ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

P.PORTO

M

MESTRADO
ENGENHARIA INFORMÁTICA

Framework para Gestão de Dispositivos Móveis: módulo inventário

Gabriel Moisés Semedo Monteiro

Doutor João Paulo Ferreira de Magalhães

AGRADECIMENTOS

Queria retribuir aqui com agradecimentos a todos que fizeram parte desta fase tão importante da minha vida. Foi um longo percurso e muitas vezes dava por mim a pensar nos sentimentos de lembrança e gratidão que iria tentar exprimir quando chegasse este momento. Muito por culpa de todos que de uma forma ou de outra acompanharam-me durante esta caminhada.

Foram muitos amigos e colegas com os quais pude partilhar momentos, ideias e conhecimentos que certamente contribuíram para o enriquecimento deste trabalho. Muitos deles estavam sempre atentos e curiosos com o seu progresso e resultado final. Muito obrigado a todos.

Gostaria de deixar um agradecimento muito especial ao Professor Doutor João Paulo Magalhães, o mentor e orientador deste projeto. Tive a sorte de puder trabalhar consigo como meu orientador na fase de conclusão do 1º ciclo acadêmico e quando surgiu novamente a oportunidade não pensei duas vezes. Muito obrigado pela confiança, pelo constante apoio e por sempre ter acreditado neste projeto.

Agradeço a toda comunidade da Escola Superior de Tecnologia e Gestão, aos colegas e docentes que fizeram parte da minha vida acadêmica. Um agradecimento especial à Professora Doutora Dorabela Gamboa, por todas as palavras de apoio e incentivo, e por se ter mostrado sempre disponível em ajudar.

Por último queria deixar um agradecimento muito importante às pessoas que sempre estiveram comigo e partilharam os meus sacrifícios, alegrias e conquistas, etapas após etapas desta jornada. Começando pelos meus pais, a razão pela qual tudo isso foi possível e que de forma incansável sempre me apoiaram e me encorajaram a continuar. Os meus irmãos e a minha namorada que sempre me deram ânimo para superar, principalmente nas horas mais difíceis. Quero dedicar este trabalho a todos vocês e agradeço do fundo do meu coração por fazerem parte da minha vida.

RESUMO

O número e diversidade de dispositivos móveis utilizados em contexto pessoal e contexto empresarial é um fenómeno em franco crescimento. Este fenómeno conhecido como *Bring Your Own Device* (BYOD) é um dos desafios que as empresas têm de enfrentar. Se por um lado o BYOD promove a satisfação pessoal e a produtividade, por outro, a gestão de dispositivos móveis e a segurança dos dados tornam-se mais complexas.

Existem várias soluções no mercado, apontadas como partes vitais da infraestrutura empresarial, no processo de gestão de dispositivos móveis (*Mobile Device Management* – MDM). Uma vez que tendem a agregar inúmeras funcionalidades, são conhecidos alguns problemas associados a estas soluções, como por exemplo a invasão da privacidade nos dispositivos pessoais e a possibilidade de disrupção com outras soluções em uso nas empresas, para além de outros. Para evitar esta situação propomos a criação de uma *framework* modular para a gestão de dispositivos móveis, com uma abordagem focada na segurança dos dados organizacionais e na separação do contexto *user vs enterprise*.

A *framework* será baseada em Web Services, e o objetivo é o de permitir que uma empresa adote ou integre apenas os serviços necessários, reduzindo o número de aplicações em uso e o custo de propriedade. O primeiro módulo a implementar é o de inventariação de dispositivos móveis.

Palavras-chave: *Bring Your Own Device; Gestão de Dispositivos Móveis; Segurança de dados; Sistemas Legacy; It Managers; Web Services; Base de dados.*

ABSTRACT

The number of personal mobile devices used inside company's network continues to grow. This phenomenon known as Bring Your Own Device (BYOD) is one of the challenges the companies have to face sooner or later. In one hand, BYOD improve staff satisfaction and productivity, in the other, the mobile devices management and data security get more complex.

There are several solutions in the market pointed as vital part of enterprises infrastructure for mobile devices management (MDM), however, some issues arise with these solutions. Privacy invasion in personal devices, and disruption with legacy systems due to several functionalities aggregation is some of them. To avoid this problem, we propose a modular framework for mobile devices management, with an approach focused in organizations data security and users versus enterprises context separation.

This will be a Web Service based framework, and the goal is to allow that companies can use or adopt only the necessary services, cutting unnecessary applications and reducing costs of ownership. The first module to develop is an inventory system for mobile devices.

Keywords: *Bring Your Own Device; Mobile Device Management; Data Security; Legacy Systems; IT Managers; Web Services; Database systems.*

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ACRÓNIMOS

API – *Application Program Interface*

CPU – *Central Processing Unit*

GDPR – *General Data Protection Regulamentation*

GPRS – *General Packet Radio Service*

HD – *Hard Drive*

HTTP – *Hypertext Transfer Protocol*

IDE – *Integrated Development Environment*

IoT – *Internet of Things*

IP – *Internet Protocol*

IT – *Information Technology*

JSON – *JavaScript Object Notation*

MDM – *Mobile Device Management*

NAC – *Network Access Control*

RAM – *Random Access Memory*

REST – *Representative State Transfer*

SaaS – *Software-as-a-Service*

SDK – *Software Development Kit*

SSID – *Service Set Identifier*

SO – Sistema Operativo

SOAP – *Simple Object Access Protocol*

URL – *Universal Resource Locator*

VoIP – *Voice over Internet Protocol*

VPN – *Virtual Private Network*

WS – Web Services

XML – *Extensible Markup Language*

1. INTRODUÇÃO

O aumento deslumbrante da adoção de dispositivos móveis excede toda a tecnologia que até então tinha sido disponibilizada ao público em geral. Frequentemente são apresentados novos modelos no mercado, muitos dotados com funcionalidades e características que superam versões anteriores. De acordo com o relatório sobre o tráfego de dados móveis da Cisco (Sports Video Group, 2017), para o ano de 2020 os utilizadores de telemóveis atingirão 5.5 biliões, representado cerca de 70% da população mundial. Prevê-se ainda, que o número de *smartphones* atingirá os 72% do total dos dispositivos móveis globais. Um crescimento significativo quando comparado com o ano de 2015, onde este valor era de 36%. Esta afluência rapidamente começou a ter impactos na vida quotidiana das organizações e se recuarmos alguns anos atrás, podemos ver que vários estudos já davam indicações da "avalanche" de dispositivos móveis que iriam entrar porta adentro nas organizações. Um estudo da ABI Research publicado em (ABI Research, 2012) indica que no ano de 2017 haveria cerca de 2.4 biliões de funcionários a usarem os seus *smartphones* no ceio das empresas. Este estudo previa uma taxa de crescimento de cerca de 17%, que representava quase três vezes mais *smartphones* do que os funcionários utilizavam na altura.

Outro estudo da Juniper Networks publicado em (Rose, 2013) indica, após inquérito a 4000 utilizadores de dispositivos móveis e profissionais, que 41% dos utilizadores usavam os seus dispositivos móveis para fins profissionais (*Bring Your Own Device* – BYOD) sem permissão da empresa.

Estudos mais recentes, como o feito pela empresa Gartner (Gartner, 2013), revelam que 38% das organizações tencionavam deixar de fornecer dispositivos móveis aos seus trabalhadores (*Corporate Owned Personally Enabled* – COPE). Este mesmo estudo previa que durante o ano de 2017, cerca de metade das organizações iriam exigir aos seus funcionários que usassem os seus próprios dispositivos dentro das mesmas. Tratando-se do BYOD, o padrão de escolhas adotadas pelas companhias revela que este fenómeno poderá deixar de ser algo que até então era preciso/necessário, para passar a ser obrigatório.

Se por um lado estudos dizem que BYOD está associado a uma melhoria da satisfação dos indivíduos levando a um aumento de produtividade e a uma poupança por parte das organizações, por outro são referidos os elementos de distração e de acréscimo de risco. Em todo o caso, e sendo algo difícil de travar é de senso comum que a melhor forma de lidar com o fenómeno é através da gestão eficiente dos dispositivos móveis. Assim, e de forma a apoiar as empresas na gestão e controlo destes dispositivos, existem no mercado várias

abordagens/soluções de gestão de dispositivos móveis (MDM) que tentam minimizar o impacto e os riscos associados ao BYOD. A maiora das soluções refere uma facilidade de integração e operacionalização que nem sempre se verifica. Estas acabam por impor restrições na utilização dos dispositivos móveis que são pertença dos funcionários. Essas restrições passam por exemplo pela seleção e restrição das aplicações que podem ser usadas até à criação de ambientes paralelos no mesmo dispositivo obrigando o utilizador a selecionar o ambiente em que se encontra (e.g., pessoal ou profissional). Estas restrições levam-nos a dizer que estas soluções acabam por ser disruptivas, e que dificultam a integração e uso. Esta opinião faz-nos levantar um conjunto de questões: e se as abordagens seguidas até então por estas soluções não forem as melhores a seguir? E se uma empresa precisar de colmatar somente um problema específico? Fará sentido adotar um sistema que agrega várias funcionalidades? A nível de custos, quais serão as consequências? Faz sentido que um funcionário seja "obrigado" a cumprir políticas que põem em causa a usabilidade do seu telemóvel e a privacidade dos seus dados num contexto extraorganizacional?

Com base nestas questões, o objetivo deste trabalho é especificar uma solução dirigida às organizações e que seja capaz de endereçar necessidades como as de inventário, monitorização, configuração, ativação remota de controlos e gestão de aplicações. Tal aplicação deverá estar provida de mecanismos automáticos, que lhe permitam de forma autonoma fazer a gestão dos dispositivos móveis em função do contexto. A título de exemplo, e para facilitar a compreensão dos objetivos, pretende-se que a solução seja capaz de perceber o contexto em que o dispositivo se encontra (e.g. casa, local de trabalho, local público) e em função disso adotar medidas de gestão do dispositivo tais como: que aplicações são possíveis de executar, que tratamento será dados aos dados enviados e recebidos pelo dispositivo, entre outros.

A solução será baseada em *Web Services* mapeados em diversos módulos, que de forma combinada serão capazes de automatizar o funcionamento do sistema de gestão de dispositivos móveis e colmatar as necessidades conforme o ambiente onde se encontra o dispositivo. Considerando a quantidade de trabalho subjacente à implementação da solução, e para efeitos de prova de conceito, neste projeto será implementado o módulo de inventário. Este módulo tem como propósito demonstrar:

- A facilidade de integração com sistemas existentes – através da utilização de *Web Services* capazes de funcionar como *middleware*, para que de forma transparente seja possível fazer a ponte entre os dispositivos móveis e os sistemas *legacy*;

- A possibilidade de o dispositivo perceber o seu contexto e atuar em conformidade com este. Por exemplo, o dispositivo quando ligado à rede empresarial poderá enviar dados para o servidor central e quando ligado a uma rede externa não atuará ou adequará o tipo de dados a enviar;
- A obtenção de relatórios globais – através da recolha e do armazenamento constante de um conjunto de informações relevantes sobre cada dispositivo que se encontrar em funcionamento dentro da rede empresarial;
- O consumo adequado dos recursos existentes nos dispositivos (ex: RAM, *hard drive*, CPU). Este acaba por ser um aspecto importantes uma vez que se trata de dispositivos em que por vezes os recursos são limitativos.

1.1. Estrutura do Relatório

Este relatório esta dividido em capítulos. O capítulo 2 apresenta uma análise comparativa das soluções de gestão de dispositivos móveis existentes no mercado, indicando as suas características e as suas limitações. No capítulo 3 é apresentada a proposta de criação de uma solução de gestão dos dispositivos, que segue a abordagem necessária para solucionar as limitações encontradas nas soluções existentes. No capítulo número 4 são apresentadas as tecnologias e ferramentas utilizadas na prova de conceito, e as razões que ditaram estas escolhas. O quinto capítulo apresenta os passos associados à implementação do módulo de inventário que funciona como prova de conceito desta solução. No capítulo 6 é feito uma avaliação do sistema implementado e os resultados obtidos. Finalmente, no capítulo 7 é feito um apanhado sobre o trabalho desenvolvido, os resultados alcançados e algumas dificuldades encontradas durante a sua elaboração. No final, são apresentadas algumas ideias e melhorias que possam ditar os caminhos a seguir em termos de trabalho futuro.

2. ESTADO DA ARTE

Os termos *segurança da informação* e *segurança informática* são frequentemente utilizados nos mesmos contextos, acabando por se sobreporem; por vezes, torna-se difícil distingui-los. Estes dois termos partilham objetivos comuns no que diz respeito à proteção da confidencialidade, integridade e disponibilidade da informação (Y. & Kim, 2007). Para ajudar a perceber as suas diferenças é preciso focarmos nas metodologias e áreas que cada um atua. O primeiro preocupa-se com a prevenção do acesso, utilização, modificação e divulgação não autorizada dos dados, independentemente do estado ou formato em que se encontram (ex.: papel, digital, etc.). O segundo por sua vez garante a disponibilidade e o correto funcionamento do sistema informático sem se preocupar com a informação que é processada ou armazenada pelos computadores.

Há vários anos que os profissionais defendem que a confidencialidade, integridade e disponibilidade são os princípios básicos para a segurança da informação e ainda hoje este cenário mantêm-se inalterado. Com a competitividade crescente dos mercados, a informação tornou-se cada vez mais num dos patrimónios mais importantes para qualquer organização que se queira afirmar ou estar passos à frente da concorrência.

Confidencialidade

O termo confidencialidade é utilizado na prevenção do acesso à informação por parte de pessoas ou sistemas que não têm autorização para o fazer. A autorização de acesso é normalmente concedida pelo proprietário ou dono da informação. Os mecanismos principais de proteção da confidencialidade nos sistemas de informação são a criptografia e o controlo de acesso. Por outro lado, a confidencialidade poderá estar comprometida quando se trata de sistemas mal administrados, ou redes de comunicação inseguras onde possam existir intrusos e *softwares* maliciosos (*malwares*) a operar (Y. & Kim, 2007).

Integridade

A integridade diz respeito à originalidade ou qualidade da informação. Significa que esta não pode ser modificada sem a autorização ou sem o devido conhecimento do proprietário/dono da informação. No contexto de segurança de informação, este termo refere-se não só à integridade da informação em si, mas também à origem da informação. Os mecanismos de proteção da integridade podem ser agrupados em dois tipos: mecanismos de prevenção e

mecanismos de deteção. Como a própria palavra diz, o primeiro tem como objetivo prevenir que a informação seja alterada sem o conhecimento ou consentimento prévio; e o segundo tem como objetivo detetar esta alteração, quando o primeiro mecanismo falhar (Y. & Kim, 2007).

Disponibilidade

Quando a disponibilidade da informação é afetada, a confidencialidade e a integridade perdem relevância. Ou seja, se no final as entidades envolvidas não conseguem ter acesso à informação a questão da confidencialidade e integridade perde expressão. Para isso, é necessário que o sistema de informação utilizado para guardar/processar a informação, bem como os canais de acesso estejam a funcionar corretamente e se mantenham sempre disponíveis. Existem vários fatores que podem afetar a disponibilidade. Um dos mais relevantes e frequentes é o ataque informático conhecido como *denial-of-service* (DoS) (Qadir & Quadri, 2016).

A Figura 1 ilustra os componentes da segurança de informação conhecido como CIA tríade (*Confidentiality, Integrityand, Availability*).

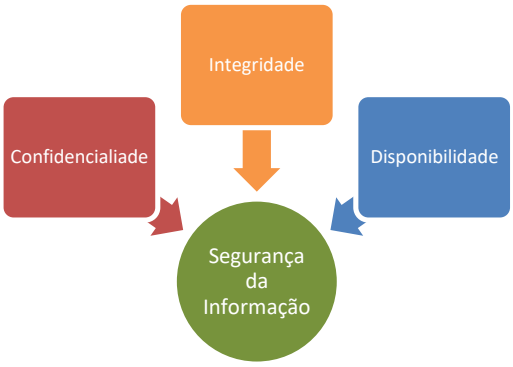


Figura 1 - CIA tríade

Existem ainda outros elementos defendidos por autores como (Y. & Kim, 2007) como sendo atributos importantes a ter em conta na segurança da informação. Entre estes atributos temos: Autenticidade e o não repúdio.

Autenticidade

Com os sistemas de informação a cada vez mais suportar o funcionamento dos negócios é extremamente importante que as entidades/sistemas envolvidas no acesso à informação sejam

quem verdadeiramente dizem ser. Durante o processo de autenticação (onde é feita a identificação e posterior autorização) para o acesso à informação, é necessário validar com elevado grau de certeza que a entidade declarada na fase de identificação pertence a uma das partes que pode ter acesso à informação.

Não Repúdio

O não repúdio assegura que ambas as partes envolvidas na troca de informação não possam negar o facto de terem enviado ou recebido a informação. Em ambientes de comércio eletrónico as assinaturas e certificados digitais são exemplos de tecnologias utilizadas para garantir o não repúdio da informação.

2.1. Mobilidade e Desafios à Segurança Informática

Mais do que uma simples tendência, o BYOD já é considerado um requeristo nos modelos de negócios atuais. A maioria das empresas reconhece no BYOD vantagens como a satisfação do *staff*, diminuição dos custos, aumento da motivação, aumento da comunicação/colaboração e consequentemente aumento da produtividade. Independentemente da posição das empresas, a utilização dos dispositivos móveis em contextos organizacionais é uma realidade. Esta adesão em massa é conhecida no meio informático como consumerização de IT e tem-se revelado um verdadeiro desafio para os responsáveis de IT e equipas de segurança informática. Com o BYOD os profissionais de IT passam a ter uma diversidade de dispositivos para gerir informações sensíveis da empresa passam a estar dispersos por dispositivos vulneráveis à perda, extravio ou ataques informáticos.

Associado a estes desafios e ao crescente fenómeno do *Internet of Things* (IoT), aumenta a procura pelos sistemas *Network Access Control* (NAC) para solucionar as necessidades dentro das redes organizacionais (Black Box Network Services, 2010). Estas soluções permitem reforçar a segurança de uma rede de dados restringindo o acesso apenas a dispositivos que cumprem as políticas de segurança definidas. De uma forma geral, estes sistemas implementam funcionalidades como: controlo de acesso, gestão de políticas da empresa, serviços de rede *guest*, restrição de dados por utilizador, sistemas *anti-ameaça* (*firewall*, antivírus).

Os sistemas NAC são ideais para empresas que necessitam de controlos dentro das redes organizacionais. Contudo, alguns administradores de IT continuam a levantar questões

sobre a praticabilidade de soluções NAC em cenários de rede que estão em constantes mudanças, acedidas por um grande número de utilizadores e na qual a diversidade de dispositivos móveis é grande, tal como acontece com o BYOD. De acordo com (Sobers, s.d.), uma das limitações destes sistemas é que são focados somente na autenticação dos utilizadores e no controle de acesso. Não possuem funções para detetar ou responder a comportamentos anormais de dispositivos quando estes já se encontram dentro da rede organizacional.

Nos cenários onde os perímetros de segurança tendem a ser incertos, as soluções de segurança que funcionam de forma centralizada não são as mais adequadas. Desta feita, as metodologias de *Endpoint Security* ganham cada vez mais importância pois complementam as soluções centralizadas com camadas adicionais de proteção para todos os *endpoints* – servidores, *desktops*, portáteis, *smartphones*, e outros dispositivos da chamada IoT. Além de pontos de saída de dados sensíveis (corporativos e pessoais), estes dispositivos são também possíveis pontos de entrada de ameaças e ataques à rede organizacional. Por norma, as soluções de *Endpoint Security* funcionam num modo cliente – servidor. Em cada *endpoint* é instalado uma aplicação cliente, e do lado do servidor existe uma aplicação de segurança responsável pela autenticação e atualização da aplicação cliente, quando necessário. Muitas destas soluções funcionam em modo *Software-as-a-Service* (SaaS) (Rouse, 2015), onde tanto a aplicação de segurança como o servidor são mantidos remotamente pela empresa. Seja qual a alternativa implementada, (Waltermire & Harrington, 2015) dizem que a avaliação da postura do *endpoint* geralmente inclui:

- Recolha de informações de um determinado *endpoint*;
- Disponibilização destas informações em repositórios de dados apropriados para avaliação;
- Verificação se a postura do *endpoint* está em conformidade com os padrões e políticas da organização.

Assente nos problemas da mobilidade e na necessidade de os controlar, independentemente das metodologias ou soluções adotadas, é reconhecida a importância de uma gestão eficiente de dispositivos móveis (*Enterprise Mobility Management*). Esta gestão engloba os aspetos relacionados com a segurança, gestão dos dispositivos (*Mobile Device Management*), gestão das aplicações (*Mobile Application Management*), controlo de custos e integração de soluções alinhadas com a infraestrutura IT existente na empresa.Os

administradores de IT precisam de ser proactivos na gestão de riscos em tempo real, em ambientes onde o objetivo é poder garantir um certo nível de segurança, conciliado à capacidade de os trabalhadores tirarem o máximo proveito dos seus dispositivos para criarem valores dentro das organizações.

2.2. Sistemas de MDM

Com o objetivo de dar respostas a estas questões e atacar diretamenteos os desafios do BYOD, surgiram no mercado várias soluções de gestão de dispositivos móveis denominadas por *Mobile Device Management* - MDM.

A evolução das soluções MDM é caracterizada, segundo a Zenprise (Zenprise, 2012) em duas fases:

- MDM 1.0 (*Say "Yes" to Mobile*) – onde o objetivo é fornecer a gestão do ciclo de vida dos dispositivos, efectuando tarefas básicas como: aprovisionamento, bloqueio e *wipe* ("apagar dados") dos dispositivos;
- MDM 2.0 (*Put Mobile to Work*) – é a fase seguinte, onde o objectivo é tirar partido das potencialidades dos dispositivos móveis, de forma a fazê-los trabalhar/criar valor dentro das organizações.

Independentemente das fases, estas soluções permitem de forma geral: definir políticas de segurança; fazer o inventário (*hardware*, *software*) dos dispositivos; fazer configurações (rede wireless, VPN); fazer a gestão das aplicações dos dispositivos (Mobile Applications Management – MAM); fazer a gestão de conteúdo dos dispositivos (Mobile Content Management – MCM); fazer o *wipe* (total ou *enterprise*) ao dispositivo; enviar notificações para os dispositivos; despoletar alertas sobre o não cumprimento de políticas.

A Gartner tem apresentado anualmente um estudo (nos seus *Magic Quadrants*), em que classifica numa matriz as empresas líderes do mercado em soluções MDM. A classificação é feita quanto à "capacidade de satisfazer as necessidades atuais" e "abrangência/plenitude da visão" (Gartner, 2017). De acordo com a Gartner, em (Redman, Girard, & Wallin, 2011), uma boa solução MDM deve fornecer uma linha base comum entre diferentes plataformas para definir, conter, validar, aplicar e atualizar políticas para *gateways*,VPNs, *proxies*, NACs, certificados, conteúdos, controlo de versões, *backups*, *updates*, *wipe* e inúmeras outras áreas que entram no espaço dos dispositivos móveis.

Sem dúvida que uma das características a ter em conta na escolha de uma solução MDM deve ser a variedade de dispositivos/plataformas que suporta. E neste caso, deve pelo menos garantir o suporte a dispositivos com sistemas Android e iOS, que são os líderes do mercado (Statcounter, 2018). Com base nesta característica selecionou-se no *MagicQuadrant*, cinco soluções que tentam cobrir o maior número de dispositivos diferentes, incluindo também plataformas *desktop*/*laptop*, que na nossa perspetiva é importante ter em conta. Desta seleção analisaram-se quais os sistemas que funcionam em modo *Software-as-a-Service* e *cloud-based*, embora algumas disponibilizam versões que possam ser instaladas localmente. Pudemos também notar que a maioria fornece as funcionalidades referidas como importantes numa solução de MDM, mas nenhum deles tem a especificidade de funcionamento por módulos, ou seja, agregam um conjunto de funcionalidades num único produto. O resultado desta análise é apresentado nas Tabelas 1, 2 e 3 e tem como origem o estudo apresentado em (PC Magazine, 2018).

