Universidade do Minho
Escola de Engenharia

Ailton Moreira da Veiga

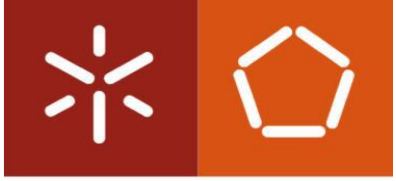
Assistente Pessoal Hospitalar

Assistente Pessoal Hospitalar

Ailton Moreira da Veiga

UMinho | 2017

Outubro 2017



Universidade do Minho
Escola de Engenharia

Ailton Moreira da Veiga

Assistente Pessoal Hospitalar

Dissertação de Mestrado

Mestrado integrado em Engenharia e Gestão de Sistemas de Informação

Trabalho efetuado sob a orientação do

Professor Doutor Manuel Filipe Santos

e do

Professor Doutor Carlos Filipe Portela

Outubro 2017

DECLARAÇÃO

Nome: Ailton Moreira da Veiga

Endereço eletrónico: amv265@hotmail.com

Telefone: 926028389

Título de Residência: CX1915040

Título da dissertação: Assistente Pessoal Hospitalar

Orientadores:

Professor Doutor Manuel Filipe Santos

Professor Doutor Carlos Filipe Portela

Ano de conclusão: 2017

Designação do Mestrado: Ciclos de estudos conducentes ao Grau de Mestre em Engenharia e Gestão de Sistemas de Informação

Área de Especialização: Engenharia e Gestão de Sistemas de Informação

Escola: Escola de Engenharia da Universidade do Minho

Universidade do Minho, _____/_____/_____

Assinatura: _____

Agradecimentos

Dedico este espaço a todos aqueles que deram o seu contributo direto ou indiretamente para que esta dissertação fosse realizada. A todos eles deixo aqui o meu sincero agradecimento.

Em primeiro lugar gostaria de agradecer aos meus orientadores o Professor Doutor Manuel Filipe Santos e o Professor Doutor Carlos Filipe Portela pela disponibilidade, dedicação, atenção e ajuda prestadas durante a realização do trabalho, visto que sem essa contribuição não seria possível, um muito obrigado. Estou grato por ambos e também pela liberdade de ação que me foi atribuído, que acabou por se tornar decisivo para que este trabalho contribuísse para o meu desenvolvimento pessoal.

Um agradecimento especial aos meus colegas do curso e amigos que me acompanharam durante esta longa caminhada académica, e principalmente aqueles que tornaram o meu processo de adaptação e integração mais fácil à nova realidade.

A nível pessoal, um profundo agradecimento à minha família e de forma particular à minha mãe Maria Isabel Moreira que foi e continua a ser meu pilar, pela confiança em mim depositado e ao apoio incondicional que me deu durante todo o meu percurso académico. Também não poderia deixar de agradecer ao meu irmão Amilton Moreira, à minha namorada Mariana Bispo, ao Júlio Barros, ao Neryvaldo Galvão pela amizade, motivação e confiança, que eles deram e que fazem deles pessoas muito importantes para mim.

Para terminar, quero destacar as pessoas que fizeram parte deste percurso que partilhei momentos inesquecíveis e que são companheiros para a vida Mário Cardozo, João Azevedo, Tiago Cruz, Rui Ribeiro, Carlos Miranda, Bruno Carneiro, Sérgio Fernandes.

A todos um obrigado pelo vosso apoio sobretudo pela vossa amizade!

Resumo

O crescimento exponencial dos dispositivos móveis (*smartphones* e *tablets*) nos últimos anos, tem vindo a adquirir cada vez mais espaço de destaque nas nossas atividades do dia-a-dia, quer a nível pessoal bem como a nível profissional. Neste contexto as organizações na área da saúde não são uma exceção, na medida que os dispositivos móveis estão a ser cada vez mais utilizados pelos colaboradores das unidades hospitalares. Para melhorar a qualidade dos serviços prestados e no cumprimento da missão do Centro Hospitalar do Porto (CHP), foi definido como uma das melhorias gerais proporcionar aos seus utentes uma melhoria na qualidade dos serviços prestados. Para atingir a melhoria pretendida no CHP, realizou-se o levantamento dos principais problemas que os utentes têm quando dirigem ao CHP. Para colmatar esses problemas a solução proposta foi: encontrar/criar um artefacto do género assistente pessoal hospitalar para ajudar os utentes quando dirigirem para o CHP.

Embora o mercado dos dispositivos móveis encontra-se a crescer muito nos últimos anos, existe um grande entrave no mercado de desenvolvimento de aplicações para os dispositivos móveis devido a heterogeneidade das plataformas móveis existentes (*Android*, *iOS*, *Windows Phone*, entre outros). Devido a diversificação das plataformas móveis, é exigido cada vez mais maior esforço no desenvolvimento das aplicações móveis, de modo que obriga várias vezes o desenvolvimento das mesmas aplicações para cada plataforma móvel em particular. Com o intuito de resolver este entrave, as abordagens de desenvolvimento de aplicações móveis multiplataformas ganharam relevância, pois possibilitam o desenvolvimento de aplicações móveis para as diferentes plataformas móveis a partir de apenas um único código fonte.

O principal objetivo desta dissertação é encontrar uma solução face aos diversos problemas que os utentes têm quando estes deslocam ao CHP. Com o problema identificado, a solução encontrada foi a criação de um artefacto do género assistente pessoal hospitalar de modo a introduzir melhorias na qualidade dos serviços prestados aos utentes no CHP.

Palavras-Chaves: BYOD, dispositivos móveis, desenvolvimento multiplataforma, *healthcare*, *smartphones*

Abstract

The exponential growth of mobile devices (smartphones and tablets) in recent years have been acquiring increasingly prominent space in our day-to-day, both personally and professionally. In this context organizations in healthcare are not an exception, as mobile devices are being increasingly used by employees of hospitals. To improve the quality of services and the fulfillment of the mission of the Hospital of Porto (CHP), it was defined as one of the general improvements to provide its users with an improvement in the quality of the services provided. To achieve the desired improvement in the CHP, the survey was carried out of the main problems users have when attending the CHP. To address these problems, the proposed solution was to find / create a personal hospital assistant device to assist users when driving to the CHP.

Although the market for mobile devices is growing in recent years, there is a major obstacle in the application development market for mobile devices because of the heterogeneity of existing mobile platforms (Android, iOS, Windows Phone, etc.). Due to the diversification of mobile platforms, it is increasingly required increased effort to develop mobile applications, so requiring often the development of such applications for each mobile platform. To solve this obstacle, the cross-platform mobile application development approaches have gained importance, as they allow the development of mobile applications for different mobile platforms from a single source.

The main objective of this dissertation is to find a solution to the various problems that the users have when they move to the CHP. With the problem identified, the solution was to create an artifact of the hospital gender personal assistant to improvements in the quality of services provided to users in the CHP.

Keywords: BYOD, healthcare, mobile devices, multiplatform development, smartphones

Índice

Lista das Figuras

Lista de Gráficos

Lista de Tabelas

Lista de Sigla e Acrônimos

AIDA – Agência para Integração, Difusão e Arquivo de Informação Médica

AJAX – Asynchronous JavaScript and XML

API – Application Programing Interface

ARC – Automatic Reference Counting

BYOD – Bring Your Own Device

CHP – Centro Hospitalar do Porto

CPU – Central Processing Unit

CSS– Cascading Style Sheets

DIS – Department Information System

DMZ – Demilitarized Zone

DVM – Dalvik Virtual Machine

GPS – Global Positioning System

HIPPA – Health Insurance Portability and Accountability Act

HIS – Hospital Information System

HTML – Hyper Text Markup Language

IDC – International Data Corporation

JSON – JavaScript Object Notation

JVM – Java Virtual Machine

LIS – Laboratory Information System

MAM – Mobile Application Management

MAS – Multi-Agent System

MCDT – Meios Complementares de Diagnostico e Terapêutica

MDM – Mobile Device Management

MIM – Mobile Information Management

MMDCP – Medical Mobile Data Collecting Platform

MSS – Management Support System

MVC – Model View Controller

NFC – Near Field Communication

OS – Sistema Operativo

PCE – Processo Clinico Eletrónico

PDA – Personal Digital Assistant

PSS – Proportional Set Size

RIS – Radiology Information System

SAPE – Sistema de Apoio à Prática de Enfermagem

SMA – Sistema Multi Agente

SDK – Software Development Kit

SOA – Service Oriented Architecture

SOAP – Simple Object Access Protocol

SONHO – Sistema de Gestão de Doentes Hospitalares

REST – Representational State Transfer

UBS – Universal Bus Service

UDDI – Universal Description Discovery Language

UI – User Interface

UIS – Universal Identification Service

USS – Unique Set Size

VPN – Virtual Private Network

XML - eXtensible Markup Language

WSDL - Web Service Definition Language

1. Introdução

1.1. Contextualização

Este projeto enquadra-se no processo de restruturação e melhoramento da infraestrutura informática do Centro Hospitalar do Porto (CHP), de modo a acompanhar o processo de evolução das tecnologias de informação e comunicação. Como tal, tendo em conta as melhorias que o CHP pretende atingir para proporcionar uma melhor qualidade de serviços prestados aos seus utentes, a realização deste projeto de dissertação constitui uma das melhorias que o hospital pretende alcançar.

1.2. Motivação

Atualmente, os dispositivos móveis estão a mudar a forma como as pessoas interagem entre si e com o mundo, grande parte devido à sua portabilidade e às inúmeras aplicações que existem para estes tipos de dispositivos móveis. A utilidade e a agilidade das aplicações móveis no dia-a-dia dos utilizadores dos dispositivos móveis são cada vez mais indispensáveis tanto a nível pessoal como também a nível profissional.

No *healthcare* o mesmo se aplica, de modo que se têm notado um crescimento na utilização dos dispositivos móveis pelas organizações do *healthcare* com o intuito de permitir que os colaboradores desenvolvam as suas atividades com maior eficácia e eficiência. A introdução da utilização dos dispositivos móveis no *healthcare* provocou um aumento na qualidade de prestação dos seus serviços por parte dos colaboradores das organizações.

No contexto do assistente pessoal, de forma a proporcionar um serviço de melhor qualidade aos utentes, a criação do artefacto (protótipo funcional) vem facilitar a vida dos utentes quando estes deslocam-se ao CHP, proporcionando-lhes uma melhor estadia.

1.3. Objetivos

Esta subsecção encontra-se divida em três pontos, sendo o primeiro a identificação do objetivo principal da realização desta dissertação, o segundo a identificação dos objetivos específicos da dissertação e por fim a identificação dos resultados esperados com realização da mesma.

a) Objetivo principal

O principal objetivo inerente a esta dissertação de investigação é:

- Encontrar uma solução do género assistente pessoal para área hospitalar, de modo a proporcionar uma melhor qualidade de serviços prestados aos utentes no CHP.

b) Objetivos específicos

Para a atingir o objetivo proposto para esta dissertação, este foi divido em duas categorias: uma a nível das tecnologias e outra a nível dos testes e análise dos resultados.

- A nível das tecnologias o foco é a análise comparativa das tecnologias de desenvolvimento de aplicações móveis, bem como o desenvolvimento de um protótipo funcional da solução;
- A nível dos testes e análise dos resultados, o foco é a realização dos testes da solução. Este consiste em certificar que a solução se encontra apta para ser utilizada pelo público. A análise dos resultados visa analisar o valor que a solução pode acrescentar ao centro hospitalar e como a mesma pode ajudar a colmatar os problemas identificados no referido centro hospitalar, no que diz respeito ao processo de atendimento dos utentes e não só.

c) Resultados Esperados

Como resultado esperado da realização desta dissertação, era a criação de um artefacto, uma aplicação *pervasive* protótipo do género de assistente pessoal. Com o desenvolvimento da solução, era esperado que os utentes passam a usufruir de um conjunto de benefícios à chegada do CHP, tais como, a gestão do processo de atendimento dos utentes, gestão de senhas, entre outras, de modo a evitar as longas filas de esperas nos hospitais.

1.4. Estrutura da Dissertação

Esta dissertação encontra-se dividida em sete capítulos. No **primeiro capítulo** é apresentado uma breve introdução ao tema da dissertação, bem como as principais motivações que levaram ao desenvolvimento deste trabalho e os objetivos propostos.

O **segundo capítulo** é composto pelas metodologias de investigação que foram utilizadas neste projeto de dissertação. Para o desenvolvimento desta dissertação foram utlizadas três metodologias o ***Design Science Research*** (DSR), o **Apoio à Decisão** (DS) e o ***Scrum***. As metodologias DSR e DS foram cruzadas com o intuito de conseguir se efetuar uma melhor análise para obter melhores resultados, enquanto que o Scrum foi utilizado essencialmente durante a conceção da solução.

O **terceiro capítulo** apresenta o estudo do estado da arte (revisão de literatura) onde é apresentado uma breve introdução sobre os dispositivos e plataformas móveis, o conceito ***Bring Your Own Device*** em português "traga o seu próprio dispositivo" no geral e especificamente no *healthcare*. Será apresentado um caso de estudo sobre a utilização dos dispositivos móveis no *healthcare*. Será feita uma análise sobre as metodologias de desenvolvimento de aplicações e por fim uma análise às tecnologias de desenvolvimento de aplicações móveis.

O **quarto capítulo** contém os requisitos funcionais e não funcionais da aplicação, bem como a arquitetura da solução. Neste capítulo encontra-se especificados os requisitos funcionais da solução, bem como os não funcionais. Ainda, é apresentado e especificado a arquitetura da solução.

O **quinto capítulo** é o capítulo de implementação e testes. O capítulo começa com a escolha das tecnologias de desenvolvimento selecionadas para o desenvolvimento da solução em si, seguindo de uma breve explicação da solução desenvolvida. Seguidamente, é apresentado um conjunto de testes que foram realizados à solução de modo a certificar que este consegue cumprir os requisitos que foram definidos no início da dissertação.

O **sexto capítulo** é o capítulo de monotorização e avaliação dos resultados da solução. Trata-se de um capítulo que foi introduzido devido à metodologia de investigação *Design Science Research*

(DSR), que indica que é importante no final do desenvolvimento do projeto, efetuar a monitorização do mesmo e de igual modo efetuar um conjunto de testes, para posteriormente proceder à avaliação e comunicação dos resultados obtidos.

O **sétimo e último capítulo** apresenta as conclusões sob a forma de síntese de todo trabalho realizado e os resultados alcançados. Ainda no mesmo capítulo são apresentadas algumas sugestões para os trabalhos futuros.

2. Abordagem Metodológica

Para a realização desta dissertação, a metodologia de investigação que foi utilizada é o *Design Science Research* (DSR). Mas, no entanto, foram também utilizadas outras metodologias (métodos) tais como o *Apoio à Decisão* (que será cruzado com o DSR para obter melhor análise dos resultados) e o *Scrum* (durante o desenvolvimento da parte prática desta dissertação).

A identificação e definição de uma questão de investigação é de especial relevância no desenvolvimento de um projeto, e a realização desta dissertação não era uma exceção. A questão de investigação identificada para esta dissertação foi:

- Como um assistente pessoal hospitalar pode introduzir melhoria na qualidade dos serviços prestados aos utentes?

Para obter as respostas aos problemas levantados pela questão de investigação, foi utilizada a metodologia de investigação DSR, juntamente com a metodologia de apoio à decisão. A metodologia DSR a seguir apresentada identifica todos os passos necessários para criar artefactos necessários para resolver os problemas levantados pela questão de investigação.

2.1. Design Science Research

Os investigadores da área de Sistemas de Informação (SI) no início da década de 1990 começaram a despertar o interesse de pesquisas pelo *Design Science* (DS). Nessa altura, havia um acordo básico das pesquisas anteriores sobre as principais diferenças entre o DS e outros paradigmas (Gregor & Hevner, 2013).

A metodologia de investigação *Design Science Research* (DSR) apresenta um paradigma de investigação na qual o investigador responde às questões relevantes para os problemas humanos com base na criação de artefactos inovadores, para contribuir assim com novos conhecimentos para a prova científica. Os artefactos desenvolvidos devem ser úteis e fundamentais para a compreensão do problema.

"The fundamental principle of design science research is that knowledge and understanding of a design problem and its solution are acquired in the building and application of an artifact".

A citação acima afirma que o principio fundamental da pesquisa científica é o conhecimento e a compreensão do problema e as soluções que são adquiridos na construção e aplicação do artefacto, (Gregor & Hevner, 2013; Hevner & Chatterjee, 2010; Walls, Widmeyer, & El Sawy, 1992).

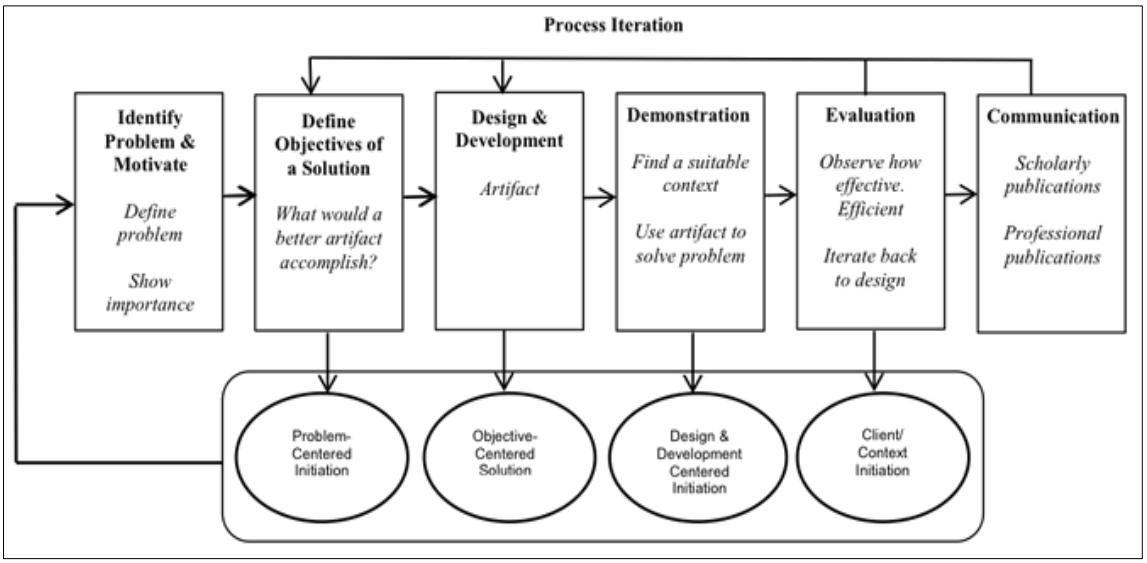


Figura 1 - Ciclo da Metodologia Design Science Research retirado de (Hevner & Chatterjee, 2010)

Design Science Research Methodology (DSRM) Figura 1 incorpora princípios, práticas e procedimentos que são necessários para realização de trabalhos de investigação e atende aos seguintes objetivos:

- Conceder um modelo de processo nominal para investigação no DS;
- Conceder um modelo mental para a apresentação e avaliação da investigação no DS em Sistemas de Informação (SI).

