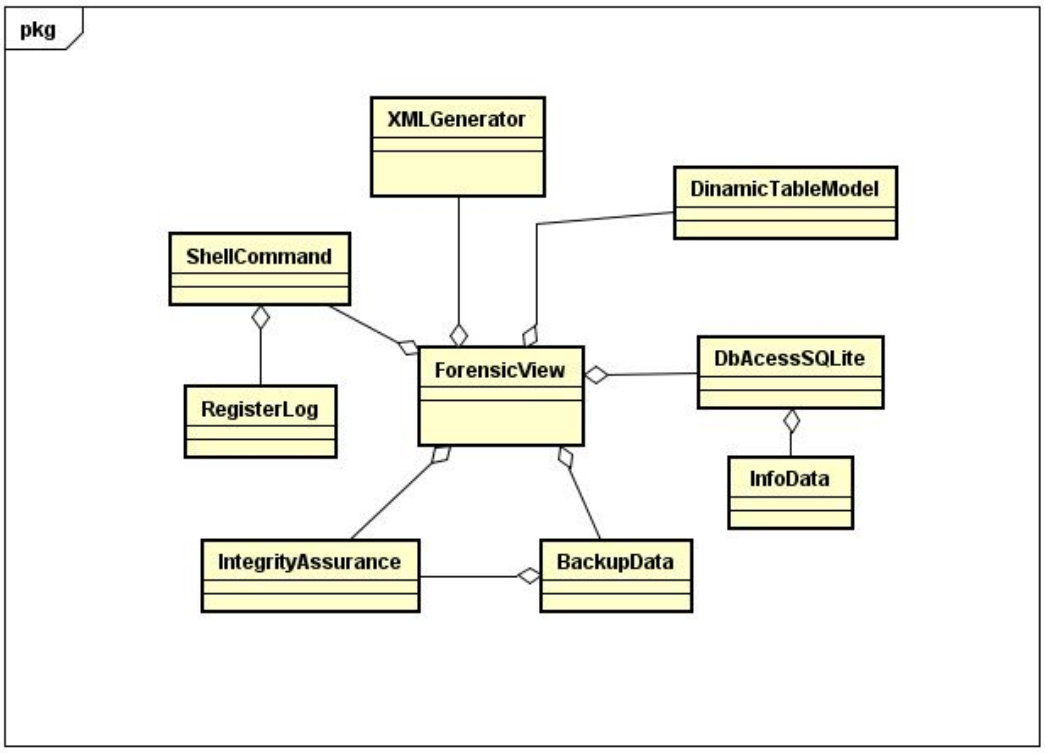

Figura 4: Diagrama de Classe

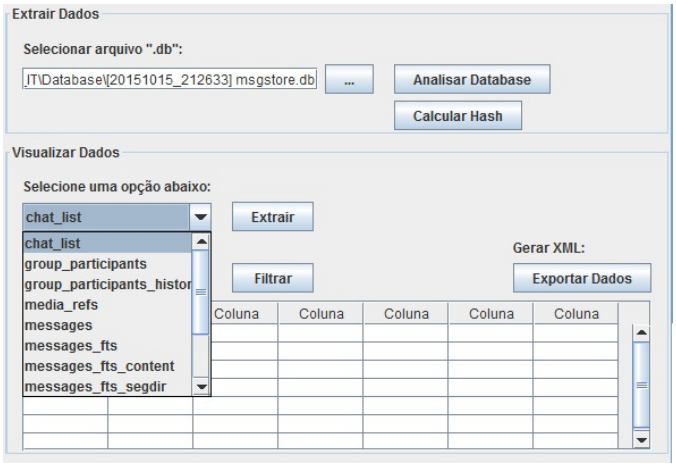

Figura 7: Tabelas da base de dados analisada

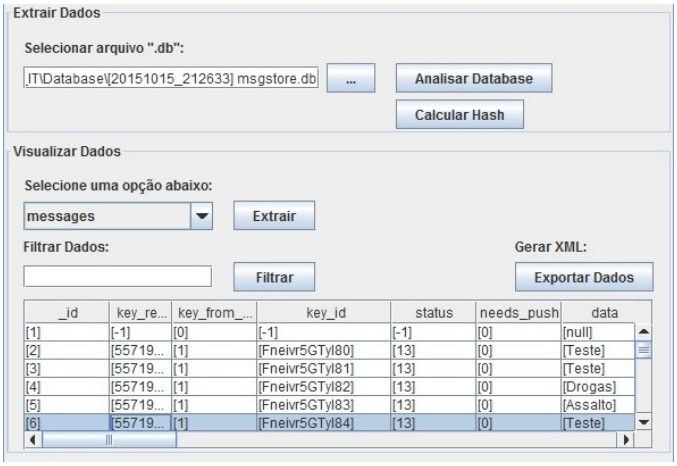

Figura 8: Tabela que apresenta os dados extraídos

`Filtrar`.

7.6.2 *Camada de Negócio - Forensic Business*

A camada de negócio forense (*Forensic Business*) foi construída com foco nas operações relacionadas a técnicas forense, tanto as realizadas no dispositivo periciado quanto a verificação de integridade do arquivo, tais como:

- Clonagem dos dados: reconhece o dispositivo conectado ao computador com acesso via terminal e realiza a clonagem do dados requeridos do dispositivo periciado;
- *Backup* de segurança: é realizada uma copia de segurança do arquivo clonado visando ter um *backup* dos dados;
- Registro de *log*: grava as operações realizadas no terminal de comando como forma de verificação do processo de clonagem e ainda entendimento de possíveis erros;
- Garantia de Integridade: é gerado valor hexadecimal através de um algoritmo de *hash* MD5 para o arquivo clonado ou um arquivo selecionado pelo usuário.