Vendors	UserSelf-Registration	RemoteLock	RemoteWipe	Granular Selective Wipe
VMwareAirWatch	✓	✓	✓	✓
CitrixXenMobile	✓	✓	✓	✓
IBM MaaS360	✓	✓	✓	✓
SOTI MobiControl	✓	✓	✓	✓
Microsoft Intune	✓	✓	✓	✓

Tabela 1 - Soluções de MDM visionárias e líderes de mercado (parte 1)

Vendors	Single Sign-On (SSO) for AllApps	Geofencing	Knox Support	Mobile Expense Management (MEM)
VMwareAirWatch	✓	✓	✓	✓
CitrixXenMobile	✓	✓	✓	✘
IBM MaaS360	✓	✓	✓	✓

SOTI MobiControl	✓	✓	✓	✖
Microsoft Intune	✓	✓	✓	✖

Tabela 2 - soluções de MDM visionárias e líderes de mercado (parte 2)

Vendors	Works ByModule	Mobile PlatformSupport	Desktop/Laptop Support	Price (per device, per month)
VMwareAirWatch	✖	Android, iOS, Windows 10 Mobile	macOS, Windows 10	± 3,67 €
CitrixXenMobile	✖	Android, BlackBerry, iOS, Windows 10 Mobile	macOS, Windows 10	± 1,91 €
IBM MaaS360	✖	Android, BlackBerry, iOS, Windows 10 Mobile	macOS, Windows 10	± 3,39 €
SOTI MobiControl	✖	Android, BlackBerry, iOS, Windows 10 Mobile	macOS, Linux, Windows 10	± 3,39 €
Microsoft Intune	✖	Android, iOS, Windows 10 Mobile	macOS, Windows 10	± 5,09 €

Tabela 3 - soluções de MDM visionárias e líderes de mercado (parte 3)

As funcionalidades disponibilizadas pelas soluções de MDM são relevantes para o fenómeno BYOD, mas é reconhecido no meio a existência de algumas limitações. Uma das limitações mais relevantes é o facto de não existir a devida separação do contexto *user versus enterprise*. Isto é, quem utiliza o seu dispositivo no âmbito do MDM fica sujeito às regras aplicadas pelas organizações, o que muitas vezes não faz sentido e levanta aspetos de foro legal. Outra limitação é que uma vez aplicados os perfis definidos nas soluções de MDM, o

dispositivo fica sempre limitado a esses perfis, afetando as questões de usabilidade. Por último, a capacidade de fazer *wipe* ou *lock* remoto nem sempre funciona quando os dispositivos estão “fora da rede” empresarial.

3. SOLUÇÃO PARA GESTÃO DE DISPOSITIVOS MÓVEIS – PROPOSTA

Na Figura 2 é ilustrado o *framework* que propomos no âmbito deste trabalho. Esta é uma solução dirigida às organizações e que consiste num conjunto de módulos que contém a lógica e o tratamento de dados resultantes da interação entre os dispositivos móveis e as aplicações de inventário, monitorização, gestão de aplicações, gestão de configurações entre outras. O objetivo é criar uma camada transparente aos dispositivos móveis e às aplicações eventualmente existentes nas empresas, de forma a cobrir necessidades de gestão dos dispositivos, levados a cabo pelo fenómeno BYOD dentro de uma organização.

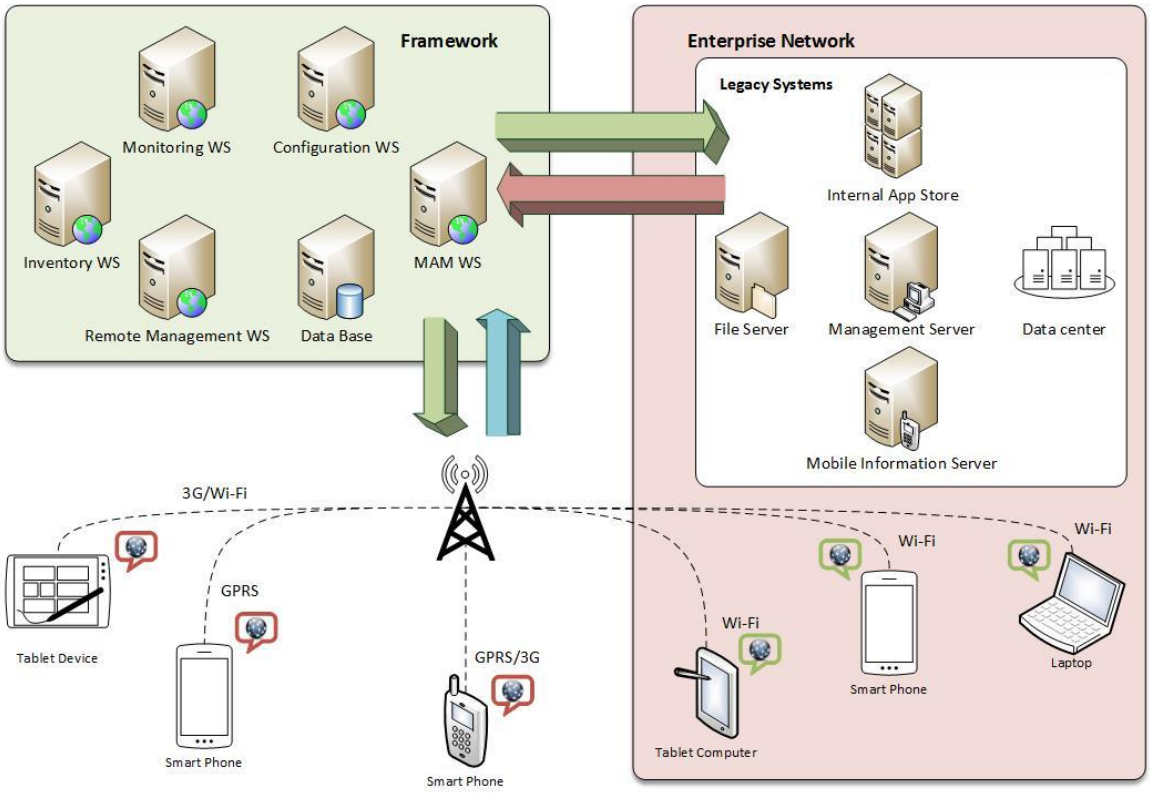


Figura 2 – Diagrama geral da Framework proposta

A funcionalidade delegada a cada módulo resulta de um processo de comunicação entre dois elementos. O primeiro trata-se de um "agente" que é instalado nos dispositivos e que dependendo dos casos, será o responsável pela recolha de dados, aplicação de configurações, instalação de aplicações; o segundo elemento é um *Web Service* alojado na *Cloud*, encarregue de receber e/ou enviar notificações para o agente, fazer o tratamento de

dados recebidos e a sua integração com a infraestrutura existente nas empresas. Desta forma cada módulo terá a liberdade de funcionar de forma singular, com responsabilidades e objetivos próprios no processo de gestão dos dispositivos móveis. A solução terá aplicação a diferentes tipos de dispositivos desde *smartphone*, *tablets*, *laptops*, e será independente da marca e sistema operativo que neles opera.

Proteção vs Privacidade

A separação do contexto *user vs enterprise* não é um processo tão linear quanto parece. Tratando-se de dispositivos pessoais, por vezes torna-se num processo complexo distinguir quando são utilizados em contexto pessoal e contexto empresarial. O ideal seria uma abordagem que fosse capaz de preservar a privacidade dos funcionários e ao mesmo tempo protegesse os dados da organização. As abordagens seguidas até então pelas soluções MDM são constantemente questionadas pela comunidade, uma vez que põe em causa a privacidade dos funcionários, com posturas "agressivas" na gestão dos dispositivos.

As empresas devem centrar os esforços na criação de mecanismos de acesso e proteção aos dados de forma bidirecional, ou seja, todos os dados recolhidos nos dispositivos devem ser protegidos e unicamente acedidos por pessoal autorizado, e ao mesmo tempo, toda a informação caracterizada com sendo património da organização deve permanecer em segurança nos dispositivos pessoais dos funcionários independentemente onde estes estiverem.

Para endereçar estes problemas a solução que propomos define dois pontos essenciais e complementares:

- **Separação de contexto:** A rede empresarial representa a fronteira que define o contexto de utilização onde poderá existir a interação/comunicação com os dispositivos móveis. Deste modo a monitorização, recolha de dados, configurações e outras ações só serão feitas quando o dispositivo se encontrar ligado à rede (ou redes) para a qual o utilizador aceitou a recolha. Quer isto dizer que, a rede é a peça chave que determina o contexto em que poderá existir a interação com os dispositivos móveis. Qualquer outra rede é considera propriedade alheia no qual deverão ser respeitadas mantendo a privacidade dos utilizadores.

- **Segurança implícita:** Os mecanismos de segurança devem estar implícitos na comunicação de dados feita entre os dispositivos, a *framework* e os sistemas *legacy* existentes nas empresas. Para isso, será utilizado algoritmos de cifra assegurando que os dados enviados em ambas direções sejam previamente cifrados com uma chave fornecida pela organização. Desta forma, consegue-se limitar o acesso em caso de interceção por parte dos atores maliciosos, garantindo que tanto os dados da organização como os dos funcionários permaneçam mais seguros.

Atualização e manutenção

Considerando o ritmo ao qual novos dispositivos são lançados, os sistemas operativos são atualizados e as aplicações são desenvolvidas para uso comercial, muitas empresas descobrem que seus sistemas de MDM não conseguem atender de forma continua às necessidades e com os níveis de segurança exigidos. Tal como referido em (Business, 2018), existe a necessidade de uma atualização constante destes sistemas MDM, o que muitas vezes se traduzem em custos adicionais para as organizações.

A arquitetura de funcionamento por módulos proposta por esta *framework* trará como vantagem o processo de manutenção e orquestração dos vários serviços, tornando-se assim menos complexos visto que será possível adicionar ou remover módulos dependendo das necessidades ou falhas que se pretendem colmatar. Este processo poderá ser feito sem comprometer o funcionamento geral do sistema, permitindo fazer um isolamento de problemas e consequentemente obter uma recuperação mais rápida em caso de falhas. Esta arquitetura permitirá que uma empresa adote ou integre apenas os serviços necessários, reduzindo o número de aplicações em uso e o custo de propriedade.

Consumo mínimo de recursos dos dispositivos

Para completar o *puzzle*, outro objetivo desta solução é que as aplicações criadas para os dispositivos móveis sejam bastantes minimalistas no que diz respeito ao consumo de recursos, de processamento e autonomia. Para dar resposta a esta questão, a solução encontrada e que está em linha com o que foi adotado por outros autores (Genevey & Dominguez, 2017) passa por adaptar os agentes instalados nos dispositivos para que funcionem em modo *background*, isto é, sem que sejam pedidas interações por parte dos utilizadores, e com a capacidade de se tornarem inativos (*sleepmode)* durante períodos de

inatividade. Com isso consegue-se garantir que as aplicações sejam menos consumidoras de recursos possíveis, evitando situações como drenagem de bateria, memória ou CPU dos dispositivos.

3.1. Módulos que compõem o *Framework*

Tratando-se de um sistema cujo principal papel é a gestão dos dispositivos móveis, consideramos que existem alguns serviços que seriam indispensáveis. Contudo, como já foi referido, caberá sempre à organização analisar quais os módulos a implementar de forma a endereçar às suas necessidades.

Módulo de Inventário

O módulo de inventário visa a recolha de várias informações dos dispositivos, o seu tratamento e a sua disponibilização para que possa ser integrada em bases de dados do inventário. A recolha das informações vai desde o modelo, *serial number*, capacidade de armazenamento total/disponível, versões de aplicações, sistema operativo instalado, datas de instalações/atualizações entre outras informações relevantes para que os administradores estejam a par das mudanças que vão ocorrendo e tenham uma ideia clara do potencial móvel em uso na organização.

Módulo de Monitorização

A monitorização constante dos dispositivos irá permitir aos responsáveis de IT estarem um passo à frente na deteção de possíveis problemas que possam surgir no seio da organização.

Este módulo permitirá uma análise da exposição ao risco a partir do *tracking* de aplicações não permitidas dentro da *network* empresarial, detetando possíveis ameaças em caso de aplicações malcomportadas. Será ainda possível filtrar/cruzar informações e obter as marcas ou modelos predominantes. Outro ponto importante será na sinalização de situações de incumprimento das políticas internas da empresa.

Módulo de Configuração

Será a partir do módulo de configuração que será possível definir políticas e restrições baseadas por exemplo no sistema operativo, tipo ou grupo de utilizadores. Será também

possível garantir aos funcionários o acesso a sites, conteúdos e aplicações internas, *email*, rede *wireless*, VPN, entre outras, sem que haja necessidade de interação por parte dos utilizadores.

Módulo de Gestão de Aplicações

É da responsabilidade do módulo de gestão de aplicações a gestão e o controlo no acesso a aplicações que os funcionários poderão ter em uso dentro na rede empresarial. Com a panóplia de aplicações existentes torna-se fundamental a disponibilização de catálogos com aplicações aprovadas e seguras que não põem em risco os dados da organização e os dispositivos móveis. Será também necessário desenvolver e gerir *App Stores* internas onde os funcionários poderão escolher e descarregar aplicações específicas para uso interno.

A implementação destes módulos irá possibilitar a obtenção de relatórios globais, permitindo que os administradores/responsáveis de IT tenham o conhecimento de todo o potencial de dispositivos móveis existentes dentro da empresa. Deverá ser ainda possível enviar notificações, despoletando alertas em caso de incumprimento de políticas internas, ou quando detatado anomalias no funcionamento dos dispositivos, casos em que os dados da empresa poderão estar em risco.

3.2. Processo de *Enrollment*

A Figura 3 ilustra a sequência de passos que deverão ser executados durante o acesso e registo dos utilizadores (seus dispositivos) na rede empresarial. A solução deverá implementar um processo de *enrollment* simples, mas com o enquadramento necessário para que os funcionários tenham o conhecimento das políticas de acesso implementadas pela empresa.

Figura 3 - Processo de registo

De forma sequencial, a Figura 3 ilustra os seguintes passos:

1. **Acesso à rede**

Quando um utilizador se liga à rede da empresa é informado que para continuar terá que ter a aplicação instalada no seu dispositivo. É apresentado uma ligação da diretoria onde deverá aceder para descarregar a aplicação.

2. **Instalação e termos de utilização**

Neste ponto é apresentado de forma explícita o enquadramento GDPR (*General Data Protection Regulamentation*), que mostra como a empresa encara a política de gestão e controlo de dados. Neste enquadramento deverá ser apresentado também a lista de dados que serão recolhidos e a sua finalidade. Feito isso, o utilizador lê e aceita os termos e condições, dando também o consentimento para a recolha de dados quando o dispositivo estiver conectado às redes da empresa, salientando que a empresa só fará a recolha nas redes aceites pelo utilizador.

3. **Gestão / recolha de dados**

Chegada a esta fase, são aplicadas as configurações necessárias no equipamento e ativada a recolha de dados que será executada em intervalos de tempo previamente definidos. Desta forma é dada como concluída o processo de registo/ativação do dispositivo na rede da empresa.

Sob o ponto de vista das organizações, a recolha de dados nos dispositivos permitirá a criação e implementação de um conjunto de estratégias não só de gestão, mas principalmente de auxílio na realização de tarefas diárias, permitindo que os funcionários possam criar valores dentro da organização.

Sendo esta uma solução dirigida às organizações, torna-se evidente a necessidade de existir um conjunto de políticas e normas que os funcionários terão de cumprir para terem o acesso à rede e informações privilegiadas a partir dos seus dispositivos. Estas normas e políticas devem ser previamente definidas pela empresa e expostas de forma clara aos funcionários para que tenham o devido conhecimento, tornando também patente a necessidade de instalação da aplicação agente nos dispositivos.

4. FUNDAMENTOS TEÓRICOS

Ao longo desta sessão são apresentados os fundamentos teóricos e as razões que ditaram a escolha das metodologias, ferramentas e plataformas para a implementação da prova de conceito apresentada no capítulo 5.

4.1. Sistema de Inventário como Prova de Conceito

Para que uma empresa tenha o conhecimento minucioso dos dispositivos em funcionamento na sua rede interna e poder avaliar melhor os riscos e as vantagens do BYOD é preciso começar por fazer o levantamento detalhado das características destes dispositivos.

Este sistema é o alicerce para tudo o que se segue, alimentando o *framework* com informações relevantes (*hardware* e *software*), para que a organização possa estar ciente das mudanças e ter uma visão constante sobre as alterações que vão ocorrendo em cada dispositivo.

Esta é a principal razão pela qual foi escolhida a implementação do módulo de inventário como prova de conceito. Futuramente este módulo irá servir como base na implementação dos restantes módulos que fazem parte deste *framework*.

4.2. Escolha da Plataforma de Desenvolvimento *Mobile*

Para a escolha da plataforma de desenvolvimento a utilizar na prova de conceito, foi necessário fazer um pequeno estudo inicial para se identificar os líderes do mercado das aplicações e desenvolvimento *mobile*. Sem margens de dúvida que o Android e o iOS dominam esta área.

Independentemente dos prós e contras, estudos mostram que a primeira tem uma quota de mercado de cerca de 74.73% (Statcounter, 2018), o que se traduz numa comunidade maior de *developers*/*testers,* e quantidade de informação disponibilizada.

Tendo em consideração estes dados, e tratando-se do trabalho final do segundo ciclo da formação superior, o caminho a seguir foi escolher a plataforma "mais fechada" no qual requeria mais esforço e dedicação, adicionando a aliciante ideia de que após a implementação da prova de conceito para a plataforma iOS, num trabalho futuro seria menos atribulado fazer a mesma implementação para a plataforma Android.

4.3. Programas de Desenvolvimento iOS

A Apple disponibiliza programas diferentes de desenvolvimento dos quais podemos nos registar como programadores (Apple, 2018):

- *University Program* – este foi o programa de desenvolvimento utilizado para a implementação deste caso de estudo. Além de ser gratuito e direcionado às instituições educativas, permite o acesso às versões *Gold Master* de SDK (*Software Development Kit*) do iOS; fóruns de desenvolvimento da Apple; documentação e exemplos em linha; *deploy* de aplicações em dispositivos da marca.
- *Individuals/Organizations Program* – este programa tem um custo de desenvolvimento anual de 99 dólares, e é direcionado a indivíduos/proprietários em nome individual ou organizações interessadas em desenvolver aplicações para distribuição na App Store. Além das vantagens do programa anterior, conta também com o acesso a versão beta de sistemas operativos; suporte técnico oficial de código; ferramentas de desenvolvimento e testes.
- *Enterprise Program* – por último, este é o programa direcionado às instituições corporativas e tem um custo de 299 dólares anuais por membro de equipa. Este programa está orientado ao desenvolvimento de aplicações proprietárias projetadas e distribuídas exclusivamente a funcionários de organizações. Inclui também as vantagens do programa anterior, bem como o acesso a ferramentas de desenvolvimento e testes para a distribuição interna.

4.4. Tarefas de Longa Duração (modo *background*)

Em iOS apenas alguns tipos de aplicações têm permissões para executarem em modo *background* sem que sejam suspensas pelo sistema operativo. São elas, aplicações de conteúdos áudio; aplicações de localização e navegação; aplicações de VoIP (*Voice over Internet Protocol*); aplicações que necessitam constantemente de fazer *download* e processamento de conteúdos; e aplicações que necessitam regularmente de receber atualizações a partir de acessórios externos (Apple, 2017).

Tal como será apresentado no capítulo seguinte, a configuração escolhida para permitir que o agente funcione em *backgound* foi VoIP. Uma das características relevantes a ter em conta nestas aplicações é o facto de serem relançadas sempre após o *boot* do sistema

operativo, garantindo que os serviços VoIP continuem disponíveis. Tirando partido desta característica, não será necessário reativar manualmente o agente (ou a partir de *pushnotification* por exemplo) sempre que o dispositivo é reiniciado.

4.5. Web Services

De forma simplificada, um WS é um programa criado para realizar uma determinada operação e disponibilizado na internet para que clientes ou outras aplicações o possam aceder e usufruir dos seus serviços (Monteiro, 2010). Existem duas arquiteturas diferentes para a criação de WS. A mais antiga e tradicional que tem uma abordagem baseada em padrões denomina-se *Simple Object Access Protocol* (SOAP) e a mais recente e definida como sendo mais simples é a *Representative State Transfer* (REST).

O SOAP tornou-se num protocolo *standard* para troca de mensagens baseadas no padrão XML e uma das suas vantagens é que não está ligado a nenhum protocolo de envio de mensagens em particular. Estas podem ser enviadas por qualquer protocolo que possa transmitir XML. Por sua vez, o REST tem na sua essência o protocolo HTTP, no qual é utilizado para o envio de mensagens, tirando partido de alguns métodos como GET, POST, PUT e DELETE, para obter, enviar ou apagar dados (Monteiro, 2010). Esta arquitetura foi adotada por várias empresas de topo no mercado, na área de fornecimento de serviços Web, como é o caso da Amazon, Google, Yahoo e Facebook devido à sua facilidade de utilização, em detrimento da primeira (De Luca, Epicoco, Lezzi, & Aloisioa, 2012).

Na implementação do WS do módulo de inventário, a arquitetura utilizada foi REST. No cenário apresentado na prova de conceito, faz sentido a implementação de um WS baseado nesta arquitetura, uma vez que este WS se limita à escuta e armazenamento dos dados enviados pelo agente. Desta forma, a implementação da comunicação entre estas duas partes seria menos complexa.

4.6. Arquitetura de Transmissão de Dados

O Java Script Object Notations (JSON) foi a arquitetura utilizada para o envio de informações entre o agente e o WS. Este formato de troca de dados computacionais é baseado em textos estruturados, fáceis de interpretar e descodificar (Grahl et al., 2017). O facto de ser independente de plataformas ou linguagens de programação torna-se muitas vezes ideal na interação entre sistemas diferentes que têm necessidades de comunicarem entre si através do

envio de dados. Uma das vantagens do JSON sobre o XML (que por sua vez é a linguagem de troca de dados mais antiga e popular) é que consegue ser mais pequeno, leve e muito mais rápido nos processos de serialização e desserialização de dados. Estudos também comprovaram que JSON supera XML no que diz respeito à utilização dos recursos do CPU (Joshi, 2017), tornando-o mais atrativo em ambientes onde os recursos computacionais escasseiam-se com alguma facilidade, como é o caso dos dispositivos móveis.

4.7. Cifra de Dados

Os algoritmos de cifra de dados encontram-se divididos em dois grupos. Os simétricos ou chave secreta, e os assimétricos que também são conhecidos como chave pública. A diferença é que na primeira o processo de codificação e descodificação dos dados é feita usando a mesma chave. Enquanto na segunda a codificação e descodificação é feita usando chaves diferentes, ou seja, uma chave pública e uma chave privada (Singh & Supriya, 2013).

Questões importantes a ter em conta no processo de escolha de algoritmos de cifra são: tempo de computação, memória utilizada e os *bytes* resultantes. Estudos comprovaram que técnicas de cifra assimétricas são cerca de 1000 vezes mais demorosas do que as simétricas (Singh & Supriya, 2013), o que as torna impraticáveis quando se trata em quantidades enormes de dados ou em ambientes onde os recursos são escassos. Com base nestas informações o foco foi optar por um algoritmo de cifra simétrico por bloco (pondo de parte os algoritmos do mesmo grupo que funcionam por *stream*), uma vez que os dados a cifrar são previamente conhecidos.

Dentro deste grupo de algoritmos, podemos destacar três: *Advanced Encryption Standard* (AES); *Digital Encryption Standard* (DES) e *Triple Digital Encryption Standard* (3DES). Foi feita uma análise comparativa entre estes algoritmos, tendo em conta fatores como: tamanho da chave; tipo de cifra; tamanho do bloco de dados; segurança; possibilidades de chaves, etc., e chegou-se à conclusão que o AES seria a melhor opção em relação às outras duas (Singh & Supriya, 2013).

O AES processa os dados em blocos de 128 bits e pode ser configurado para utilizar três tamanhos de chaves diferentes, resultando nos algoritmos AES-128, AES-192 e AES-256 respetivamente, indicando em bits o tamanho das chaves (Townsend Security, 2016).
O algoritmo utilizado para garantir a segurança dos dados recolhidos nos dispositivos foi AES-256. No capítulo seguinte serão apresentados os passos da sua implementação.

5. FRAMEWORK GESTÃO DE DISPOSITIVOS MÓVEIS EM FUNÇÃO DO CONTEXTO: MÓDULO DE INVENTÁRIO

Como prova de conceito, neste capítulo são apresentados os passos que foram necessários para a criação do módulo de inventário, em particular a implementação dos dois elementos que compõem este módulo. O agente foi arquitetado para a plataforma do sistema operativo iOS utilizando a linguagem de programação Objective-C e o programa de desenvolvimento (*UniversityProgram*) fornecido pela Apple. Por sua vez o WS foi implementado em REST, utilizando a linguagem de programação PHP. Como motor da base de dados foi utilizado o MySQL para criar a estrutura que armazena os dados recolhidos dos dispositivos.

5.1. Modo de Funcionamento

A estrutura e o funcionamento do módulo de inventário são partilhados pelos dois componentes que implementam a sequência lógica, começando pela obtenção dos dados até a sua disponibilização para consulta. O agente instalado nos dispositivos é o responsável pela recolha e o envio de informações detalhadas dos dispositivos. OWS, por sua vez, garante a receção dos dados, o tratamento e posterior armazenamento na base de dados de inventário. Na Figura 4 apresenta-se o *use case* que dá a conhecer as funcionalidades deste módulo.