No processo de DS, estão presentes seis etapas: **Identificação do Problema e Motivação** (***Identify Problem & Motivate***), **Definição dos Objetivos da Solução** (***Define Objectives of a Solution***), **Design e Desenvolvimento** (***Design & Development***), **Demonstração** (***Demonstration***), **Avaliação** (***Evaluation***) e **Comunicação** (***Communication***). A metodologia assim projetada satisfaz efetivamente aos objetivos e tem o potencial de ajudar a aceitação da investigação no DS na disciplina de SI (Gregor & Hevner, 2013; Hevner & Chatterjee, 2010; March & Smith, 1995; Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007).

a) **Identificação do Problema e Motivação** (***Identify Problem & Motivate***) – Definir o problema especifico da investigação e justificar o valor da solução. A definição do problema será utilizada para desenvolver um artefacto que pode eventualmente fornecer uma solução e ser útil para atomizar o problema conceitualmente, de modo que a solução possa capturar toda a complexidade. Justificar o valor da solução visa realizar duas coisas essencialmente motivar o investigador e o público da investigação a procurar uma solução e a aceitar os resultados, e ajudar a perceber o raciocínio associado à compreensão do problema pelo investigador. Os recursos necessários para esta atividade incluem o conhecimento do estado do problema e a importância da solução.

b) **Definição dos Objetivos da Solução** (***Define Objectives of a Solution***) – Inferir os objetivos da solução a partir da definição do problema e do conhecimento do que é possível e viável. Os objetivos podem ser quantitativos, por exemplo, em termo de que uma solução desejada seria melhor do que os atuais, ou qualitativa, por exemplo, uma descrição de como um novo artefacto deverá suportar as soluções do problema até agora não abordado. Os objetivos devem ser inferidos racionalmente a partir da especificação do problema. Os recursos necessários para esta atividade incluem o conhecimento do estado do problema e as soluções atuais caso existam e a sua eficácia.

c) **Design e Desenvolvimento** (***Design & Development***) – Criar os artefactos. Estes artefactos são potencialmente construções, modelos, métodos ou instâncias (cada um definido amplamente) ou novas propriedades de recursos técnicos, sociais e/ou internacionais. Conceitualmente, um artefacto da investigação do Design pode ser qualquer objeto desenhado com a contribuição da investigação que está incorporada no projeto. Esta atividade inclui a determinação da funcionalidade desejada do artefacto e a sua arquitetura, e de seguida para criar o artefacto real. Os recursos necessários para passar de Objetivos **(b)** para Design e Desenvolvimento **(c)** incluem o conhecimento da teoria que pode ser levado a uma solução.

d) **Demonstração** (***Demonstration***) – Demonstrar a utilidade do artefacto para resolução de uma ou mais instâncias do problema. Isto poderia envolver a sua utilização na experimentação, simulação, caso de estudo, provas ou outras atividades apropriadas. Os

recursos necessários para a demonstração incluem o conhecimento efetivo de como utilizar o artefacto para resolver o problema.

e) **Avaliação** (***Evaluation***) – Observar e medir o quão bem é o artefacto na resolução do problema. Esta atividade envolve a comparação dos objetivos da solução com os resultados reais observados da utilização do artefacto na demonstração. Requer conhecimentos de métricas e técnicas de análise relevantes. Dependendo da natureza do local do problema e do artefacto, a avaliação pode assumir muitas formas. Poderia incluir itens como a comparação da funcionalidade do artefacto com os objetivos da solução da atividade acima em **(b)**, medidas objetivas de desempenho quantitativo, como por exemplo, os orçamentos ou itens produzidos, resultados das investigações de satisfação e *feedback* de clientes ou simulações. Poderia incluir medidas quantificáveis do desempenho do sistema, como o tempo de resposta ou a disponibilidade. Conceitualmente, tal avaliação poderia incluir qualquer evidência empírica apropriada ou prova lógica. No final desta atividade, os investigadores podem decidir se é necessário iterar novamente a terceira atividade, de modo a tentar melhorar a eficácia do artefacto ou para continuar a comunicação e deixar as melhorias para os projetos subsequentes. A natureza do local de investigação pode determinar se tal iteração é viável ou não.

f) **Comunicação** (***Communication***) – Comunicar o problema e a sua importância, o artefacto, a sua utilidade e novidade, o rigor do seu desenho e sua eficácia para os investigadores e outros públicos relevantes, como profissionais praticantes, quando apropriado. Em investigações académicas, os investigadores podem utilizar a estrutura desse processo para estruturar o artigo, assim como a estrutura nominal de um processo de investigação empírica (definição do problema, revisão de literatura, desenvolvimento de hipóteses, recolha de dados, análise dos resultados, discussão e conclusão). É uma estrutura comum para trabalhos empíricos de investigação. A comunicação requer conhecimento da cultura disciplinar (Bello Garba, Armarego, Murray, & Kenworthy, 2015; Hevner & Chatterjee, 2010).

Este processo encontra-se estruturado na ordem nominal sequencial, mas, porém, não há expectativas de que os investigadores sigam sempre a ordem sequencial da primeira etapa **(a)** até

a sexta etapa **(f)**. Na realidade, os investigadores podem começar em qualquer etapa e seguir em frente. Uma abordagem centrada no problema é a base da sequência nominal, que inicia na primeira etapa **(a).** Os investigadores podem utilizar esta sequência caso a ideia para a investigação resultou-se da observação do problema ou da investigação futura sugerida num artigo de um projeto anterior. Uma solução centrada nos objetivos inicia-se na segunda etapa **(b)** e pode desencadear-se numa indústria ou necessidade de investigação que deve ser abordada através do desenvolvimento de um artefacto. Uma abordagem centrada no *design* e no desenvolvimento inicia -se na terceira etapa **(c)**. Como tal, resultaria na existência de um artefacto que ainda não tenha sido formalmente pensado como uma solução para o domínio do problema explícito no qual será utilizado. De igual modo, tal artefacto poderia vir de outro domínio da investigação, podendo já ter sido utilizado para resolução de um problema diferente, ou pode ter aparecido como uma ideia análoga. Por último, uma solução iniciada pelo cliente/contexto poderia basear-se na observação de uma solução prática que já funciona. Para esta solução geralmente inicia-se na quarta etapa **(d)**, resultando numa solução de DS se os investigadores trabalharem para trás, de modo a aplicar o rigor ao processo de forma retroativa (Bello Garba et al., 2015; Hevner & Chatterjee, 2010).

2.2. Scrum

O *Scrum* é uma *framework* que é utilizada para resolver complexos ao mesmo tempo, de forma criativa e produtiva e no fim, é entregado produtos com o maior valor possível (Scrum Guides, 2016).

O *Scrum* é uma *framework* que tem sido utilizada para gerir o desenvolvimento de produtos complexos desde o início da década de 1990. Este não é um processo ou uma técnica para construir produtos; contrariamente, é uma *framework* dentro da qual pode ser utilizada por diversos processos e técnicas. O *Scrum* torna clara a eficácia relativa das práticas de gestão e de desenvolvimento de produtos de cada um, para que seja possível melhorar (Scrum Guides, 2016).

O *Scrum* é (Scrum Guides, 2016):

- Leve;
- Simples de perceber;
- Difícil de dominar.

A *framework Scrum* é constituída por Equipas *Scrum* e suas funções associadas (ou papéis de cada indivíduo), eventos, artefactos e regras. Cada componente dentro da *framework* tem um propósito específico e é essencial para o sucesso e utilização do *Scrum*. As regras do *Scrum* unem os eventos, funções e artefactos, gere as relações e interações entre eles (Scrum Guides, 2016).

O *Scrum* fundamenta na teoria de controlo de processo empírico (empirismo). O Empirismo afirma que o conhecimento vem da experiência e da tomada de decisão baseada no que é conhecido. O *Scrum* utiliza uma abordagem iterativa e incremental para otimizar a previsibilidade e o controlo do risco. Existem três pilares que sustentam qualquer implementação do controlo de processo empírico que são: **transparência**, **inspeção** e **adaptação** (Scrum Guides, 2016).

- **Transparência** – Os aspetos importantes do processo devem ser visíveis para aqueles que são responsáveis pelo resultado. A transparência exige que esses aspetos sejam definidos por um padrão comum de forma a que os observadores tenham uma perceção comum sobre o que está a ser visualizado.

- **Inspeção -** Os utilizadores do *Scrum* devem inspecionar com regularidade os artefactos *Scrum* e respetivo progresso em direção ao objetivo da *sprint* para detetar variações indesejáveis. A inspeção não deve ser tão frequente que atrapalhe a execução do trabalho. As inspeções são mais vantajosas quando realizadas diligentemente por inspetores qualificados no trabalho sujeito a verificação.

- **Adaptação** – Se um inspetor determinar que um ou mais aspetos de um processo se desviaram para além dos limites aceitáveis, e que o produto resultante será inaceitável, o processo ou o material em produção têm de ser ajustado. O ajustamento deve ser feito o mais rapidamente possível para minimizar desvios adicionais.

A equipa *Scrum* é constituída pelo ***Product Owner***, a **Equipa de Desenvolvimento** e **Scrum *Master*** (Scrum Guides, 2016).

- ***Product Owner*** – é o responsável por maximizar o valor do produto e do trabalho da equipa de desenvolvimento. A forma como isso é concretizado pode variar bastante entre organizações, equipas Scrum e indivíduos. Este também é o responsável pela gestão do *Product Backlog*.

- **Equipa de Desenvolvimento** – é formada por profissionais que fazem o trabalho para entregar um incremento "*Done*" (feito) potencialmente utilizável do produto, no final de cada *sprint*. Apenas membros da equipa de desenvolvimento criam o incremento. As equipas de desenvolvimento são estruturadas e capacitadas pela organização para organizarem e gerirem o seu próprio trabalho. A sinergia daí resultante, otimiza a eficiência e a eficácia global da equipa de desenvolvimento.

- ***Scrum Master*** – é o responsável por garantir que o *Scrum* é compreendido e colocado em prática. Este garante que a equipa *Scrum* adere à teoria, práticas e regras do *Scrum*. O *Scrum Master* é um líder-servo da equipa *Scrum*. O *Scrum Master* ajuda aqueles que não pertencem à equipa *Scrum* a entender que interações são e não são úteis à equipa *Scrum*. O *Scrum Master* ajuda todas as pessoas a modificar essas interações para maximizar o valor criado pela equipa *Scrum*.

O *Scrum* prescreve quatro eventos formais para inspeção e adaptação, conforme indicado anteriormente. Os eventos *Scrum* são:

- **Planeamento da *Sprint*** (*Sprint Planning*);
- **Reunião de Sincronização Diária** (*Daily Scrum*);
- **Revisão da *sprint*** (*Sprint Review*);
- **Retrospetiva da *sprint*** (*Sprint Retrospective*).

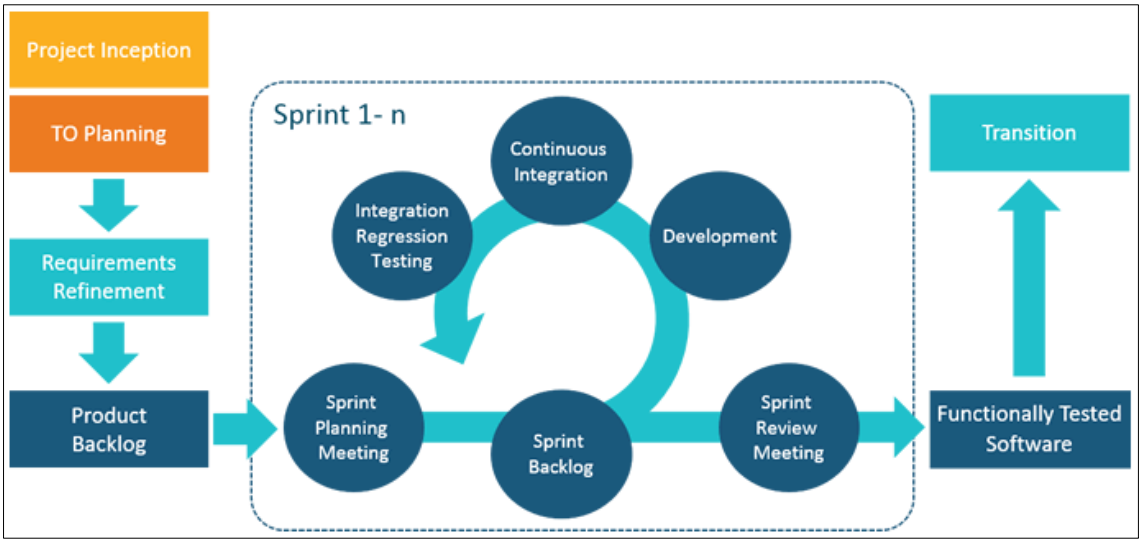


Figura 2 - Metodologia Scrum retirado de (Scrum Guides, 2016)

- ***Sprint*** – é o elemento vital do *Scrum*, com duração limitada de um mês ou menos, durante o qual se cria um incremento potencialmente comercializável. As *sprints* terão de ter durações consistentes durante todo o esforço de desenvolvimento. Inicia-se uma nova *sprint* imediatamente após a conclusão da *sprint* anterior. As s*prints* contêm e consistem na reunião de planeamento da *sprint*, nas reuniões diárias ("*Daily Scrums*"), no trabalho de desenvolvimento per si, na revisão da *sprint* e na retrospetiva de *sprint*.

- **Planeamento da *Sprint*** – efetua-se o planeamento do trabalho a executar na *sprint* (reunião de planeamento da *sprint*). É criado um plano, recorrendo ao trabalho colaborativo de toda a equipa *Scrum*. O *Scrum Master* garante que o evento acontece e que os participantes entendem o seu propósito. O *Scrum Master* ensina a equipa a mantê-lo dentro do limite temporal estabelecido.

- **Reunião de Sincronização Diária –** a reunião de sincronização diária é uma reunião de duração limitada a 15 minutos para a equipa de desenvolvimento sincronizar atividades e criar um plano para as próximas 24 horas. Isto é feito inspecionando o trabalho desde a última reunião de sincronização diária e prevendo o trabalho que poderá ser realizado até à próxima. A reunião de sincronização diária acontece, como o próprio nome indica, diariamente, sempre à mesma hora e no mesmo local para reduzir complexidade. O

Scrum Master assegura que a equipa tem a reunião, mas a equipa é que é responsável por conduzi-la. O *Scrum Master* ensina a equipa a manter esta reunião dentro da duração limite de 15 minutos. O *Scrum Master* reforça a regra de que apenas os membros da equipa de desenvolvimento participam na reunião de sincronização diária. As reuniões de sincronização diárias melhoram a comunicação, eliminam outras reuniões e identificam impedimentos ao desenvolvimento para que possam ser removidos. Além disso, a reunião de sincronização diária promove a tomada de decisões de forma rápida e melhora o nível de conhecimento geral da equipa de desenvolvimento. Esta reunião é fulcral no processo de inspeção e adaptação.

- **Revisão da *Sprint*** – no final da *sprint* ocorre uma reunião denominada revisão da *sprint* que serve para inspecionar o incremento e adaptar o *Product Backlog* se necessário. Durante a revisão da *sprint* a equipa *Scrum* e os restantes *stakeholders* colaboram sobre o que foi feito durante a *sprint*. Com base nesse resultado e quaisquer outras alterações ao *Product Backlog* durante a *sprint*, os participantes colaboram nas próximas atividades que possam ser executadas para otimizar valor. Esta é uma reunião informal, não uma reunião de ponto de situação, e a apresentação do incremento serve para fomentar colaboração e partilha de *feedback*.

- **Retrospetiva da *Sprint* –** a reunião de retrospetiva da *sprint* é uma oportunidade para a equipa *Scrum* se inspecionar a si própria e criar um plano de melhoramentos a serem executados durante a próxima *sprint*. A retrospetiva do *sprint* ocorre depois da revisão da *sprint* e antes do próximo planeamento da *sprint* (Scrum Guides, 2016).

2.3. Apoio à Decisão

Sistema de apoio à decisão não tem uma definição universal aceitável, mas de acordo com o livro de Turban, existem duas definições aceitáveis.

Sistema de apoio à decisão (***Decision Support System* - DSS**) são sistemas computacionais interativos que ajudam as pessoas a utilizar os dados e modelos para a resolução de problemas não estruturados. É a combinação dos recursos intelectuais dos indivíduos com as capacidades computacionais do computador para melhorar a qualidade das decisões (Turban & Sharda, 2010).

O Sistema de apoio à decisão foi introduzido por Simon (1977), no qual ele identificou três fases principais: ***Intelligence***, ***Design*** e ***Choice***. Mais tarde, juntamente com o Turban, foi adicionado uma quarta fase ***Implementation***. ***Monitoring*** é considerado a quinta fase – uma forma de *feedback*. Mas, no entanto, às vezes ***Monitoring*** pode ser visto com a fase da ***Intelligence*** aplicada à fase de ***Implementation***. O modelo de Simon é a caracterização mais concisa e completa do processo de tomada de decisão racional (Turban & Sharda, 2010).