Esta camada é composta pelas seguintes classes: *Shell-*

Command, *BackupData*, *RegisterLog* e *IntegrityAssurance*, responsáveis, respectivamente, pelas ações supracitadas.

A camada Visual (*Forensic View*) passa como parâmetro o *shell script* selecionado para a classe `ShellCommand` que por sua vez é invocada ao acionar o botão `Clonar` da interface gráfica. Esta classe é composta pelo método `ExecCommandAdbShell`, responsável por executar o *script*, que:

- contém os comandos para acessar o `adb shell`;
- acessa o dispositivo periciado;
- realiza a clonagem da base de dados SQLite para o SD card;
- invoca a classe *RegisterLog* para que todos os comandos executados no terminal *shell* sejam gravados em arquivo, assim como todos as saídas do terminal.

Este método utiliza a classe *ProcessBuilder* e *Process* para acessar o terminal e executar o *shell script*. Para obter a saída do terminal, o método utiliza as classes *StringBuilder* e *BufferedReader*, como pode ser visto no trecho de código na Figura 10. A chamada do método termina quando o *shell script* é finalizado, resultando na criação do arquivo clone da base de dados do dispositivo ou em algum erro no processamento do script, em ambos os casos os comandos são registrados em *log*.

```
ProcessBuilder pb = new ProcessBuilder(commands);
pb.directory(new File("C:/Users/Italo/Documents/Forensic_IT/Shell"));
pb.redirectErrorStream(true);
Process process = pb.start();
//Read output
StringBuilder out = new StringBuilder();
BufferedReader br = new BufferedReader(new InputStreamReader(process.getInputStream()));
String line = null, previous = null;
while ((line = br.readLine()) != null)
    if (!line.equals(previous)) {
        previous = line;
        out.append(line).append('\n');
    }
log.writeLogPB(out.toString(), log.formatDate(), commands);
```

Figura 9: Execução do shell script

A Figura 10 representa o Diagrama de Sequência do processo de clonagem para melhor entendimento do fluxo de operações e interações do sistema.

A classe `BackupData` é responsável por realizar duas cópias do arquivo clonado do SD card para uma pasta do sistema. A primeira cópia é realizada para que o arquivo seja utilizado na extração e o perito não precise acessar o SD card, limitando o contato com o dispositivo. A segunda cópia, é um arquivo de segurança que é gerado, caso o arquivo da primeira cópia seja corrompido, sendo assim possível ter os dados originais inalterados.

Esta classe é composta por três métodos: *copyFile*, *backupFile* e *formatDate*. Os dois primeiros executam funções parecidas de cópia de arquivo, diferenciando apenas o local de destino e a nomenclatura dos mesmo. O arquivo de *backup* é nomeado com a data do momento exato da cópia obtido através do método *formatDate*. No final do processo de cópia, um objeto da classe *IntegrityAssurance* é invocado para realizar o cálculo do *hash* do arquivo recém clonado, e por fim adicionar o valor gerado no final do arquivo de *log* referente ao processo de clonagem daquele arquivo.

A Figura 11 representa o Diagrama de Sequência do processo de *backup* para melhor entendimento do fluxo de operações e interações do sistema.

A classe *RegisterLog* é responsável por realizar o registro das operações do *shell* em *log*. Essa classe é composta por dois métodos, o "*writeLog*"e o "*formatDate*". O método *formatDate* é responsável por capturar a data, no formato "YYYYMMdd_HHmmss", no momento que é invocado e converter a data para *String*, para que seja utiliza posteriormente na criação do arquivo de *log*. O *writeLog* é o método que realiza a criação do arquivo de *log*. Esta recebe como parâmetros a saída do terminal *shell*, a data do momento da execução do *shell script* e uma lista com os comandos que foram executados pela classe invocadora. O *log* é gravado em arquivo texto utilizando as classes *FileWriter* e *PrintWriter*.

A Figura 12 representa o *log* gerado após o processo de clonagem e *backup*. As primeiras informações são referentes aos comandos executados no terminal pelo arquivo *shell script*, desde o comando executado até o tempo total do processo e quantidade de dados clonados. As últimas linhas representam o valor do *hash* daquele arquivo específico.

```
Command exec:  sh clone_wa.sh

Result:

List of devices attached
S557020643f2f   device

556+0 records in

556+0 records out

284672 bytes transferred in 0.013 secs (21897846 bytes/sec)

--------------
Hash MD5:
3e649ef8781846de37d29d7fe205a39c
```

Figura 12: Log gerado pela ferramenta

A classe `IntegrityAssurance` é responsável por verificar a integridade dos arquivos periciados, através do algoritmo *hash* MD5. Esta classe é composta por duas assinaturas do método *gerarHashMD5*. Ambas são responsáveis por calcular o hash de um arquivo recebido como parâmetro. O primeiro é chamado pela classe *Backup* e recebe além do arquivo a ser verificado, o *log* do processo de clonagem do mesmo. No fim do calculo, o valor do *hash* é escrito no final do arquivo de *log*, visando prover um maior número de dados sobre o processo para futuras consultas. O segundo é acionado pelo botão `Calcular Hash` e retorna o valor do *hash* para qualquer arquivo selecionado pelo usuário, conforme pode ser visto na Figura 13. Desta forma é possível validar a integridade do arquivo.