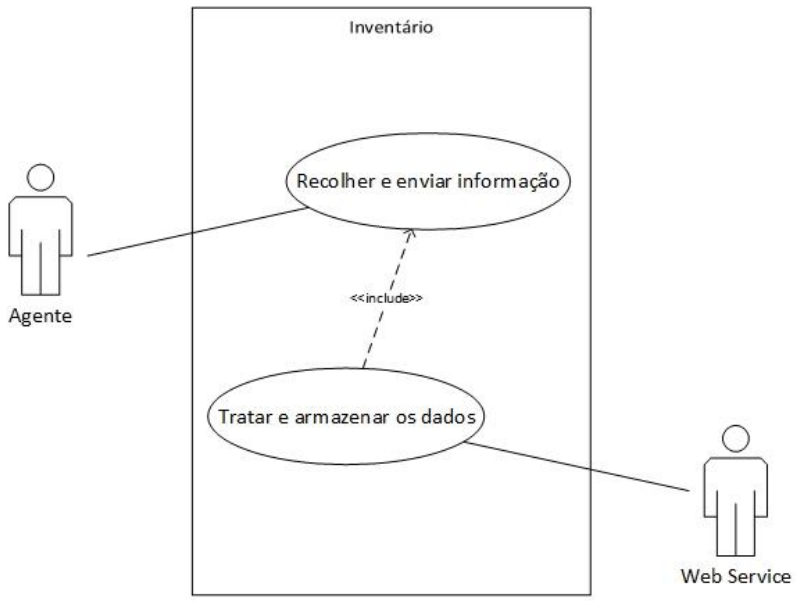


Figura 4 - Use case do funcionamento do módulo inventário

Como se pode analisar pelo diagrama ilustrado na Figura 4, este módulo implementa duas funcionalidades principais que estão diretamente ligadas às ações dos dois atores deste sistema. A funcionalidade "Recolher e enviar informação" é essencial para o comportamento do segundo ator (WS) na fase de "Tratar e armazenar os dados".

Assente no princípio da separação do contexto *user vs entreprise* onde foi definido que os dados serão recolhidos somente quando o dispositivo estiver ligado à rede empresarial, cabe ao agente ter esse conhecimento e decidir se a recolha e o envio da informação poderão ou não ser iniciadas. Por sua vez, o WS limitar-se-á à escuta da informação enviada pelo agente, desencadeando a execução da segunda funcionalidade no momento da receção desta informação. Deste modo, a comunicação será feita num único sentido e estará a cargo da ação do agente. Uma descrição mais pormenorizada do *use case* pode ser consultada no Apêndice A.

A Figura 5 visa facilitar a leitura e interpretação da interação entre os dois atores. Nela podemos ver o fluxo de dados enviados pelo agente, passando pelo WS até ser armazenado na base de dados de inventário.

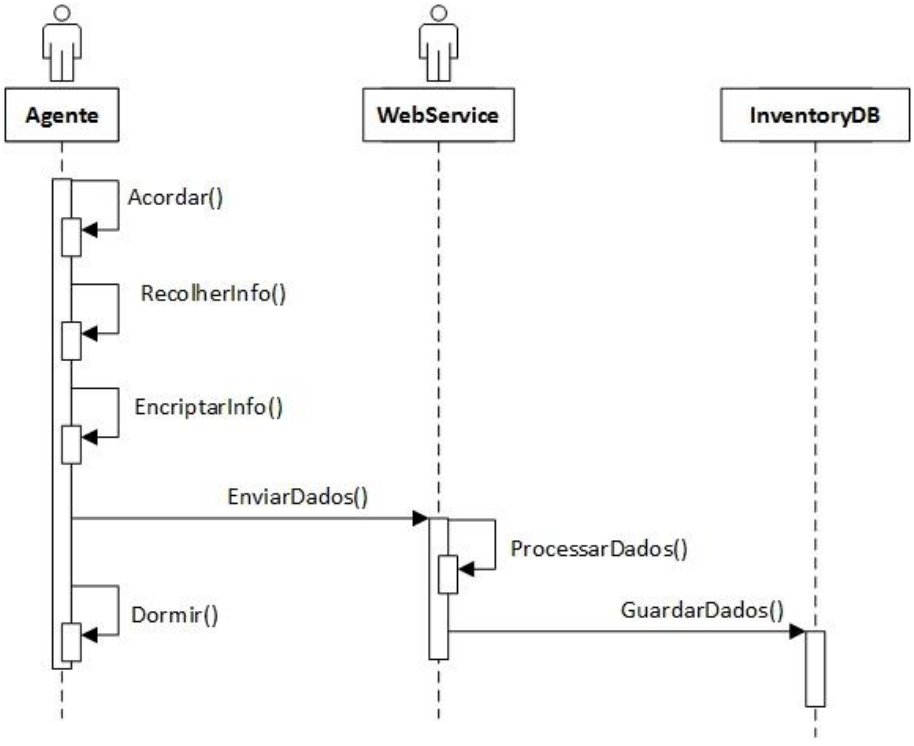


Figura 5 - Interação entre os atores

5.2. Identificação e Caracterização dos Dados

Após a definição da estrutura e modo de funcionamento deste módulo, o passo seguinte foi identificar os dados para recolha e possíveis de serem integrados num sistema de inventário. Estas informações foram divididas em três grupos:

Característica e estado dos dispositivos – neste grupo são apresentadas especificações a nível do *hardware* e o estado de funcionamento dos dispositivos. Existem informações que são estáticas, como é o caso do modelo, capacidade total do disco, etc. E outras que vão variando com o tempo (ex.: memória RAM, versão do SO). Com estas informações consegue-se por exemplo criar configurações para grupos específicos de dispositivos consoante a marca ou o modelo. Será também possível definir alertas/notificações a serem enviadas quando os dispositivos se encontrarem perante situações de carência de memória RAM ou espaço em disco.

Aplicações instaladas – neste grupo são apresentadas informações sobre as aplicações que se encontram instaladas nos dispositivos e as versões de cada uma delas. Normalmente estas aplicações podem ser do SO ou instaladas pelo utilizador. Independentemente dos casos, elas podem ser identificadas pelo nome ou por um identificador (ID). Importa referir que para a prova de conceitos foi recolhido informações sobre o máximo número de aplicações possíveis; contudo a ideia é que a *framework* só faça o rastreio a aplicações malcomportadas e previamente identificadas pela empresa como potenciais riscos para o funcionamento da rede interna.

Processos a correr – neste último grupo, são identificadas informações sobre os processos que estejam a correr nos dispositivos. Cada processo tem associado a si um nome, um estado, uma *flag* que indica se é ou não um processo do SO e a data de início. Esta informação será relevante para identificar por exemplo casos de incumprimentos de políticas no que diz respeito a não utilização de determinadas aplicações no seio empresarial.

Na Tabela 4 é apresentada uma breve descrição destes dados.

	Nome	Tipo	Descrição
Característica e estado dos dispositivos	Nome do dispositivo *[devicename]*	String	Indica o nome associado ao dispositivo identificado na rede empresarial. Ex.: iPhone Pedro.
	Modelo do dispositivo *[devicemodel]*	String	Representa o modelo do dispositivo. Ex.: iPhone.
	Nome do sistema operativo *[systemname]*	String	Indica o nome do sistema operativo instalado no equipamento. Ex.: iPhoneOS.
	Versão do sistema operativo *[systemversion]*	String	Indica a versão do sistema operativo. Ex.: iOS 7.1.2.
	Espaço total do disco *[totaldiskspace]*	String	Indica o espaço total do *hard drive* (HD) do dispositivo. Ex.: 13,6GB.
	Espaço livre no disco *[freediskspace]*	String	Indica o espaço disponível no HD. Ex.: 594 MB.
	Memória física *[physicalmemory]*	String	Representa a RAM total do dispositivo. Ex.: 2GB RAM.
	Memória do utilizador *[usermemory]*	String	Indica a memória RAM disponível ao utilizador.
	Memória disponível *[freememory]*	String	Parte da memória RAM livre e que pode ser utilizada a qualquer altura pelo SO.
	Memória alocada pelo sistema operativo *[wiredmemory]*	String	Parte da memória RAM utilizada pelo SO para guardar informações consideradas críticas. Ex.: *kernel* e outras aplicações chave.
	Memória ativa *[activememory]*	String	Parte da memória RAM onde se encontra informações em uso ou informações que foram usadas recentemente.
	Memória inativa *[inactivememory]*	String	Parte da memória RAM utilizada para armazenar informações acedidas com mais frequência pelo utilizador.
	IP do dispositivo *[ipcell]*	String	Indica o IP do dispositivo móvel na rede GPRS.
	IP do dispositivo na rede Wi-Fi *[ipwifi]*	String	Indica o IP do dispositivo na rede Wi-Fi da empresa.
	MAC Address Wi-Fi *[wifimacaddress]*	String	Identificador do MAC Address da rede sem fios do dispositivo dentro da rede empresarial.

	Bluetooth MAC Address *[bluetMacAddress]*	String	Identificador do MAC Address do *Bluetooth* do dispositivo.
	Identificador da rede Wi-Fi *[ssid]*	String	SSID da rede a que o dispositivo se encontra ligado.
	Conectada via USB *[isUsbPlugged]*	int	Indica se o dispositivo se encontra conectado a um computador ou em modo carregamento via USB.
	Conectado à rede 3G *[is3GNetworkActivated]*	int	Indica se o dispositivo se encontra ligado à rede de terceira geração.
Aplicações instaladas	Nome da aplicação *[appname]*	String	Indica o nome da aplicação instalada no dispositivo.
	Versão da aplicação *[appversion]*	String	Indica a versão da aplicação instalada no dispositivo.
Processos a correr	Nome do processo *[procename]*	String	Indica o nome do processo a correr no dispositivo.
	Estado do processo *[procstat]*	String	Indica o estado do processo.
	Flag do processo *[procflag]*	String	*Flag* que indica o tipo de processo a correr (processo do SO, processo do utilizador, etc.).
	Tempo do processo *[proctime]*	String	Indica a data do início do processo.

Tabela 4 - Informação recolhida dos dispositivos

5.3. Estrutura e Criação da Base de Dados

Tendo em conta os dados acima identificados para recolha nos dispositivos, começou-se por desenhar a estrutura da base de dados que iria armazenar estas informações. A criação da mesma seguiu as boas práticas, na qual as entidades, atributos e relacionamentos foram sendo refinados ao longo dos processos de normalização. No final as tabelas resultantes foram nomeadas com o prefixo "Inv" para melhor serem identificadas como sendo tabelas do módulo de inventário. No Apêndice B podem ser consultados os comandos implementados para a criação de cada uma destas tabelas.

Tabela `InvDevice`

Nesta tabela são armazenadas informações estáticas que permitem identificar cada dispositivo em funcionamento dentro da rede empresarial. Cada entrada nesta tabela diz respeito a um só dispositivo. Ela é composta pelos seguintes campos:

- **Id_device (PK)**
- AppId
- Wifimacaddress
- Model
- SysName
- TotalDiskSpace
- PhysicalMemory
- BluetMacAddress

O campo `Id_device` representa a chave primária (PK – *Primary Key*) que permite identificar inequivocamente cada linha da tabela. Como forma de identificar unicamente cada aplicação agente instalada nos dispositivos, foi ainda necessário adicionar a esta tabela o campo `AppId`. No Apêndice D será abordado em mais detalhes a importância deste campo nas operações com a base de dados.

Tabela `InvDeviceStatus`

Nesta tabela encontram-se armazenadas informações que estão em constante alteração e que indicam o estado de cada dispositivo num determinado momento. Ela é composta pelos seguintes campos:

- **Id_status (PK)**
- **Id_device (FK)**
- SysVersion
- FreeDiskSpace
- IpWifi
- IpCell
- UserMemory
- WiredMemory
- ActiveMemory
- InactiveMemory
- FreeMemory
- IsUsbPlugged
- Is3GNetworkActivated
- DeviceName
- Date

Nesta tabela `Id_status` é a chave primária. O campo `Id_device` é a chave estrangeira (FK – *Foreign Key*) que faz referências à tabela `InvDevice`, permitindo identificar cada dispositivo.

Tabela `InvInstalledApps`

Esta tabela armazena a lista de aplicações instaladas em cada dispositivo num determinado momento. É composta pelos seguintes campos:

- **Id_app (PK)**
- **Id_device (FK)**
- AppName
- AppVersion
- Date

O campo `Id_app` é a chave primária desta tabela, e o `Id_device` é a chave estrangeira que identifica os dados da tabela `InvDevice`.

Tabela `InvRunningProcess`

Por último, nesta tabela são armazenadas informações sobre os processos a correr nos dispositivos num determinado momento. É composta pelos campos:

- **Id_process (PK)**
- **Id_device (FK)**
- ProcName
- ProcStat
- ProcFlag
- ProcTime
- Date

O campo `Id_process` é a chave primária que permite identificar cada entrada na tabela. O `Id_device` é a chave estrangeira que faz referência aos dados da tabela `InvDevice`.

A Figura 6 mostra o modelo relacional da base de dados do módulo de inventário e o relacionamento entre as tabelas acima apresentadas. Esta base de dados foi desenvolvida de modo a ter um funcionamento simples, seguro e que facilite as tarefas de pesquisas.

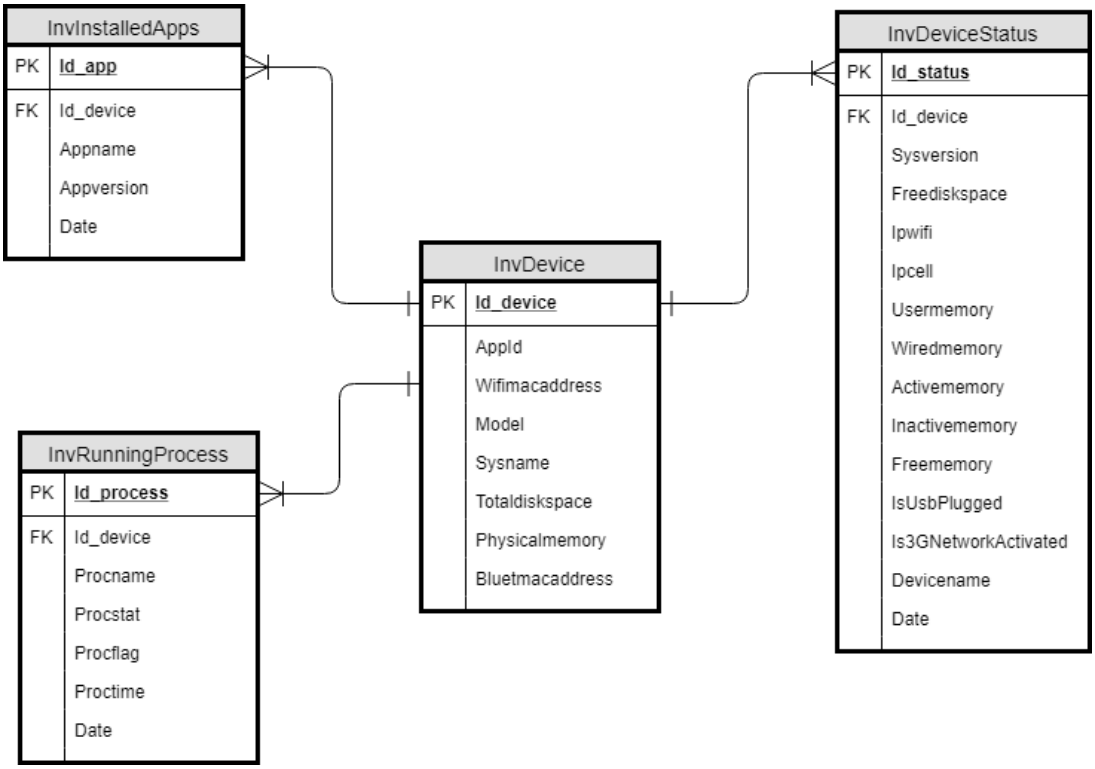


Figura 6 - Modelo ER da base de dados

5.4. Implementação do Agente

Nesta secção serão apresentados os passos utilizados na implementação e configuração do agente. Por questões de apresentação, será dado mais ênfase aos pontos considerados essenciais para o correto funcionamento do agente e como tal alguns métodos/variáveis e configurações secundárias serão omissas. Contudo, no Apêndice C é apresentado em mais detalhes a descrição de alguns dos métodos aqui omitidos.

5.4.1. Modo *Background* usando VoIP

Para permitir que o agente funcione como uma aplicação VoIP foram necessárias algumas configurações. O primeiro passo foi criar uma conexão remota ao *host* (endereço do WS na *cloud*) e utilizar objetos do tipo *stream* para receber ou criar os dados que seriam enviados remotamente. Para isso, foi necessário recorrer às classes `CFStream`, `NSStream` e à função `CFStreamCreatePairWithSocketToHost`, disponíveis pela API (Apple, 2018). A primeira permite-nos criar a conexão com o WS e de seguida podemos converter os *streams* resultantes para `NSStream` e desta forma escrever os dados recolhidos do dispositivo. Isso tudo é possível através da utilização da função indicada que recebe como parâmetro o endereço remoto do WS, o número de porto para a conexão e ambas as *streams* de leitura e de escrita.

A Figura 7 ilustra a declaração das variáveis e métodos no ficheiro `ViewController.h` do projeto no Xcode para permitir esta configuração.

```
11 @interface ViewController : UIViewController <NSStreamDelegate> {
12
13     //Variávies globais
14     NSInputStream *inputStream;
15     NSOutputStream *outputStream;
16 }
17
18 //Métodos de configuração VoIP
19 - (void)initNetworkCommunication;
20 - (void)stream:(NSStream *)theStream handleEvent:(NSStreamEvent)streamEvent;
```

Figura 7 - Declaração de variáveis

O primeiro método (`initNetworkCommunication`) tem como objetivo configurar e iniciar a comunicação com o WS através das classes e da função acima indicada. Outro detalhe importante é que este método é o responsável pela suspensão do agente durante intervalos de tempo configuráveis, garantindo que o SO não o termine em caso de escassez de memória ou espaço em disco. Neste caso, o intervalo de tempo utilizado foi de 600 segundos, equivalentes a 10 minutos.

A Figura 8 apresenta a implementação do primeiro método (`ViewController.m`).

```
- (void)initNetworkCommunication
{
    //1 - Configuração da função com o endereço, porto e as streams de leitura e escrita
    CFReadStreamRef readStream;
    CFWriteStreamRef writeStream;
    CFStreamCreatePairWithSocketToHost(NULL, (CFStringRef)@"188.93.231.75", 80, &readStream, &writeStream);

    //2 - Cast de CFStream para NSStream
    inputStream = (__bridge_transfer NSInputStream *)readStream;
    outputStream = (__bridge_transfer NSOutputStream *)writeStream;

    //3 -  Indica o tipo de serviço VoIP na stream de entrada
    [inputStream setProperty:NSStreamNetworkServiceTypeVoIP forKey:NSStreamNetworkServiceType];

    //4 -  Delega quem recebe as mensagens input e output stream
    [inputStream setDelegate:self];
    [outputStream setDelegate:self];

    //5 - Indica o ciclo de execução das streams de input e ouput
    [inputStream scheduleInRunLoop:[NSRunLoop currentRunLoop] forMode:NSDefaultRunLoopMode];
    [outputStream scheduleInRunLoop:[NSRunLoop currentRunLoop] forMode:NSDefaultRunLoopMode];

    //6 - Abre a conexão para ambos os streams
    [inputStream open];
    [outputStream open];

    //7 - Suspende o agente durante um tempo determinado, indicado em segundos (600s)
    [[UIApplication sharedApplication] setKeepAliveTimeout:600 handler:^{
        if (outputStream)
        {
            //Verifica se o dispositivo se encontra conectado à rede empresarial
            if([SSID isEqualToString:[self getSSIDInfo]])
            {
                //... Aqui faz-se a recolha e o envio de dados
            }
        }
    }];
}
```

Figura 8 - Método que inicia o processo de comunicação com o WS

Por sua vez, o segundo método é delegado a receber as constantes atualizações que vão ocorrendo com a conexão. Consoante o evento recebido, poderemos fazer um tratamento diferente. No nosso caso, uma boa altura para enviar o pedido de escrita dos dados recolhidos é quando esse evento for do tipo `NSStreamEventHasSpaceAvailable` (Apple, 2013). Fazendo um ligeiro recuo, podemos ver que na Figura 7 foi adicionado o protocolo `NSStreamDelegate` (Apple, 2018) à assinatura da interface. Este protocolo permite indicar à aplicação de que vai ser implementado o método encarregue de receber e controlar os eventos de *stream* resultantes da conexão com o *host*. Na Figura 9 podemos ver a implementação do segundo método.

```objc
#pragma mark - NSStreamDelegate

- (void)stream:(NSStream *)theStream handleEvent:(NSStreamEvent)streamEvent
{
    //Analisa o evento recebido
    switch (streamEvent) {

        case NSStreamEventHasSpaceAvailable:

            //Boa altura para recolher e enviar os dados
            if(theStream == outputStream)
            {
                //Verifica se o dispositivo se encontra conectado à rede empresarial
                if([SSID isEqualToString:[self getSSIDInfo]])
                {
                    //... Aqui faz-se a recolha e o envio de dados
                }
            }

            break;

            //Continuação...
    }

}
```

Figura 9 - Método que avalia o estado da conexão

De forma a garantir que estas configurações sejam sempre reiniciadas após o *boot* do sistema operativo, foi necessário alterar o método `viewDidLoad` que é o primeiro a ser chamado durante este processo, adicionando uma invocação do método`iniNetworkCommunication`. Assimconseguimos permitir que o agente continue a funcionar como uma aplicação VoIP, assegurando a conexão com o *host*.

```objc
- (void)viewDidLoad {
    [super viewDidLoad];
    //Invoca a função sempre quando o sistema é reiniciado
    [self initNetworkCommunication];
}
```

Figura 10 - Método que inicia o agente durante o boot do sistema

Depois de terminar estas implementações a nível de código, é necessário ainda fazer uma última configuração na aplicação para que possamos ter o *Background Modes* ativo. Para isso selecionamos o projeto no Xcode, depois o separador *Capabilities* e é apresentado uma lista de opções de configuração que podemos adicionar à aplicação. Desta lista vamos procurar pelo *Background Modes*, ativá-lo e selecionar a opção *Voice over IP*. A Figura 11 mostra-nos o resultado esperado desta configuração.

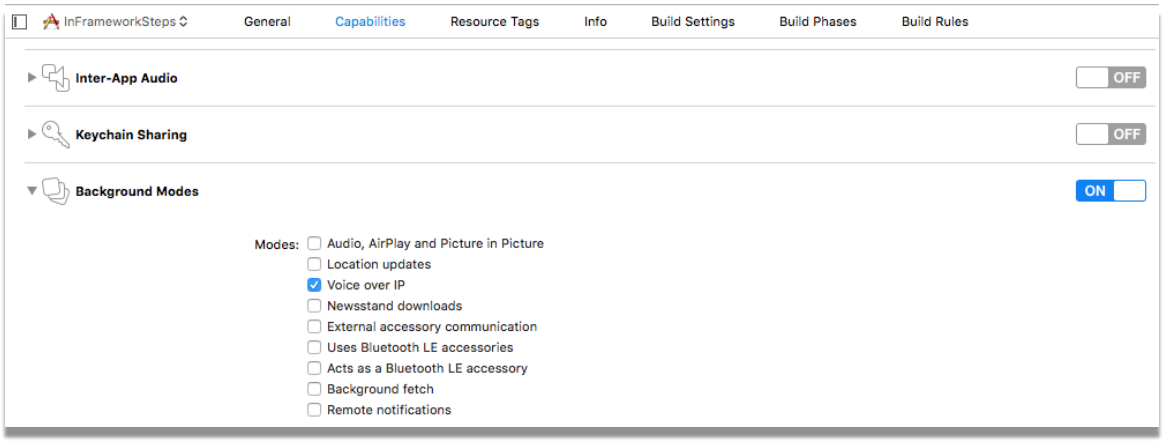


Figura 11 - Ativar o modo Background

Para concluir, podemos validar que no ficheiro `Info.plist` do projeto foi adicionado esta entrada, o que irá informar ao sistema operativo de que o agente irá fornecer serviços de VoIP e que como tal necessita de funcionar em modo *background*.