1- **Inteligência *(Intelligence)*** – esta fase envolve a busca de condições que exigem decisões;
2- **Design *(Design)*** – esta fase envolve inventar, desenvolver e analisar as possíveis alternativas de ações (soluções);
3- **Escolha *(Choice)*** – esta fase envolve a seleção de um curso de ação disponível;
4- **Implementação *(Implementation)*** – esta fase envolve a adaptação do curso de ação selecionado para a situação da decisão (resolução do problema ou oportunidade de explorar) (Turban & Sharda, 2010).

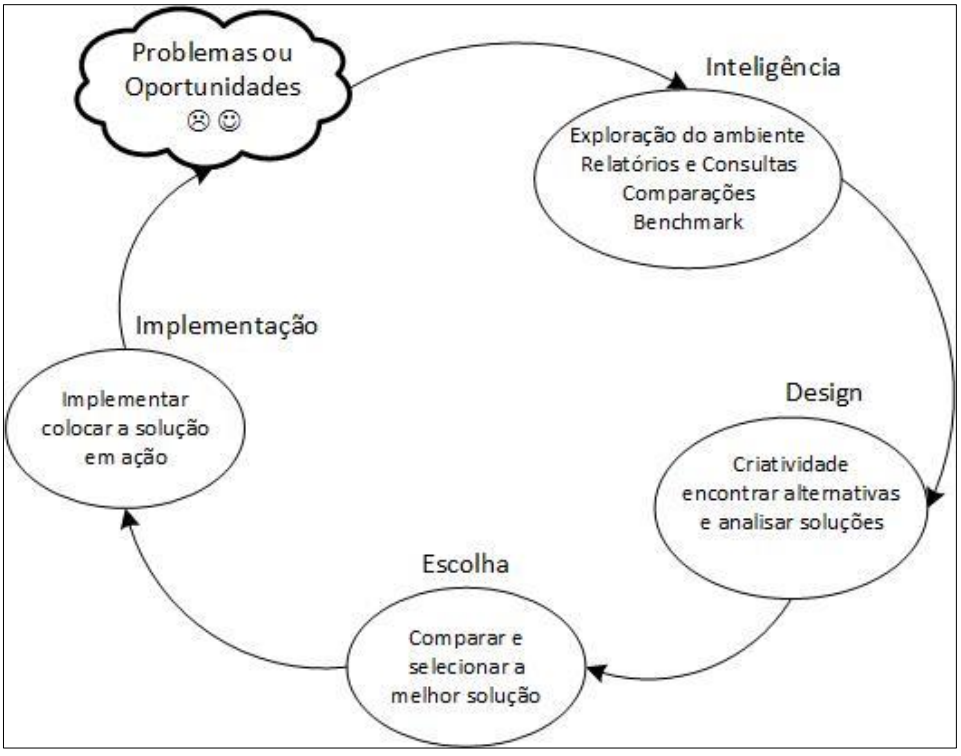


Figura 3 - Metodologia de apoio à decisão adaptado de Santos 2016

O processo de tomada de decisão inicia na fase ***Intelligence***, na qual o investigador analisa a realidade, identificando e definindo o problema. A propriedade do problema também é estabelecida. No ***Design*** é construído um modelo que representa o sistema. Isto é possível com base nas suposições que simplificam a realidade e anota as relações entre todas as variáveis. O modelo é então validado, e os critérios são determinados num conjunto de princípios para a avaliação dos cursos alternativos de ação que são identificados. Muitas vezes, o processo de desenvolvimento de modelos identifica soluções alternativas e vice-versa (Turban & Sharda, 2010).

A fase ***Choice*** inclui a seleção de uma solução proposta para o modelo (não necessariamente para o problema que representa). A solução é testada para determinar a sua viabilidade. Caso a proposta for aceitável, passa-se para a próxima fase, a Implementação (***Implementation)*** da decisão (não necessariamente de um sistema). Uma implementação bem-sucedida resulta na solução do problema real. Em caso de falha, leva a retomar a uma fase anterior do processo. Na verdade, é possível retornar a uma fase anterior durante qualquer uma das três fases (Turban & Sharda, 2010).

a) Fase de Inteligência (*Intelligence*)

A inteligência na tomada de decisão é um processo que percorre todo o ambiente de forma continua ou intermitente. Este processo geralmente inclui várias atividades que estão voltadas para à identificação das situações de problemas ou de oportunidades. Por vezes, pode incluir a monitorização (***Monitoring***) dos resultados da fase de implementação (***Implementation***) de um processo de tomada de decisão (Turban & Sharda, 2010).

A identificação do problema (ou oportunidade) na fase da inteligência inicia-se com a identificação dos objetivos e metas organizacionais relacionados com a questão em causa e determinar se vão cruzar. Os problemas ocorrem devido à insatisfação do "*status quo*". A insatisfação é o resultado de uma diferença entre o que as pessoas desejam (ou esperam) e o que está a acontecer. Numa primeira fase, o investigador procura determinar se existem problemas, identificando os seus sintomas e determinar a sua magnitude definindo os explicitamente. Muitas vezes, o que pode ser descrito como um problema pode ser apenas um sintoma de um problema. Isto acontece porque existem muitos fatores que estão inter-relacionados que geralmente complicam os

problemas do mundo real, o que torna frequentemente difícil de fazer a distinção entre os sintomas e o problema real (Turban & Sharda, 2010).

b) Fase de Design (*Design*)

A fase de *Design* envolve o encontro ou o desenvolvimento e a análise de possíveis cursos de ações. Estas incluem a compreensão do problema e o teste da viabilidade das soluções. Nesta fase, um modelo do problema de tomada de decisão é construído, testado e validado (Turban & Sharda, 2010).

A modelação envolve a conceitualização do problema e a sua abstração de forma quantitativa e/ou qualitativa. Para um modelo matemático, são identificadas as variáveis bem como as respetivas relações mutuas. Simplificações devem ser feitas, sempre que necessários, com o objetivo de encontrar um equilíbrio adequado entre o nível de simplificação do modelo e a representação da realidade que deve ser obtido entre o "*trade-off*" do custo-benefício. Modelos mais simples levam a menos custos de desenvolvimento e manipulação, e a uma solução mais rápida, mas é menos representativo do problema real. Os dados são mais fáceis de obter (Turban & Sharda, 2010).

O processo de modelação é uma combinação da arte e da ciência. Na ciência, existem muitas classes de modelos padrões disponíveis, e com a prática, o analista pode determinar o modelo que é aplicável a determinada situação. Na arte, a criatividade e delicadeza são necessárias para determinar quais são as suposições simplificadoras que podem funcionar, como combinar os recursos apropriados das classes do modelo e integrar modelos para obtenção de soluções válidas. Os modelos possuem variáveis de decisão que descrevem as alternativas entre as quais um investigador deve escolher uma variável de resultado ou um conjunto de variáveis de resultado que descreve o objetivo ou o objetivo do problema de tomada de decisão e as variáveis, ou parâmetros incontroláveis que descrevem o ambiente. O processo de modelação envolve a determinação das relações entre as variáveis (Turban & Sharda, 2010).

c) Fase de Escolha (*Choice*)

A escolha é um ato crítico da tomada de decisão. Esta fase é aquela em que a decisão real e o compromisso de seguir determinado curso de ação são escolhidos. A fronteira entre as fases de ***Design*** e ***Choice*** é muitas vezes pouco clara porque certas atividades podem ser realizadas durante ambas e também porque o investigador pode frequentemente retroceder para as atividades da fase *Choice* para projetar novas atividades (por exemplo, gerar novas alternativas ao realizar uma avaliação das existentes). Esta fase inclui a procura, avaliação e recomendação de uma solução adequada para um modelo. Uma solução para o modelo é um conjunto específico de valores para as variáveis de decisão em uma alternativa selecionada (Turban & Sharda, 2010). É de notar que, resolver um modelo não é o mesmo que resolver o problema que o modelo representa. A solução para o modelo fornece uma solução recomendada para o problema. O problema é considerado resolvido apenas se a solução recomendada foi implementada com sucesso (Turban & Sharda, 2010).

A resolução de um modelo de tomada de decisão envolve a busca de um curso de ação apropriado. As abordagens de pesquisa incluem **técnicas analíticas** (resolução de uma fórmula), **algoritmos** (procedimentos passo-a-passo), **heurísticas** (regras do polegar) e **procura cega** (tiro no escuro, idealmente de forma lógica). Cada alternativa deve ser avaliada. Se uma alternativa apresentar múltiplos objetivos, todos eles devem ser examinados e equilibrados uns com os outros. A **análise de sensibilidade** é utilizada para determinar a robustez de qualquer alternativa. Ligeiras alterações nos parâmetros devem idealmente levar a ligeiras ou inexistentes alterações na alternativa escolhida. A análise "*What if*" é utilizada para explorar grandes mudanças nos parâmetros. A busca de metas ajuda o investigador a determinar os valores das variáveis de decisão para atender os objetivos específicos (Turban & Sharda, 2010).

d) Fase de Implementação (*Implementation*)

A implementação de uma solução proposta para um problema é, na verdade, o início de uma nova sequência de introdução de mudanças. Estas mudanças devem ser geridas. As expectativas dos utilizadores devem ser geridas como parte de gestão das mudanças (Turban & Sharda, 2010). A definição da implementação é um bocado complicado porque a implementação é um processo longo que envolve limites vagos. A fase de **implementação** envolve a colocação de uma solução de recomendação para o trabalho, não é necessário implementar um sistema computacional.

Muitas questões de implementação genérica, como a resistência ao grau de mudança de suporte da alta gestão e treinamento do utilizador são importantes para lidar com o *Management Support Systems* (MSS). Algumas decisões são testadas experimentalmente pelas pessoas responsáveis por esses aspetos de tomada de decisão antes da decisão ser implementada globalmente (Turban & Sharda, 2010).

A metodologia de apoio à decisão apresentado tem o propósito de cruzar com a metodologia DSR com o intuito de proporcionar melhor análise dos problemas e a consequente tomada de decisão com o intuito de obter melhores resultados. Estas metodologias serão conjugadas para produzir trabalho de investigação com melhor qualidade, i.e., a conjugação das duas metodologias é feita com o intuito de tomar a melhor decisão possível no momento da tomada de decisão no desenvolvimento do artefacto de modo a solucionar a questão de investigação. Assim sendo, sentiu-se a necessidade de criar um mapeamento entre as metodologias apresentadas para a realização desta dissertação. A Tabela 1, apresenta o mapeamento entre as duas metodologias.

Tabela 1 - Mapeamento entre as metodologias de investigação

	DSR	**Apoio à Decisão**
Fases	Identificação e Motivação do Problema	Inteligência
	Definição dos Objetivos da Solução	
	Design e Desenvolvimento	*Design* e Escolha
	Demonstração	Implementação e Monitorização
	Avaliação	
	Comunicação	

Na primeira fase da dissertação, procedeu-se à identificação do problema e motivação, bem como a definição dos objetivos da solução (correspondente à metodologia DSR) o que na metodologia de apoio à decisão corresponde a fase da inteligência. De acordo com este mapeamento, procedeu se então ao estudo sobre o estado da arte do tema e ao desenvolvimento da revisão de literatura.

Na segunda fase da dissertação, procedeu-se à conceção dos protótipos não funcionais, correspondente a fase *design* de ambas as metodologias. De seguida, efetuou-se à escolha do *design* de um dos protótipos apresentados para a criação do artefacto proposto (protótipo funcional).

Na terceira fase procedeu-se a demonstração, avaliação e comunicação (DSR) dos resultados conseguidos o que na metodologia apoio à decisão corresponde às fases de implementação da solução e monotorização da mesma.

No caso metodologia *Scrum*, esta foi aplicada durante a execução da parte prática desta dissertação. Basicamente, o *Scrum* foi utilizado para efetuar toda a gestão e o desenvolvimento da parte prática da dissertação.

3. Revisão de Literatura

A realização do estudo sobre o estado da arte constitui uma base teórica de suporte para o trabalho prático a realizar. Este capítulo foca-se no desenvolvimento da revisão de literatura no qual aborda temas tais como os dispositivos e plataformas móveis, o conceito *Bring Your Own Device* (BYOD), a utilização dos dispositivos móveis no *healthcare,* as metodologias de desenvolvimento de aplicações móveis e as *frameworks* disponíveis para o desenvolvimento de aplicações móveis multiplataformas. Para isso, foi necessário realizar pesquisas sobre os artigos relacionados com o desenvolvimento de aplicações móveis, da utilização dos dispositivos móveis no *healthcare* e do BYOD. As pesquisas dos artigos foram realizadas nos repositórios *online*, tais como: Scholar Google, Scopus, Science Direct, IEEE, Web of Science e b-on.

As pesquisas dos artigos foram feitas com base nas seguintes *keywords* (palavras-chaves) e no cruzamento das mesmas:

- *Bring your own device;*
- *Bring your own device in healthcare;*
- *Comparison of cross-platform approaches;*
- *Cross-platform development;*
- *Cross-platform frameworks;*
- *Healthcare;*
- *Mobile device in healthcare;*
- *Mobile application development.*

O processo de seleção dos artigos foi realizado com base nas seguintes características:

- Número de citações do artigo;
- Ano em que o artigo foi escrito;
- Leitura do *Abstract* do artigo;
- Ano de publicação do artigo.

3.1. Introdução

O mercado dos dispositivos móveis tem evoluído muito nos últimos anos, devido à popularidade e massificação dos *smartphones* e *tablets*, o que veio a dar origem a um novo mercado das aplicações móveis para estes dispositivos. Trata-se de um mercado que já se encontra muito consolidado e essas aplicações são indispensáveis nas nossas atividades do dia-a-dia atualmente.

Ao longo deste capítulo serão abordados vários temas relacionados com o desenvolvimento e modelação de aplicações móveis bem como o conceito BYOD, também conhecido por "traga o seu próprio dispositivo" em Português, a utilização de dispositivos móveis no *healthcare* e o desenvolvimento multiplataformas de aplicações móveis. Este capítulo inicia-se com a apresentação de algumas aplicações do género assistente pessoal, depois segue-se uma análise dos principais dispositivos e plataformas móveis presentes no mercado atual. Em seguida, é apresentado o conceito BYOD na qual é apresentado os seus benefícios, os desafios e problemas que existem no ambiente BYOD, bem como também as soluções propostas para colmatar tais desafios e problemas. Será também apresentado o BYOD no *healthcare*. Depois é apresentado casos práticos de plataformas do *healthcare* que utilizam de dispositivos móveis, tais como o *Medical Mobile Data Collecting Platform* (MMDCP) implementada em alguns hospitais na Polónia e a Agência para Integração, Difusão e Arquivo de Informação Médica (AIDA) utilizada em Portugal, na qual futuramente a solução será executada.

O estudo sobre o desenvolvimento de aplicações móveis é próximo tema a ser abordado sendo que estas podem seguir duas abordagens: abordagem nativa ou abordagem multiplataforma (com maior destaque para a abordagem multiplataforma). A abordagem multiplataforma encontra-se divida em quatro tipos de abordagens diferentes: Web, híbrida, compilação-cruzada e interpretada. Ainda, é realizado um estudo sobre os critérios de seleção e a análise comparativa entre as abordagens nativa e multiplataforma.

Seguidamente, é introduzido o padrão *Model Views Controller* (MVC), que é utilizado pela maioria das *frameworks* de desenvolvimento multiplataforma. Também, é apresentado a análise sobre as tecnologias de desenvolvimento multiplataforma (*frameworks*), bem como a análise

comparativa entre as mesmas. Após a análise das tecnologias, é apresentada a arquitetura geral de desenvolvimento de aplicações móveis multiplataformas.

a) Assistente Pessoal

Assistente Pessoal ou Assistente Virtual é uma aplicação capaz de realizar um conjunto de tarefas com objetivo de ajudar os utilizadores nas suas atividades diárias. Atualmente existem inúmeras aplicações do género assistente pessoal, cada uma com seu modelo de negócio específico. Embora existem diferentes tipos de aplicações do género assistente pessoal, todos têm algo em comum, que é ajudar os utilizadores nas suas atividades diárias de forma eficiente.

b) Trabalhos Relacionados

Nesta subsecção será apresentada alguns dos projetos encontrados sobre o Assistente Pessoal (Tabela 2, Tabela 3, Tabela 4, Tabela 5 e Tabela 6). Para este efeito, foi necessário efetuar uma a análise às diferentes soluções encontradas, para verificar as possíveis funcionalidades que podem ser desenvolvidas a partir das aplicações já existentes na área do Assistente Pessoal. As aplicações encontradas na área do Assistente Pessoal são: ***Google Now*** **(*Google Assistant*)**, ***Siri***, ***myCol***, ***Cortana*** e ***Speaktoit*** que de seguida serão apresentadas nas tabelas seguintes.