7.6.3 Camada de Acesso a Dados - Forensic Data Access

A camada de acesso a dados forense (*Forensic Data Access*) é responsável por gerenciar o acesso e a extração de dados da base de dados SQLite além de fornecer os dados para preencher a *grid* e ainda gerar o arquivo XML. Ela é composta pelas seguintes classes: *DBAcessSQLite*, *InfoData* e *DinamicTableModel*.

`DBAcessSQLite` é a classe onde ocorre o processo de extração dos dados da base de dados SQLite. Esta classe tem um método *extractData* que recebe como parâmetro o diretório onde se encontra o arquivo da base de dados. Este parâ-

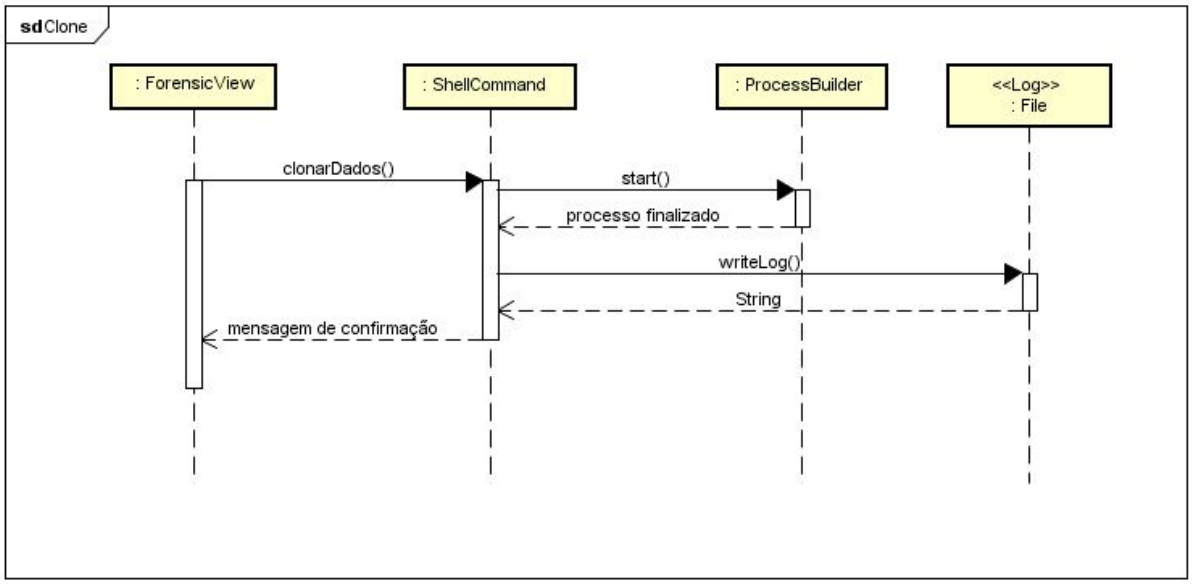

Figura 10: Diagrama de Sequência do processo de clonagem

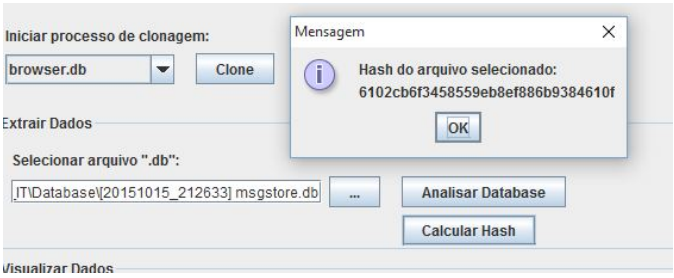

Figura 13: Cálculo do hash

metro é utilizado para que seja iniciado o *driver* do SQLite em tempo de execução e estabelecida a conexão com a base de dados. O método envia uma consulta para a base de dados e salva o retorno como uma lista de *InfoData*, conforme trecho de código da Figura 14. Assim que toda a tabela é percorrida a classe retorna a lista para popular a *grid* com os dados extraídos.

```
int iTimeout = 30;
        ArrayList<InfoData> ListData = new ArrayList<InfoData>();
try {

    PreparedStatement stmt = conn.prepareStatement("SELECT * FROM "+tblName);
    ResultSet res = stmt.executeQuery();
    while (res.next()){
        InfoData data = new InfoData();
        for(int m=1; m <= tblCount ;m++){
        data.setDados(res.getString(m));
        }  ListData.add(data);
    }
```

Figura 14: Consulta na base de dados SQLite

A Figura 15 representa o Diagrama de Sequência do processo de extração de dados para melhor entendimento do fluxo de operações e interações do sistema.

A classe *InfoData* foi criada com o intuito de ser parametrizada com os dados extraídos da base de dados e que serão populados na *grid*. Esta classe foi desenvolvida para receber os valores dinamicamente, sem a limitação da quantidade de variáveis possíveis.