Information Property List	Dictionary	(15 items)
Localization native development re...	String	en
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	Gabriel.InFramework
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1
Application requires iPhone enviro...	Boolean	YES
Required background modes	Array	(1 item)
Item 0	String	App provides Voice over IP services
Launch screen interface file base...	String	LaunchScreen
Main storyboard file base name	String	Main
Required device capabilities	Array	(1 item)
Supported interface orientations	Array	(3 items)

Figura 12 - Validar o serviço VoIP ativo

5.4.2. Cifra de Dados

Tal como foi referido no capítulo anterior, o algoritmo utilizado nesta prova de conceitos para garantir a segurança dos dados recolhidos nos dispositivos foi AES-256. Para a sua

implementação foi necessário utilizar a classe `NSData` da API e a criação de dois métodos: `AES256EncryptWithKey:key` e `AES256EncryptForDictionary:dictionary`.

```
11 @interface NSData (Encryption)
12
13 // Encripta dados utilizando a key indicada e devolve um NSData
14 - (NSData *)AES256EncryptWithKey:(NSString *)key;
15 // Recebe um NSDictionary encripta e devolve o resultado
16 - (NSDictionary *)AES256EncryptForDictionary:(NSDictionary *)dictionary;
```

Figura 13 - Declaração dos métodos de cifra

O primeiro método é a chave principal deste algoritmo, visto ser nela onde todo o processo é executado. Foi implementado para funcionar de forma genérica, uma vez que se limita a receber a chave e os bytes e devolver um objeto do tipo `NSData` que contém os dados cifrados (caso contrário devolve nulo). É também neste método onde é feito a configuração da chave a utilizar, o tamanho do bloco, entre outros parametros. Na Figura 14 é apresentada a sua implementação.

```objc
/** Faz encriptação de dados utilizando a chave indicada **/
- (NSData *) AES256EncryptWithKey:(NSString *)key {

    //Configura uma chave para o tamanho de 256 bits
    char keyPtr[kCCKeySizeAES256 + 1];
    bzero(keyPtr, sizeof(keyPtr));
    [key getCString:keyPtr maxLength:sizeof(keyPtr) encoding:NSUTF8StringEncoding];

    NSUInteger dataLength = [self length];
    //Indica o tamanho do buffer para bloco de 128 bits
    size_t bufferSize = dataLength + kCCBlockSizeAES128;
    //Instancia o apontador (buffer) para o resultado da encriptação
    void * buffer = malloc(bufferSize);
    //Variavel que guarda o numero de bytes encriptados
    size_t numBytesEncrypted = 0;
    //Faz a encriptação criando um objecto do tipo CCCryptorStatus
    CCCryptorStatus cryptStatus = CCCrypt(kCCEncrypt,
                                          kCCAlgorithmAES128,
                                          kCCOptionPKCS7Padding,
                                          keyPtr,
                                          kCCKeySizeAES256,
                                          NULL,
                                          [self bytes],
                                          dataLength,
                                          buffer,
                                          bufferSize,
                                          &numBytesEncrypted);

    //Verifica se correu com sucesso
    if (cryptStatus == kCCSuccess) {
        //Devolve um objecto NSData com o bytes encriptados
        return [NSData dataWithBytesNoCopy:buffer length:numBytesEncrypted];
    }
    //Liberta o apontador
    free(buffer);
    return nil;
}
```

Figura 14 - Método que cifra os dados

Por sua vez, o segundo método desempenha também um papel importante neste processo uma vez que é o responsável por receber o JSON que contém os dados recolhidos. O seu objetivo é iterar sobre o objeto recebido, obtendo o SSID da rede (que representa a chave a ser utilizada na cifra) e os valores de cada uma das entradas no objeto JSON. Depois disso, chama a função anterior passando-lhe a chave e os bytes a serem cifrados. Este processo é repetido para todas as entradas existentes no objeto, devolvendo no final um JSON com todos os valores cifrados.

5.4.3. Recolha e Envio de Dados

Para a recolha de dados do dispositivo foi necessário implementar um conjunto de métodos. Durante este processo, cada um deles é invocado de forma sequencial por um método agregador principal, que no final faz o mapeamento dos dados obtidos para criar um objeto do tipo JSON que será posteriormente cifrado e enviado ao WS através da conexão criada no ponto anterior.

É importante também referir que foi utilizada a classe externa `Reachability` (GitHub, 2018) disponibilizada pela Apple, para o acesso a informações relacionadas com o tipo de rede em que o dispositivo se encontra ligado a cada momento.

Com base nas informações pré-definidas para recolha, foi definido no ficheiro de cabeçalho (`ViewController.h`) um conjunto de métodos principais.

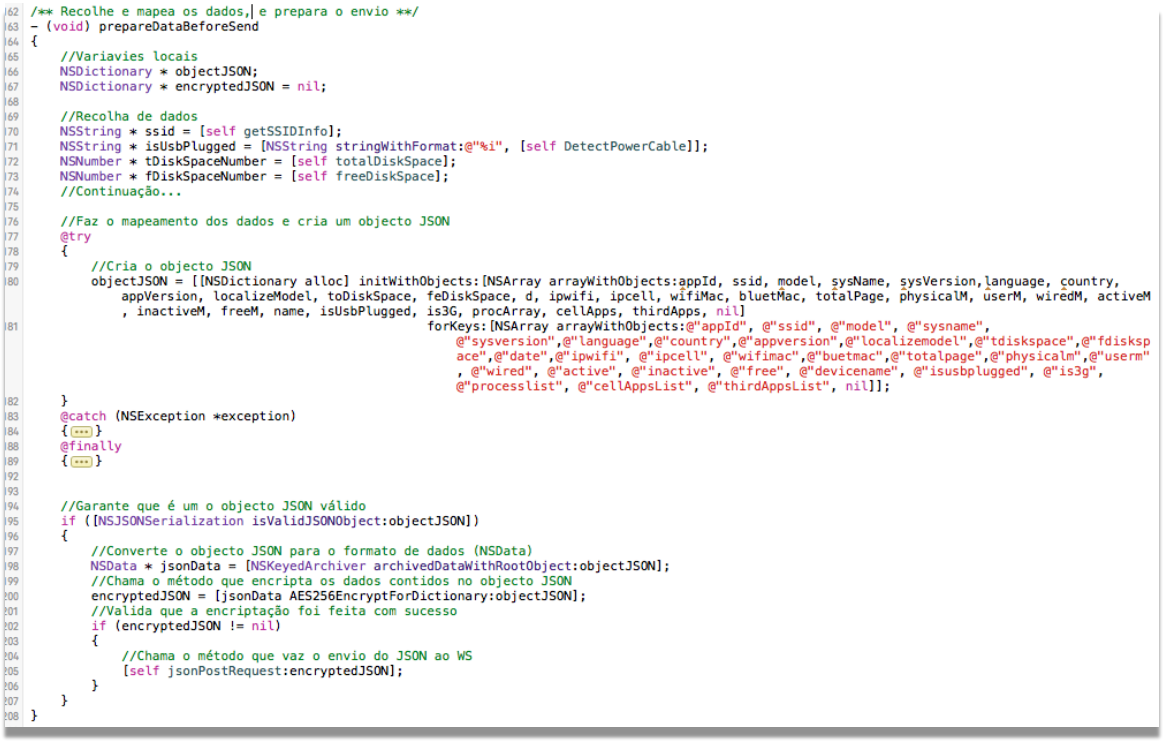
```
/*** Métodos para recolha de dados ***/

// Devolve o SSID da rede wireless
- (NSString *) getSSIDInfo;
// Verifica se o dispositivo está ligado por cabo
- (int) detectPowerCable;
// Indica o tipo de rede que o dispositivo está ligado (WWAN/WIFI)
- (int) configureOutPutReachability:(Reachability *)reachability;
// Devolve o espaço total do disco
- (NSNumber *) getTotalDiskSpace;
// Devolve o espaço livre em disco
- (NSNumber *) getFreeDiskSpace;
// Devolve o endereço IP
- (NSDictionary *) getIPAddress;
// Devolve o Mac Address WIFI do device
- (NSString *) getMacAddress;
// Devolve informações sobre a memória RAM
- (NSDictionary *) getMemoryInfo;
// Devolve a lista de processos a correr
- (NSMutableArray *) getRunningProcesses;
// Devolve a lista de aplicações do SO
- (NSMutableArray *) getCellPhoneInstalledApplication;
// Devolve a lista de aplicações instaladas pelo utilizador
- (NSMutableArray *) getThirdPartyInstalledApplications;
// Faz o mapeamento dos dados e prepara o envio
- (void) prepareDataBeforeSend;
// Envia os dados recolhidos ao WS
- (void) jsonPostRequest:(NSDictionary *) jsonData;
```

Figura 15 - Declaração dos métodos

Com a exceção dos dois últimos métodos, todos os outros foram implementados para funcionarem de forma independente e com objetivos específicos. Quando são invocados, executam as tarefas a que foram incumbidas e devolvem o resultado.

O método `prepareDataBeforeSend` tem o conhecimento de todos os outros métodos e sabe a qual deles recorrer durante o processo de recolha, fazendo depois o devido tratamento e mapeamento da informação obtida. Este método é fundamental no processo de sincronização dos dados, garantido no final a transformação dos mesmos num objeto JSON que será cifrado e passado como parâmetro para o último método, que por sua vez trata do envio.

Na Figura 16 podemos ver os pontos essenciais da implementação do método `prepareDataBeforeSend`.



```
/** Recolhe e mapea os dados, e prepara o envio **/
- (void) prepareDataBeforeSend
{
    //Variavies locais
    NSDictionary * objectJSON;
    NSDictionary * encryptedJSON = nil;

    //Recolha de dados
    NSString * ssid = [self getSSIDInfo];
    NSString * isUsbPlugged = [NSString stringWithFormat:@"%i", [self DetectPowerCable]];
    NSNumber * tDiskSpaceNumber = [self totalDiskSpace];
    NSNumber * fDiskSpaceNumber = [self freeDiskSpace];
    //Continuação...

    //Faz o mapeamento dos dados e cria um objecto JSON
    @try
    {
        //Cria o objecto JSON
        objectJSON = [[NSDictionary alloc] initWithObjects:[NSArray arrayWithObjects:appId, ssid, model, sysName, sysVersion,language, country,
            appVersion, localizeModel, toDiskSpace, feDiskSpace, d, ipwifi, ipcell, wifiMac, bluetMac, totalPage, physicalM, userM, wiredM, activeM
            , inactiveM, freeM, name, isUsbPlugged, is3G, procArray, cellApps, thirdApps, nil]
                                                forKeys:[NSArray arrayWithObjects:@"appId", @"ssid", @"model", @"sysname",
                                                    @"sysversion",@"language",@"country",@"appversion",@"localizemodel",@"tdiskspace",@"fdisksp
                                                    ace",@"date",@"ipwifi", @"ipcell", @"wifimac",@"buetmac",@"totalpage",@"physicalm",@"userm"
                                                    , @"wired", @"active", @"inactive", @"free", @"devicename", @"isusbplugged", @"is3g",
                                                    @"processlist", @"cellAppsList", @"thirdAppsList", nil]];
    }
    @catch (NSException *exception)
    {...}
    @finally
    {...}


    //Garante que é um o objecto JSON válido
    if ([NSJSONSerialization isValidJSONObject:objectJSON])
    {
        //Converte o objecto JSON para o formato de dados (NSData)
        NSData * jsonData = [NSKeyedArchiver archivedDataWithRootObject:objectJSON];
        //Chama o método que encripta os dados contidos no objecto JSON
        encryptedJSON = [jsonData AES256EncryptForDictionary:objectJSON];
        //Valida que a encriptação foi feita com sucesso
        if (encryptedJSON != nil)
        {
            //Chama o método que vaz o envio do JSON ao WS
            [self jsonPostRequest:encryptedJSON];
        }
    }
}
```

Figura 16 - Método de sincronização do processo de recolha de dados

Finalmente, chegado a esta fase cabe ao método `jsonPostRequest:jsonData` configurar o protocolo HTTP para o pedido de envio dos dados, indicando: tipo de pedido (POST), o objeto a ser enviado (JSON), o endereço do WS, o corpo e o cabeçalho da mensagem. O envio é feito através da classe `NSURLConnection` disponibilizada pela API que nos permite solicitar o envio de um objeto para um determinado endereço URL (Apple, 2017). Na Figura 17 podemos ver melhor este processo.

```
/** Envia os dados (JSON) ao WS **/
- (void) jsonPostRequest:(NSDictionary *) jsonData
{
    //Converte o objecto JSON para NSData
    NSData * data = [NSJSONSerialization dataWithJSONObject:jsonData options:kNilOptions error:NULL];
    //Cria um objecto do tipo NSURL com o endereço do WS
    NSURL * url = [NSURL URLWithString:SERVER_URL];
    //Instancia um objecto NSMutableURLRequest para solicitar o pedido de envio de dados
    NSMutableURLRequest * request = [NSMutableURLRequest requestWithURL:url];
    //Indica ao protocolo HTTP que o tipo de pedido será POST
    [request setHTTPMethod:@"POST"];
    //Indica ao protocolo HTTP que o conteúdo será do tipo JSON
    [request setValue:@"application/json; charset=utf-8" forHTTPHeaderField:@"Content-Type"];
    //Adiciona os dados ao corpo da mensagem
    [request setHTTPBody:data];
    //Adiciona o cabeçalho da mensagem
    [request addValue:@"postValues" forHTTPHeaderField:@"METHOD"];

    NSURLResponse * response;

    NSError * err;

    //Envia os dados ao WS e guarda o resultado do envio
    NSData * urlData = [NSURLConnection sendSynchronousRequest:request returningResponse:&response error:&err];
    //Converte o resultado para o formato string
    NSString * getData = [[NSString alloc] initWithData:urlData encoding:NSUTF8StringEncoding];
    //Como dados de teste pode-se consultar o resultado o envio
    NSLog(@"RESPONSE DATA: %@", getData);

}
```

Figura 17 - Método de envio de dados

5.4.4. *Deploy* do Agente no Dispositivo

Todas as aplicações desenvolvidas para correrem em sistemas operativos iOS devem ter uma assinatura e um provisionamento para poderem ser instaladas e executados em dispositivos físicos (Apple, 2018). Este provisionamento é o processo de preparação e configuração que garante esta execução e também permite que a aplicação possa utilizar alguns serviços.

Felizmente, nas versões mais recentes do Xcode todos estes passos podem ser feitos de forma automática, utilizando a informação fornecida para: criar o perfil de provisionamento da equipa de desenvolvimento e atribuí-lo ao projeto criado; gerar automaticamente o certificado de desenvolvimento; fazer o registo do dispositivo, criando um identificador único que permitirá que a aplicação possa correr num dispositivo da Apple (por exemplo: iPhone, iPad).

Basicamente estes passos resumem-se no seguinte: criar um certificado para indicar ao Xcode e ao Mac que temos o direito de poder executar a aplicação num dispositivo móvel e ainda disponibilizá-la na App Store; criar um identificador da nossa aplicação e do dispositivo onde será instalada.

Para prosseguir com a configuração, é necessário primeiro conectar o dispositivo ao Mac e no navegador de projetos do Xcode selecioná-lo a partir da lista apresentada. Fazendo

isso, o Xcode assume que pretendemos utilizar o dispositivo selecionado para desenvolvimento e faz o seu registo automático.

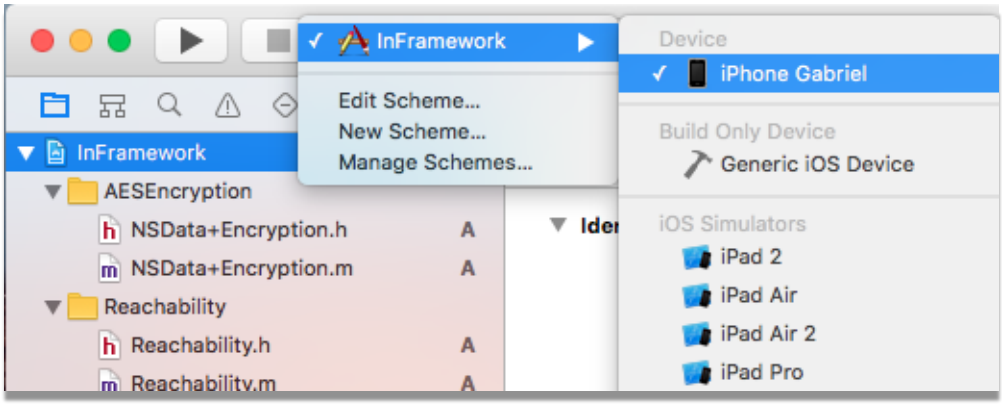


Figura 18 - Deploy da aplicação agente no dispositivo

O passo seguinte é fazer um clique no botão "*Build and Run*" para que o Xcode possa tratar de toda a configuração necessária e fazer a instalação da nossa aplicação no dispositivo selecionado. Feito isso, o agente é automaticamente executado tal como podemos ver nas imagens seguintes.

Figura 19 - Ícone da aplicação agente instalada

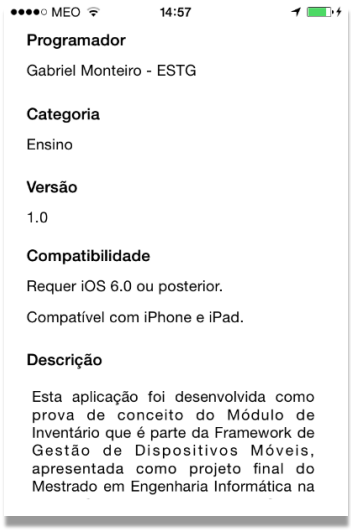


Figura 20 - Interface descritiva da aplicação agente

Concluído com sucesso a configuração e instalação da aplicação agente, pode-se avaliar que foi implementada uma interface gráfica simples, que visa descrever o âmbito e propósito da sua implementação.

Terminado este processo, podemos ainda a partir de um *browser* aceder ao portal de *developer* da Apple (https://developer.apple.com/account/) para verificarmos estas etapas e perceber melhor as configurações criadas automaticamente pela ferramenta Xcode.

As figuras 21, 22 e 23 ilustram estas configurações a partir do portal.

Figura 21 - Registo do dispositivo no portal da Apple

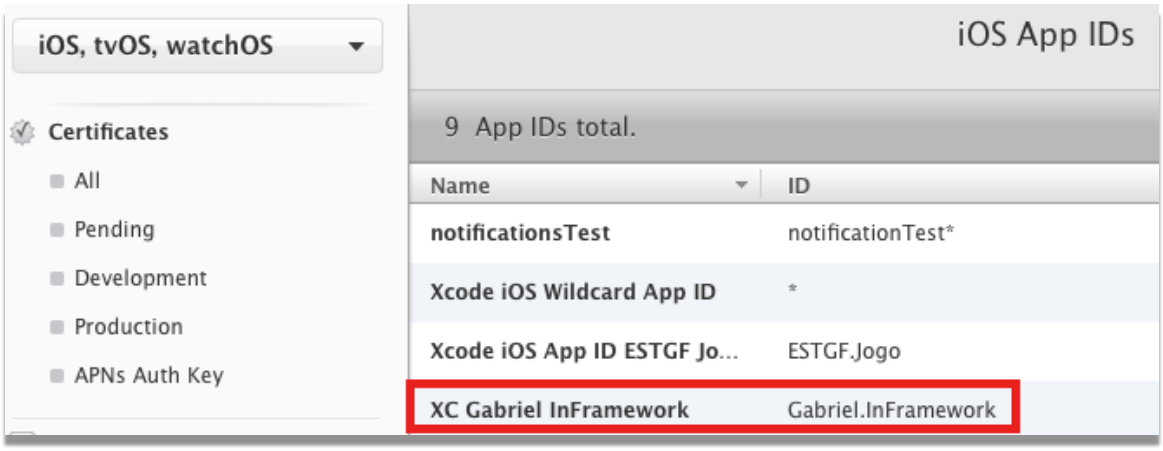


Figura 22 - Registo da aplicação agente

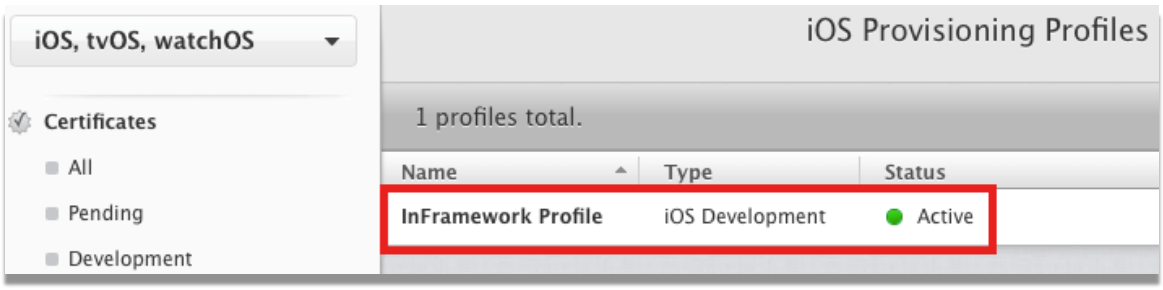


Figura 23 - Perfil de provisionamento

5.5. Implementação do Web Service

Tal como foi apresentado no capítulo 4, para a implementação do WS, foi utilizada a arquitetura REST. Desta forma conseguimos tirar partido do protocolo HTTP e dos seus métodos para facilitar o envio de dados do agente para o WS.

Foi criado um ficheiro denominado `webserviceInFW.php`, onde foram implementados vários blocos de código PHP que definem a sequência de passos executados pelo WS durante o processo de receção, tratamento e armazenamento dos dados.

De forma a tornar menos massudo ao leitor, serão apresentados aqui alguns pontos importantes desta implementação, deixando o resto para ser consultado em mais detalhes no Apêndice D.

O primeiro passo foi identificar, validar e obter o pacote de dados com o JSON enviado pelo agente, que contem no cabeçalho da mensagem a indicação `postValues`. Neste passo, foram utilizadas as funções do PHP (PHP Group, 2018) `fopen`, `fgets` e `json_decode`, para obter um apontador e descodificar os dados.

```php
<?php
/* Verifica se o cabeçalho da mensagem enviada
pela app Agente contém o valor "postValues" */
if ($_SERVER['HTTP_METHOD'] === 'postValues') {

    $body;

    /* Verifica e obtem o conteudo JSON enviado */
    if($_POST == null){
      $handle  = fopen('php://input', 'r');
      $rawData = fgets($handle);
      $body = json_decode($rawData);
    } else {
      $body == $_POST;
    }

    //Continuação ...

```

Figura 24 - Receção dos dados no WS

O passo seguinte foi criar e configurar a conexão com a base de dados onde será armazenada as informações dos dispositivos. Para isso foi necessário utilizar a função `mysql_connect` asd (PHP Group, 2018), indicando-lhe o servidor, o *username* e a *password*. Esta função abre uma conexão com um servidor MySQL utilizando a informação que lhe é passada como parâmetro e no final devolve um *link* desta conexão. Depois disso, foi necessário selecionar a base de dados utilizando o método `mysql_select_db`.

```php
    //Conexão com a base de dados
    $username = "gabriel";

    $password = "gabriel0102";

    $database = "dmpereir_gabriel";

    $link = mysql_connect('localhost', $username, $password);

    @mysql_select_db($database) or die ("Unable to find database");

    //Continuação ...

```

Figura 25 - Conexão à base de dados do módulo de inventário

Tendo em mãos os dados enviados e a conexão com a base de dados, o passo seguinte foi validar que o processo de descodificação correu com sucesso e fazer o mapeamento de

toda a informação contida no JSON, de forma a preparar os pedidos de inserção na base de dados.

```
31      //Verifica o processo de "decode" do JSON recebido
32      switch (json_last_error())
33      {
34              case JSON_ERROR_NONE:
35
36                  if(count($body) > 0)
37                  {
38
39                      //Mapeamento dos dados contidos no JSON
40                      $date = $body ->{'date'};
41                      $physicalm = $body ->{'physicalm'};
42                      $localizemodel = $body ->{'localizemodel'};
43                      $userm = $body ->{'userm'};
44                      $wired = $body ->{'wired'};
45
46                      //Continuação ...
```

Figura 26 - Mapeamento dos dados

Por fim, foi necessário construir e executar uma série de *statements* contendo os pedidos de inserção nas tabelas `InvDevice`, `InvDeviceStatus`, `InvRunningProcess` e `InvInstalledApps`. Desta forma conseguimos garantir que todas as entradas nas tabelas correspondam a cada um dos elementos contidos no JSON enviado pelo agente, durante o processo de recolha de informação no dispositivo.

```
78          //Verifica se e a primeira vez que recebe dados deste dispositivo
79          if($num == 0)
80          {
81
82              //Insere na tabela InvDevice
83              $insertInvDevice_query = "INSERT INTO InvDevice VALUES ('',
84              '$wifimac','$model',
85              '$sysname','$tdiskspace',
86              '$physicalm','$buetmac')";
87
88              mysql_query($insertInvDevice_query);
89              echo mysql_error($link);
90          }
```

Figura 27 -Inserção dos dados nas tabelas

Com isso, conclui-se a apresentação dos passos elaborados durante a implementação desta prova de conceito. Sendo este um processo de elevado conteúdo técnico, tentou-se dar mais ênfase aos pontos essenciais, seguindo uma sequência lógica para que no final o leitor possa ter uma ideia clara não só da complexidade e do funcionamento do módulo, mas também da importância de implementação de uma *framework* deste nível.

No capítulo seguinte será feito uma avaliação do sistema, fazendo uma analogia dos resultados obtidos nesta prova de conceito com os objetivos propostos inicialmente.

6. AVALIAÇÃO DO SISTEMA

Neste capítulo, faz-se uma avaliação do sistema implementado, começando pelos resultados da recolha de dados, até o armazenamento final na base de dados do módulo de inventário. Os resultados desta avaliação foram obtidos a partir da execução de *queries,* utilizando o motor de base de dados MySQL.

A seguir, é feita uma avaliação dos recursos computacionais e consumo de energia que a aplicação agente exerce sobre os dispositivos onde foram instalados para testes. Para o efeito, foram utilizados os indicadores fornecidos pelo *debuger* do Xcode que apresentam os valores de consumo de memória, CPU, energia, quando a aplicação é executada a partir deste IDE.

Mais uma vez, é importante sublinhar que foi dado mais ênfase aos pontos-chave desta implementação, como forma a facilitar a leitura e compreensão do leitor.