Tabela 2 - Assistente Pessoal Cortana adaptado de (Trusted Reviews, 2016)

Nome	**Cortana**
Descrição	Assistente Pessoal da Microsoft
Funcionalidades	Efetuar chamadas, enviar mensagens de texto, efetuar pesquisas na web (utilizando o *Bing*), apresentar eventos locais, procurar por restaurantes e bares, adicionar/alterar eventos do calendário, efetuar anotações, apresentar relatórios meteorológicos, calcular o tempo levará para chegar a casa/trabalho, oferecer dicas de viagem, identificar a música, definir lembretes quando você conversar com uma pessoa.
Sistema Operativo	*Windows Phone*

Tabela 3 - Assistente Pessoal do Google Now (Google Assistant) adaptado de (Trusted Reviews, 2016)

Nome	**Google Now (Google Assistant)**
Descrição	Assistente Pessoal da Google para os dispositivos *Android*
Funcionalidades	Apresentar relatório meteorológicos, obter as direções, obter as informações dos transportes públicos, indicar como chegar ao destino de carro ou a pé, ou de *train*/autocarro, definir lembretes sobre eventos do calendário, obter notificação de email e avisos, obter atualizações das equipas de desportos, atualizações sobre as ações, apresentar as informações com base em suas pesquisas na internet, permitir pesquisar na internet com voz / digitação, lançar assistência contextual com base no que está no ecrã (*Android* 6.0), reconhecer músicas, reproduzir músicas, ranking de aplicações, realizar comandos SMS/chamadas, cálculos.
Sistema Operativo	*Android*

Tabela 4 - Assistente Pessoal Hospitalar myCol adaptado de (Core Communique, 2016; myCol, 2016)

Nome	**myCol**
Descrição	Assistente pessoal que facilita a sua estadia no Hospital
Funcionalidades	Gerir o processo de admissão, permite reduzir o tempo de espera para obter uma cama, e o tempo de espera nas filas, efetuar a pré-aprovações de seguros com as companhias de seguros de saúde, agilizar o processo no momento de saída do hospital, efetuar o acompanhamento de qualquer consultas pós-cuidado. Efetuar pesquisas baseada nos mapas: informações genuínas, verificadas e imparcial de todas as instalações médicas na cidade, independentemente do tamanho, para que o utilizador possa fazer uma escolha informada; *Digitize Health Records*: permite guardar e partilhar todos os registos de saúde do utilizador. Lembretes: das consultas médicas ou qualquer medicação.
Sistema Operativo	*Android* e *iOS*

Tabela 5 - Assistente Pessoal da Apple adaptado de (Trusted Reviews, 2016)

Nome	**Siri**
Descrição	Assistente Pessoal para os dispositivos *iOS*
Funcionalidades	Efetuar pesquisas na internet, envio de SMS, serviços de chamadas, abrir outras aplicações, efetuar chamadas *FaceTime*, envio de *tweets*, atualizar o status do facebook, sugerir aplicações e contatos, configurar/alterar eventos do calendário, consultar os detalhes do calendário, definir alarmes, alterar as configurações do telefone, consultar os detalhes do filme, procurar direções no Apple Maps, solicitar informações dos transportes públicos, localizar as atrações/restaurantes próximos, ler mensagens de texto, ler e-mails não lidos, apresentar fotos por data e local.
Sistema Operativo	*iOS*

Tabela 6 - Assistente Pessoal Speaktoit adaptado de (Trusted Reviews, 2016)

Nome	**Speaktoit**
Descrição	Assistente Pessoal que realiza as suas atividades a base de reconhecimento de voz
Funcionalidades	Apresentar relatório meteorológico, adicionar/alterar eventos no calendário, definir lembretes, gerir a lista dos contactos, realizar comandos SMS/Chamadas, efetuar navegação no mapa (obter direções), permitir pesquisas na internet com voz / digitação, executar outras aplicações e modificar as configurações do sistema (apenas possível no Android)
Sistema Operativo	*Android*, *iOS* e *Windows Phone*

3.2. Dispositivos e Plataformas Móveis

"Mobility can be defined as the capability of being able to move or be moved easily. In the context of mobile computing, mobility pertains to people's use of portable and functionality powerful mobile devices that offer the ability to perform a set of application functions untethered, while also being able to connect, to obtain data from and provide data to other users, applications and systems"(Hernandez, Viveros, & Rubio, 2013). Segundo o autor, a mobilidade é a capacidade de mover ou ser movido facilmente. No caso da computação móvel, a mobilidade refere-se à utilização de dispositivos móveis poderosos com enorme capacidade computacional para realização das diversas tarefas.

Os dispositivos móveis de pequenas dimensões, inteligentes e com grande capacidade de computação estão a tornar-se cada vez mais presentes e essenciais nas diversas atividades no nosso dia-a-dia, tanto a nível pessoal como a nível profissional. Estes dispositivos estão em constante evolução e crescimento no mercado, quer a nível do *hardware,* quer a nível do *software*, ou seja, dispositivos com melhores *hardware* requer *software* mais sofisticados. O sistema operativo móvel é o responsável pela gestão do *hardware* do dispositivo. Atualmente os principais sistemas operativos móveis são: o *Android*, o *iOS* e o *Windows Phone*, sendo que o *Android* detém a maior cota do mercado 87.6% (linha verde), seguido pelo *iOS* com 11.7% (linha azul), *Windows Phone* com 0.4% (linha roxa) e por fim tem-se os outros sistemas operativos (*BlackBerry OS*, *Symbian OS*) 0.3% (linha castanha). Esses dados são referentes ao estudo realizado pela ***International Data Corporation*** (IDC) em agosto de 2016, os resultados do estudo são apresentados nas figuras (Figura 4 e Figura 5) com mais detalhes. Esses resultados encontram se agrupados por quadrimestre. (Q1, Q2, Q3, Q4), referente ao período compreendido entre o ano de 2013 e 2016.

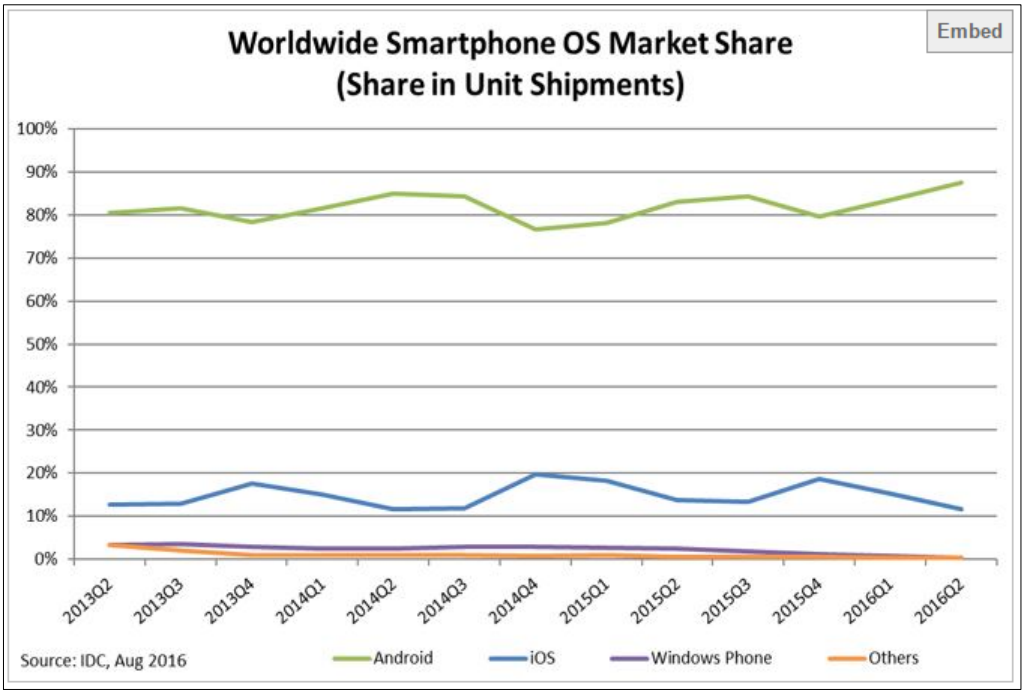


Figura 4 - Smartphone OS Market share Q2 2016 retirado de (IDM International Data Corporation, 2016)

Period	Android	iOS	Windows Phone	Others
2015Q3	84.3%	13.4%	1.8%	0.5%
2015Q4	79.6%	18.6%	1.2%	0.5%
2016Q1	83.4%	15.4%	0.8%	0.4%
2016Q2	87.6%	11.7%	0.4%	0.3%

Source: IDC, Aug 2016

Figura 5 - Smartphone OS Market Q2 2016 retirado de (IDM International Data Corporation, 2016)

3.2.1. Dispositivos Móveis

Vários tipos de dispositivos móveis foram introduzidos nos anos 1990s, entre os quais tem-se o *Personal Digital Assistant, Smartphones, Tablet, Ultra-mobile PC e Wearable Computers* (Nosrati, Karimi, & Hasanvand, 2012). Neste trabalho será abordado apenas três dos dispositivos móveis anteriormente referidos.

a) Personal Digital Assistant

O *Personal Digital Assistant* (PDA) conhecido também como *palmtop* ou assistente de dados pessoais é um dispositivo móvel cuja a principal função é efetuar a gestão dos dados pessoais. Em

1984 **Psion** lançou o primeiro dispositivo móvel "PDA", o **Organizer II**. Em 1991, seguido pelo **Psion** serie 3, começaram a aparecer PDAs com estilos mais familiares (Nosrati, Karimi, et al., 2012). Atualmente os PDAs dispõe de grandes capacidades de processamento, incorporado com funções tais como gestor de agenda e de escritório, com o acesso fácil à internet, intranet e com redes informáticas através do Wi-Fi (Arjun Goud, Vijaya, Rao, & Rao, 2013). Embora os dispositivos moveis PDAs, terem crescido muito, estes começaram a ficar obsoletas com a chegada e adoção dos s*martphones*, que é uma mistura de telemóvel com o PDA, que apresenta maior portabilidade o que levou a ganhar cada vez mais utilizadores (Arjun Goud et al., 2013; Nosrati, Karimi, et al., 2012).

b) Smartphones

O *smartphone* é um telemóvel construído a partir de um sistema operativo (SO) móvel, com maior capacidade computacional, mais avançado e mais capacidade de conexão do que um simples telemóvel. O primeiro *smartphone* era a combinação de PDA com as funcionalidades de um telemóvel. Algumas funções foram adicionadas nos modelos posteriores tais como *media players* portáteis, camara digital compactas *low-end,* camara de vídeos, unidade de navegação *Global Positioning Systems* (GPS), ecrãs sensíveis ao toque (*touchscreen*) de alta resolução e foram incorporadas *browsers* para a navegação em sites e páginas web otimizadas para dispositivos móveis. O Wi-Fi fornece acesso aos dados de alta velocidade e banda larga móvel. Os *smartphones* geralmente correm um dos seguintes sistemas operativos: *Palm OS*, *Windows Mobile* (*Phone Edition*), *RIM OS*, *Symbian OS*, *Linux*, *Android*, *iOS* e *BlackBerry OS*. Atualmente os sistemas operativos mais utilizados nos s*martphones* modernos são: *Android* da Google, *iOS* da Apple e *Windows Phone* da Microsoft (Arjun Goud et al., 2013; Nosrati, Karimi, et al., 2012).

c) Tablet

Os *tablets* (7.0" a 12.5") são dispositivos de dimensão maiores do que os *smartphones* (4.0" a 6.0"). e os PDAs. São tipos de dispositivos móveis integrados com um ecrã de toque plana e são operados a base de toques no ecrã. Estes não apresentam um teclado físico no próprio dispositivo, mas possuem um teclado virtual embutido no ecrã. O conceito de *tablet* originou nos séculos XIX e XX, mas apenas como protótipos e conceito de ideias. O primeiro *tablet* comercial baseado neste conceito apareceu no final do século XIX. A Apple lançou o *iPad* com sistema operativo e

tecnologia *touchscreen* em 2010 e tornou-se o primeiro dispositivo móvel *tablet* a atingir fins comerciais em todo mundo. Após este sucesso, muitos outros fabricantes têm vindo a produzir os seus próprios *tablets*, como é o caso da Samsung, HTC, Motorola, RIM, Sony, Amazon, JP, Microsoft, entre outros (Nosrati, Karimi, et al., 2012).

3.2.2. Sistemas Operativos Móveis

"A mobile software platform is defined as the combination of an operating system for a collection of compatible mobile devices with a set of related software development libraries, application programming interfaces (API), and programming tools" (Hernandez et al., 2013).

Atualmente existem uma vasta gama de sistemas operativos para dispositivos móveis tais como: *Android*, *Bada*, *BlackBerry*, *iOS*, *Palm OS*, *Tizen*, *Symbian*, *Windows Phone*, entre outros. Os sistemas operativos para os dispositivos móveis mais conhecidos e mais utilizados são: *Android*, *iOS*, *Windows Phone* e *BlackBerry OS*. Neste projeto apenas serão abordados os quatros sistemas operativos mais conhecidos e utilizados.

a) Android

O sistema operativo *Android* foi desenvolvido pela *Open Handset Alliance* (OHA), liderado pela Google. OHA é um consórcio de 65 empresas de *hardware*, *software* e serviços de telecomunicações que tem o objetivo de desenvolver uma plataforma *standard* (padrão) para os dispositivos móveis (Hammershoj, Sapuppo, & Tadayoni, 2010; Hee-Yeon Cho, Choon-Sung Nam, & Dong-Ryeol Shin, 2010; Hernandez et al., 2013; Kaur & Sharma, 2014).

O *Android* é um sistema operativo móvel *open source* baseado no *kernel* do Linux, que apresenta uma abordagem unificada para o desenvolvimento de aplicações para dispositivos móveis, isto é, as aplicações *Android* executam em diferentes dispositivos com o sistema operativo *Android* (Hernandez et al., 2013; Oliver & Earl, 2009).

A primeira versão beta do *Android Software Development Kit* (SDK) foi lançado pela Google em 2007, mas a sua versão comercial o *Android* 1.0 só foi lançada em outubro de 2008 (Nosrati, Karimi, et al., 2012). A plataforma *Android* para além de oferecer o sistema operativo e o seu

ambiente de desenvolvimento, também oferece uma máquina virtual (*Dalvik Virtual Machine* - DVM) que executa o código da aplicação, ou seja, as aplicações *Android* são executadas na instância do DVM (Gronli, Hansen, Ghinea, & Younas, 2014; Liu & Yu, 2011). O DVM foi redesenhado e otimizado pela Google para os recursos de *hardware* dos dispositivos móveis. Este pode apresentar várias instâncias num dispositivo e cada instância é executada separadamente em um processo (Liu & Yu, 2011).

A arquitetura do *Android* OS é constituída por quatro camadas: ***Application Layer*** (Camada de Aplicação), ***Application Framework*** (*Framework* da Aplicação), ***Android Runtimes and Libraries*** (Android Runtimes e Bibliotecas) e ***Linux Kernel*** (Kernel do Linux), conforme consta na Figura 6.

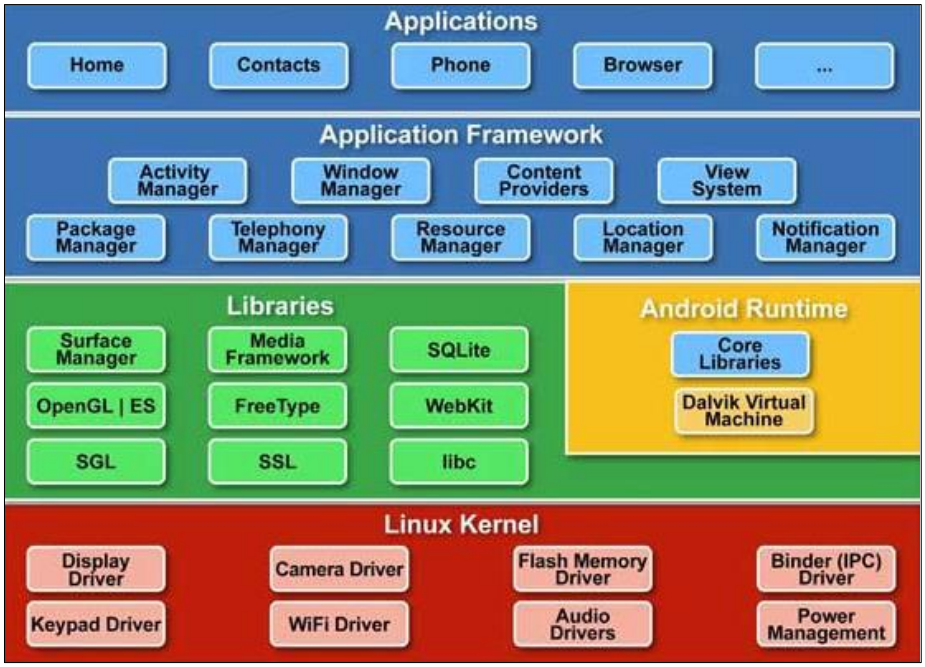


Figura 6 - Arquitetura do Android retirado de (Gronli et al., 2014; Kaur & Sharma, 2014)

- ***Application Layer***

Application Layer é a camada de superior da arquitetura do SO *Android* que utiliza as aplicações instaladas (Ex: telefone, caixa de correio, entre outras). A maioria destas aplicações são nativas tais como Google *Maps*, camara, *browser*, SMS, contactos, calendários (Kaur & Sharma, 2014).

- ***Application Framework***

Application Framework é a camada que permite o acesso às APIs que são utilizadas para o desenvolvimento de aplicações *Android*. Esta camada oferece diferentes pacotes (*packages*) dos serviços das aplicações. Nesta camada estão presentes os gestores (*managers*) que permitem o acesso aos dados. Estes gestores são: (Chao Wang, Wei Duan, Jianzhang Ma, & Chenhui Wang, 2011; Kaur & Sharma, 2014)

- ✓ Gestor de Atividades – controla e faz a gestão correta de todas as atividades e trata do ciclo de vida das aplicações;
- ✓ Gestor de Recursos – permite o acesso a *non-code resources* (ex. gráficos);
- ✓ Gestor de Notificações – permite que todas as aplicações apresentam as alertas personalizadas na barra de estados;
- ✓ Gestor de Localização – permite ativar alertas quando o utilizador entra ou saí de uma determinada localização geográfica;
- ✓ Gestor de *Packages* – permite recuperar os dados instalados no dispositivo;
- ✓ Gestor de Telefone – permite gerir as configurações de ligação a rede e todas as informações sobre os serviços do dispositivo;
- ✓ Gestor de Janelas – permite criar *views* e *layouts* nas janelas;
- ✓ Fornecedor de Conteúdos – permite que uma aplicação tenha acesso aos dados das outras aplicações, ou seja, partilha de dados.

- ***Android Runtimes and Libraries***

Android Runtimes é a camada que é responsável pela execução de todas as aplicações *Android* (Chao Wang et al., 2011). Apesar das aplicações *Android* serem desenvolvidas em linguagem Java, estas não são executadas por uma máquina virtual tradicional do Java. O SO *Android* possui uma máquina virtual específica, que foi desenvolvida por uma equipa de engenheiros da Google, com o intuito de obter o consumo mínimo de memória e o isolamento de processos, denominado de *Dalvik Virtual Machine* (DVM) (Kaur & Sharma, 2014). Esta máquina virtual executa ficheiros no formato DEX (.dex). A Figura 7, ilustra a execução do DVM.