DinamicTableModel é uma classe responsável por definir a estrutura da *grid*. Esta classe será invocada e receberá como parâmetro uma lista de *InfoData* populada com os dados extraídos da base de dados pela classe *DBAcessSQLite*. Em seguida, percorre a lista e exibe os dados na *grid*. Essa classe foi configurada de forma a receber dados de qualquer tabela e apresentar as colunas dinamicamente independente da quantidade de linhas ou colunas da tabela de origem, como pode ser visto no trecho de código do método *getValueAt* na Figura 16.

```
public ArrayList getValueAt(int linha, int coluna) {

    ArrayList<String> ar = new ArrayList<String>();
    InfoData i = (InfoData) linhas.get(linha);
    ar.add(i.getDados().get(coluna));//}
    return ar;
```

Figura 16: Método getValue do DinamicTableModel

A última classe do pacote DAO é o *XMLGenerator*, que é invocada ao acionar o botão `Exportar Dados` da interface, e é responsável por receber uma lista da classe *InfoData* com os mesmos dados que popularam a tabela e gerar um arquivo XML. Para a formatação do arquivo foi utilizado a classe *XStream* da biblioteca *com.thoughtworks.xstream*. Essa biblioteca reduz a quantidade de código necessário para efetuar a formatação do arquivo, é necessário apenas chamar o método *toXML* da classe *XStream* e escrever o retorno desse método no arquivo destino, como mostra a Figura 17.

7.7 Requisitos

7.7.1 Acesso Root

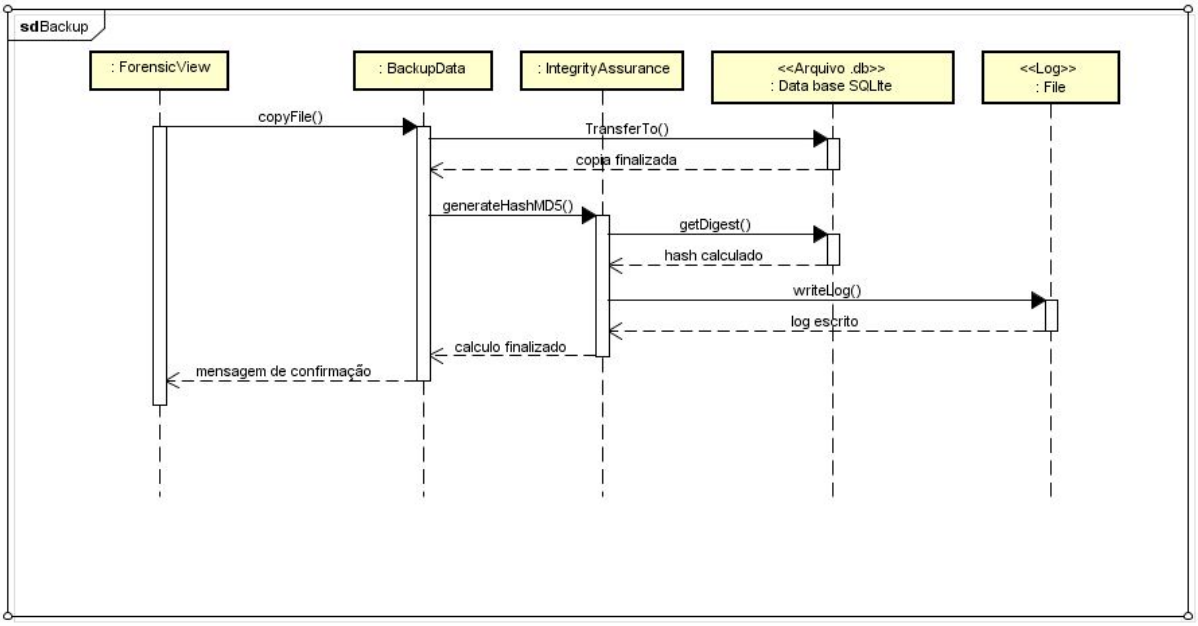

Figura 11: Diagrama de Sequência do processo de backup

```
String dataInXML = xstream.toXML(data);
try{
PrintWriter arquivo = new PrintWriter(new BufferedWriter(new FileWriter(path_xml)));
arquivo.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");  //cabeçalho do xml
arquivo.println(dataInXML);
arquivo.close();
```

Figura 17: Formatação do arquivo XML

A maior parte dos arquivos relevantes para o perito forense em um dispositivo Android, como os arquivos de base de dados, se encontram inicialmente inacessíveis pelo terminal *shell*. Para que o *shell* tenha permissão de acesso a todos os diretórios e arquivos da memória do dispositivo é preciso que seja dado uma permissão de *root*, ou seja uma permissão irrestrita.

Em termos de computação forense em dispositivos móveis, o acesso *root* tem duas grandes vantagens:

- Clonar quase todos os diretórios e arquivos do sistema de armazenamento do dispositivo é muito simples, como os diretórios de `/data` foco deste trabalho;
- Quando se utiliza técnicas de extração física, nem sempre é possível montar alguns sistemas de arquivos adquiridos, tais como YAFFS2. Se o `adb` está sendo executado com permissões de *root*, você pode rapidamente extrair uma cópia lógica do sistema de arquivos.