6.1. Característica dos Dispositivos

Para efeitos de teste e avaliação preliminar, a aplicação agente foi instalada em dois dispositivos da marca Apple com as seguintes características (GSMArena, 2018):

Modelo	**Apple iPhone 4**
Lançamento	**Anunciado:** Junho de 2010
	***Status*:** Disponível desde Junho de 2010
Network	**Tecnologia:** GSM / HSPA
Plataforma	**SO:**iOS 4, *upgrade* até iOS 7.1.2
	***Chipset*:** Apple A4
	CPU:1.0 GHz Cortex-A8
	GPU: PowerVR SGX535
Disco	***Slot* para cartão:** n/a
	Interna: 8/16/32 GB
Memória	**RAM:** 512 MB
Comunicação	**WLAN:**Wi-Fi 802.11 b/g/n, *hotspot*
	Bluetooth:v2.1, A2DP
	GPS: Sim, com A-GPS
	Radio: n/a
	USB: v2.0
Bateria	Bateria Li-Po 1420 mAh não removível
	***Stand-by*:** até 300 h (2G) / até 300 h (3G)
	Conversa:até 14 h (2G) / até 7 h (3G)
	Música:até 40 h

Tabela 5 - Características do dispositivo 1

Modelo	**Apple iPhone 4s**
Lançamento	**Anunciado:** Outubro de 2011
	***Status*: Disponível desde Outubro de 2011**
Network	**Tecnologia:** GSM / CDMA / HSPA / EVDO
Plataforma	**SO:** iOS 5, *upgrade* até iOS 9.3.5
	***Chipset*: Apple A5**
	CPU: Dual-core 1.0 GHz Cortex-A9
	GPU: PowerVR SGX543MP2
Disco	***Slot* para cartão:** n/a
	Interna: 8/16/32/64 GB
Memória	**RAM:** 512 MB
Comunicação	**WLAN:** Wi-Fi 802.11 b/g/n, *hotspot*
	Bluetooth:v4.0, A2DP, LE
	GPS: Sim, com A-GPS, GLONASS
	Radio: n/a
	USB: v2.0
Bateria	Bateria Li-Po 1432mAh (5.3 Wh) não removível
	***Stand-by*: até 200 h (2G) / até 200 h (3G)**
	Conversa: até 14 h (2G) / até 8 h (3G)
	Música: até 40 h

Tabela 6 - Características do dispositivo 2

6.2. Envio de Dados em Formato JSON

A utilização do JSON permitiu ao agente compactar blocos de dados recolhidos num formato leve e estruturado de fácil interpretação/descodificação por parte do WS. Os dados contidos nesta estrutura foram organizados por um conjunto de pares "chave-valor", onde cada valor é identificado por uma chave única dentro da estrutura. Cada um destes pares pode ser designado como elemento simples ou compostos. Nos simples, cada nó valor é constituído por um só elemento. Enquanto nos compostos cada nó valor é constituído por uma subestrutura de pares "chave-valor".

Nas ilustrações seguintes, podemos analisar alguns exemplos de elementos simples adicionados ao JSON. Estes elementos permitiram ao agente etiquetar os dados que possuem

um valor único em cada momento de recolha. Como exemplo destes dados temos: nome do dispositivo, modelo, IP do telemóvel.

Figura 28 - Elementos simples contidos no JSON

Do lado do WS, o acesso a qualquer um destes elementos contidos na lista é feita de forma direta, ou seja, basta pesquisar pela chave pretendida (`devicename`, `ipcell`, etc.), obtém-se o respetivo valor. Por outro lado, a definição de elementos compostos permitiu ao agente agrupar em listas, as informações relativas a processos e aplicações instaladas nos dispositivos. Tratando-se de listas dinâmicas, que na prática armazenam vários objetos do mesmo tipo, do lado do WS, o acesso a todos os objetos foi feito sempre de forma cíclica, garantindo que todos iriam corresponder a uma entrada nas respetivas tabelas.

Figura 29 - Elementos compostos contidos no JSON

Como podemos ver, o elemento `processlist` é constituído por vários grupos de objetos do mesmo tipo, onde cada um tem um conjunto de pares "chave-valor". Neste caso, em cada um destes grupos temos a informação de um processo que esteja a correr no dispositivo no momento da recolha de informação: o nome, o identificador, data de início, estado e a *flag* que indica se é um processo iniciado pelo utilizador ou pelo SO.

Em ambos os exemplos foram ilustrados os elementos JSON no estado cifrado e não cifrado de forma a facilitar a compreensão do leitor.

6.3. Armazenamento dos Dados

Como resultado final do processo de recolha, foram inseridos milhares de registos nas tabelas da base de dados do módulo de inventário mapeados pelo WS. Tal com foi apresentado no capítulo anterior, todas as informações relevantes foram cifradas de forma a garantir a segurança no acesso às mesmas, possibilitando que somente os indivíduos com as devidas

permissões o possam fazer, neste caso, os administradores que tiverem a chave correta para descodificar as referedias informações.

6.3.1. Tabelas resultantes

Ao longo desta subsecção são apresentados o resultado da recolha de dados com recurso à listagem do conteúdo das tabelas.

Tabela `InvDevice`

O comando executado a seguir visa recolher as seguintes informações dos dispositivos: identificador, modelo, endereço MAC da rede sem fios, nome do sistema operativo, espaço total do HD, total da memória RAM, e endereço MAC do *Bluetooth*.

```
SELECT id_device, model, wifimacaddress, sysname, totaldiskspace,
physicalmemory, bluetmacaddress

FROM InvDevice
```

id_device	model	wifimacaddress	sysname	totaldiskspace	physicalmemory	bluetmacaddress
1	KBveEQZQt	xCf1/4iQH2wi5UD0el	rGiJ1YnbKn	kS5cxb8fxRJrK	nxjC5I86wym8Y9A	xjXrftkXj4YBJ61hegw7y9
10	iPhone	02:00:00:00:00:00	iPhone OS	5	504	02:00:00:00:00:ffffffff

Figura 30 - Tabela InvDevice com os dados

Nesta primeira tabela, no qual são armazenadas informações estáticas de cada dispositivo, conseguimos comparar os dados na forma cifrada e *plaintext*, uma vez que só temos dois dispositivos inseridos. Podemos identificar os dispositivos através dos seus identificadores que são 1 e 10 respetivamente. Uma vez inseridos nesta tabela, onde são gerados automaticamente os identificadores, será mais fácil identificá-los nas restantes tabelas que usam o campo `id_device` como chave estrangeira (FK).

Tabela `InvDeviceStatus`

Nesta tabela, o comando executado irá devolver a informação sobre o estado dos dispostivos a partir dos seguintes campos: identificador do dispositivo, versão do sistema operativo, espaço

livre no HD, IP da rede sem fios, memória RAM disponível, ligação via USB (ativa/inativa), nome do dispositivo. São apresentados os primeiros três registos, ordenados de forma ascendente pela chave primária da tabela.

```
SELECT id_device, sysversion, freediskspace, ipwifi, freememory, isUsbPlugged, devicename

FROM InvDeviceStatus

ORDER BY id_status ASC

LIMIT 3
```

id_device	sysversion	freediskspace	ipwifi	freememory	isUsbPlugged	devicename
1	WW3IYtKHry0u7i	m5hRETqCNsQjzbZQI	cb8UQ1PLm9r	VEIuugdCxJoEUGRL	m5hRETqCNsQjzbZQhX	3QDJvIFemd8yq+b
1	WW3IYtKHry0u7i	m5hRETqCNsQjzbZQI	cb8UQ1PLm9r	k3HMMQUiUYQIWKE	m5hRETqCNsQjzbZQhX	3QDJvIFemd8yq+b
1	WW3IYtKHry0u7i	FF+qhzyAyjmunu2fum	RqrpoZY0vkep	xIWmkhIWgIJAWihm(	m5hRETqCNsQjzbZQhX	3QDJvIFemd8yq+b

Figura 31 - Tabela InvDeviceStatus com os dados

Tabela `InvInstalledApps`

Para apresentar as informações sobre as aplicações instaladas, é executado o comando que devolve todos os campos existentes nesta tabela: identificador da aplicação, identificador do dispositivo, nome da aplicação, versão da aplicação e a data de recolha. Esta pesquisa devolve os primeiros três registos, ordenados de forma ascendente pela chave primária da tabela.

```
SELECT * FROM InvInstalledApps

ORDER BY id_app ASC

LIMIT 3
```

id_app ▲ 1	id_device	appname	appversion	date
1	1	u64wlxM1xAzQ9QZESRu4qA==	RqrpoZY0vkepXpkI+F8MIg==	IltRnjI4OWyIpV53glt8ypUJX//LMWA9B07n
2	1	DtrUJg7SPI2fdnI8LHeASw==	RqrpoZY0vkepXpkI+F8MIg==	IltRnjI4OWyIpV53glt8ypUJX//LMWA9B07n
3	1	HoEHbrHPP9ktamxF/gsJHg==	RqrpoZY0vkepXpkI+F8MIg==	IltRnjI4OWyIpV53glt8ypUJX//LMWA9B07n

Figura 32 - Tabela InvInstalledApps com os dados

Tabela InvRunningProcess

À semelhança da tabela anterior, aqui são apresentados todos os campos referentes aos processos em executação nos dispositivos num determinado momento. Esta informação é apresentada pelos campos: identificador do processo, identificador do dispositivo, nome do processo, estado do processo e data de início de execução. Mais uma vez, são apresentados os primeiros três registos, ordenados de forma ascendente pela chave primária da tabela.

```
SELECT * FROM InvRunningProcess

ORDER BY id_process ASC

LIMIT 3
```

id_process 1	id_device	procname	procstat	proctime
1	1	+KdK9SRCF14nXTp7l	FF+qhzyAyjmunu2fum›	mqnf+be31S74ZqLXqPcM9sbBNOcLe
2	1	Uj6zQzyz3k1r9a3jRWl	FF+qhzyAyjmunu2fum›	mqnf+be31S74ZqLXqPcM9sbBNOcLe
3	1	81/Ww6nTeKGXEOU6	FF+qhzyAyjmunu2fum›	mqnf+be31S74ZqLXqPcM9i45tOGY1l

Figura 33 - Tabela InvRunningProcess com os dados

Devido ao volume de dados introduzidos pelo WS, à exceção da primeira tabela as restantes foram obtidas limitando a pesquisa para apresentar somente as três primeiras entradas em cada uma delas. Este primeiro teste visa apresentar o resultado final desta prova de conceito, assegurando que a base de dados deste módulo seja alimentada via WS com as informações recolhidas em cada um dos dispositivos onde o agente estiver instalado.

Por último, é importante também apresentar o número de registos inseriros em cada uma das tabelas do módulo de inventário, no momento da execução destes testes. Os comandos seguintes devolvem o somatório de registos inseridos em cada uma delas:

```
SELECT COUNT (*) FROM InvDevice
```

COUNT(*)
2

Figura 34 - Número de registos na tabela InvDevice

```
SELECT COUNT (*) FROM InvDeviceStatus
```

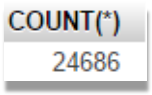


Figura 35 - Número de registos na tabela InvDeviceStatus

```
SELECT COUNT (*) FROM InvInstalledApps
```

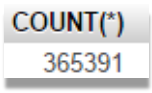


Figura 36 - Número de registos na tabela InvInstalledApps

```
SELECT COUNT (*) FROM InvRunningProcess
```

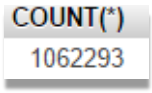


Figura 37 - Número de registos na tabela InvRunningProcess

Desta forma, conseguimos manter a base de dados constantemente atualizada com informações relevantes, permitindo aos administradores e à organização terem uma visão constante sobre as alterações que vão ocorrendo em cada dispositivo e estarem alertas em caso de riscos ou falhas.

6.4. Análise de dados

Como é impossível a compreensão da informação recolhida sem se passar pelo processo de descodificação, para os testes seguintes, serão apresentados os dados em *plaintext*. Deste modo, conseguimos executar pesquisas que simulam o acesso à informação por parte de administradores durante o processo de consulta e gestão dos dispositivos móveis na rede empresarial.

Qual o estado da memória RAM dos dispositivos entre as 16h e as 17h do dia 10/4/2017?

```
SELECT devicename, usermemory, wiredmemory, activememory,
inactivememory, date

FROM InvDeviceStatus

WHERE STR_TO_DATE (date,'%Y-%m-%d %H:%i') >= '2017-04-10 16:00:00'

AND STR_TO_DATE (date,'%Y-%m-%d %H:%i') <= '2017-04-10 17:00:00'
```

devicename	usermemory	wiredmemory	activememory	inactivememory	date
iPhone de Moises	419	38.12 %	39.10 %	17.75 %	2017-04-10 16:04:23 +0000
iPhone de Moises	419	37.81 %	39.03 %	18.03 %	2017-04-10 16:14:23 +0000
iPhone Gabriel	435	31.42 %	38.54 %	18.12 %	2017-04-10 16:23:50 +0000
iPhone Gabriel	436	31.02 %	38.42 %	19.65 %	2017-04-10 16:23:53 +0000
iPhone de Moises	419	37.97 %	38.73 %	17.38 %	2017-04-10 16:24:24 +0000
iPhone Gabriel	444	25.00 %	43.66 %	28.89 %	2017-04-10 16:32:12 +0000
iPhone de Moises	419	37.80 %	37.72 %	19.07 %	2017-04-10 16:34:24 +0000
iPhone de Moises	418	39.57 %	37.56 %	17.12 %	2017-04-10 16:40:11 +0000
iPhone Gabriel	444	25.54 %	44.80 %	26.18 %	2017-04-10 16:42:15 +0000
iPhone Gabriel	444	25.49 %	46.35 %	22.62 %	2017-04-10 16:42:43 +0000
iPhone de Moises	418	38.68 %	36.82 %	17.12 %	2017-04-10 16:49:56 +0000
iPhone Gabriel	435	29.05 %	41.76 %	22.73 %	2017-04-10 16:50:17 +0000
iPhone Gabriel	439	29.70 %	43.20 %	23.59 %	2017-04-10 16:55:43 +0000
iPhone de Moises	418	38.78 %	37.09 %	17.43 %	2017-04-10 16:58:24 +0000

Figura 38 - Estado da memória dos dispositivos

Quais são os dispositivos que estiveram conectados ao mesmo tempo via USB e pela rede de terceira geração (3G) dentro da rede empresarial entre as 11h e as 18h do dia 9/4/2017?

```
SELECT ipcell, ipwifi, isUsbPlugged, is3GNetworkActivated,
devicename, date

FROM InvDeviceStatus

WHERE is3GNetworkActivated = 1

AND isUsbPlugged = 1

AND STR_TO_DATE (date,'%Y-%m-%d %H:%i') >= '2017-04-09 11:00:00'
```

```
AND STR_TO_DATE (date,'%Y-%m-%d %H:%i') <= '2017-04-09 18:00:00'
```

ipcell	ipwifi	isUsbPlugged	is3GNetworkActivated	devicename	date
10.81.130.210	192.168.1.66	1	1	iPhone Gabriel	2017-04-09 11:02:51 +0000
10.81.130.210	192.168.1.66	1	1	iPhone Gabriel	2017-04-09 11:12:54 +0000
10.81.130.210	192.168.1.66	1	1	iPhone Gabriel	2017-04-09 11:22:57 +0000
10.81.130.210	192.168.1.66	1	1	iPhone Gabriel	2017-04-09 11:33:01 +0000
10.81.130.210	192.168.1.66	1	1	iPhone Gabriel	2017-04-09 17:56:32 +0000

Figura 39 - Dispositivos conectados via USB e 3G

Quais são as últimas dez entradas de aplicações instaladas no dispositivo com o identificador 10?

```
SELECT id_app, id_device, appname, appversion, date

FROM InvInstalledApps

WHERE id_device = 10

ORDER BY date DESC

LIMIT 10
```

id_app	id_device	appname	appversion	date
289516	10	iTunes	N/A	2017-04-11 17:16:21 +0000
289508	10	Phone	N/A	2017-04-11 17:16:21 +0000
289514	10	Music iphone	N/A	2017-04-11 17:16:21 +0000
289520	10	Instagram	N/A	2017-04-11 17:16:21 +0000
289517	10	App Store	N/A	2017-04-11 17:16:21 +0000
289512	10	Maps	N/A	2017-04-11 17:16:21 +0000
289513	10	Mail	N/A	2017-04-11 17:16:21 +0000
289510	10	YouTube	N/A	2017-04-11 17:16:21 +0000
289515	10	Videos	N/A	2017-04-11 17:16:21 +0000
289511	10	Safari	N/A	2017-04-11 17:16:21 +0000

Figura 40 - Últimas aplicações instaladas no dispositivo 2

Quais são os identificadores dos dispositivos e detalhes dos processos WhatsApp, Facebook, InFramework ou Viber que foram executados na rede empresarial a partir do dia 18/3/2017?

```
SELECT * FROM InvRunningProcess

WHERE procname

IN ('WhatsApp','Facebook','InFramework','Viber')

AND STR_TO_DATE(date,'%Y-%m-%d %H:%i') >= '2017-03-18 00:00:00'

LIMIT 8
```

id_process	id_device	procname	procstat	procflag	proctime	date
901712	1	Viber	2	16448	2017-03-19 19:13:04	2017-03-21 20:40:36 +0000
901737	10	WhatsApp	2	16384	2017-03-21 20:06:19	2017-03-21 20:40:36 +0000
901739	10	Facebook	2	16448	2017-03-21 20:39:19	2017-03-21 20:40:36 +0000
901744	10	InFramework	2	18432	2017-03-21 20:40:26	2017-03-21 20:40:36 +0000
902105	1	Viber	2	16384	2017-03-28 14:10:26	2017-03-29 23:09:54 +0000
902109	1	Facebook	2	16384	2017-03-29 15:16:32	2017-03-29 23:09:54 +0000
902111	1	WhatsApp	2	16384	2017-03-29 16:20:47	2017-03-29 23:09:54 +0000
902134	1	InFramework	2	18432	2017-03-30 00:09:46	2017-03-29 23:09:54 +0000

Figura 41 - Informação de processos a correr

6.5. Consumo de Energia e Recursos Computacionais

Devido ao modo de funcionamento peculiar do agente, foi necessário identificar o momento em que este atinge o seu ponto máximo de utilização dos recursos disponíveis. Este momento foi registado durante a fase de cifra e envio dos dados ao WS, por isso, os testes realizados tiveram o foco nestes picos de utilização de recursos e nesta secção são apresentados os registos mais elevados.

Foi também necessário testar o comportamento da aplicação durante intervalos de tempo maiores com o objetivo de garantir que durante o período de inatividade não são registados quaisquer consumos extras quer de memória, CPU ou energia. De salientar que para estes testes o agente foi configurado para executar a recolha em intervalos de 10 minutos,

ou seja, o agente permaneceu inativo durante aproximadamente 10 minutos entre cada ciclo de recolha de dados.

6.5.1. Memória RAM

Foi possível notar que nos primeiros 3 segundos de execução o agente consegue atingir o seu ponto máximo de utilização de memória RAM, alcançando uma percentagem máxima registada de 0.85% equivalente a 4,3 MB, num dispositivo com cerca de 512 MB de memória total.

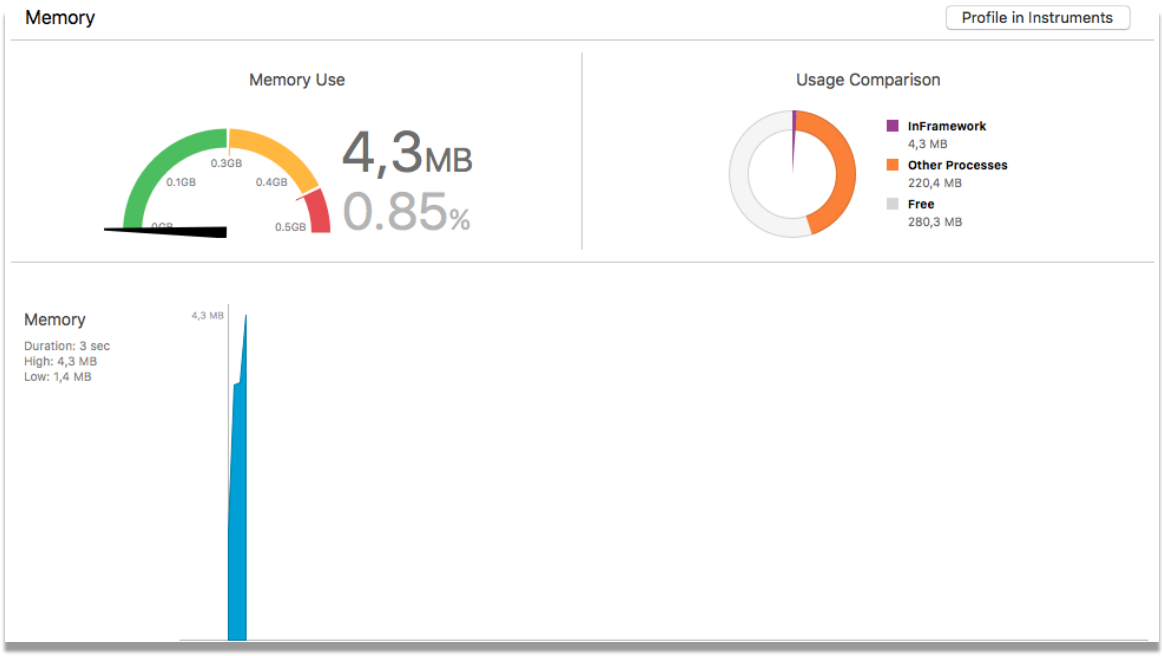


Figura 42 - Consumo de RAM momento de envio de dados

Nos instantes seguintes, o valor da utilização máxima da memória RAM permanece praticamente inalterada até o próximo ciclo de execução. Como se pode analisar na ilustração seguinte, após um período de aproximadamente 30 minutos de utilização (equivalentes a três ciclos de recolha e envio de dados) os valores registados permaneceram num intervalo de 1,4 MB para o consumo mais baixo e 4,4 MB para o maior consumo obtido.

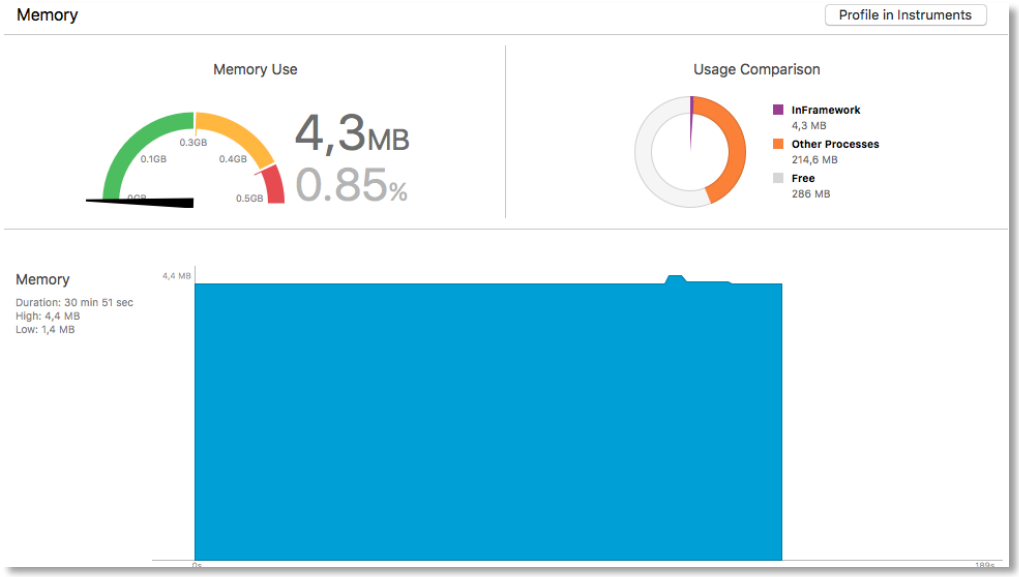


Figura 43 - Consumo de RAM intervalo maiores de execução

6.5.2. CPU

Como era de esperar, em termos de utilização do CPU o pico de consumo é muito semelhante aos registos recolhidos nos testes da memória RAM. No intervalo de 0 a 3 segundos, o agente conseguiu atingir um nível de 14% de consumo, tendo este valor mantido inalterado após alguns ciclos de recolha de dados.

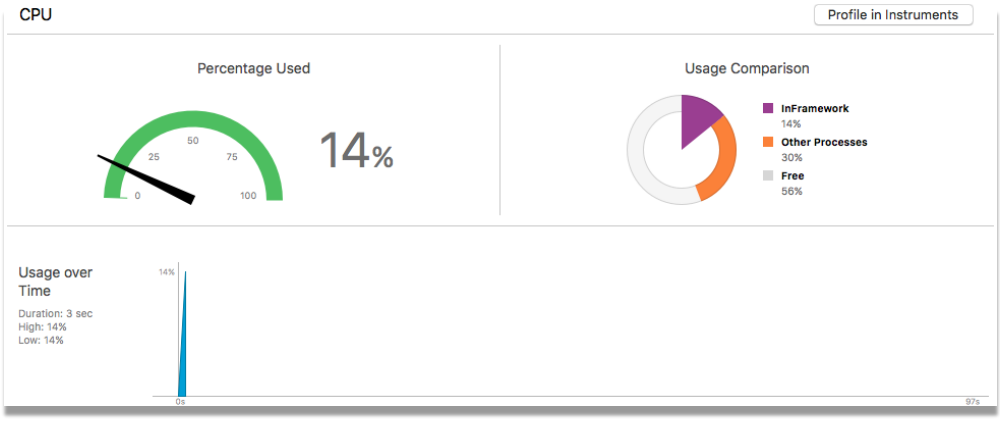


Figura 44 - Consumo CPU momento de envio de dados

Para intervalos de tempo maiores, foram registados picos maiores, chegando o agente a alcançar um consumo de cerca de 25% do CPU em intervalos de execução de aproximadamente 1 hora 45 minutos.