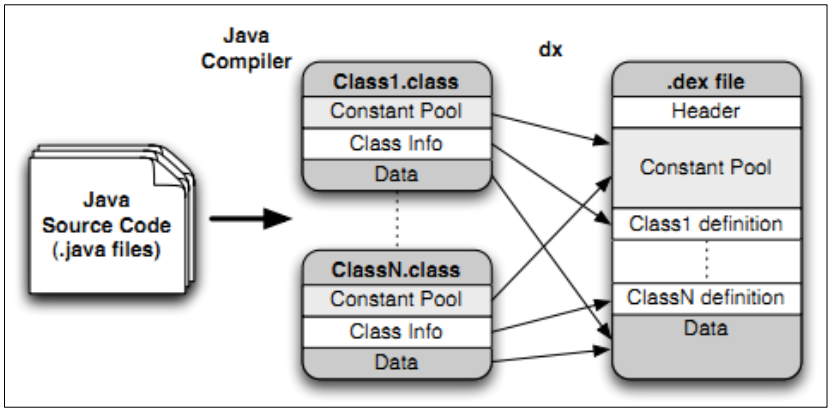


Figura 7 - Dalvik Virtual Machine retirado de (Kaur & Sharma, 2014)

Os ficheiros .class do Java são compilados pelo DVM num único ficheiro .dex que é utilizado para gerar a aplicação no formato .apk para ser instalado no dispositivo móvel com o SO *Android*.

Libraries é uma camada composta por bibliotecas escritas em C/C++ que são utilizados em vários componentes do *Android*. A partir da camada de *Application Framework*, pode-se ter o acesso a estas bibliotecas. Existem uma vasta gama de bibliotecas, tais como WEB que permitem o acesso ao browser WEB, bibliotecas 3D, *SQLite*, *FreeType*, bibliotecas para áudio e vídeo, entre outros (Chao Wang et al., 2011; Kaur & Sharma, 2014).

- ***Linux Kernel***

Linux Kernel é a camada central da arquitetura do SO *Android*, pois nesta camada estão presentes os serviços de gestão da memória, energia, segurança, entre outros serviços. Esta camada fornece um nível de abstração entre o *hardware* e o *software* do dispositivo que contém todos os *drivers* de *hardware* essenciais como a camera, o display, o teclado, entre outros (Kaur & Sharma, 2014).

b) iOS

O *iOS* é o sistema operativo desenvolvido e distribuída pela Apple Inc., inicialmente lançado em 2007 para *iPhone* e *iPod Touch*, que posteriormente foi estendido para outros dispositivos da Apple, como o *iPad* e a *Apple* TV (Nosrati, Hojat, & Hasanvand, 2012). O SO *iOS* faz a gestão do *hardware* do dispositivo e fornece as tecnologias necessárias para desenvolver aplicações nativas (Hernandez et al., 2013).

As aplicações para o *iOS* são desenvolvidas em *Objective-C* utilizando a biblioteca *Cocoa Touch.* O *Objective-C* é uma extensão da linguagem C, enquanto que, *Cocoa Touch* é uma coleção de classes. Enquanto que as linguagens C# e Java (utilizadas para o desenvolvimento de aplicações para *Windows Phone* e *Android*) são muito semelhantes na sintaxe, a biblioteca do *Objective-C* oferece uma alternativa diferente. O *Objective-C* como o próprio nome indica, suporta a programação orientada a objetos. É um superconjunto da linguagem de programação C, que oferece recursos orientados a objetos e tempo de execução dinâmica. A programação orientada a objetos do *Objective-C* é baseada na passagem de mensagens para instâncias do objeto. O *Objective-C* herda da linguagem de programação C a sintaxe, os tipos primitivos e declarações de controlo de fluxo e adiciona sintaxe para definir classes e métodos (Gronli et al., 2014).

A plataforma e a linguagem vêm a ser melhoradas continuamente ao longo dos anos, mas a mudança mais notável foi a introdução *Automatic Reference Counting* (ARC), que proporcionou a melhoria na gestão automática da memória e fez com que a quantidade de código *boilerplate* (códigos que podem ser reutilizados muitas vezes com pouca ou nenhuma alteração) reduzisse e os vazamentos na memória geral fosse menos comum (Gronli et al., 2014).

O *iOS* tem quatro camadas de abstração, as camadas inferiores contêm os serviços e as tecnologias fundamentais dos quais todas as aplicações dependem, enquanto que, as camadas superiores fornecem serviços e tecnologias mais sofisticados. A arquitetura do *iOS* é constituída por quatro camadas, conforme consta a Figura 8 ***Cocoa Touch***, ***Media***, ***Core Services*** e ***Core OS***.

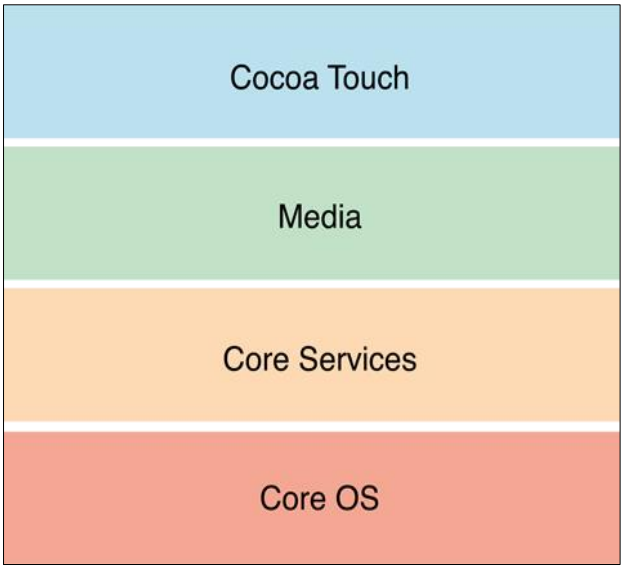


Figura 8 - Arquitetura do iOS retirado de (Apple Inc, 2014; Hee-Yeon Cho et al., 2010)

- ***Cocoa Touch***

A camada *Cocoa Touch* contém as principais *frameworks* de desenvolvimento de aplicações para o *iOS* tais como *Address Book UI Framework, EventKit UI Framework, GameKit Framework*, etc. Esta camada oferece ainda uma estrutura básica às aplicações e o suporte às principais tecnologias multitarefa (*multitasking*), o serviço de notificações da *Apple push*, as entradas baseadas no toque (*touch-based input*), as notificações locais, o *TextKit*, *AirDrop* entre outros serviços de alto nível oferecidos pelo sistema (Apple Inc, 2014; Hee-Yeon Cho et al., 2010).

- ***Media***

A camada *Media* contém as tecnologias gráficas (Ex: *UIKit graphics*, *Core Animations*, etc.), áudio (Ex: *Media Player Framework*, *Core Áudio*, etc.) e vídeo (Ex: *AVKit*, *Core Media*, etc.). estas tecnologias foram desenvolvidas para facilitar a implementação das aplicações multimédias (Apple Inc, 2014; Hee-Yeon Cho et al., 2010).

- ***Core Services***

A camada *Core Services* contém todos os serviços que são essenciais para as aplicações. Dos serviços presentes nesta camada estão o *Core Foundation* e o *Foundation Framework* que definem os tipos básicos que todas as aplicações utilizam. As principais tecnologias presentes

nesta camada são: *Grand Central Dispatch, SQLite, XML Support, In-App Purchase, iCloud Storage*, entre outros (Apple Inc, 2014; Hee-Yeon Cho et al., 2010).

- ***Core OS***

A camada *Core OS* contém todas as características de mais baixo nível que são utilizadas para implementação da maioria das tecnologias. Por vezes existem situações em que são necessários lidar explicitamente com os mecanismos de segurança ou comunicar com um acessório de *hardware* externo, são utilizados as *frameworks* desta camada, tais como: *Accelerate Framework, External Accessory Framework, Local Authentication Framework*, entre outras *frameworks* existentes nesta camada (Apple Inc, 2014; Hee-Yeon Cho et al., 2010).

c) Windows Phone

O SO *Windows Phone* foi concebido pela Microsoft, sendo que inicialmente chamava-se *Windows Mobile*, pois a interface do utilizador era muito similar à versão atual do *Windows*. As aplicações desenvolvidas para o SO *Windows Phone* são escritas em *.NET manage code.* O *manage code* é o código escrito em linguagens de programação que se encontram disponíveis para serem utilizados com a *framework* Microsoft .NET, como por exemplo o C#. O *Windows Phone* 7 suporta duas plataformas de programação muito conhecidas o *Silverlight* e o *XNA Silverlight*, que é o resultado da evolução do *Windows Presentation Foundation* (WPF). Estas plataformas de programação oferecem aos programadores a capacidade de criar novas interfaces de utilizador mais sofisticadas. O XNA é uma plataforma para o desenvolvimento de jogos da Microsoft, com suporte para tecnologias gráficas 2D e 3D. A plataforma de desenvolvimento do *Windows Phone* é o Visual Studio (Hammershoj et al., 2010).

O *Windows Phone* 8 é a versão mais recente do *Windows Phone* e foi lancada em janeiro de 2014. Nesta nova versão do *Windows Phone*, as principais novidades residem no suporte aos CPUs *quad-core e octa-core* e na capacidade de suportar resoluções mais altas (Gronli et al., 2014). A Figura 9, ilustra a arquitetura *Windows Phone* 8, e pode-se constatar na mesma, a arquitetura do *Windows Phone* 8 é composta por quatro camadas.

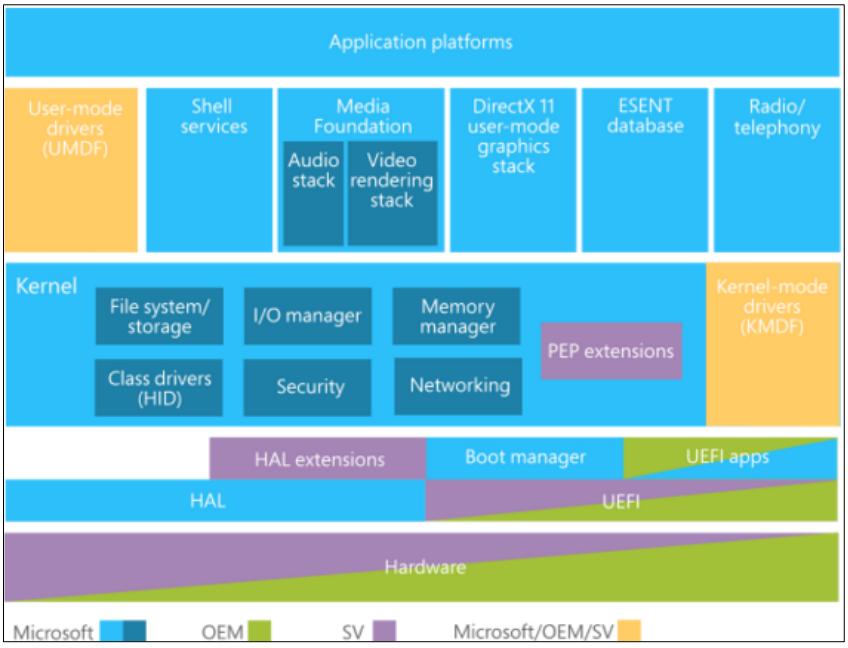


Figura 9 - Arquitetura do Windows Phone 8 retirado de (Gronli et al., 2014)

A camada inferior contém os *drivers* que são necessários para o suporte do *hardware* do dispositivo. A segunda camada contém o *kernel.* Na versão 8 do *Windows Phone* o *kernel* foi alterado para o *Windows* NT, que também é utilizado para sistema operativo *Windows* 8 para computadores, em vez do *Windows* CE que era utilizado anteriormente no *Windows Phone* 7. O SO e o *kernel* foram adaptadas do *Windows* com algumas modificações especificas para o *Windows Phone*. Esta adaptação permitiu a simplificação do processo de portabilidade de aplicações entre os computadores de mesa e os dispositivos móveis. O *Windows Phone* 8 já utiliza o ***Core System*** do *Windows*, que utiliza os recursos do sistema, nomeadamente o arranque, o gestor de *hardware* e os respetivos recursos. O ***Mobile Core*** é um conjunto de binários específicos do *Windows Phone* que é utilizado para as tarefas específicas do telefone. O ***Mobile Core*** complementa o ***Core System*** do *Windows Phone*.

A camada acima do *kernel* estão os ***Shared Services*** do sistema e as APIs que são utilizadas pelas aplicações para obter o acesso às funcionalidades subjacentes. Nesta camada estão presentes muitos dos recursos que são utilizados no *Windows* 8, tais como o sistema de ficheiros NTFS, o motor de gráficos *DirectX*, os elementos de segurança, entre outros recursos. O desenvolvimento de aplicações para *Windows Phone* é feito a partir do *Windows Phone* SDK e o *Driver Kit*, ambos oferecidos pela Microsoft.

Na camada superior estão as *frameworks* de desenvolvimento de aplicações para *Windows Phone*. Essas aplicações são implementadas com base nas linguagens de programação C#, *Visual Basic* (VB) .NET e C++ (Gronli et al., 2014).

d) BlackBerry OS

O SO *BlackBerry* e a sua plataforma de desenvolvimento foram desenvolvidas pela empresa Canadiana **Research-In-Motion (RIM)** (Hammershoj et al., 2010; Nosrati, Hojat, et al., 2012). O SO *BlackBerry* oferece uma plataforma de desenvolvimento de aplicações com suporte apenas para *Java 2 Platform Micro Edition* (J2ME). A máquina virtual Java (JVM) do SO *BlackBerry* é baseado na implementação dos J2ME da *Sun*, que é parcialmente escrita em C, C++ e *assembler.* Trata de uma implementação nativa pois encontra-se localizada no *firmware* do dispositivo, o que torna mais difícil a sua alteração. Apesar de ser difícil de alterar, este apresenta duas vantagens que são:

- O sistema operativo não precisa de ser compilado para o tipo de CPU do dispositivo;
- Ao mesmo tempo, oferece camadas de abstração de *hardware* para outras funcionalidades de *hardware* do dispositivo, como o botão de controlo de som, da comunicação radio, entre outros.

O SO *BlackBerry* vem integrado com o suporte *multitasking* com a integração de gestos (*gestures*) (Hammershoj et al., 2010).

A Figura 10 ilustra a arquitetura do SO *BlackBerry* 10, na qual pode-se ver os diferentes componentes que fazem parte da arquitetura do SO *BlackBerry*. A arquitetura do SO *BlackBerry* é constituída pelos seguintes componentes ***Application Runtimes***, ***Platform and Application Services***, ***Microkernel***, ***Bootloader*** e ***CPU Embedded bootloader***. Esses componentes trabalham juntos para garantir melhor a privacidade, a integridade e a confidencialidade das aplicações e dos seus dados. De modo a ter uma fácil interpretação da arquitetura do SO *BlackBerry*, a Tabela 7 apresenta esses componentes da arquitetura, bem como uma descrição detalhada de cada um dos componentes da arquitetura do SO *BlackBerry*.

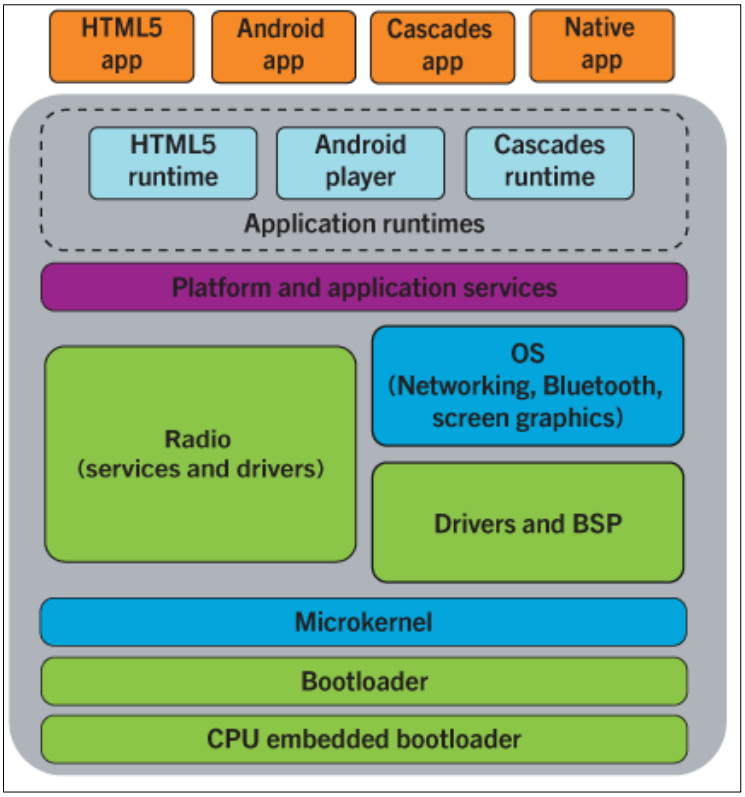


Figura 10 - Arquitetura do BlackBerry 10 OS retirado de (BlackBerry OS, 2016)

Tabela 7 - Descrição dos componentes da Arquitetura do BlackBerry OS retirado de (BlackBerry OS, 2016)

Componente	**Descrição**
App	São as aplicações nativas introduzidas pela organização ou as instaladas pelo utilizador.
Application Runtimes	Estes componentes incluem as máquinas virtuais, as bibliotecas, os serviços, o mapeamento das camadas, entre outros. Todas as aplicações são executas em caixas de proteção (*sandbox*) isoladas.
Bootloader	Este componente verifica a assinatura digital do sistema operativo antes de ser executado.
CPU Embedded Bootloader	Este componente verifica a assinatura digital do código *bootloader* antes de ser executado
Drivers and BSP	Estes componentes incluem os *drivers* e a lógica *board ring* para o suporte do *hardware* do dispositivo.

Componente	Descrição
Microkernel	Este componente indica qual é a quantidade mínima de *software* que o sistema operativo requer para ser executado
Platform and Application Services	Estes componentes incluem a gestão da segurança, gestão da instalação de *software*, os serviços de segundo plano para as aplicações, os serviços media, entre outros serviços. Estes são necessários porque as aplicações não executam serviços em segundo plano ou então para obter acesso aos componentes e serviços protegidos.
OS	São os processos do SO que existem fora do *kernel*.
Radio	Este componente inclui os *drivers*, pilhas (*stack*) e os serviços necessários para dar suporte dos subsistemas rádio para voz, dados e outros serviços.