Porém, na contramão dos benefícios citados, o processo de *rooting*[9] de um dispositivo é complexo e pode comprometer alguns arquivos importantes, além do que existe a possibilidade do dispositivo ser danificado durante o processo a ponto de não poder ser restaurado. Esta questão se torna de certa forma um ponto de maior atenção para os peritos forenses, tendo em vista que a grande maioria dos dispositivos móveis da plataforma Android não tem permissão de acesso *root* e ao mesmo tempo que muitos dados importante para a pericia só podem ser acessados dessa forma.

[9]tornar um usuário *root* padrão

7.7.2 Cabo de dados USB e o modo de depuração

Para o reconhecimento do dispositivo Android periciado pelo `adb` é necessário que o mesmo esteja conectado ao computador. Dentre as opções disponíveis, como acesso sem fio, *bluetooth* ou infravermelho, o uso do cabo de dados USB é a melhor escolha, como já afirmado na Seção 5. Além disso, é necessário que o dispositivo esteja com o modo de depuração USB ativado, o que é feito de forma manual através das configurações do próprio dispositivo.

Esta representa uma das limitações deste trabalho. Em resumo, dispositivo deve estar conectado à ferramenta de extração e deve ter o modo de depuração USB ligado.

7.7.3 SD card

O cartão sd removível é uma peça fundamental para esta ferramenta. O arquivo gerado durante o processo de clonagem do sistema de arquivos em questão é gravado no *SD card*. Está prática é dotada porque caso o arquivo de clonagem seja criado na memória do próprio dispositivo, esse arquivo poderia sobrescrever algum dado relevante para a perícia.

Além disso é fundamental que o SD Card esteja formatado antes do início do processo de clonagem dos dados da memória, para garantir que o arquivo gerado não sofra nenhuma alteração durante a sua alocação.

8. AVALIAÇÃO E TESTES

A ferramenta proposta neste trabalho tem como escopo uma área muito específica da computação, que por sua vez tem um tipo de usuário ainda mais específico - o perito forense. No âmbito nacional o maior número de profissionais com esse perfil e com "know-how" para essa validação trabalham na Polícia Federal. A acessibilidade aos profissionais

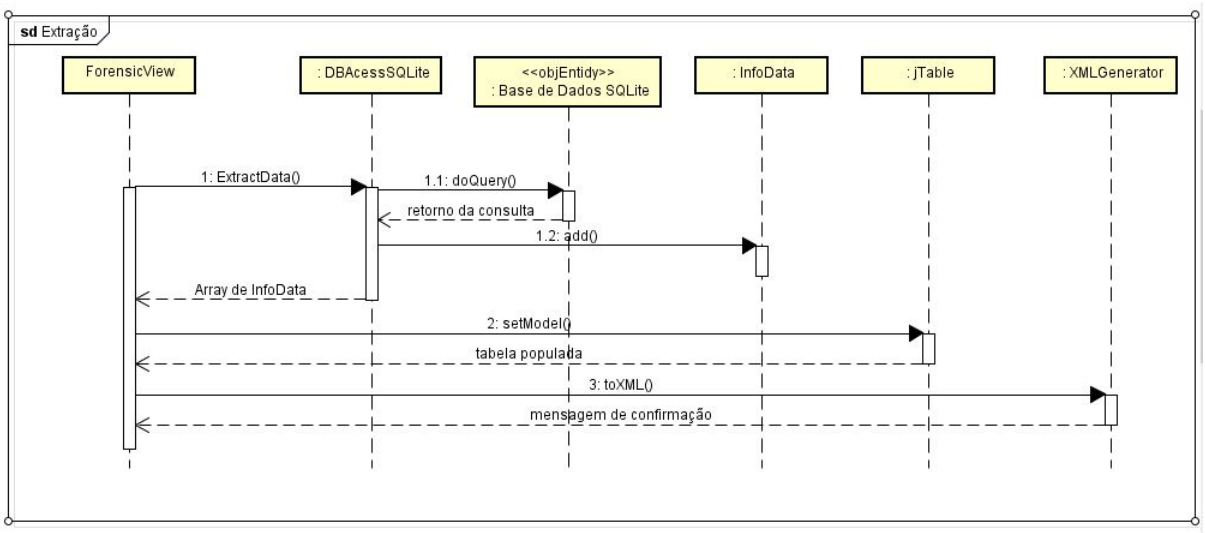

Figura 15: Diagrama de Sequência do processo de extração

deste órgão federal é um tanto quanto reduzida, por conta do número reduzido de atuantes na área e por conta da carga de trabalho dos mesmos. Diante de todas essas questões as análises e validações da ferramenta proposta se tornaram restritas no sentido de quantidade de usuários para teste.

Dessa forma, os testes funcional, de interface e positivo / negativo foram realizados pelo próprio autor do trabalho para avaliar respectivamente as regras de negócio, a navegabilidade e o tratamento para os casos de sucesso e exceção. Por conta da restrição no número de especialistas, consideramos um perito forense digital para fazer uma avaliação da ferramenta. Para proceder com a avaliação da ferramenta, desenvolvemos um questionário de apoio baseado na norma ISO/IEC 9126-1 [15].

Esta norma descreve um modelo de qualidade de software composto de duas partes. A primeira parte especifica seis características para qualidade interna e externa do software. A segunda parte especifica quatro características de qualidade em uso, sendo que qualidade em uso nada mais é do que a combinação das seis características de qualidade do produto. Vale ressaltar que as características definidas na ISO /IEC 9126 são aplicáveis a todo tipo de software.