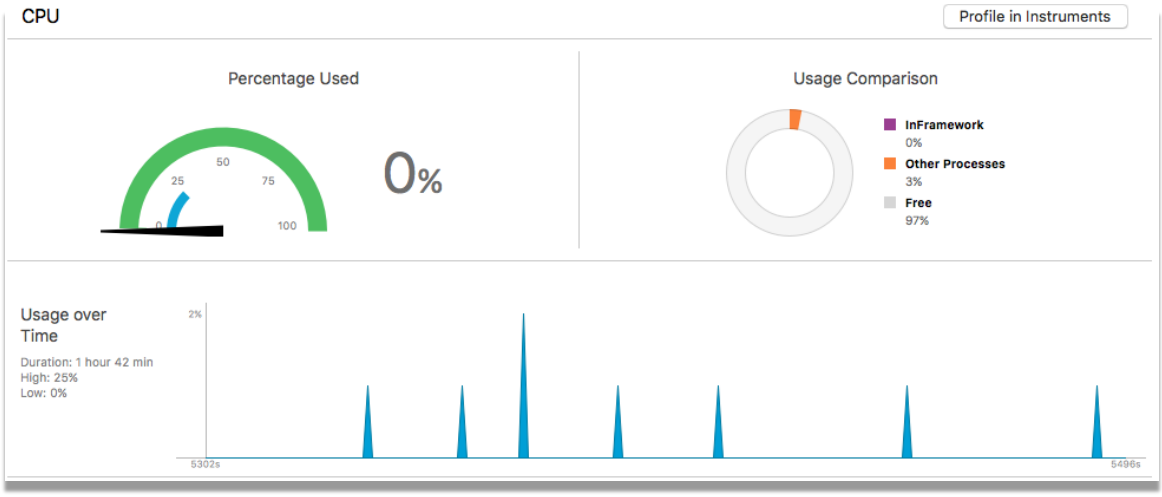


Figura 45 - Consumo CPU intervalo maiores de execução

Como podemos ver, o modo de funcionamento do agente permite-lhe libertar a utilização do CPU na sua totalidade, logo após o envio de dados, permitindo também ao SO ter mais recursos disponíveis para alocar a outros processos, caso for necessário.

6.5.3. Energia

A nível de consumos de energia, foi possível notar que o processo de cifra e envio de dados teve um impacto elevado nos dispositivos onde foi instalado. Tendo como base o mesmo tempo de execução (0 a 3 segundos), o agente conseguiu atingir o ponto máximo, apresentando uma média de 29% de custos de utilização da energia do dispositivo durante a operação. Como podemos ver, nestas duas fases (cifra e envio) o agente faz maior uso do CPU e da *network*, ditando assim os consumos significativos de energia. A ilustração seguinte apresenta o resultado deste teste.

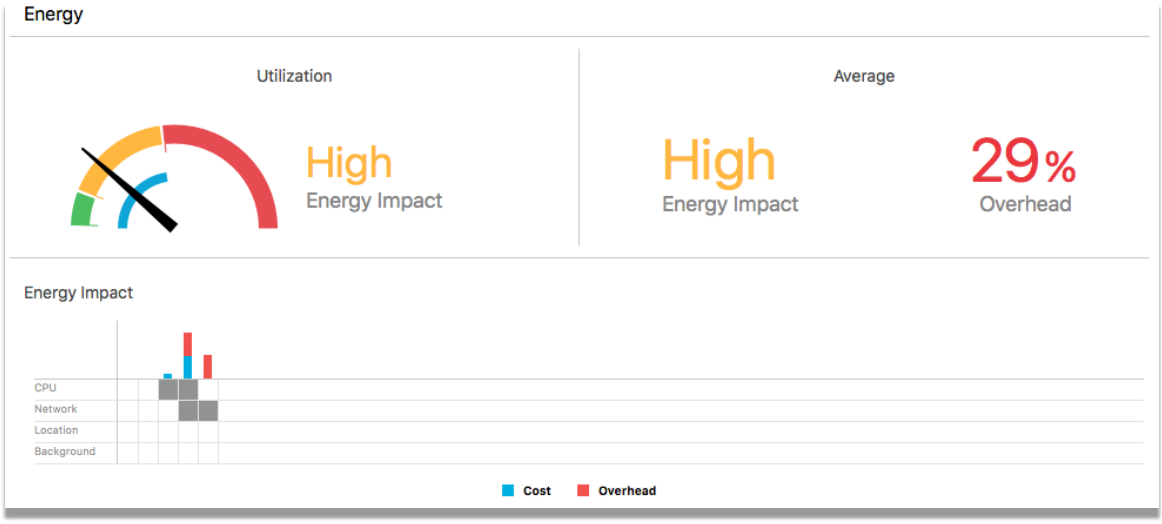


Figura 46 - Consumo de energia momento de envio de dados

Durante a fase de inatividade, como era de esperar, o valor do impacto da utilização de energia passou para zero, diminuindo também a média que passou para "baixo". Contudo, foi registado cerca de 2% de custos que de forma indireta estão relacionados com a utilização da *network* para a execução do envio de dados.

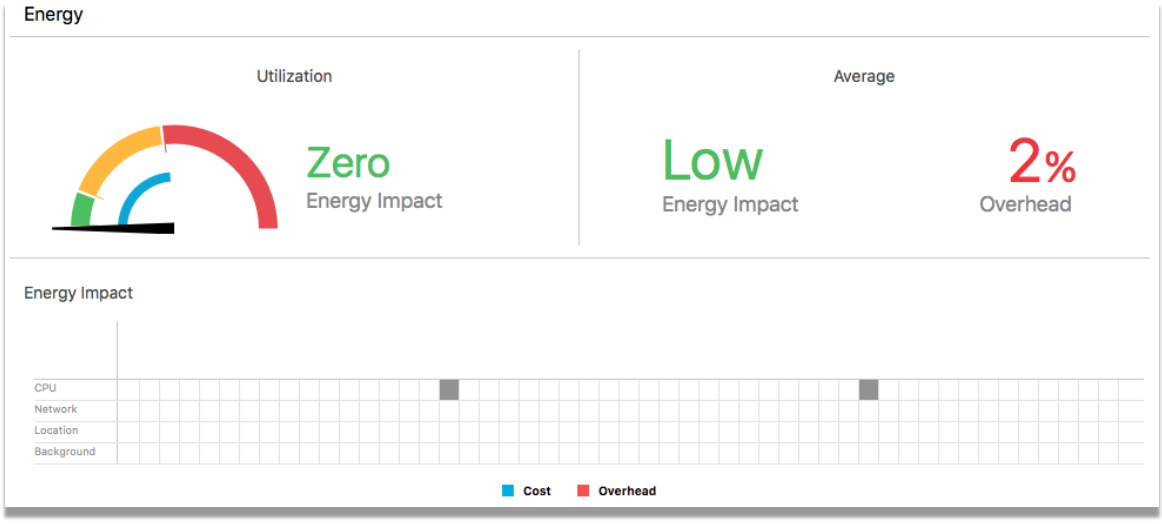


Figura 47 - Consumo de energia intervalos maiores de execução

Existem custos substanciais de consumo de energia sempre que as aplicações recorrem ao uso da *network* para executarem operações. No nosso caso, o agente faz o uso de qualquer um dos recursos disponíveis no processo de análise à rede na qual o dispositivo se encontra ligado a cada momento (rede móvel, 3G ou Wi-Fi). Por norma, estes recursos estão desligados para conservar ao máximo o tempo de vida de bateria dos dispositivos. Quando ligados, permanecem ativos não só durante o período em que são executadas as operações, mas também por um período adicional como forma de antecipar a necessidade de execução de mais tarefas, acabando por se traduzir muitas vezes em maiores consumos de energia do dispositivo (Apple, 2018).

7. CONCLUSÕES E TRABALHO FUTURO

As empresas estão cada vez mais consciencializadas dos benefícios e vantagens competitivas da mobilidade. A grande maioria identifica no conceito de BYOD um aumento da produtividade e redução de custos. Contudo este conceito traz novos desafios de gestão dos dispositivos móveis dentro das organizações e levanta várias questões ao nível de segurança da informação e gestão de dispositivos. Existem muitas soluções de gestão de dispositivos (MDM) no mercado que fornecem funcionalidades como inventariação, monitorização, controlo, segurança dos dispositivos e dados empresariais. Porém, algumas análises identificam-nas como sendo soluções fechadas e por vezes de difícil integração com o resto da infraestrutura.

Tendo em conta a quantidade e experiência de utilização com sistemas tradicionais, consideramos que estas soluções não devem ser disruptivas, mas sim de fácil integração e uso. Neste contexto consideramos um *framework* de gestão de dispositivos móveis modular baseada em Web Services e como prova de conceito foi implementado o módulo de inventário que servirá de base para o desenvolvimento dos restantes módulos que compõem esta *framework*.

Após a fase da escolha das metodologias ferramentas e plataformas, seguiu-se para a implementação do agente em iOS, responsável pela recolha e envio dos dados. Durante toda a etapa desta prova de conceitos, sem dúvida, esta foi a fase de aplicação de mais esforços, uma vez que era imprescindível a interação com tecnologias e conceitos que até então eram novos para mim. Foram necessárias muitas horas de pesquisa, estudos e testes de cada uma das funcionalidades implementadas, para garantir que no final o resultado fosse o esperado. Apesar de os conhecimentos adquiridos durante a minha formação académica contribuírem bastante para implementação de toda a lógica e codificação das funcionalidades, muitas vezes, as dificuldades surgiam durante os processos de configuração de aspetos essenciais para o correto funcionamento da aplicação. Um dos pontos a salientar durante este processo foi a configuração do agente para funcionar em modo *backgound*, que era um dos requisitos chave definidos para a *framework*.

Feito isso, a fase seguinte foi a criação da base de dados usando MySQL e implementação do Web Service em PHP responsável por receber e armazenar toda a informação recolhida pelo agente. Ultrapassados alguns obstáculos iniciais, sobre a utilização de um servidor na *cloud* onde seria possível alojar a base de dados e o WS, os passos seguintes foram menos turbulentos. Esta fase não só serviu para testar cada um dos

componentes em si, mas também para consolidar a filosofia e o modo de funcionamento do módulo de inventário.

Com a implementação deste sistema de inventário, enquadrada na *framework* de gestão de dispositivos móveis conseguimos demonstrar:

- A facilidade de integração com outros sistemas existentes – a utilização de WS permitiu a compatibilidade de aplicações desenvolvidas em plataformas diferentes e garante que este sistema possa interagir de forma eficiente com outros sistemas de inventário, monitorização, bases de dados e dispositivos móveis existentes nas organizações;
- A possibilidade de obter relatórios globais – permitiu facultar uma vasta quantidade de informação sobre a utilização dos dispositivos dentro da empresa; filtrar/cruzar informações e obter as marcas ou modelos dos dispositivos; o estado e recursos disponíveis; aplicações que a maioria dos funcionários utiliza, etc. Esta informação possibilita por exemplo, uma análise sobre a exposição ao risco quando é detetada uma aplicação malcomportada;
- Simplicidade em termos de *enrollment* com os dispositivos – uma vez instalada a aplicação agente não foi necessário efetuar mais configurações nos dispositivos.

Para além dos resultados alcançados, conseguiu-se que a aplicação agente consumisse o mínimo de recursos possíveis nos dispositivos, mantendo o seu funcionamento por um tempo indeterminado, sem ser interrompido pelo sistema operativo.

7.1. Trabalho futuro

Chegado a esta fase, consegui-se perceber que ainda há um longo caminho a percorrer e muito trabalho a fazer. Para começar, seria interessante dotar a aplicação agente com a capacidade de reajustar os intervalos de inatividade consoante os recursos disponíveis. Após a recolha da informação, o agente poderia ter uma fase intermédia onde seria possível analisá-la e decidir se é possível diminuir ou aumentar o intervalo de inatividade. Desta forma, conseguir-se-ia evitar consumos de processamento ou autonomia, durante períodos em que o dispositivo se encontrar com sobrecarga de processos.

O desenvolvimento para outros sistemas móveis seria o passo a seguir. Neste caso, a plataforma Android seria uma excelente escolha, uma vez que é a plataforma com mais quota

de mercado em aplicações e desenvolvimento *mobile*. Durante esta fase seria necessário redefinir a lista de informações a serem recolhidas pela aplicação agente, bem como as informações sobre aplicações e processos em execução nos dispositivos. Sem dúvida, estas são informações relevantes a ter em conta em sistemas deste nível.

Outro trabalho futuro visa o desenvolvimento, nos mesmos moldes, dos outros módulos. Cada um dos módulos endereçará questões específicas relacionadas com a gestão de dispositivos móveis. Nesta altura será importante implementar mecanismos de *geofencing* (Falkowski, Jurgenhake, Anacker, & Dumitrescu, 2018) permitindo que a *framework* tenha conhecimento quando um determinado dispositivo entra ou deixa os perímetros da *network* empresarial.

Posteriormente seria necessário adoptar o agente instalado nos dispositivos com funcionalidades que pudessem apoiar e gerir de forma segura, todos os trabalhos desenvolvidos pelos funcionários. A ideia seria criar um sistema fechado, protegido com um código pin onde todos os conteúdos (documentos, vídeos, fotos, etc.) captados dentro da rede empresarial como os produzidos fora seriam catalogados, cifrados e armazenados. Em caso de detecção de tentativa de acesso não autorizado, perda ou extravio do dispositivo, o agente teria a inteligência de fazer o *wipe* aos dados até então guardados. A imagem seguinte ilustra esta ideia:

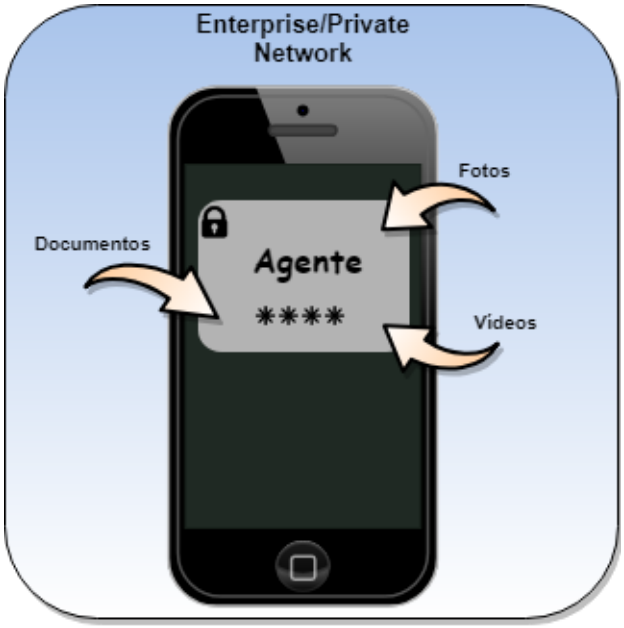


Figura 48 - Aplicação agente - trabalho futuro

O código criado ao longo deste trabalho será disponibilizado para a comunidade para futuros desenvolvimentos e melhorias.

REFERÊNCIAS

ABI Research. (29 de Agosto de 2012). *Microsoft and RIM Seeking to Grab Share from Android for the 2.4 Billion Enterprise Smartphone Subscribers in 2017*. Obtido de ABI Research: https://www.abiresearch.com/press/microsoft-and-rim-seeking-to-grab-share-from-andro/

Al Ayubi, S. U., Pelletier, A., Sunthara, G., Gujral, N., Mittal, V., & Bourgeois, F. C. (2016). A Mobile App Development Guideline for Hospital Settings: Maximizing the Use of and Minimizing the Security Risks of "Bring Your Own Devices" Policies. *JMIR Mhealth Uhealth, 4* (2).

Al-Hazaimeh, A., & Mohammad, O. (2013). A New Approach for Complex Encrypting and Decrypting Data. *International journal of Computer Networks & Communications, 5* (2), 95-103.

Andrews, J., Dark, J., & West, J. (2017). *CompTIA A+ guide to IT technical support* (Ninth edition. ed.). Boston: Cengage Learning.

Andriotis, P., Oikonomou, G., Tryfonas, T., & Li, S. (2016). Highlighting Relationships of a Smartphone's Social Ecosystem in Potentially Large Investigations. *IEEE Trans Cybern, 46* (9), 1974-1985.

Apple. (16 de Dezembro de 2013). *Setting Up Socket Streams*. Obtido de Apple: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/Streams/Articles/NetworkStreams.html

Apple. (16 de Dezembro de 2013). *Setting Up Socket Streams*. Obtido de Apple: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/Streams/Articles/NetworkStreams.html

Apple. (17 de Setembro de 2013). *Using Sockets and Socket Streams*. Obtido de Apple: https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/NetworkingTopics/Articles/UsingSocketsandSocketStreams.html

Apple. (16 de Dezembro de 2013). *Writing To Output Streams*. Obtido de Apple: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/Streams/Articles/WritingOutputStreams.html

Apple. (27 de Março de 2017). *Background Execution*. Obtido de Apple: https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html

Apple. (27 de Março de 2017). *Making HTTP and HTTPS Requests*. Obtido de Apple: https://developer.apple.com/library/archive/documentation/NetworkingInternetWeb/Conceptual/NetworkingOverview/WorkingWithHTTPAndHTTPSRequests/WorkingWithHTTPAndHTTPSRequests.html

Apple. (27 de Março de 2017). *Performance Tips*. Obtido de Apple: https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/PerformanceTips/PerformanceTips.html

Apple. (2018). *Choosing a Membership*. Obtido de Apple: https://developer.apple.com/support/compare-memberships/

Apple. (2018). *Energy and Networking*. Obtido de Apple: https://developer.apple.com/library/archive/documentation/Performance/Conceptual/EnergyGuide-iOS/EnergyandNetworking.html

Apple. (2018). *Function CFStreamCreatePairWithSocketToHost*. Obtido de Apple: https://developer.apple.com/documentation/corefoundation/1539739-cfstreamcreatepairwithsockettoho?language=objc

Apple. (2018). *Function host_statistics*. Obtido de Apple: https://developer.apple.com/documentation/kernel/1502546-host_statistics?language=objc

Apple. (2018). *Measure Energy Impact with Xcode*. Obtido de Apple: https://developer.apple.com/library/archive/documentation/Performance/Conceptual/EnergyGuide-iOS/MonitorEnergyWithXcode.html#//apple_ref/doc/uid/TP40015243-CH34-SW1

Apple. (2018). *Protocol NSStreamDelegate*. Obtido de Apple: https://developer.apple.com/documentation/foundation/nsstreamdelegate

Apple. (2018). *Run Your App On A Device*. Obtido de Apple: https://help.apple.com/xcode/mac/current/#/dev60b6fbbc7

Arakawa, K., Kido, N., Oshita, K., & Tomita, M. (2010). G-language genome analysis environment with REST and SOAP web service interfaces. *Nucleic Acids Res, 38*(Web Server issue), W700-705.

Armstrong, K. A., Semple, J. L., & Coyte, P. C. (2014). Replacing ambulatory surgical follow-up visits with mobile app home monitoring: modeling cost-effective scenarios. *J Med Internet Res, 16*(9), e213.

Ballard, B. (2015). *What you need to know about mobile device management*. Obtido de Betanews: https://betanews.com/2016/01/27/what-you-need-to-know-about-mobile-device-management/

BBVA API Market. (23 de Março de 2016). *REST API: What is it, and what are its advantages in project development?* Obtido de BBVA API Market: https://bbvaopen4u.com/en/actualidad/rest-api-what-it-and-what-are-its-advantages-project-development

Baua'a, M. M., Hailan, A. M., & Srayyih, M. N. (2014). Using MPI to Improve ROA&REST Transfer Data Services. *Procedia Computer Science, 37*, 153-159.

Ben.B. (16 de Março de 2017). *Mobile Device Management (MDM) to be removed From Spiceworks Inventory*. Obtido de Spiceworks: https://community.spiceworks.com/topic/1973241-mobile-device-management-mdm-to-be-removed-from-spiceworks-inventory

Bhanot, R., & Hans, R. (2015). A Review and Comparative Analysis of Various Encryption Algorithms. *International Journal of Security and Its Applications, 9*(4), 289-306.

Black Box Network Services. (2010). The Basics of NAC - Network Access Control.

Business. (26 de Outubro de 2018). *Best Mobile Device Management (MDM) Solutions Buying Guide*. Obtido de Business: https://www.business.com/categories/mobile-device-management-solutions/

Chan, B. (22 de Abril de 2014). *Mobile Device Management vs. Mobile Application Management: Which Way is Best?* Obtido de IBM Community: https://www.ibm.com/developerworks/community/blogs/025bf606-020a-48e9-89bf-99adda13e9b1/entry/mobile_device_management_vs_mobile_application_management_which_way_is_best?lang=en

Chang, J. (19 de Fevereiro de 2015). *5 Hidden Costs of MDM*. Obtido de Workspot: http://blog.workspot.com/5-hidden-costs-of-mdm-you-should-consider-before-purchase

Chernyshov, A., Balandina, A., Kostkina, A., & Klimov, V. (2016). Intelligence Search Engine and Automatic Integration System for Web-Services and Cloud-Based Data Pro-viders Based on Semantics. *Procedia Computer Science, 88*, 272-276.

Choo, K. K. R., & Dehghantanha, A. (2017). *Contemporary digital forensic investigations of cloud and mobile applications*. Amsterdam ; Boston ; Heidelberg ; London ; New York ; Oxford ; Paris ; San Diego ; San Francisco ; Singapore ; Sydney ; Tokyo: Elsevier : Syngress.

Cisco. (28 de Março de 2017). *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper*. Obtido de Cisco: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

Collins, L., & Ellis, S. (2015). *Mobile devices: tools and technologies*. Boca Raton, Florida: CRC Press/Taylor & Francis Group.

Colton, S., & Hunt, L. (2016). Developing a smartphone app to support the nursing community. *Nurs Manag (Harrow), 22*(9), 24-28.

Coons, S. J., Eremenco, S., Lundy, J. J., O'Donohoe, P., O'Gorman, H., & Malizia, W. (2015). Capturing Patient-Reported Outcome (PRO) Data Electronically: The Past, Present, and Promise of ePRO Measurement in Clinical Trials. *Patient, 8*(4), 301-309.

Crabtree, A. (24 de Julho de 2012). *iOS SDK: Accessing Device Data with UIDevice and NSLocale*. Obtido de Envato Tuts: https://code.tutsplus.com/tutorials/ios-sdk-accessing-device-data-with-uidevice-and-nslocale--mobile-11511

Dallimore, D. (2018). *Getting data from your REST APIs into Splunk*. Obtido de Splunk: https://www.splunk.com/blog/2013/06/18/getting-data-from-your-rest-apis-into-splunk.html

Dang-Pham, D., & Pittayachawan, S. (2015). Comparing intention to avoid malware across contexts in a BYOD-enabled Australian university: A Protection Motivation Theory approach. *Computers & Security, 48*, 281-297.

De las Cuevas, P., Mora, A. M., Merelo, J. J., Castillo, P. A., García-Sánchez, P., & Fernández-Ares, A. (2015). Corporate security solutions for BYOD: A novel user-centric and self-adaptive system. *Computer Communications, 68*, 83-95.

De Luca, V., Epicoco, I., Lezzi, D., & Aloisioa, G. (2012). GRB_WAPI, a RESTful Framework for Grid Portals. *Procedia Computer Science, 9*, 459-468.

Dhingra, M. (2016). Legal Issues in Secure Implementation of Bring Your Own Device (BYOD). *Procedia Computer Science, 78*, 179-184.

Disterer, G., & Kleiner, C. (2013). BYOD Bring Your Own Device. *Procedia Technology, 9*, 43-53.

Downer, K., & Bhattacharya, M. (2015). BYOD Security: A New Business Challenge.

Falconer, J., Gray, S., & Gaul, K. (2014). Bring your own device into problem based learning tutorials. *Med Teach, 36*(12), 1086-1087.

Falkowski, T., Jurgenhake, C., Anacker, H., & Dumitrescu, R. (2018). Feature model for the specification of industrial indoor location-based services. *Procedia Manufacturing, 24*, 141–146.

Faulds, M. C., Bauchmuller, K., Miller, D., Rosser, J. H., Shuker, K., Wrench, I., . . . Research, C. (2016). The feasibility of using 'bring your own device' (BYOD) technology for electronic data capture in multicentre medical audit and research. *Anaesthesia, 71*(1), 58-66.

Fielding, R. T. (2000). *Representational State Transfer (REST)*. Obtido de Donald Bren School of Information and Computer Sciences : https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

FinancesOnline. (2018). *20 Best Mobile Device Management Software in 2018*. Obtido de FinancesOnline: https://financesonline.com/mobile-device-management/

Fleck, C. (7 de Agosto de 2013). *The Mobile Equation: EMM=M(D+A+I+E)M= Safe(BYOD+COPE+LDD) +Productive(users)*. Obtido de Citrix: https://www.citrix.com/blogs/2012/12/05/the-mobile-equation-emmmdaiem-safebyodcopeldd-productiveusers/

G2Crowd. (2018). *Best Mobile Device Management (MDM) Software*. Obtido de G2Crowd: https://www.g2crowd.com/categories/mobile-device-management-mdm

Gadbury-Amyot, C. C., Purk, J. H., Williams, B. J., & Van Ness, C. J. (2014). Using tablet technology and instructional videos to enhance preclinical dental laboratory learning. *J Dent Educ, 78* (2), 250-258.