3.3. Abordagem *Bring Your Own Device*

A crescente evolução no desenvolvimento e adoção da iniciativa de tecnologias de informação e comunicação internacionalmente fez evoluir também as tendências do *Bring Your Own Device* (BYOD) que está a mudar de forma rápida os métodos operacionais das organizações com o intuito de obter maior eficiência e produtividade (Bello Garba et al., 2015).

Bring Your Own Device (BYOD) é um esquema que foi adotado pelas organizações, para permitir que os colaboradores levam e utilizam os seus próprios dispositivos móveis na realização das suas atividades organizacionais. Isto significa que os dispositivos móveis BYOD não só têm os dados pessoais dos colaboradores, mas também dados da organização à qual pertence o colaborador (Kestle & Self, 2013).

Esta evolução trouxe novas oportunidades para os colaboradores, uma vez que os permitiu trazer os seus próprios dispositivos para o local do trabalho e integrá-los na rede da organização, em vez de utilizar os dispositivos da organização. Os benefícios do BYOD são claros, à medida que os colaborados das organizações estão mais familiarizados e satisfeitos com a utilização dos seus

próprios dispositivos na organização, e ao mesmo tempo as organizações conseguem economizar recursos, uma vez que não tem gostos com os dispositivos caros e planos de dados. Os objetivos das organizações com a adoção do BYOD passam pelo aumento da flexibilidade, da convivência e portabilidade dos dispositivos móveis para atender aos fluxos de trabalho dos seus colaboradores, e aumentar assim a sua produtividade (Dhingra, 2016).

No entanto, para que as organizações possam beneficiar inteiramente com o sucesso do BYOD, existem várias questões relacionadas à segurança e privacidade do ambiente BYOD que devem ser analisadas e compreendidas (Bello Garba et al., 2015).

As organizações estão a aumentar a taxa de utilização do BYOD, e este aumento está assente sobre a base dos seguintes fatores-chave: código de conduta dos colaboradores, a instalação de programas de segurança e as regras de gestão cada vez mais eficientes. Estes fatores são os responsáveis pelo desempenho geral do BYOD. Segundo o estudo realizado pela ***Tech Pro Research 2*** cerca de 74% das organizações entrevistadas afirmaram que estão a usar ou então estão a planear utilizar o BYOD (Dhingra, 2016). A Figura 11, ilustra a percentagem das organizações que utilizam o BYOD nas suas atividades bem como as que ainda não utilizam, mas que estão a desenvolver um plano para utilizar e as não utilizam e nem estão a desenvolver um plano de utilizar.

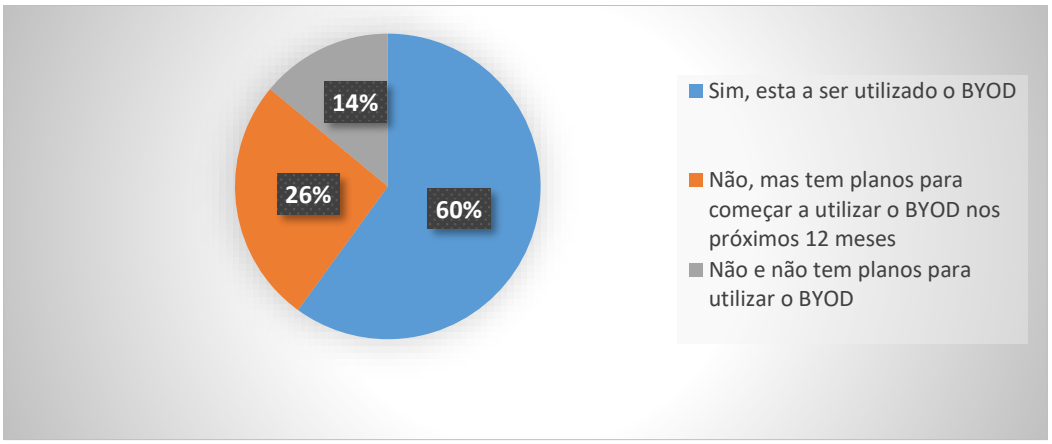


Figura 11 - Percentagem de utilização do BYOD nas Organizações adaptado de (Dhingra, 2016)

As organizações devem apresentar uma política eficiente e eficaz da utilização do BYOD e ainda devem indicar claramente todos os objetivos e as restrições quanto à utilização do BYOD nas organizações. Estas devem descrever todas as atividades que são permitidas nos dispositivos móveis quando eles estão a ser utilizados na rede da organização (Dhingra, 2016).

Muitas organizações ainda não têm uma arquitetura adequada de informações, de *hardware* e outros recursos do sistema para a utilização do BYOD. Já foram desenvolvidos vários modelos para oferecer as ferramentas de *software* e estratégias de segurança adequadas com o intuito de ultrapassar os problemas do BYOD. Atualmente, os modelos de segurança mais utilizados para BYOD são: *Mobile Device Management* (MDM), *Mobile Application Management* (MAM), *Mobile Information Management* (MIM) (Dhingra, 2016).

3.3.1. Benefícios do *Bring Your Own Device*

Esta subseção apresenta alguns dos benefícios mais notáveis do BYOD, tanto do ponto de vista dos colaboradores como também da própria organização em si. Os benefícios identificados são (Bello Garba et al., 2015; Kestle & Self, 2013):

- **Acessibilidade –** BYOD permite o acesso e a recuperação dos recursos e materiais da organização *online*. Possibilita ainda que os colaboradores tenham o acesso aos recursos de rede disponibilizados pelas organizações e que trabalhem com esses recursos ao próprio ritmo. BYOD abriu novas janelas de oportunidades para as organizações e os seus colaboradores, permitindo-lhes desenvolver as suas atividades a qualquer hora e em qualquer lugar. Com a utilização do BYOD, o conhecimento e a informação estão disponíveis para os colaboradores 24 horas por dia. A disponibilidade de redes e tecnologias sem fio reduziram as restrições físicas ou geográficas, o que permite que os colaboradores permaneçam produtivos mesmo enquanto estão fora das instalações físicas da organização.

- **Convivência e Flexibilidade** - BYOD oferece uma elevada flexibilidade no trabalho, bem como na convivência dos colaboradores da organização. Os colaboradores podem trabalhar em locais remotos, bem como podem fazer suas pausas durante o trabalho para a própria convivência. Alguns ambientes de trabalho não exigem a utilização de

tecnologias sofisticadas, pois o simples acesso a partir de um *smartphone* a uma rede (Internet) é suficiente.

- **Satisfação dos Colaboradores** – BYOD leva também a uma maior satisfação dos colaboradores nas organizações, pois os colaboradores têm a total liberdade de utilizar os dispositivos móveis que eles escolheram e investiram, ao invés do que aquele que foi escolhido pela organização. BYOD permite que os colaboradores utilizem os seus dispositivos pessoais e veio a evitar o transporte de vários dispositivos. Esta abordagem criou mais convivência para que os colaboradores desenvolvam as suas atividades e, ao mesmo tempo ter a flexibilidade para gerir os seus assuntos pessoais.

- **Aumento da Produtividade e Inovação –** Uma vez que há uma perspetiva de que os colaboradores estão cada vez mais satisfeitos e confortáveis com a utilização dos seus dispositivos móveis pessoais, eles rapidamente se tornam especialistas em utilizá-los, o que implica que sejam mais produtivos no desenvolvimento das suas atividades. As organizações acabam por beneficiar das tecnologias recentes, uma vez que a maioria dos colaboradores atualizam frequentemente para o *software* e *hardware* mais recente.

- **Redução de Custos** – BYOD pode reduzir os custos de viagem, espaço de escritório e instalações de trabalho e materiais impressos. Também pode reduzir as despesas organizacionais, transferindo os custos dos serviços de dados e de dispositivos para os colaboradores.

3.3.2. Desafios e Problemas do *Bring Your Own Device*

Tal como os outros dispositivos apresentam diversos desafios a nível da segurança, BYOD também apresenta desafios a nível de segurança semelhantes a outros dispositivos. Como apresentado anteriormente pode-se constatar que BYOD também levantam alguns problemas de segurança e desafios, devido à sua dupla finalidade (utilização pessoal e profissional) (Wang, Wei, & Vangury, 2014).

A prática do BYOD está a tornar-se numa "regra" versus "exceção" nas organizações devido aos benefícios que trazem para a mesma. No entanto, embora a evidência satisfatória é ampla que mostra que BYOD pode ajudar a agregar valor e melhorar os processos de negócio das organizações, não é sem custo. O esforço de investimento e implementação de uma nova infraestrutura segura para suportar e gerir a utilização de diferentes dispositivos móveis com diferentes SO pode ser muito desafiante e muito caro. Por outro lado, há que ter muita atenção aos perigos e ameaças inerentes à segurança e à privacidade que podem afetar os recursos da rede da organização e do utilizador do dispositivo móvel (Bello Garba et al., 2015).

O principal problema do BYOD é que não existe garantia de que os riscos de segurança e privacidade podem ser eliminados. Sempre que um dispositivo móvel aceda a um recurso da rede, os vestígios de informações e dados pessoais podem ser facilmente recolhidos. Os dispositivos móveis podem ser perdidos com facilidade ou roubados devido às suas pequenas dimensões, o que pode levar à exposição das informações contidas nos mesmos. A utilização dos dispositivos móveis não controlados, bem como a utilização dos mesmos sem restrições pode causar sérios problemas para a organização (pode infetar a rede da organização com vírus e *malware*) (Bello Garba et al., 2015; Kestle & Self, 2013).

Os desafios e problemas do BYOD apresentados a seguir, foram identificados por vários autores (Bello Garba et al., 2015; Dhingra, 2016; Kestle & Self, 2013; Wang et al., 2014):

- **Ameaças e Ataques** – BYOD são dispositivos móveis, pelo que são vulneráveis a várias ameaças e ataques como todos os outros dispositivos. Essas ameaças e ataques são normalmente realizadas por meio de *malware* que são disfarçados como aplicações móveis normais, tais como jogos, um *patch* de segurança ou então outros tipos de aplicações desejadas pelo utilizador, que são baixados para o dispositivo móvel. O *malware* móvel encontra-se dividido em três categorias: vírus, cavalo de troia e *spyware*, sendo os dois últimos os mais predominantes nos dispositivos móveis.

- **Vulnerabilidades do BYOD** – BYOD integram muitas funções tais como e-mails, calendários, notas, dados pessoais do utilizador e informações confidenciais das organizações. Além disso, BYOD é utilizado nos negócios para partilhar dados sensíveis da própria organização. As informações contidas no BYOD podem incluir, mas não são

limitados a informações pessoais (como endereço da residência, contacto telefónico, fotos, lista de contacto, entre outros), correspondência de informações comerciais (e-mails, mensagens de texto, mensagens MMS, registo de chamadas, entre outros), informação de cartão de crédito e credenciais secretas (nome do utilizador e a palavra passe), ficheiros de memória ou no cartão de memória, documentos organizacionais (documentos de texto e planilhas) e localização geográfica. O centro de gestão de dados do BYOD é muito atraente para os *hackers.* As vulnerabilidades do BYOD ocorrem devido à falta de **confidencialidade**, **isolamento** e **conformidade** no próprio BYOD.

Confidencialidade – BYOD utiliza dados sensíveis das organizações, pelo que não devem ser armazenados em BYOD com formato de texto simples. Nesse aspeto torna-se essencial proteger a confidencialidade dos dados da organização em um BYOD. Para além disso, não é apenas essencial proteger a confidencialidade dos dados da organização, mas também é importante para monitorizar e rejeitar o acesso não autorizado e ilegal aos dados da organização. O acesso não autorizado provém dos colaboradores internos (colaboradores ou ex-colaboradores) quando eles não devem aceder aos dados da organização, enquanto que, o acesso ilegal provém de terceiros quando eles pretendem recuperar dados da organização armazenados em um BYOD (como por exemplo, utilizadores mal-intencionados que tentam roubar dados de um BYOD perdido).

Isolamento – BYOD são adotadas para a utilização pessoal e profissional, ambos com diferentes requisitos de segurança. Quando um BYOD é utilizado para uso pessoal, o proprietário do mesmo procura a flexibilidade e a convivência de decidir quais as aplicações que podem ser baixados e quais podem ser instalados. Por outro lado, quando é utilizado em serviço da organização, é essencial que as organizações garantam que os dispositivos móveis estejam em conformidade com as políticas de segurança da organização. O isolamento do espaço é uma característica desejada e torna possível aplicar políticas de segurança diferentes no espaço pessoal e organizacional de um BYOD.

Conformidade – BYOD são extensões das organizações, mas, no entanto, é difícil de garantir que os estes cumpram com as políticas de segurança da organização. Apesar de existirem já algumas soluções BYOD, todos eles têm limitações ao aplicar políticas de

segurança em um BYOD. Alem disso, BYOD são dispositivos móveis pessoais que são adotados para a utilização profissional nas organizações e é impraticável auditar manualmente o dispositivo móvel pessoal de um colaborador devido à propriedade e também à grande quantidade de dispositivos. As opções automáticas alternativas devem ser exploradas para aplicar as políticas de segurança das organizações em um BYOD.

- **Descoberta do BYOD –** uma vez que o BYOD faz parte da própria organização, torna-se essencial efetuar a monitorização e o rastreamento do mesmo. Essa descoberta do BYOD pode ser dividida em duas categorias: o **sistema de descoberta BYOD baseado em agentes** e o **sistema de descoberta BYOD à base de digitalização**.

  O **sistema de descoberta BYOD baseado em agentes** requer um agente (uma aplicação para dispositivo móvel) instalada em um BYOD. Este será o responsável para relatar o estado do dispositivo móvel para o sistema central de gestão da rede, como por exemplo o *Mobile Device Management* (MDM). De igual modo, o agente acaba por ser o representante dos administradores do sistema ao estabelecer certas políticas de segurança em um BYOD. O sistema de descoberta BYOD baseado em agentes é fácil de ser utilizado e não causa muita sobrecarga de comunicação na rede da organização, mas, no entanto, este requer que seja instalado uma aplicação em um BYOD primeiro.

  O **sistema de descoberta BYOD baseado em digitalização** não requer a instalação de qualquer tipo de aplicação em um BYOD. Neste caso, as ferramentas de digitalização das redes como o *nmap* (*Network Mapper*) são utilizadas para detetar o BYOD com base na sua impressão digital. Como o BYOD está normalmente ligado a rede da organização através do Wi-Fi, torna-se assim possível realizar a sua digitalização. Por outro lado, a digitalização pode ser baseada em interfaces Bluetooth no BYOD. Esta abordagem apresenta uma limitação, pois só funciona em pequenas áreas e é difícil de dimensionar para grandes áreas de trabalho. A digitalização de uma rede pode levar muito tempo a ser implementado, e por outro lado adiciona tráfego extra a rede da organização.

3.3.3. Soluções do *Bring Your Own Device*

Atualmente existem algumas soluções BYOD implementadas que visam ultrapassar os desafios e os problemas enfrentado pelo BYOD. Tais soluções são:

- ***Virtual Private Networks*** **(VPN)** – numa rede organizacional típica, existe uma zona desmilitarizada (DMZ - servidor que serve de zona neutra e evita a entrada de utilizadores da rede pública à rede interna da organização) normalmente utilizada para proteger a rede interna da organização. As redes Wi-Fi, redes móveis tais como 3G e 4G são necessárias para utilizar a VPN para conectar à rede interna da organização através de uma rede pública. Mas, no entanto, utilizar o DMZ ou a VPN não ajuda a proteger o BYOD, pois os proprietários do BYOD são colaboradores e eles não têm problemas para ligar os seus dispositivos móveis à internet. Por outro lado, VPN garante uma conexão segura entre BYOD e a rede da organização, mas não consegue proteger os dados armazenados em BYOD (Kestle & Self, 2013; Wang et al., 2014).

- ***Mobile Device Management*** **(MDM) -** é uma solução para proteger o BYOD, com suporte ao controlo total do dispositivo móvel baseado em *software* completos que podem ser utilizados pelas organizações para bloquear, controlar, criptografar e aplicar políticas de segurança nos dispositivos móveis. Atualmente, no mercado existem algumas soluções MDM tais como: *FiberLink Maas360*, *Zenprise MobileManager*, *AirWatch MDM* e *MobileIron* (Dhingra, 2016; Scarfo, 2012; TechTarget, 2016; Wang et al., 2014).

- ***Mobile Application Management*** **(MAM) -** é uma solução para proteger BYOD semelhante ao MDM, mas em vez de ter o controlo total do dispositivo móvel, esta solução é aplicada apenas para as aplicações específicas num dispositivo. Esta solução possibilita bloquear, controlar e proteger apenas as aplicações da organização, enquanto que as outras aplicações são da responsabilidade do utilizador do BYOD (Dhingra, 2016; Scarfo, 2012; TechTarget, 2016).

- ***Mobile Information Management*** **(MIM) -** é uma solução BYOD que pode ser descrita como todos os serviços baseados na cloud como por exemplo o Dropbox, que permite a sincronização de ficheiros e documentos em diferentes dispositivos (Dhingra, 2016; Scarfo, 2012; TechTarget, 2016).

- ***Mobile Virtual Machines* (MVM) –** as máquinas virtuais têm sido muito utilizadas em ambientes *desktop* e na *Cloud Computing*, pois elas fornecem formas eficazes de fazer a separação entre o espaço e os dados. Da mesma forma, como os dispositivos móveis estão a tornar-se cada vez mais poderosos, o desenvolvimento de máquinas virtuais para os dispositivos móveis, estão a tornar-se cada vez mais viáveis. Estas podem ser utilizadas para separar o espaço pessoal e o espaço organizacional em um BYOD. O ***VMware Horizon Mobile*** é uma abordagem do MVM (Kestle & Self, 2013; Wang et al., 2014).