A norma permite que diversas validações de qualidade, no que diz respeito a software, sejam realizadas. Aqui destacamos o seguinte trecho da norma que define melhor essa questão:

> Esta parte da NBR ISO/IEC 9126 permite que a qualidade do produto de software seja especificada e avaliada em diferentes perspectivas pelos envolvidos com aquisição, requisitos, desenvolvimento, uso, avaliação, apoio, manutenção, garantia de qualidade e auditoria de software. Ela pode, por exemplo, ser utilizada por desenvolvedores, adquirentes, pessoal de garantia de qualidade e avaliadores independentes, particularmente os responsáveis por especificar e avaliar qualidade do produto de software.

A importância da qualidade de software está relacionada a entrega de um produto que satisfaça as reais necessidades do usuário. Vale ressaltar que nem sempre a necessidade explícita do usuário reflete suas necessidade reais em sua totalidade. Além disso, um software de qualidade tem uma fácil usabilidade, corretude no funcionamento e é de fácil manutenção. As visões de qualidade mudam durante o projeto e dessa forma é necessário definir as perspectivas de qualidade a fim de que seja gerenciado de modo apropriado em cada estágio do ciclo de vida do software [7].

Os requisitos de qualidade externa são analisados sobre o ponto de vista das necessidades de qualidade dos usuários, incluindo também os requisitos de qualidade em uso do software. Os requisitos de qualidade interna especificam o nível de qualidade sob o ponto de vista interno do produto. Pode ser incluído modelos estáticos e dinâmicos, documentos e código-fonte [15].

8.1 Resultados

Para os teste da ferramenta foi utilizado um *smartphone* Samsung Galaxy Mini portador de versão 2.3 da plataforma Android. Assim como citado anteriormente, para a avaliação desta ferramenta foi desenvolvido um questionário baseado na ISO/IEC 9126-1 e aplicado a um perito lotado na Superintendência da Policia Federal - Regional Bahia.

O modelo de qualidade de software descrito pela ISO pode ser visto na Figura 18.

A seguir serão apresentados os resultados obtidos da análise do perito em conjunto com comentários do autor.

Entendemos que os testes e a validação são fundamental após o desenvolvimento de uma solução. Entretanto, para a ferramenta descrita aqui neste texto estas etapas não poderiam ser feita por qualquer usuário, uma vez que há termos e processos que só podem ser plenamente avaliados por um usuário que esteja no contexto da área de forense digital. Assim, focamos em uma avaliação, feita por um profissional da área e seguimos com testes não repetidos, realizados junto a um usuário especialista.

Para tanto, foi agendada uma visita a Superintendência da Polícia Federal - Regional Bahia, para avaliação da solução, com o objetivo de avaliarmos:

- adequação da ferramenta;

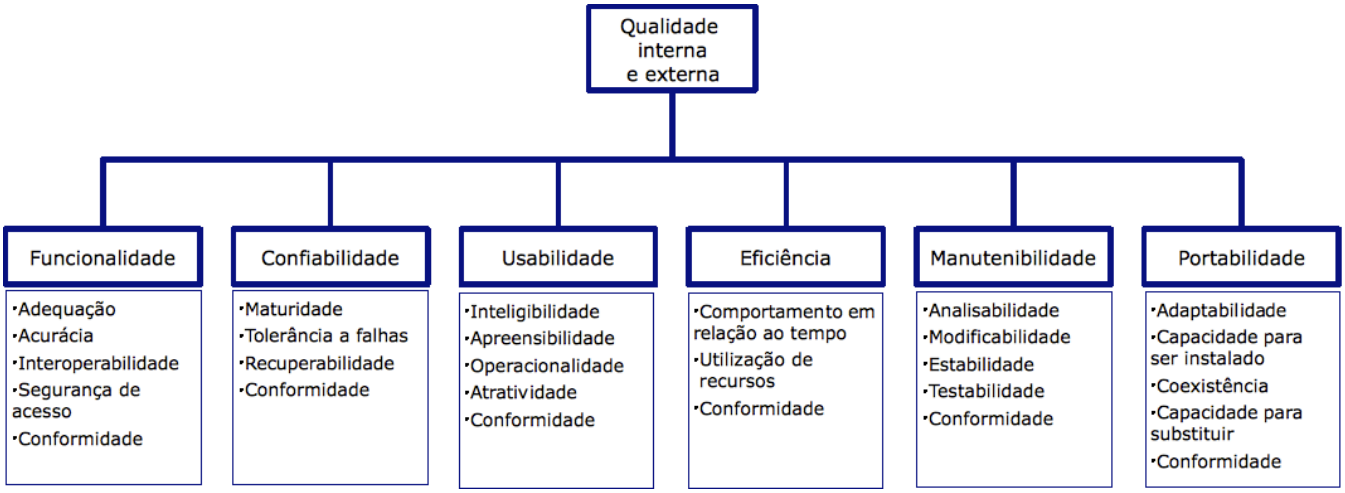

Figura 18: Modelo de Qualidade para qualidade externa e interna

- necessidade de ferramentas forenses;
- diferencial da ferramenta;
- funcionamento geral e específico;
- avaliação de objetivos atendidos para validarmos se o objetivo geral e específico da aplicação foram contemplados.

A apresentação da ferramenta foi seguida de uma avaliação feita pelo perito e por uma entrevista, a fim de contextualizar o perfil do usuário e confrontar os resultados do questionário com os resultados observados através dos testes do usuário especialista.