Galera, E., Kristine Mae, & Llantos, E., Orven (2017). Mobile Web Energy Monitoring System Using DFRduino Uno. *Procedia Computer Science, 124*, 706–713.

Gallagher, M. (8 de Março de 2016). *Gathering system information in Swift with sysctl*. Obtido de Cocoa with Love: https://www.cocoawithlove.com/blog/2016/03/08/swift-wrapper-for-sysctl.html

Gartner. (1 de Maio de 2013). *Gartner Predicts by 2017, Half of Employers will Require Employees to Supply Their Own Device for Work Purposes*. Obtido de Gartner: http://www.gartner.com/newsroom/id/2466615

Gartner. (13 de Janeiro de 2014). *Gartner Says Less Than 0.01 Percent of Consumer Mobile Apps Will Be Considered a Financial Success by Their Developers Through 2018*. Obtido de Gartner: https://www.gartner.com/newsroom/id/2648515

Gartner. (6 de Junho de 2017). *Magic Quadrant for Enterprise Mobility Management Suites*. Obtido de Gartner: https://www.gartner.com/doc/reprints?id=1-42A6QBL&ct=170607&st=sb

Genevey, S., & Dominguez, C. (20 de Julho de 2017). *Improving Battery-Powered Device Operation Time Thanks To Power Efficient Sleep Mode*. Obtido de Design & Reuse: https://www.design-reuse.com/articles/42405/improving-battery-powered-device-operation-time-thanks-to-power-efficient-sleep-mode.html

GitHub. (2018). *ARC and GCD Compatible Reachability Class for iOS and MacOS.* . Obtido de GitHub: https://github.com/tonymillion/Reachability

Gonzalez, A. R., Callahan, A., Cruz-Toledo, J., Garcia, A., Egana Aranguren, M., Dumontier, M., & Wilkinson, M. D. (2014). Automatically exposing OpenLifeData via SADI semantic Web Services. *J Biomed Semantics, 5*, 46.

Goshwe, Y., Nentawe. (2013). Data Encryption and Decryption Using RSA Algorithm in a Network Environment. *13*.

Google. (2018). *Geofencing API*. Obtido de Google: https://developers.google.com/location-context/geofencing/

GSMArena. (2018). *Apple iPhone 4 vs 4s* . Obtido de GSMArena: https://www.gsmarena.com

Grahl, M., Bluhm, T., Grün, M., Hennig, C., Holtz, A., Krom, J. G., . . . Werner, A. (2017). Archive WEB API: A web service for the experiment data archive of Wendelstein 7-X. *Fusion Engineering and Design, 123*, 1015-1019.

Hamann, R. (18 de Agosto de 2014). *iOS, Android e Windows Phone: números dos gigantes comparados* . Obtido de Tecmundo: https://www.tecmundo.com.br/sistema-operacional/60596-ios-android-windows-phone-numeros-gigantes-comparados-infografico.htm

Hess, K. (11 de Junho de 2012). *10 BYOD mobile device management suites you need to know*. Obtido de ZDNet: https://www.zdnet.com/article/10-byod-mobile-device-management-suites-you-need-to-know/

Hewlett-Packard. (2017). Security Begins At The Endpoint. 2-7.

Hughes, J. (2015). Bring your own device or bring your own disruption? How to handle the mulitude of challenges. *Health Manag Technol, 36* (4), 14-15.

Intermec Technologies Corporation. (2009). Lowering Total Cost of Ownership Through Mobile Device Management.

Jamal, S., & Deters, R. (2011). Using a Cloud-Hosted Proxy to support Mobile Consumers of RESTful Services. *Procedia Computer Science, 5*, 625-632.

Joshi, V. (18 de Janeiro de 2017). *Why You Should Be Using JSON VS XML*. Obtido de Cloud Elements: https://blog.cloud-elements.com/using-json-over-xml

Katayama, T., Nakao, M., & Takagi, T. (2010). TogoWS: integrated SOAP and REST APIs for interoperable bioinformatics Web services. *Nucleic Acids Res, 38*(Web Server issue), W706-711.

Keeper Security. (2016). How to Provision Employees in a BYOD World.

Kishore, A. (22 de Abril de 2011). *Monitoring RAM, CPU and Battery Usage On Your iPhone*. Obtido de Switching To Mac: https://www.switchingtomac.com/tutorials/iphone/monitoring-ram-cpu-and-battery-usage-on-your-iphone/

Klein, F., Severijns, C., Albiez, D., Seljutin, E., Jovanovic, M., & Eyvazi Hesar, M. (2016). The Hygiene Games. *Stud Health Technol Inform, 225*, 658-662.

Lewis, N. (6 de Julho de 2016). *A Look at Gartner's Network Access Control Market Guide*. Obtido de Solutions Review: https://solutionsreview.com/wireless-network/a-look-at-gartners-network-access-control-market-guide/

Kusuma, R. (20 de Junho de 2013). *iOS Silent Push notifications*. Obtido de HayaGeek: http://hayageek.com/ios-silent-push-notifications/

Maddineni, V. S. K., & Ragi, S. (2011). Security Techniques for Protecting Data in Cloud Computing.

Madduri, R. K., Sulakhe, D., Lacinski, L., Liu, B., Rodriguez, A., Chard, K., . . . Foster, I. T. (2014). Experiences Building Globus Genomics: A Next-Generation Sequencing Analysis Service using Galaxy, Globus, and Amazon Web Services. *Concurr Comput, 26*(13), 2266-2279.

Mahajan, P., & Sachdeva, A. (2013). A Study of Encryption Algorithms AES, DES and RSA for Security. *Global Journal of Computer Science and Technology Network, Web & Security, 13*.

Martinez, K., Borycki, E., & Courtney, K. L. (2017). Bring Your Own Device and Nurse Managers' Decision Making. *Comput Inform Nurs, 35*(2), 69-76.

Matthews, J. (2017). Challenges to Implementing Network Access Control.

McClean, S., & Crowe, W. (2017). Making room for interactivity: using the cloud-based audience response system Nearpod to enhance engagement in lectures. *FEMS Microbiol Lett, 364* (6).

McDonald, C. (21 de Novembro de 2013). *More than one billion BYOD users predicted by 2018*. Obtido de ComputerWeekly: https://www.computerweekly.com/news/2240209481/More-than-one-billion-BYOD-users-predicted-by-2018

McLean, K. J. (2016). The Implementation of Bring Your Own Device (BYOD) in Primary [Elementary] Schools. *Front Psychol, 7*, 1739.

Medcalf, R., Loucks, J., Buckalew, L., & Faria, F. (2013). O impacto financeiro da consumerização de TI.

Milius, R. P., Heuer, M., George, M., Pollack, J., Hollenbach, J. A., Mack, S. J., & Maiers, M. (2016). The GL service: Web service to exchange GL string encoded HLA & KIR genotypes with complete and accurate allele and genotype ambiguity. *Human Immunology, 77*(3), 249-256.

Miller, M. A., Schwartz, T., Pickett, B. E., He, S., Klem, E. B., Scheuermann, R. H., . . . O'Leary, M. A. (2015). A RESTful API for Access to Phylogenetic Tools via the CIPRES Science Gateway. *Evol Bioinform Online, 11*, 43-48.

Mohamed, K., & Wijesekera, D. (2012). Performance Analysis of Web Services on Mobile Devices. *Procedia Computer Science, 10*, 744-751.

Monteiro, G. M. (2010). Uma implementação baseada no paradigma SOA. 1-62.

Morrow, B. (2012). BYOD security challenges: control and protect your most sensitive data. *Network Security, 2012*(12), 5-8.

Motulsky, A., Wong, J., Cordeau, J. P., Pomalaza, J., Barkun, J., & Tamblyn, R. (2017). Using mobile devices for inpatient rounding and handoffs: an innovative application developed and rapidly adopted by clinicians in a pediatric hospital. *J Am Med Inform Assoc, 24* (e1), e69-e78.

Mouli, V. R., & Jevitha, K. P. (2016). Web Services Attacks and Security- A Systematic Literature Review. *Procedia Computer Science, 93*, 870-877.

Mowafia, Y., Abou-Taira, I., Dhiah el Diehn, Zmilya, A., Al-Aqarbeha, T., Abilovb, M., & Dmitriyevrb, V. (2015). Exploring a Context-based Network Access Control for Mobile Devices. *Procedia Computer Science*.

Muchow, J. (24 de Janeiro de 2014). *iOS 7: Base64 Encode and Decode NSData and NSString Objects*. Obtido de iOS Developer Tips: http://iosdevelopertips.com/core-services/encode-decode-using-base64.html

Namiot, D., & Sneps-Sneppe, M. (2013). Geofence and Network Proximity. 117-127.

National Institute of Standards and Technology. (2009). DISCUSSION PAPER: The Transitioning of Cryptographic Algorithms and Key Sizes.

Nordahl, M., & Magnusson, B. (2015). A lightweight Data Interchange Format for Internet of Things in the PalCom Middleware Framework. *Procedia Computer Science, 56*, 284-291.

Oracle. (2018). *Database Administrator's Guide*. Obtido de Oracle: https://docs.oracle.com/cd/B28359_01/server.111/b28310/tables003.htm#ADMIN11004

Pascucci, M. (Janeiro de 2018). *Comparing the leading mobile device management products*. Obtido de TechTarget: https://searchsecurity.techtarget.com/feature/Comparing-the-best-mobile-device-management-products

PC Magazine. (8 de Janeiro de 2018). *The Best Mobile Device Management (MDM) Solutions of 2018*. Obtido de PC Magazine: https://www.pcmag.com/article/342695/the-best-mobile-device-management-mdm-software

Phifer, L. (Março de 2013). *Mobile device management checklist*. Obtido de TechTarget: https://searchmobilecomputing.techtarget.com/tip/Mobile-device-management-checklist

PHP Group. (2018). *Filesystem Functions*. Obtido de PHP Group: http://php.net/manual/en/function.file-get-contents.php

PHP Group. (2018). *MySQL Functions*. Obtido de PHP Group: http://php.net/manual/en/function.mysql-connect.php

Parra, J., Hossain, M. A., Uribarren, A., & Jacob, E. (2014). RESTful discovery and eventing for service provisioning in assisted living environments. *Sensors (Basel), 14*(5), 9227-9246.

Patel, R. K., Sayers, A. E., Patrick, N. L., Hughes, K., Armitage, J., & Hunter, I. A. (2015). A UK perspective on smartphone use amongst doctors within the surgical profession. *Ann Med Surg (Lond), 4*(2), 107-112.

Payant, R. P. (2016). *Emergency management for facility and property managers*. New York: McGraw-Hill Education.

Pitchaiah, M., Philemon, D., & Praveen. (2012). Implementation of Advanced Encryption Standard Algorithm. *International Journal of Scientific & Engineering Research, 3*(3).

Pugliese, L., Woodriff, M., Crowley, O., Lam, V., Sohn, J., & Bradley, S. (2016). Feasibility of the "Bring Your Own Device" Model in Clinical Research: Results from a Randomized Controlled Pilot Study of a Mobile Patient Engagement Tool. *Cureus, 8* (3), e535.

Qadir, S., & Quadri, S. M. K. (2016). Information Availability: An Insight into the Most Important Attribute of Information Security. *Journal of Information Security, 07*(03), 185-194.

Quizlet. (2018). *Segurança - Confiabilidade. Integridade. Disponibilidade. Autenticidade*. Obtido de Quizlet: https://quizlet.com/25083520/seguranca-confiabilidade-integridade-disponibilidade-autenticidade-flash-cards/

R & G Technologies. (2015). *Mobile Device Management (MDM)*. Obtido de R & G Technologies: https://rgtechnologies.com.au/resources/mobile-device-management/

Rouse, M. (Setembro de 2006). *Network Access Control (NAC)*. Obtido de TechTarget: https://searchnetworking.techtarget.com/definition/network-access-control

Rouse, M. (Julho de 2015). *Endpoint security management*. Obtido de TechTarget: https://searchsecurity.techtarget.com/definition/endpoint-security-management

Redman, P., Girard, J., & Wallin, L. O. (2011). Magic Quadrant for Mobile Device Management Software.

Rieders, L., & Monroy, M. (2014). Creating Your Practice's 'Bring Your Own Device' Policy. *Med Econ, 91*(20), 44-45.

Rose, C. (2013). BYOD: An Examination Of Bring Your Own Device In Business. *The Clute Institute, 17*.

Saha, A., & Sanyal, S. (2015). Review of Considerations for Mobile Device based Secure Access to Financial Services and Risk Handling Strategy for CIOs, CISOs and CTOs.

Salah, K., & Kama, N. (2017). Inter-service provider charging protocol: a solution to address range anxiety of electric vehicle owners. *Energy Procedia, 136*, 157-162.

Sana Ben Abdallah Ben Lamine, Hajer Baazaoui Zghal, Michael Mrissa, & Guegan, C. G. (2017). An ontology-based approach for personalized RESTful Web service discovery. *Procedia Computer Science, 112*, 2127–2136.

Schafer, J. (2013). The risks of BYOD (bring your own device) in the workplace. *J Mich Dent Assoc, 95*(11), 16.

Singh, G., & Supriya. (2013). A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security. *International Journal of Computer Applications, 67*.

Smith, R., Taylor, B., Bhat, M., Silva, C., & Cosgrove, T. (2017). Magic Quadrant for Enterprise Mobility Management Suites.

Sneps-Sneppe, M., & Namiot, D. (2013). Smart Cities Software: Customized Messages for Mobile Subscribers. 25-36.

Sobers, A. (s.d.). BYOD and the Mobile Enterprise. Organisational challenges and solutions to adopt BYOD.

Solution Review. (2018). Mobility Management Buyer's Guide. 2-26.

Snow Software AB. (2015). Mobile Device Inventory – the first step in enterprise mobile management.

Solbrig, H. R., Prud'hommeaux, E., Grieve, G., McKenzie, L., Mandel, J. C., Sharma, D. K., & Jiang, G. (2017). Modeling and validating HL7 FHIR profiles using semantic web Shape Expressions (ShEx). *Journal of Biomedical Informatics, 67*, 90-100.

Song, D., X., Wagner, D., & Perrig, A. (s.d.). Practical Techniques for Searches on Encrypted Data.

SOPHOS. (2018). The State of Endpoint Security Today.

Symantec. (2010). Meeting the Challenges of Endpoint Security.

Salesforce. (2018). *Power Your Business with the Best of the Web.* Obtido de Salesforce: https://www.salesforce.com/saas/

Schinsky, H. (19 de Outubro de 2012). *Tutorial: Apple Push Notifications with PhoneGap - Part 1*. Obtido de Devgirl: http://devgirl.org/2012/10/19/tutorial-apple-push-notifications-with-phonegap-part-1/

Schulz, M. (Junho de 2013). *Pros and cons of mobile device management software*. Obtido de TechTargets: https://searchmobilecomputing.techtarget.com/tip/Pros-and-cons-of-mobile-device-management-software

Sheldon, R. (Julho de 2013). *BYOD vs. COPE: Why corporate device ownership could make a comeback*. Obtido de TechTarget:

https://searchmobilecomputing.techtarget.com/feature/BYOD-vs-COPE-Why-corporate-device-ownership-could-make-a-comeback

Sports Video Group. (13 de Fevereiro de 2017). *Cisco: 70 Percent of Global Population Will Be Mobile Users*. Obtido de Sports Video Group: https://www.sportsvideo.org/2017/02/13/cisco-70-percent-of-global-population-will-be-mobile-users/

Staines, R. (26 de Setembro de 2018). *How Much Does MDM or EMM Software Cost?* Obtido de Samsung: https://insights.samsung.com/2018/09/26/how-much-does-mdm-or-emm-software-cost/

Statcounter. (Outubro de 2018). *Mobile Operating System Market Share Worldwide*. Obtido de Statcounter: http://gs.statcounter.com/os-market-share/mobile/worldwide

Tarkoma, S. (2014). *Smartphone energy consumption : modeling and optimization*. United Kingdom ; New York: Cambridge University Press.

Taylor, R. H., Rose, F., Toher, C., Levy, O., Yang, K., Buongiorno Nardelli, M., & Curtarolo, S. (2014). A RESTful API for exchanging materials data in the AFLOWLIB.org consortium. *Computational Materials Science, 93*, 178-192.

Techotopia. (27 de Outubro de 2016). *An Overview of iOS 5 iPhone Multitasking*. Obtido de Techotopia: https://www.techotopia.com/index.php/An_Overview_of_iOS_5_iPhone_Multitasking

Telecom Expense Management Industry Association. (2013). BYOD: Dos and Don'ts. 1-12.

Thomson, G. (2012). BYOD: enabling the chaos. *Network Security, 2012*(2), 5-8.

Tokuyoshi, B. (2013). The security implications of BYOD. *Network Security, 2013*(4), 12-13.

Townsend Security. (2016). AES Encryption.

VDC Research Group. (2010). Total Cost of Ownership Models For Mobile Computing and Communication Platforms. *3*.

Vodafone. (2018). *Avoid the iceberg: see your device management costs clearly*. Obtido de Vodafone: https://www.vodafone.com/business/news-and-insights/blog/gigabit-thinking/avoid-the-iceberg-see-your-device-management-costs-clearly

W3School. (2018). *JSON vs XML*. Obtido de w3school: https://www.w3schools.com/js/js_json_xml.asp

Waltermire, D., & Harrington, D. (2015). Endpoint Security Posture Assessment: Enterprise Use Cases. *Internet Engineering Task Force*.

Wang, J. (17 de Junho de 2010). *iPhone SDK: First Steps With JSON Data Using the Twitter API*. Obtido de Envato Tuts: https://code.tutsplus.com/tutorials/iphone-sdk-first-steps-with-json-data-using-the-twitter-api--mobile-1181

Waugh, Z. (14 de Março de 2009). *Programmatically retrieving IP Address of iPhone*. Obtido de ZachWaugh: https://zachwaugh.com/posts/programmatically-retrieving-ip-address-of-iphone

WeboPedia. (2018). *Endpoint Security*. Obtido de WeboPedia: https://www.webopedia.com/TERM/E/endpoint_security.html

Wenderlich, R. (12 de Abril de 2011). *How to Write an iOS App That Uses a Web Service*. Obtido de RayWenderlich: https://www.raywenderlich.com/3088-how-to-write-an-ios-app-that-uses-a-web-service

White, R., & Downs, T. E. (2015). *How computers work: the evolution of technology* (Tenth edition. ed.). Indianapolis, IN: Que.

Whitman, E., Michael, & Mattord, J., Herbert. (2014). Principles Of Information Security.

Wilke, A., Bischof, J., Harrison, T., Brettin, T., D'Souza, M., Gerlach, W., . . . Meyer, F. (2015). A RESTful API for accessing microbial community data for MG-RAST. *PLoS Comput Biol, 11*(1), e1004008.

Wodehouse, C. (2018). *SOAP vs. REST: A Look at Two Different API Styles*. Obtido de Upwork: https://www.upwork.com/hiring/development/soap-vs-rest-comparing-two-apis/

Y., S. F., & Kim, P. T.-h. (2007). IT Security Review: Privacy, Protection, Access Control, Assurance and System Security. *International Journal of Multimedia and Ubiquitous Engineering, 2*.

Yuan, T., Tang, Y., Wu, X., Zhang, Y., Zhu, H., Guo, J., & Qin, W. (2014). Formalization and Verification of REST on HTTP Using CSP. *Electronic Notes in Theoretical Computer Science, 309*, 75-93.

Yüce, Y. K., Gülkesen, K. H., & Barcın, E. N. (2012). Balancing Autonomy and Security Over Geotracking Patients with Alzheimer's Using a Personalized Geotracking System with Social Support Network. *Procedia Computer Science, 10*, 1064-1072.

Zahadat, N., Blessner, P., Blackburn, T., & Olson, B. A. (2015). BYOD security engineering: A framework and its analysis. *Computers & Security, 55*, 81-99.

Zenprise. (2012). The Shift to MDM 2.0. 1-8.

Zhao, L., Li, P., Yang, L., Xavior, A., Cai, J., & You, L. (2017). Management strategies of Bring Your Own Device. *MATEC Web of Conferences, 100*.

APÊNDICE A – DESCRIÇÃO DOS *USE CASES*

Os *use cases* descrevem a essência do sistema na perspetiva dos utilizadores/atores e revelam as funcionalidades desejadas. Nesta secção é feita uma descrição das funcionalidades do módulo de inventário apresentadas no ponto 5.1.

Nome	**Recolher e enviar informação**
Actor	Agente
Tipo	Primário
Pré-requisitos	O dispositivo tem que estar conectado à rede da empresa.
Pós-requisitos	É enviado ao Web Service do módulo de inventário a informação recolhida do dispositivo.
Descrição	A aplicação passa para o estado ativo e verifica se o dispositivo se encontra conectado à rede da empresa. Se sim, recolhe os dados, cifra e envia ao WS e volta a passar para o estado inativo.

Tabela 7 - Funcionalidade recolha e envio de informação

Nome	**Tratar e armazenar os dados**
Actor	Web Service
Tipo	Primário
Pré-requisitos	Receção da informação enviada pela aplicação agente.
Pós-requisitos	A informação é armazenada na base de dados do módulo de inventário.
Descrição	O WS recebe a informação enviada pela aplicação agente, faz o tratamento dos dados e armazena nas respetivas tabelas da base de dados.

Tabela 8 - Funcionalidade tratar e armazenar dados

APÊNDICE B – CRIAÇÃO DA BASE DE DADOS

Neste apêndice são apresentados os comandos MySQL utilizados para a criação de cada uma das tabelas que compõem a base de dados do módulo de inventário. O objetivo é unicamente complementar a descrição da implementação da estrutura de dados apresentada na secção 5.3.

```
-- Estrutura da tabela InvDevice
CREATE TABLE InvDevice
(
    Id_device mediumint NOT NULL AUTO_INCREMENT,
    AppId varchar (100),
    Wifimacaddress varchar (60),
    Model varchar (50),
    Sysname varchar (50),
    TotaldiskSpace varchar (50),
    Physicalmemory varchar (50),
    Bluetmacaddress varchar (60),
    PRIMARY KEY (Id_device)

);
```

InvDevice	
PK	Id_device
	AppId
	Wifimacaddress
	Model
	Sysname
	Totaldiskspace
	Physicalmemory
	Bluetmacaddress

Figura 49 - Criação da tabela InvDevice

```
-- Estrutura da tabela InvDeviceStatus
CREATE TABLE InvDeviceStatus
(
    Id_status mediumint NOT NULL AUTO_INCREMENT,
    Id_device mediumint NOT NULL,
    Sysversion varchar (50),
    FreediskSpace varchar (50),
    Ipwifi varchar (50),
    Ipcell varchar (50),
    Usermemory varchar (50),
    Wiredmemory varchar (50),
    Activememory varchar (50),
    Inactivememory varchar (50),
    Freememory varchar (50),
    IsUsbPlugged varchar (50),
    Is3GNetworkActivated varchar (50),
    Devicename varchar (60),
    Date varchar (60),
    PRIMARY KEY (Id_status),
    FOREIGN KEY (Id_device) REFERENCES InvDevice (Id_device)
);
```

InvDeviceStatus	
PK	Id_status
FK	Id_device
	Sysversion
	Freediskspace
	Ipwifi
	Ipcell
	Usermemory
	Wiredmemory
	Activememory
	Inactivememory
	Freememory
	IsUsbPlugged
	Is3GNetworkActivated
	Devicename
	Date

Figura 50 - Criação da tabela InvDeviceStatus

```
-- Estrutura da tabela InvInstalledApps
CREATE TABLE InvInstalledApps
(
    Id_app mediumint NOT NULL AUTO_INCREMENT,
    Id_device mediumint NOT NULL,
    Appname varchar (60),
    Appversion varchar (50),
    Date varchar (60),
    PRIMARY KEY (Id_app),
    FOREIGN KEY (Id_device) REFERENCES InvDevice (Id_device)
);
```

InvInstalledApps	
PK	Id_app
FK	Id_device
	Appname
	Appversion
	Date

Figura 51 - Criação da tabela InvInstalledApps

```
-- Estrutura da tabela InvRunningProcess
CREATE TABLE InvRunningProcess
(
    Id_process mediumint NOT NULL AUTO_INCREMENT,
    Id_device mediumint NOT NULL,
    Procname varchar (60),
    Procstat varchar (50),
    Procflag varchar (50),
    Proctime varchar (60),
    Date varchar (60),
    PRIMARY KEY (Id_process),
    FOREIGN KEY (Id_device) REFERENCES InvDevice (Id_device)
);
```

InvRunningProcess	
PK	Id_process
FK	Id_device
	Procname
	Procstat
	Procflag
	Proctime
	Date

Figura 52 - Criação da tabela InvRunningProcess

Sintaxes MySQL

Foram usados alguns parâmetros para melhorar a usabilidade e mesmo a segurança das tabelas criadas. Aqui serão apresentados rapidamente alguns destes parâmetros.