- ***Cisco BYOD Smart Solutions* -** é uma solução BYOD que fornece uma infraestrutura completa, incluindo pontos de acesso, controladores, gestores segurança e gestão de rede para BYOD. Esta solução é possível aplicar controlos de acesso atribuindo diferentes tipos de permissões (tais como acesso total, acesso parcial, apenas internet, acesso negado) que podem ser aplicadas em um BYOD. Mas no entanto, a solução não fornece suporte para os clientes BYOD, tal como o isolamento de espaço e a proteção dos dados, pelo que esta solução deve ser integrada com outras soluções apresentadas (como por exemplo MDM, MAM, MIM, entre outras) para proporcionar os recursos de segurança desejado nos BYODs (Bello Garba et al., 2015; Kestle & Self, 2013; Yong Wang et al., 2014).

3.3.4. BYOD no *Healthcare*

Segundo o estudo realizado por Moyer (Moyer, 2013), ele concluiu que atualmente, existe um número significativo de hospitais que já adotaram a abordagem BYOD nas suas políticas. O estudo ainda revela que cerca de 85% dos 130 hospitais que foram pesquisados já adotaram e apoiam a utilização do BYOD na realização das suas atividades. Contudo a tendência é de que mais unidades hospitalares adirem para a utilização dos BYOD nas suas atividades.

As políticas BYOD nos hospitais já apresentam benefícios claros, nomeadamente a redução de custos a curto prazo nas unidades hospitalares. Mas, no entanto, custos associados aos potenciais riscos do BYOD, tais como falhas de segurança, roubo e perda de dados hospitalares, aparentam superar significativamente os potenciais benefícios aos profissionais do *healthcare* que utilizam os seus próprios dispositivos móveis nos ambientes ***Health Insurance Portability and Accountability Act*** (**HIPAA**) (Moyer, 2013).

Segundo artigo apresentado por Lund & Dunbrack, quando se trata de BYOD no *healthcare*, normalmente estas apresentam grandes desafios em quatros áreas distintas. As áreas identificadas pelos autores são (Lund & Dunbrack, 2015):

- O ambiente de regulamentação da saúde (regulamentos federais e estatuais, tais como **HIPAA** e as preocupações de privacidade);
- A segurança do dispositivo móvel e a segurança do acesso aos dados clínicos e aplicações clínicas, incluindo os desafios únicos que os clínicos têm em ambientes restritos e estéril;
- Atualizações, alterações das políticas e problemas de suporte, tais como problemas de *Mobile Device Management* (MDM) que podem surgir com os clínicos que utilizam os dispositivos móveis para aceder a rede em vários hospitais onde desenvolvem as suas atividades;
- As implicações financeiras da utilização da voz e dos dados, incluindo as implicações financeiras para o porte de número de telefone móvel fornecido pela organização para o dispositivo móvel pessoal de um clínico.

a) Ambiente de Regulamentação da Saúde

A área da saúde (medicina) é conhecida pela sua elevada complexidade e alto nível de exigência. Com a introdução do BYOD no *healthcare*, agora a administração das unidades hospitalares tem de lidar com o cruzamento dos dados dos dispositivos móveis com a privacidade e os requisitos da regulamentação. O **HIPAA** é a entidade que faz a regularização do acesso e a divulgação dos dados clínicos, mas existem outras legislações que regulam a proteção dos dados das organizações. Os profissionais de saúde que utilizam BYOD no desenvolvimento das suas atividades, colocam grande foco sobre esses regulamentos, independentemente de a organização ter ou não uma política de BYOD implementada. A maior preocupação neste aspeto são as regularizações que lidam com os dados clínicos, pelo fato de estes apresentarem consequências negativas se tais dados forem comprometidos (Lund & Dunbrack, 2015).

b) Segurança do dispositivo, dos conteúdos organizacionais e do acesso às aplicações organizacionais

A tendência do BYOD, trouxe novas implicações a nível da segurança, tal como o *malware* a que os dispositivos móveis estão sujeitos que podem comprometer os dados dos clínicos quando estes são conectados à rede da organização. Por outro lado, existem riscos associados à perda ou roubo desses dispositivos, pois vários incidentes de *hacking* em grande escala ocorridos em 2015, resultou da perda ou roubo dos dispositivos móveis. No contexto do *healthcare*, essas perdas colocam os dados dos utentes em risco por meio da divulgação não autorizada dos mesmos, e a própria organização da saúde pode enfrentar a possibilidade real contestação e rígidas sanções (Lund & Dunbrack, 2015).

Tanto nas organizações do *healthcare*, como em outras organizações, a utilização dos dispositivos móveis por parte dos colaboradores estimulam a necessidade de estabelecer e gerir novas políticas de segurança na utilização desses dispositivos que não são propriedades da organização e não são geridos diretamente pelo departamento de Tecnologias e Informação (TI) da organização. Essas preocupações intensificaram-se com a entrada do decreto estabelecido pelo **HIPAA Omnibus Rule**. Este decreto tem novos requisitos mais rigorosos sobre as notificação de violação da privacidade e da utilização mínima e divulgação de informações (Lund & Dunbrack, 2015).

As políticas de segurança devem considerar o ambiente complexo que é o *healthcare*. A tendência do BYOD levantou a necessidade para que a administração das organizações do *healthcare* criem um processo de avaliação da própria rede da organização, à infraestrutura subjacente para assegurar a proteção dos dados, das aplicações e dos dispositivos móveis que são utilizados dentro da organização. Assim, as organizações devem estabelecer políticas e procedimentos de segurança móveis para garantir a conformidade da utilização dos dispositivos móveis entre todos os colaboradores da área da saúde (Lund & Dunbrack, 2015).

c) Atualizações e alterações de políticas

Em muitas organizações, a utilização dos dispositivos móveis no local do trabalho deixou de ser uma ferramenta de comunicação bilateral para uma ferramenta que utiliza as bases do conhecimento da própria organização, tais como as aplicações e outros dados da organização. As organizações do *healthcare* são obrigadas a atualizar as tecnologias e os processos sobre as informações eletrônicas e os pontos de acesso. Estas atualizações são uma aposta significativa quando se trata da privacidade dos utentes, por isso a administração das organizações do *healthcare* deve avaliar o ambiente da organização e os casos da utilização do BYOD para analisar os possíveis riscos com novas atualizações. Existe uma mentalidade dentro da área TI "vamos economizar mais com a estratégia do BYOD", mas isso nem sempre é o que acontece se a estratégia não for bem estudada e implementada corretamente (Lund & Dunbrack, 2015).

Na realidade, as organizações do *healthcare* devem determinar antecipadamente como irá ser oferecido o suporte aos colaboradores que utilizam o BYOD. Segundo o estudo realizado pela *International Data Corporation* (IDC), revela que na melhor das hipóteses, a maioria das organizações do *healthcare* ainda prestam suporte limitado para as aplicações da área. O impacto do tempo de inatividade entre os colaboradores pode causar danos nas prioridades da organização tanto a nível do atendimento aos utentes, o acesso eficiente e eficaz no registo de dados, pelo que a produtividade dos colaboradores pode ser afetada negativamente. O suporte para os dispositivos móveis utilizados pelos colaboradores é um requisito que as organizações do *healthcare* pretendem abraçar com êxito do BYOD (Lund & Dunbrack, 2015).

d) As implicações financeiras da utilização da voz e dados móveis

Os profissionais do *healthcare*, em particular os clínicos muitas vezes preferem utilizar os seus próprios dispositivos em vez de utilizar os dispositivos oferecidos pela organização. Os clínicos e a organização do *healthcare* estão vinculados para enfrentar este dilema de como manter o número clínico, enquanto alberca a utilização dos dispositivos pessoais. Muitas vezes, a organização do *healthcare* é confrontada com o incorrer de custos financeiros no processo de transição para os dispositivos. Mas, no entanto, a expectativa para a organização do *healthcare* é que pode existir no mínimo o aumento da utilização do BYOD pelos colaboradores de modo a compensar os custos financeiros (Lund & Dunbrack, 2015).

As soluções BYOD no *healthcare* não diferem muito das soluções BYOD apresentadas anteriormente. A grande diferença é que na área da saúde existe uma entidade reguladora **HIPPA** que regula e gere as normas sobre a utilização dos BYOD no *healthcare*. No ambiente de regulamentação da saúde, as soluções são implementadas ao nível da rede, dos dispositivos, dos utilizadores de acordo com as normas do **HIPAA**, para ajudar a organização a minimizar o nível de infrações que podem ocorrer com a utilização do BYOD (Lund & Dunbrack, 2015). As soluções *Mobile Device Management* (MDM) e *Mobile Application Management* (MAM) são os pilares essenciais no ambiente holístico BYOD para *healthcare* e outras organizações é claro. Geralmente essas soluções permitem que as organizações tenham maior controlo sobre os dispositivos móveis e as informações da organização, mesmo independentemente se os dispositivos pertencerem à organização ou ao colaborador da organização. As soluções do MDM e MIM concentram desde a gestão e segurança dos dispositivos móveis e as informações pessoais nesses dispositivos. Com essas soluções, as organizações do *healthcare* tornam-se mais eficientes na gestão e segurança dos dados clínicos dos utentes e dos colaboradores da organização, pois têm maior controlo, gestão e segurança dos dados e dos dispositivos BYOD (Lund & Dunbrack, 2015).

3.3.5. Análise SWOT do BYOD

Esta subsecção apresenta a análise SWOT, identificando as forças (Strength), as fraquezas (Weakness), as oportunidades (Opportunities) e as ameaças (Threats) que estão subjacentes à utilização do BYOD nas organizações. Esta análise tem como principal finalidade identificar os principais pontos fortes e as oportunidades que a abordagem BYOD tem nas organizações, mas também, pretende-se identificar os principais pontos fracos e as ameaças inerentes à utilização do BYOD nas organizações. A Tabela 8 apresenta de forma sucinta essas características do BYOD nas organizações.

Tabela 8 - Análise SWOT do BYOD adaptado de (Nitish Kirtiraj Shah, sem data)

Strength	**Weakness**
Possibilidade de expansão; Aumento da produtividade; Aumento da satisfação moral dos colaboradores; Flexibilidade da cultura e do trabalho; Redução do custo de TI das organizações; Infraestrutura de TI simplificada; Colaboradores convivem mais ao utilizar os seus próprios dispositivos; Traz melhor ferramenta de negócios eficiente.	Questões legais, éticas e as leis sobre a privacidade; Problemas de compatibilidade; Falhas de segurança; Conectar e gerir e diferentes tipos de aplicações(s) /conexão de rede / sistema operativo com a rede organizacional é complexa.
Opportunities	**Threat**
Capacidade de adaptação de acordo com as exigências futuras; Melhorar a comunicação e a colaboração dos funcionários; Crescimento no *middleware* e tecnologia de gestão de dispositivos móveis; Crescimento no desenvolvimento de *software* para dispositivos BYOD; Grande demanda na tecnologia de desenvolvimento de *software* de gestão de memória, antivírus e *anti spyware*; Mais inteligência e abertura de negócios para novas ideias; Grande demanda por ferramenta de colaboração unificada para gerir diferentes tipos de aplicações / conexão de rede / sistema operativo com sistema (s) organizacional, bem como dispositivo(s) do (s) colaborador(es).	Privacidade e segurança (Ex., Confidencialidade, Integridade e Disponibilidade); Regulamentos governamentais com impactos na utilização de dispositivos móveis nas organizações; Roubo e extravio de dados ou dispositivos; A atitude habitual afeta a produtividade; O equilíbrio da vida profissional pode ser afetado; Violação de segurança em dados da organização.

Atualmente, o ambiente organizacional e a concorrência econômica estão se tornando globais, os clientes e a utilização de tecnologias estão a mudar rapidamente e essas mudanças implicam que as organizações encontrem formas de construir planos estratégicos adequados e decisões corretas para tornar os negócios eficazes e influenciar a concorrência. De acordo com a análise SWOT apresentada na Tabela 8, é possível ver quais são os fatores que mais condicionam as políticas BYOD nas organizações, ou os fatores que são chaves para a sucesso da organização que decidam adotar esta política.

3.4. Dispositivos Móveis no *Healthcare* (caso prático)

Esta secção apresenta dois casos práticos de plataformas que possibilitam a utilização dos dispositivos móveis nas unidades hospitalares. Primeiro será apresentado uma plataforma que se encontra implementado em alguns hospitais na Polónia. Depois será apresentado uma outra plataforma, que existe em Portugal e no qual a aplicação irá funcionar futuramente no CHP.

3.4.1. *Medical Mobile Data Collecting Platform* (MMDCP)

A medicina já é e ainda continua a ser um domínio cada vez mais complexo. Desde o processo de registo de atendimento de um utente até à prestação do tratamento adequado tornou-se cada vez mais complexo e mais sofisticado, e quaisquer erros ou imprecisões podem levar a consequências significativas bem como para o utente como também para os profissionais do *healthcare* (Brzeziński, Kobusińska, Kobusiński, Stroiński, & Szałkowski, 2013).

O ***Medical Mobile Data Collecting Platform*** (**MMDCP**) é uma plataforma que visa reduzir a taxa de erro e acelerar o processo de recolha dos dados dos utentes. É uma plataforma de utilização pessoal dos profissionais do *healthcare* que devem ter o acesso aos dados clínicos a partir de qualquer local do estabelecimento do *healthcare*. Esta solução distingue-se pela mobilidade que traz aos profissionais do *healthcare*, pela sua flexibilidade, simplicidade na integração e manutenção (Brzeziński et al., 2013).

A plataforma proposta para os profissionais do *healthcare* MMDCP é baseada no *Service Oriented Architecture* (SOA) (Brzeziński et al., 2013). De acordo com a arquitetura SOA, as funcionalidades

do sistema são distribuídas pelas aplicações independentes chamados *Web Services*. Os *Web Services* aqui constituem o processo de negócio, principalmente nas funcionalidades mais complexas e sofisticadas. O resultado da arquitetura baseado em SOA do MMDCP, as funcionalidades da plataforma proposta para os profissionais do *healthcare* podem ser facilmente expandidas sem alterar as funcionalidades já oferecidas, o que é importante no caso dos sistemas médicos e não só. Além disso, devido à arquitetura orientada a serviços do MMDCP, é possível integrar de forma fácil e exequível a plataforma proposta com vários sistemas *Hospital Information System* (HIS) (Brzeziński et al., 2013). A Figura 12, ilustra a arquitetura da plataforma MMDCP proposta pelos autores (Brzeziński et al., 2013). Conforme pode se constatar na figura abaixo apresentada, a plataforma é composta pelas seguintes partes: utilizador (*user*), motor de trabalho ROsWeL (ROsWeL *workflow engine*), serviços médicos (*Medical Services*), adaptadores dos serviços (Services adapters), serviço de *buffer* (*buffer services*), adaptador (*adapter*) e base de dados HIS (*HIS database*).

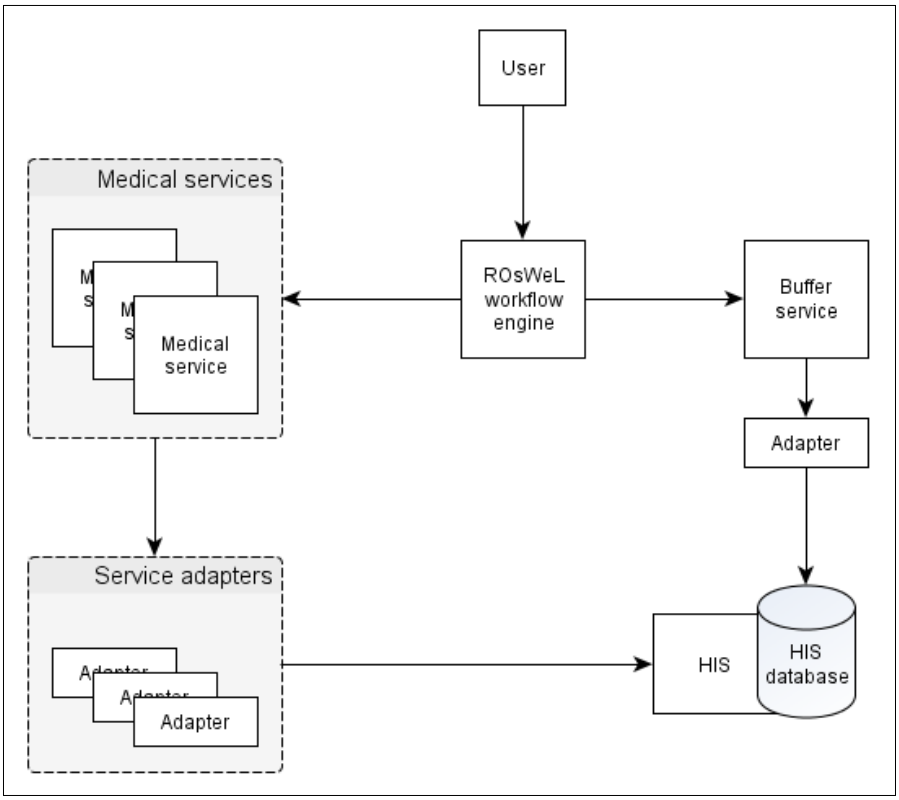


Figura 12 - Arquitetura do Mobile Medical Business Process Platform retirado de (Brzeziński et al., 2013)

Na arquitetura da plataforma presente na Figura 12, é de realçar a importância do **ROsWeL** (ambiente de execução dos processos de negócios médicos) que é o responsável pelo processamento da lógica e da comunicação entre os componentes do sistema e os dispositivos móveis dos utilizadores finais, que inclui as informações de como os dados são apresentados, quais e como as informações devem ser introduzidas sistema (Brzeziński et al., 2013).

A plataforma MMDCP proposta, ainda é constituída por um outro componente, um dispositivo móvel. A solução proposta, o dispositivo móvel com aplicação móvel tem como principal objetivo a inicialização da execução dos processos de negócios. A característica mais importante da referida aplicação é a sua capacidade de gerar a interface do utilizador com base nos dados conduzidos pelo mecanismo **ROsWeL** durante a execução do processo de negócios. Esta interface permite que os profissionais do *healthcare* introduzem os dados através de um teclado, camera ou um leitor de laser dedicado integrado com o dispositivo ou conectado com um dispositivo periférico. No estado atual de desenvolvimento do MMDCP, o utilizador pode introduzir vários tipos de dados tais como: strings, imagens e códigos de barras (Brzeziński et al., 2013).