O questionário é apresentado no Anexo e conta com perguntas que devem ser respondidas com notas de 1 (um) a 5 (cinco), onde 1 representa a menor nota e 5 a maior.

No que diz respeito a característica da `funcionalidade` do software, o mesmo foi avaliado com nota 5 no que diz respeito a dispor ao usuário as funcionalidades que foram propostas, além de ter uma boa interação com o sistema especificado sendo avaliado com nota 4.

Em termos de `confiabilidade` a ferramenta foi avaliada com duas notas 5 para essas subcaracterísticas, tanto na frequência de erros quanto no modo que reage quando os mesmos ocorrem. O que foi percebido é que os erros avaliados (possíveis e mais frequentes, de acordo com a avaliação do perito) são tratados através mensagens de alerta para cada operação. No ponto de visto do autor da ferramenta, pode-se afirmar que a maioria das situações possíveis de falha foram mapeadas e tratadas, evitando que o sistema pare de funcionar ou se comporte de forma indevida. Durante a avaliação junto ao perito, nenhuma situação de erro foi reproduzida. Entretanto, testes de estresse realizado por múltiplos usuários poderiam ajudar neste processo.

Do ponto de vista da `usabilidade` a ferramenta recebeu nota 5 em todos as subcaracterísticas, tanto no que diz respeito ao entendimento do conceito e aplicação da ferramenta, quanto na facilidade de uso da ferramenta, por conta da facilidade de operar a mesma. Este ponto foi um dos focos do projeto, e a parte visual teve uma atenção especial. Durante a avaliação da ferramenta, foi sugerido uma melhoria pelo perito, para que sempre que uma ação for executada, uma mensagem de confirmação seja exibida, informando qual pode ser a próxima ação a ser tomada, tornando o software mais intuitivo. Esta melhoria já foi implementada.

Considerando o requisito de `eficiência`, todas as subcaracterísticas receberam nota 5. Tanto o tempo de resposta quanto a velocidade da execução e o uso dos recursos é bastante otimizado. Nesse ponto vale ressaltar que a velocidade de execução está diretamente ligada ao tamanho do arquivo a ser clonado ou extraído, já que são realizadas operações de cópia de dados e consultas em bases de dados.

No que diz respeito a `manutenibilidade`, a avaliação não pode ser feita pelo perito, já que aspectos de implementação do código java não foram discutidos. Por outro lado, o *script* utilizado foi apresentado e alguns aspectos de implementação discutidos e explicados.

E por fim temos a análise da `portabilidade` da aplicação. Nesse quesito obteve uma boa nota, recebendo um 4 para `adaptabilidade`, já que a ferramenta pode ser implantada tanto em ambiente com uma distribuição Linux quanto num ambiente com sistema operacional Windows. Por conta da compatibilidade mais direta com Linux em termos de estrutura e comandos, a instalação é mais fácil neste ambiente. Para o ambiente com os sistemas operacionais da Microsoft são necessárias algumas configurações adicionais por conta de comandos executados pelo *shell script*, que não são nativos a este ambiente.

De modo geral a ferramenta foi avaliada de forma positiva, tendo em vista a que ela se propõe, tendo boa aceitação por parte do usuário final, o perito forense. Além de pertencer a uma área de poucos incentivos no âmbito nacional do desenvolvimento de *softwares*.

9. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresenta a ferramenta Forensic I.T. (*Forensic Investigation Tool*) de auxílio a perícia forense, que busca ajudar o perito nas fases de extração e dispondo os dados de maneira facilitada à análise do processo pericial. A ferramenta dispõe ao usuário final um processo automatizado de clonagem de dados de um dispositivo móvel da

plataforma Android. Além disso a ferramenta executa operações de verificação de integridade, através de algoritmo *hash*, extração dos dados de qualquer base de dados SQLite, filtragem dos dados exibidos e exportação das informações no formato XML.

A ferramenta apresenta uma interface de fácil entendimento ao usuário, conforme avaliação do usuário, e teve uma boa avaliação no que diz respeito a qualidade de *software*, após apresentado a um usuário especialista na área de forense digital.

Como trabalhos futuros, podemos mencionar inicialmente uma melhoria no processo de clonagem, onde o método de obtenção das bases de dados a serem extraídas seria alterado para que a ferramenta pudesse identificar e listar em tempo de execução todos os arquivos `.db` do dispositivo periciado. Essa melhoria requer um estudo mais aprofundado de *shell script* principalmente, melhorando o formato atual do *script*. Atualmente, essas opções de bases de dados a serem extraídas são fixas em código.

10. REFERÊNCIAS

[1] E. Alecrim. Android supera 80% das vendas de smartphones e windows phone continua avançando. `https://tecnoblog.net/145067/vendas-smartphones-terceiro-trimestre-2013/`.

[2] Android. Android developer tools. `http://developer.android.com/tools/`.

[3] Cellebrite. Home page. `http://www.cellebrite.com`.

[4] S. Chessman. dd. *Linux Journal*, 32, Dezembro 1996.

[5] M. Costa. *Computação Forense - A Analise Forense no Contexto da Resposta a Acidentes Computacionais*. MILLENNIUM EDITORA, 2011.

[6] P. Da Silva Eleuterio and M. Machado. *Desvendando a Computação Forense*. NOVATEC, 2011.