CREATE TABLE

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
        (create_definition, …)
```

Esta é a sintaxe mais simplificada segundo a documentação da Oracle (Oracle, 2018), e indica a execução do comando de criação de uma tabela. Esta sintaxe pode ainda ser complementada com alguns comandos opcionais:

- TEMPORARY – indica que a tabela a criar é temporária, o que significa que ela expira assim que a sessão no MySQL terminar. É ideal para usar em casos de testes.

- IF NOT EXISTS – esta condição permite uma verificação prévia da existência da tabela a ser criada, evitando uma interrupção do script, causada por erro.

É importante referir que os nomes das tabelas é *case sensitive*. Ou seja, para o MySQL `tbl_name` e `Tbl_Name` são dois nomes totalmente distintos. Cada tabela é composta ainda por uma ou mais colunas, cada qual com suas definições. Nas tabelas acima apresentadas, foram usadas definições do tipo `mediumint` e `varchar` para indicar que as colunas recebem valores de tipo inteiros e lista de caracteres respetivamente.

NOT NULL

No desenho e arquitetura das tabelas foi definido que algumas colunas não poderiam ser deixadas em branco (valor nulo). Para evitar que isto aconteça, é utilizado este parâmetro. O sistema irá devolver um erro, caso não sejam preenchidos os campos que são configurados com este parâmetro.

AUTO_INCREMENT

Este parâmetro é utilizado para conciliar a criação automática dos campos numéricos com uma identificação única dentro de cada tabela (chave primária). Desta forma consegue-se evitar possíveis dores de cabeça quando é necessário aplicar uma ação a um determinado registo.

PRIMARY KEY

Este é o parâmetro usado para definir uma coluna como chave primária da tabela, ajudando a identificar um registo em relação aos outros dentro da tabela. Os valores de uma chave primária têm que ser únicos e não podem ser nulos.

FOREIGN KEY

Por último, este parâmetro permite identificar os campos que são chaves primárias em outras tabelas. Este é um processo muito importante na definição das relações entre as tabelas de uma base de dados. É usado em conjunto com o parâmetro REFERENCES que permite referenciar o nome da tabela e da respetiva coluna.

APÊNDICE C – IMPLEMENTAÇÃO DO AGENTE

No desenvolvimento da aplicação agente foi implementado um conjunto de métodos que são utilizados durante o processo de recolha de informações dos dispositivos. Nesta secção iremos descrever alguns destes métodos.

Importa referir que o agente foi desenvolvido para correr nas versões 6 ou superiores da plataforma iOS e na sua implementação foi utilizada a ferramenta Xcode na versão 7.1.3. Este IDE (*Integrated Development Environment*) pode ser descarregado a partir da Mac App Store, e além do iOS é utilizado também no desenvolvimento de aplicações para as plataformas macOS, watchOS e tvOS. Suporta uma variedade de linguagens de programação tais como C, C++, Objective-C, Java, Python, e inclui ferramentas de testes e análises de código.

Os dois primeiros métodos aqui descritos são `getTotalDiskSpace` e `getFreeDiskSpace`. Como o próprio nome indica, estes métodos são os responsáveis por obter informações sobre a dimensão total e disponível no disco instalado nos dispositivos. Em ambas implementações foi necessário criar um apontador do tipo `NSDictionary` que a partir da classe `NSFileManager` e do método `attributesOfFileSystemForPath:error:` obtém os atributos do sistema de ficheiros. Foi ainda necessário utilizar a função `NSHomeDirectory()` que indica a diretoria da aplicação agente. A imagem seguinte mostra a implementação do primeiro método.

```
240  //Método que retorna a dimensão do disco
241  - (NSNumber *) getTotalDiskSpace
242  {
243      //Cria um apontador do tipo NSDictionary que contém os atributos do file system
244      NSDictionary *fattributes = [[NSFileManager defaultManager]
245          attributesOfFileSystemForPath:NSHomeDirectory() error:nil];
246      //Retorna a dimensão
247      return [fattributes objectForKey:NSFileSystemSize];
248  }
```

Figura 53 - Obter a dimensão do disco

O que diferencia os dois métodos é a utilização da "chave" que permite indicar o valor a obter a partir do dicionário de dados (`fattributes`). No primeiro caso, a variável `NSFileSystemSize` é a chave cujo valor retornado na alínea 247 corresponde à dimensão do sistema de ficheiros.

Por sua vez, no método `getFreeDiskSpace`, a chave utilizada foi a variável `NSFileSystemFreeSize`, retornando assim o espaço livre em disco. A imagem seguinte ilustra esta implementação.

```
250 //Método que retorna o espaço livre em disco
251 - (NSNumber *) getFreeDiskSpace
252 {
253     //Cria um apontador do tipo NSDictionary que contém os atributos do file system
254     NSDictionary *fattributes = [[NSFileManager defaultManager]
255         attributesOfFileSystemForPath:NSHomeDirectory() error:nil];
256     //Retorna o espaço livre
257     return [fattributes objectForKey:NSFileSystemFreeSize];
258 }
```

Figura 54 - Obter o espaço livre em disco

Para obter informações sobre a memória RAM dos dispositivos, foi implementado o método `getMemoryInfo`. Na implementação deste método há que salientar a utilização da aplicação `sysctl` (Gallagher, 2016) e da função `host_statistics` (Apple, 2018). A primeira permite ler/alterar em *runtime* os parâmetros do núcleo do sistema operativo de um computador (*kernel*) e é utilizada em sistemas operativos com base em Unix, como é o caso do iOS. A segunda é uma função que devolve os dados estatísticos sobre a memória virtual do dispositivo.

```
410  /* Método que devolve informações da memória RAM */
411  - (NSDictionary *)getMemoryInfo
412  {
413      /*** Variáveis locais ***/
414      size_t length;      //Tamanho do buffer
415      int mib[6];         //Array de informações a obter
416      int result;         //Buffer resultado
417      int pagesize;       //Buffer tamanho página
418      NSDictionary *dict; //Resultado com info RAM
419
420      /*** Obter dimensão da página de memória ***/
421      mib[0] = CTL_HW;
422      mib[1] = HW_PAGESIZE;
423      //Indica o tamanho do buffer
424      length = sizeof(pagesize);
425      //Guarda na variável "pagesize" a dimensão da página de memória
426      if (sysctl(mib, 2, &pagesize, &length, NULL, 0) < 0)
427      {
428          perror("getting page size");
429      }
430      // Converte "pagesize" para o formato string
431      NSNumber *pageNumber = [NSNumber numberWithInt:pagesize];
432      NSString *pageString = [pageNumber stringValue];
```

Figura 55 - Informações da RAM (página de memória)

Utilizando os parâmetros certos, a aplicação `sysctl` analisa a informação pretendida e em caso de sucesso, devolve os dados relativos à dimensão da página de memória, a memória física e memória do utilizador. Como podemos ver na alínea 426, a `sysctl` recebe um *array* de inteiros (*Management Information Base*) com a indicação dos dados que se quer obter; a dimensão desse mesmo *array*; a variável para onde serão mapeados os dados e o seu tamanho. Neste caso, a informação obtida é a dimensão da página de memória do dispositivo, que será mapeada para a variável `pagesize`.

O passo seguinte foi repetir a mesma ação para obter as informações relativas à memória física e à memória do utilizador. As imagens seguintes ilustram este processo.

```
431     /*** Obter memória física ***/
432     mib[0] = CTL_HW;
433     mib[1] = HW_PHYSMEM;
434     length = sizeof(result);
435     //Guarda em "result" informações da memória física
436     if (sysctl(mib, 2, &result, &length, NULL, 0) < 0)
437     {
438         perror("getting physical memory");
439     }
440
441     NSNumber *pM = [NSNumber numberWithInt:result];
442     int var1 = [pM intValue];
443     //Memoria fisica em MB
444     var1 = (var1 / 1024)/1024;
445     //Converte para o formato string
446     NSString *pMString = [NSString stringWithFormat:@"%d",var1];
```

Figura 56 - Informações da RAM (memória física)

```
448     /*** Obter memória do utilizador ***/
449     mib[0] = CTL_HW;
450     mib[1] = HW_USERMEM;
451     length = sizeof(result);
452     //Guarda em "result" informações da memória de utilizador
453     if (sysctl(mib, 2, &result, &length, NULL, 0) < 0)
454     {
455         perror("getting user memory");
456     }
457
458     NSNumber *uM = [NSNumber numberWithInt:result];
459     int var2 = [uM intValue];
460     //Memória do utilizador em MB
461     var2 = (var2 / 1024)/1024;
462     //Converte para o formato string
463     NSString *uMString = [NSString stringWithFormat:@"%d",var2];
```

Figura 57 - Informações da RAM (memória de utilizador)

De notar que a única diferença é que nestes últimos foi utilizada a mesma variável (`result`) para conter os dados obtidos pela aplicação `sysctl`. No final, as informações foram convertidas para um formato `NSString` para facilitar o mapeamento na lista com a informação a retornar.

De seguida foi então utilizada a função `host_statistics` indicando-lhe: o porto de onde a informação deve ser obtida (`mach_host_self()`); o tipo de estatísticas pretendido que neste caso é da memória virtual (`HOST_VM_INFO`); a variável que irá conter a informação pretendida (`vmstat`); e a dimensão devolvida (`count`). Este processo é executado na alínea 469 da imagem seguinte.

```
465     /*** Obter informações sobre o estado da memória ***/
466     mach_msg_type_number_t count = HOST_VM_INFO_COUNT;
467     vm_statistics_data_t vmstat;
468     //Guarda na variável "vmstat" as estatísticas da memória virtual
469     if (host_statistics(mach_host_self(), HOST_VM_INFO, (host_info_t)&vmstat, &count)
470         != KERN_SUCCESS)
471     {
472         printf("Failed to get VM statistics.");
473     }
```

Figura 58 - Obter dados estatísticos da memória virtual

Em caso de sucesso, a variável `vmstat` irá conter a estrutura com a informação sobre a estatística da memória virtual do dispositivo. O passo seguinte é extrair esta informação, obtendo assim os valores da memória total, ativa, inativa, disponível e alocada pelo sistema operativo.

```
475     //Memória total
476     double total = vmstat.wire_count + vmstat.active_count + vmstat.inactive_count
477         + vmstat.free_count;
478     //Memória alocada pelo SO
479     double wired = vmstat.wire_count / total;
480     //Memória ativa
481     double active = vmstat.active_count / total;
482     //Memória inativa
483     double inactive = vmstat.inactive_count / total;
484     //Memória disponível
485     double free = vmstat.free_count / total;
486     //Converte os resultados para formato string
487     NSNumber *totalMDouble = [NSNumber numberWithDouble:total];
488     NSString *totalMString = [totalMDouble stringValue];
489     NSString * wString = [NSString stringWithFormat:@"%0.2f %%", wired * 100.0];
490     NSString * aString = [NSString stringWithFormat:@"%0.2f %%",active * 100.0];
491     NSString * iString = [NSString stringWithFormat:@"%0.2f %%",inactive * 100.0];
492     NSString * fString = [NSString stringWithFormat:@"%0.2f %%",free * 100.0];
```

Figura 59 - Mapear os dados obtidos da memória virtual

Por fim, todos os valores obtidos foram mapeados para uma estrutura de pares chave-valor do tipo `NSDictionary` e retornados. A utilização deste tipo de estrutura permite que a

obtenção de cada um dos valores contidos neste objeto seja feita de forma direta, a partir de cada uma das chaves.

```
494         //Mapeamento dos dados recolhidos
495         dict = [[NSDictionary alloc] initWithObjects:
496         [NSArray arrayWithObjects:pageString, pMString, uMString, wString,
497             aString, iString, fString, nil]
498         forKeys:[NSArray arrayWithObjects:@"PageSize", @"PhysicalMemory",
499             @"UserMemory", @"wired", @"active", @"inactive", @"free", nil]];
500
501       //Retorna a informação da RAM
502         return dict;
503
504 }
```

Figura 60 - Devolve a informação da RAM

Outro método que vamos detalhar é o `getThirdPartyInstalledApplications`. Este método foi implementado para recolher a lista de algumas aplicações instaladas pelo utilizador. A Apple têm uma política rígida no que diz respeito à proteção da privacidade dos utilizadores e isso também se reflete às aplicações. Cada uma delas é instalada em "capsulas" isoladas e com acessos restritos ao exterior. A solução encontrada foi utilizar o método `canOpenURL` que indica se um esquema de URL pode ser manipulado por alguma aplicação instalada no dispositivo. Foi necessário criar uma lista com o nome das aplicações que pretendíamos validar e outra com os URL's possíveis de serem manipulados pelas respetivas aplicações.

```
621 /* Método que devolve informações da memória RAM */
622 - (NSMutableArray *)getThirdPartyInstalledApplications
623 {
624     /*** Variáveis locais ***/
625     //Array dinâmico para conter a lista de apps
626     NSMutableArray * thirdAppsList = [[NSMutableArray alloc] init];
627     //String que indica as versões das apps
628     NSString * appversion = @"N/A";
629     //Array estático de apps a validar
630     NSArray * apps = [NSArray arrayWithObjects:@"Facebook", @"Chrome", @"Twitter",
631         @"Skype", @"AirSharing", @"Tweetie", @"Google Drive", @"Instagram", nil];
632     //Array estático de URL's
633     NSArray * linkArray = [NSArray arrayWithObjects:@"fb://profile", @"googlechrome://",
634         @"twitter://", @"skype:magnatron.nl?call", @"airsharing://", @"tweetie://",
635         @"googledrive://www.google.com", @"instagram://", nil];
```

Figura 61 - Lista de possíveis aplicações

Nas alíneas 630 e 633 podemos ver as variáveis de tipo `NSArray` criadas para conter o nome de algumas destas aplicações e a lista com possíveis URL's. O passo seguinte foi iterar sobre a segunda lista (`linkArray`) utilizando o método `canOpenURL` para validar os que podem ser manipulados. Se esta condição se verificar, então é criado um objeto do tipo `NSDictionary` com os pares chave-valor (nome e versão) e mapeados para a variável `thirdAppsList`. Por sua vez, esta é a variável que irá conter a lista final das aplicações instaladas pelo utilizador.

É importante referir que devido à restrição da Apple no acesso externo não foi possível extrair informações sobre as versões de cada uma das aplicações detetadas. Por isso, nesta fase optou-se por assinalar cada versão com o valor "N/A", indicando não aplicável ou indisponível. Este é um dos pontos que poderá ser melhorado no trabalho futuro.

```
638     //Ciclo para iterar o array de URL's
639     for (NSUInteger i = 0; i < [linkArray count]; i++) {
640
641         //Verifica se existe alguma app capaz de manipular a URL
642         if ([[UIApplication sharedApplication] canOpenURL:[NSURL URLWithString:
643         [linkArray objectAtIndex:i]]])
644         {
645             //Cria um objeto chave-valor (Nome - Versão)
646             NSDictionary * dict = [[NSDictionary alloc] initWithObjects:
647                 [NSArray arrayWithObjects:[apps objectAtIndex:i], appversion, nil]
648                     forKeys:[NSArray arrayWithObjects:@"AppName", @"AppVersion", nil]];
649
650             //Adiciona à lista final
651             [thirdAppsList addObject:dict];
652         }
653     }
654
655     //Verfica não foi detetada nenhuma app devolve null
656     if ([thirdAppsList count] == 0) {
657         return nil;
658     }
659     //Retorna a lista de apps
660     return thirdAppsList;
661 }
```

Figura 62 - Devolve as aplicações detetadas no dispositivo

Por último, é feita uma validação à lista `thirdAppsList` para verificar se o número de elementos contidos é zero. Se esta condição se verificar, quer dizer que não foi detetada nenhuma das aplicações pretendidas, e neste caso é retornado nulo (alínea 656). Caso contrário, na última alínea do método é retornada a lista com as aplicações e assim conclui-se este processo.

Todos os métodos implementados para o funcionamento da aplicação agente seguem lógicas e comportamentos semelhantes aos que aqui foram descritos. Tal como foi referido na secção 5.4.3, o processo de invocação e sincronização dos dados recolhidos por cada um deles é efetuado pelo método `prepareDataBeforeSend`. No final os dados são convertidos no formato JSON, cifrados e enviados ao Web Service.

APÊNDICE D – IMPLEMENTAÇÃO DO WEB SERVICE

Nesta secção iremos analisar em mais detalhes a implementação do Web Service do módulo de inventário. Tal como foi dito no capítulo 5, este WS foi desenvolvido em REST utilizando a linguagem de programação PHP. O *software* utilizado no desenvolvimento do WS foi o NotePad++. Este é um editor de texto/código "*Open Source"*acedido gratuitamente na internet, fácil e prático de manusear e que já trás por omissão uma variedade enorme de linguagens de programação que podem ser utilizadas (C, C#, Java, Ruby, SQL, etc.).

Após os passos iniciais apresentados no ponto 5.5, foi necessário validar o processo de descodificação e iterar sobre cada um dos elementos contidos no JSON enviado pela aplicação agente, fazendo o respetivo mapeamento dos dados. Na Figura seguinte pode-se verificar este processo.

```
//Verifica o processo de "decode" do JSON recebido
switch (json_last_error())
{
    case JSON_ERROR_NONE:

        if(count($body) > 0)
        {

            //Mapeamento dos dados contidos no JSON
            $date = $body ->{'date'};
            $physicalm = $body ->{'physicalm'};
            $localizemodel = $body ->{'localizemodel'};
            $userm = $body ->{'userm'};
            $wired = $body ->{'wired'};
            $active = $body ->{'active'};
            $inactive = $body ->{'inactive'};
            $free = $body ->{'free'};
            $wifimac = $body ->{'wifimac'};
            $language = $body ->{'language'};
            $sysversion = $body ->{'sysversion'};
            $totalpage = $body ->{'totalpage'};
            $sysname = $body ->{'sysname'};
```

Figura 63 - Validação do JSON e mapeamento dos dados

```
//Mapeamento dos dados contidos no JSON (continuação)
$sysname = $body ->{'sysname'};
$tdiskspace = $body ->{'tdiskspace'};
$ipwifi = $body ->{'ipwifi'};
$ipcell = $body ->{'ipcell'};
$bluetmac = $body ->{'bluetmac'};
$country = $body ->{'country'};
$fdiskspace = $body ->{'fdiskspace'};
$devicename = $body ->{'devicename'};
$model = $body ->{'model'};
$plist = $body ->{'processlist'};
$ssid = $body ->{'ssid'};
$cellAppsList = $body ->{'cellAppsList'};
$thirdAppsList = $body ->{'thirdAppsList'};
$isUsbPlugged = $body ->{'isusbplugged'};
$is3gNetworkActivated = $body ->{'is3g'};
$appId = $body -> {'appId'};
```

Figura 64 - Validação do JSON e mapeamento dos dados (parte 2)

Feito isso, o passo seguinte foi validar se os dados mapeados pertencem a um novo dispositivo em funcionamento na rede da empresa. De recordar que chegado a este ponto todos os dados mapeados pelo WS encontram-se cifrados com uma chave secreta que só pode ser acedida pelos administradores da empresa. Por isso, a solução encontrada foi criar um campo que pudesse identificar unicamente cada agente e desta forma podemos garantir que cada dispositivo fica diretamente associado a um único agente. Este identificador é o campo `AppId` que é gerado automaticamente no momento de criação de cada agente e à exceção dos restantes é mantido num formato *plain text*, permitindo este processo de validação. A imagem seguinte mostra este processo.

```
44  //Comandos SQL
45  $query = "SELECT * FROM InvDevice WHERE appId = '$appId'";
46
47  $result = mysql_query($query) or die (mysql_error());
48
49  $num = mysql_numrows($result);
50
51  //Verifica se é a primeira vez que recebe dados deste dispositivo
52  if($num == 0)
53  {
54
55      //Insere na tabela InvDevice
56      $insertInvDevice_query = "INSERT INTO InvDevice VALUES ('','$appId',
57      '$wifimac','$model',
58      '$sysname','$tdiskspace',
59      '$physicalm','$buetmac')";
60
61      mysql_query($insertInvDevice_query);
62      echo mysql_error($link);
63  }
```

Figura 65–Inserção na tabela InvDevice

Foi executada uma pesquisa à tabela `InvDevice` por uma entrada correspondente ao campo `AppId` e devolvido um resultado (alíneas 45, 47 e 49). A condição seguinte verifica se o resultado obtido é zero. Se se verificar, então quer dizer que ainda não existem registos de dados recolhidos deste dispositivo e neste caso é necessário introduzir esta informação na tabela `InvDevice` (alíneas 56 e 61).

De seguida, foi necessário voltar à mesma tabela e obter o valor do campo que identifica o novo dispositivo inserido anteriormente (`Id_device`). Este é o campo que nos vai permitir proceder com a atualização nas restantes tabelas, criando a relação desejada entre elas. Nas alíneas 66 e 67 da imagem seguinte podemos ver a criação e execução deste comando.

```
65  //Query para obter o identificador do dispositivo
66  $id_device_query = "SELECT id_device FROM InvDevice WHERE appId = '$appId'";
67  $result_id_device = mysql_query($id_device_query) or die (mysql_error());
68  if (!$result_id_device)
69  {
70      echo "DB Error: could not query id device from InvDevice table\n";
71      echo "MySQL Error: " . mysql_error();
72      exit();
73  }
```

Figura 66 - Obter o identificador do novo dispositivo

Tendo em mãos o valor que identifica o dispositivo obtido da tabela `InvDevice`, o passo seguinte foi então atualizar as restantes tabelas, começando pela `InvDeviceStatus`. Aqui são armazenadas informações que indicam o estado de cada dispositivo num determinado momento: espaço em disco, estado da memória RAM, entre outros.

```
75  //Valida o resultado da pesquisa pelo identificador
76  while ($row = mysql_fetch_assoc($result_id_device))
77  {   //Obtem o id
78      $idDevice = $row['id_device'];
79
80      //Cria o comando para inserir na tabela InvDeviceStatus
81      $insert1_query = "INSERT INTO InvDeviceStatus VALUES ('','$idDevice','$sysversion',
82      '$fdiskspace','$ipwifi','$ipcell','$userm','$wired', '$active','$inactive','$free',
83      '$isUsbPlugged','$is3gNetworkActivated','$devicename','$date')";
84      //Executa o comando
85      mysql_query($insert1_query);
86      echo mysql_error($link);
```

Figura 67 - Inserção na tabela InvDeviceStatus

A próxima tabela atualizada foi a `InvRunningProcess`. A cada ciclo de recolha de dados o WS recebe uma lista de processos a correr no dispositivo. Por isso, para implementar de forma correta a inserção dos dados nesta tabela, foi necessário iterar sobre cada um dos elementos da lista obtendo os valores para a criação e execução dos comandos de inserção.

```
88  //Ciclo para atualizar a tabela InvRunningProcess
89  for($i = 0; $i < count($plist); $i++)
90  {
91      //Obtem cada um dos valores existentes na lista a partir da chave
92      $pname = $plist[$i] ->{'ProcessName'};
93      $pstat = $plist[$i] ->{'ProcesStat'};
94      $pflag = $plist[$i] ->{'Flag'};
95      $ptime = $plist[$i] ->{'StartTime'};
96      //Cria o comando para inserir na tabela de processos
97      $insert3_query = "INSERT INTO InvRunningProcess VALUES ('','$idDevice',
98                        '$pname','$pstat','$pflag','$ptime','$date')";
99      //Executa o comando
100     mysql_query($insert3_query);
101     echo mysql_error($link);
102 }
```

Figura 68 - Inserção na tabela InvRunningProcess

Por último, foram implementados os comandos de inserção na tabela `InvInstalledApps`. Este processo foi elaborado em dois passos, uma vez que a informação a ter em conta para esta tabela é contida em duas listas: a primeira com a informação das aplicações do sistema operativo e a segunda com informações das aplicações instaladas pelo

utilizador. Seguindo a mesma lógica anterior, foi necessário iterar sobre cada uma das listas, obtendo todos os valores para a criação e execução dos comandos.

```
104  //Ciclo para atualizar a tabela InvInstalledApps (aplicações do SO)
105  for($i = 0; $i < count($cellAppsList); $i++)
106  {
107      //Obtem cada um dos valores existentes na lista a partir da chave
108      $appname = $cellAppsList[$i] ->{'AppName'};
109      $appversion = $cellAppsList[$i] ->{'AppVersion'};
110      //Cria o comando para inserir na tabela
111      $insert4_query = "INSERT INTO InvInstalledApps VALUES ('','$idDevice',
112                          '$appname','$appversion','$date')";
113      //Executa o comando
114      mysql_query($insert4_query);
115      echo mysql_error($link);
116  }
```

Figura 69 - Inserção na tabela InvInstalledApps (aplicações do SO)

```
118  //Ciclo para atualizar a tabela InvInstalledApps (aplicações do utilizador)
119  for($i = 0; $i < count($thirdAppsList); $i++)
120  {
121      //Obtem cada um dos valores existentes na lista a partir da chave
122      $appname = $thirdAppsList[$i] ->{'AppName'};
123      $appversion = $thirdAppsList[$i] ->{'AppVersion'};
124      //Cria o comando para inserir na tabela
125      $insert5_query = "INSERT INTO InvInstalledApps VALUES ('','$idDevice',
126                          '$appname','$appversion','$date')";
127      //Executa o comando
128      mysql_query($insert5_query);
129      echo mysql_error($link);
130  }
```

Figura 70 - Inserção na tabela InvInstalledApps (aplicações do utilizador)