[7] L. dos Anjos and H. de Moura. Um modelo para avaliação de produtos de software. 2004.

[8] B. Elgin. Google buys android for its mobile arsenal. `http://www.bloomberg.com/bw/stories/2005-08-16/google-buys-android-for-its-mobile-arsenal`, Agosto 2005.

[9] F. Falcão. Imei, o que é? para que serve? `"http://www.guiadopc.com.br/artigos/36364/imei-e-serve.html"`, Março 2014.

[10] I. Forense. Historia informática forense. `http://informaticaforense1.blogspot.com.br/2013/11/historia-informatica-forense.html`, 2013.

[11] Forensicswikise. Metadata. `http://forensicswiki.org/wiki/Metadata`, 2013.

[12] H. S. Gomes. Whatsapp é o 4º maior aplicativo da internet móvel do brasil. online, Fevereiro 2015.

[13] A. Hoog. *Android Forensics: Investigation, Analysis and Mobile Security for Google Android*. Android Forensics: Investigation, Analysis, and Mobile Security for Google Android. Elsevier Science, 2011.

[14] W. Inc. Whatsapp messenger. `https://www.whatsapp.com`.

[15] ISO/IEC. *ISO/IEC 9126-1. Engenharia de software - Qualidade de produto*. ISO/IEC, 2003.

[16] Karpersky. Kaspersky security bulletin 2014. overall statistics for 2014. online, 2014.

[17] J. Lessard and G. C. Kessler. Android forensics: Simplifying cell phone examinations. *Small Scale Digital Device Forensics Journal*, 4(1), Setembro 2010.

[18] S. Melo. *Computação Forense com Software Livre*. Alta Books, Rio de Janeiro, RJ, 2009.

[19] C. Osborne. For investigators, a better way to extract data from mobile devices. online, 2012.

[20] R. Rivest. The md5 message-digest algorithm. `https://tools.ietf.org/html/rfc1321`, Abril 1992. Request for Comments: 1321.

[21] A. Rubin. Where's my gphone? `http://googleblog.blogspot.com.br/2007/11/wheres-my-gphone.html`, Novembro 2007.

[22] SQLite. Single-file cross-platform database. `https://www.sqlite.org/onefile.html`.

[23] W. Stalings. *Criptografia e Segurança de Redes. Principios e Praticas*. Prentice-Hall, 4 edition, 2007.

[24] R. VARGAS and C. QUEIROZ. *Investigação e Perícia Forense Computacional*. Brasport, 2010.

ANEXO

Nesta seção apresentamos o questionário aplicado na visita à Superintendência da Polícia Federal Regional Bahia, com o objetivo de validar a ferramenta.

Conforme mencionado na Seção 8, o questionário foi acompanhado e complementado por entrevista, para que os critérios das notas atribuídas pudesse ser avaliado com maior clareza.

As notas foram atribuídas de 1(um) à 5(cinco), onde 1 representa a menor nota e 5 a maior. Optamos por não usar classificações descritivas como "bom", "ruim", "ótimo", por entendermos que estas questões estão cercadas de subjetividade, o que tornaria mais difícil a avaliação das respostas obtidas.

As questões que se referiam ao quesito "Manutenibilidade" foram removidas do questionário, porque apesar de o usuário ter tido acesso ao código fonte, até mesmo para esclarecer eventuais questões adicionais, o objetivo principal não era que o mesmo avaliasse o aspecto de manutenção no código. Além disso, antes da entrevista o perfil do usuário não era conhecido.

A. QUESTÕES

1. Funcionalidade
   (a) Adequação: Propõe-se a fazer o que é apropriado?

1	2	3	4	5

   (b) Acurácia: Faz o que foi proposto de forma correta?

1	2	3	4	5

   (c) Interoperabilidade: Interage com os sistemas especificados?

1	2	3	4	5

   (d) Conformidade: Está de acordo com as normas e leis?

1	2	3	4	5

   (e) Segurança de acesso: Evita acesso não autorizado aos dados?

1	2	3	4	5

2. Confiabilidade
   (a) Maturidade: Com que frequência apresenta falhas?

1	2	3	4	5

(b) Tolerância a falhas: Ocorrendo falhas, como ele reage?

1	2	3	4	5

(c) Recuperabilidade: É capaz de recuperar dados em caso de falha?

1	2	3	4	5

3. Usabilidade

(a) Intelegibilidade: É fácil entender o conceito e a aplicação?

1	2	3	4	5

(b) Apreensibilidade: É fácil aprender a usar?

1	2	3	4	5

(c) Operacionalidade: É fácil de operar e controlar?

1	2	3	4	5

4. Eficiência

(a) Tempo: Qual é o tempo de resposta medido?

1	2	3	4	5

(b) Tempo: Qual a velocidade de execução?

1	2	3	4	5

(c) Recursos: Quanto recurso usa?

1	2	3	4	5

(d) Recursos: Durante quanto tempo usa os recursos?

1	2	3	4	5

5. Portabilidade

(a) Adaptabilidade: É fácil adaptar a outros ambientes?

1	2	3	4	5

(b) Capacidade para ser instalado: É fácil instalar em outros ambientes?

1	2	3	4	5

(c) Conformidade: Está de acordo com padrões de portabilidade?

1	2	3	4	5

(d) Capacidade para substituir: É fácil usar para substituir outro?

1	2	3	4	5