Nesta sequência, a aplicação móvel foi batizada com o nome de "***Thin Client***", o que significa que até mesmo um dispositivo móvel muito simples e barato pode desempenhar o papel proposto aos dispositivos móveis na plataforma. O ***Thin Client*** está disponível para as plataformas móveis *Android* e *Windows Phone* e também para PC, sendo que este último utiliza um *browser* web para executar. É importante realçar que independentemente do dispositivo móvel e da sua qualidade, o design da interface gráfica da aplicação proposta é praticamente o mesmo (Brzeziński et al., 2013). A Figura 13, ilustra a arquitetura apresentada para o ***Thin Client.***

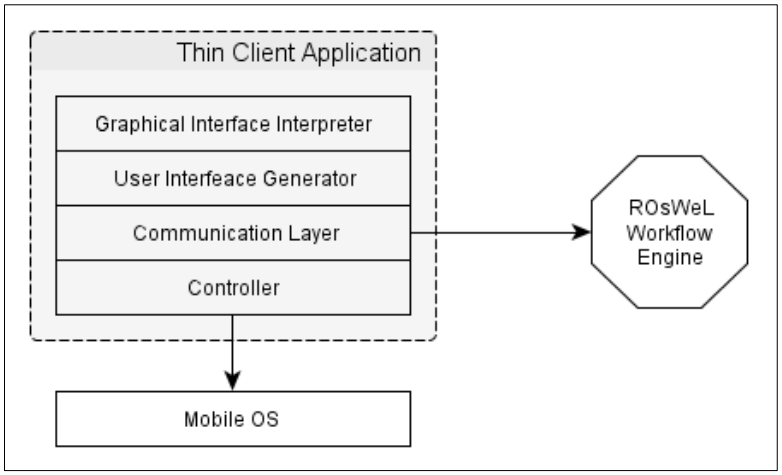


Figura 13 - Arquitetura do Thin Client Application retirado de (Brzeziński et al., 2013)

A aplicação proposta para a plataforma tem uma arquitetura modular constituída pelos seguintes módulos: ***Controller, Communication Layer, User Interface Generator*** e ***Graphical Interface Interpreter***.

O módulo do Controlador (*Controller*) é utilizado para invocar as funcionalidades nativas dos diferentes dispositivos móveis. Este é utilizado para definir ou então obter as configurações do dispositivo móveis tais como o idioma, som, conexão e utilizá-los durante a execução do processo de negócio.

O módulo acima do *Controller***,** é o módulo do Camada de Comunicação (*Communication Layer*), cujo principal objetivo é enviar e recuperar as mensagens entre o mecanismo **ROsWeL** e a aplicação "***Thin Client***". O módulo acima do *Communication Layer*, é o módulo Gerador da interface do utilizador (*User Interface Generator**)***, cuja função é recuperar as mensagens recebidas do *Communication Layer* para gerar uma interface ideal para o dispositivo (por exemplo resolução do ecrã). Os dados da interface são descritos com recurso a um pequeno subconjunto de tags HTML5, aprimorados, mas com a ausência de algumas funcionalidades para facilitar o processo de exibição da interface do utilizador nos dispositivos mais fracos. A parte da interface do utilizador em falta, é adicionada à interface mais tarde consoante o pedido do utilizador. Por último, tem-se o módulo da aplicação móvel, que é o Interpretador Gráfico do utilizador (*Graphical Interface Interpreter*) que simplesmente desenha a interface gerada no ecrã do dispositivo móvel (Brzeziński et al., 2013).

a) Integração do MMDCP com *Hospital Information Systems*

Atualmente existem muitos Sistemas de Informação Hospitalares (HIS) no mercado. Estes sistemas são obrigados a atender às demandas do rápido acesso aos dados, alta personalização e elevados níveis de segurança. Esses requisitos resultam numa arquitetura de sistema com soluções de *software* personalizado. Muitas vezes, a integração desses sistemas acaba por ser uma tarefa muito difícil e complicada. A plataforma MMDCP fornece um mecanismo para fazer a integração com todos os sistemas HIS existentes (Brzeziński et al., 2013).

Com o objetivo de atingir a máxima interoperabilidade possível, o MMDCP não se comunica com o HIS diretamente. No caso da leitura dos dados clínicos, o **ROsWeL** invoca serviços clínicos (recursos), que são as interfaces da rede para os dados armazenados no HIS. Para além disso, para permitir a fácil integração com os vários HIS, os serviços clínicos utilizam os chamados adaptadores para ajustar o formato de dados interno do HIS (por exemplo base de dados) para aquela que é compreendida pela plataforma MMDCP. Esta abordagem permite-se assim controlar os dados que serão armazenados na base de dados do HIS e fornecendo assim um alto nível de interoperabilidade (Brzeziński et al., 2013).

A solução proposta consiste na utilização de dois *Web Services* Java, utilizando a tecnologia *Jersey* e o *Hibernate* (*framework* para o mapeamento do objeto relacional). O primeiro *Web Service* é o *Universal Identification Services* (UIS) cujo objetivo é exportar as informações da base de dados HIS para o mecanismo do processo de negócios da plataforma de modo a oferecer os objetos de dados que são necessários para os processos de negócios. Este serviço cria a interface REST para aceder aos objetos utilizando diferentes tipos de dados. O segundo *Web Service* é o *Universal Bus Service* (UBS) cujo responsabilidade é de reunir de forma universal as múltiplas informações recolhidas durante a execução dos processos de negócios (Brzeziński et al., 2013).

- ***Universal Identification Service* (UIS)**

A tarefa principal do *Universal Identification Service* (UIS) (Figura 14), é ir buscar as informações no HIS e apresenta-las no mecanismo do processo de negócios que utiliza os dados obtidos para executar as suas tarefas. O serviço proposto não interfere com a infraestrutura HIS existente, nem requer a criação de estruturas de dados redundantes. Os seus requisitos são muitos rigorosos e

simples, pois o sistema da base de dados deve cooperar com o *Hibernate*. O tipo de dados armazenados nas colunas da tabela deve ser os tipos básicos padrão do *Hibernate***.** Como resultado, o HIS o suporta também outros tipos de colunas com referências a outras tabelas, tipos de dados complexos e coleções (Brzeziński et al., 2013).

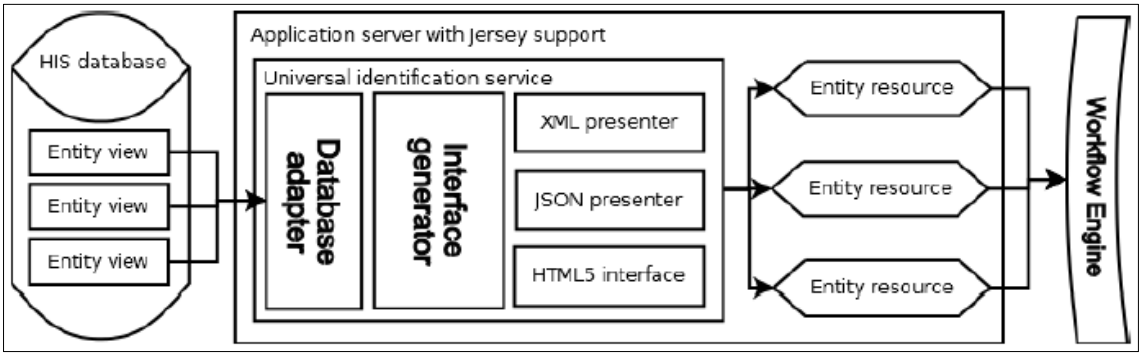


Figura 14 - Arquitetura do Universal Identification Service retirado de (Brzeziński et al., 2013)

- ***Universal Bus Service*** **(UBS)**

No *Universal Bus Service* (UBS), os processos de negócios são criados com a finalidade de executar determinado processo simultaneamente com o utilizador, e durante esse processo os dados são recolhidos. Estes dados geralmente contêm informações úteis para o HIS, e esses dados devem ser guardados de alguma forma. O UBS cuja arquitetura é apresentada na Figura 15 é a solução proposta para este problema, que é um serviço que atrasa o armazenamento de dados para o momento na qual o HIS estará pronto para aceitá-los ("*delayed write*"). A função deste serviço basicamente é recolher as meta-informações. Essas informações recolhidas podem ser utilizadas para otimizar os processos de negócios, facultar melhores níveis de segurança ou executar serviços de monitorização. Tais informações incluem o utilizador que executou o processo, o dispositivo móvel na qual foi utilizado para executar o processo, o processo que foi executado, o endereço do *host,* tempo de execução e o nome da aplicação (Brzeziński et al., 2013).

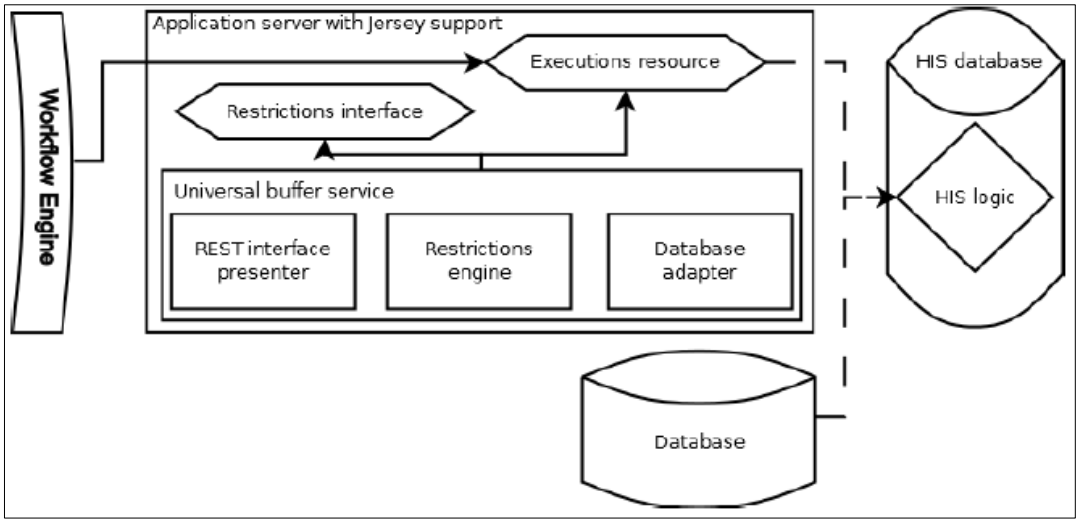


Figura 15 - Arquitetura do Universal Bus Service retirado de (Brzeziński et al., 2013)

O processo de configuração do UBS é composto por três etapas: encontrar e criar uma base de dados de acordo com o esquema oferecido, facultar os ficheiros de configuração da base de dados do *Hibernate* e adicionar restrições a nível de segurança sobre eventuais processos de negócios, os utilizadores ou os dispositivos móveis nos quais eles são executados (Brzeziński et al., 2013).

b) Processo de Recolha de dados do MMDCP

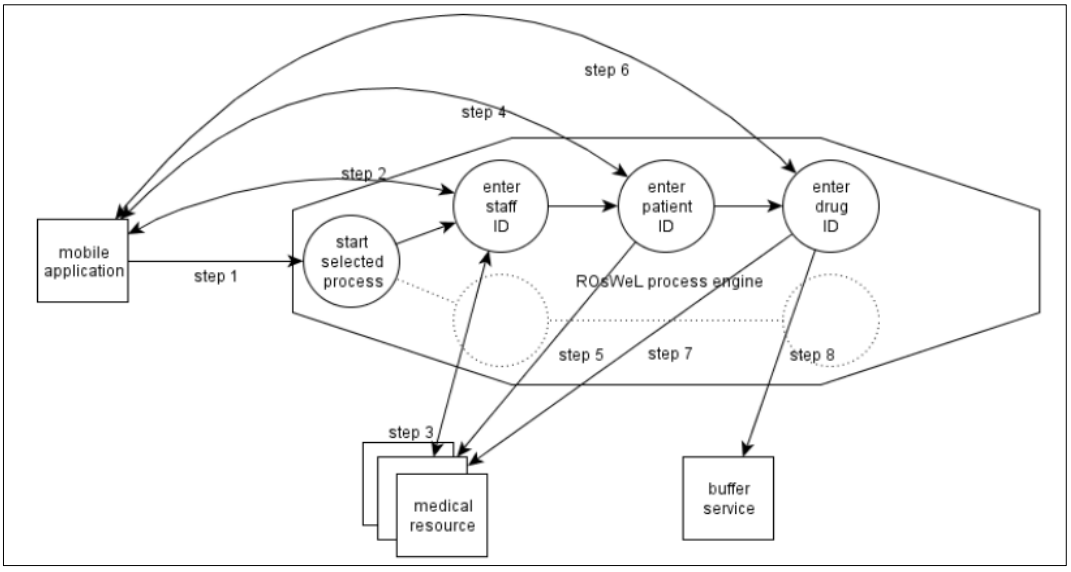


Figura 16 - Processo de Administração de Medicamento retirado de (Brzeziński et al., 2013)

O exemplo a acima apresentado (Figura 16), ilustra a utilização da plataforma MMDCP nas organizações do *healthcare*. No processo considerado na Figura 16, as informações sobre os fármacos administrados aos utentes durante o procedimento médico são recolhidas. Para começar, os profissionais da saúde utilizam uma aplicação móvel instalada em um dispositivo móvel que se conecta ao mecanismo **ROsWeL** e solicita a lista de processos disponíveis. Depois de selecionar o processo de administração do fármaco, o motor **ROsWeL** executa o primeiro passo do processo (Step 1), o qual requer a introdução do ID do colaborador da equipa médica para verificar a sua identidade (Step 2). O número de identificação pode ser introduzido manualmente (teclado) ou digitalizado a partir do código de barras existente na identificação do colaborador. De seguida, o mecanismo **ROsWeL** através de um recurso de identificação pessoal (recursos médicos disponíveis), faz uma consulta à base de dados HIS para a verificação (Step 3). Caso tal pessoa exista na base dados e tem autorização para administrar medicamentos, as informações serão enviadas de volta para o mecanismo **ROsWeL** para prosseguir, caso contrário o sistema apresenta um erro de autorização e será enviado de volta para o ***Thin Client***.

Já na etapa a seguir, deve ser introduzido a identificação do utente tal como acontece com a identificação do colaborador da equipa médica (Step 4). As informações do utente serão verificadas (Step 5) e enviadas pelo mecanismo **ROsWeL** ao cliente e é gerado a interface apropriada no dispositivo móvel do utilizador. Caso as etapas anteriores forem realizadas com sucesso, o ID do fármaco administrado pode ser agora introduzido no sistema (Step 6, Step 7 e Step 8). Para administrar o novo medicamento de forma rápida e facilmente, a plataforma MMDCP por meio do mecanismo **ROsWeL**, notifica a equipa médica e os utentes que não precisam de repetir as duas primeiras etapas do processo acima referido. Salvo em caso de mudança de utente, ou da equipa médica que participa no processo referido, é preciso introduzir novamente o ID do utente ou do colaborador da equipa médica. Finalmente com todos os *triplos* completados na forma de: **<staff_id, patient_id, drug_id>** são registados num *buffer* (Step 8) (Brzeziński et al., 2013).

A plataforma MMDCP oferece um ambiente flexível e de fácil integração, que suporta as atividades diárias dos profissionais do *healthcare* na recolha de dados clínicos. A solução proposta foi integrada com sucesso com ***Eskulap Hospital Information System***, sendo este o terceiro sistema hospitalar mais utilizado nos hospitais na Polónia (Brzeziński et al., 2013). Pelo fato de a

arquitetura e a tecnologia serem orientadas aos serviços, o processo de integração foi concluído sem grandes dificuldades e foi confirmado na prática os benefícios da plataforma. Com a implementação da solução MMDCP no **Eskulap HIS**, é possível definir novos processos de negócios, novas funcionalidades de modo enriquecer o sistema do **Eskulap HIS** e aumentando a utilização dos dispositivos móveis na mesma (Brzeziński et al., 2013).

3.4.2. Agência para Integração, Difusão e Arquivos de Informação Médica (AIDA)

Agência para a Integração, Difusão e Arquivos de Informação Médica ou simplesmente AIDA, é uma plataforma que foi desenvolvida por um grupo de investigadores da Universidade do Minho e do Centro Hospitalar do Porto (CHP), na qual já se encontra implementada em alguns hospitais em Portugal e é a plataforma na qual futuramente será utilizada para executar a solução encontrada.

A AIDA pode ser definida como uma agência que oferece aos trabalhadores eletrónicos inteligentes, mais conhecidos como agentes de *software*, com comportamento pró-ativo, que se encarregam de realizar tarefas tais como a comunicação entre os diferentes subsistemas que constitui a AIDA, para enviar e receber os dados (como por exemplo imagens, coleção de dados, prescrições médica, etc.), gerir e guardar as informações e responder aos pedidos dos utilizadores. A plataforma é baseada no paradigma orientado a agentes e tem demonstrado uma grande adaptabilidade, modularidade e efetividade através da utilização de um *Multi -Agent System* (MAS) básico que tem crescido de acordo com as necessidades particulares de cada instituição (Duarte et al., 2010).

A AIDA utiliza sistemas de agentes inteligentes que asseguram a interoperabilidade entre diferentes sistemas de informação heterogéneos (SAM, SONHO, PCE, SAPE, RIS, LIS, DIS, entre outros). O registo eletrónico da saúde é o responsável pelo armazenamento seguro e responsável de todas as informações seguras dos utentes, desde os dados pessoais, diagnósticos e até os procedimentos. O principal objetivo da AIDA é integrar, difundir e arquivar a grande quantidade de dados oriundos das diversas fontes heterogêneas (departamentos, serviços, unidades,

computadores, equipamentos médicos). A AIDA possui mecanismos para implementar a comunicação com os seres humanos com base em serviços baseados na web. A integração da AIDA com os subsistemas é feita recorrendo às ferramentas SOA e MAS (Cardoso, 2013; Marins, 2013).

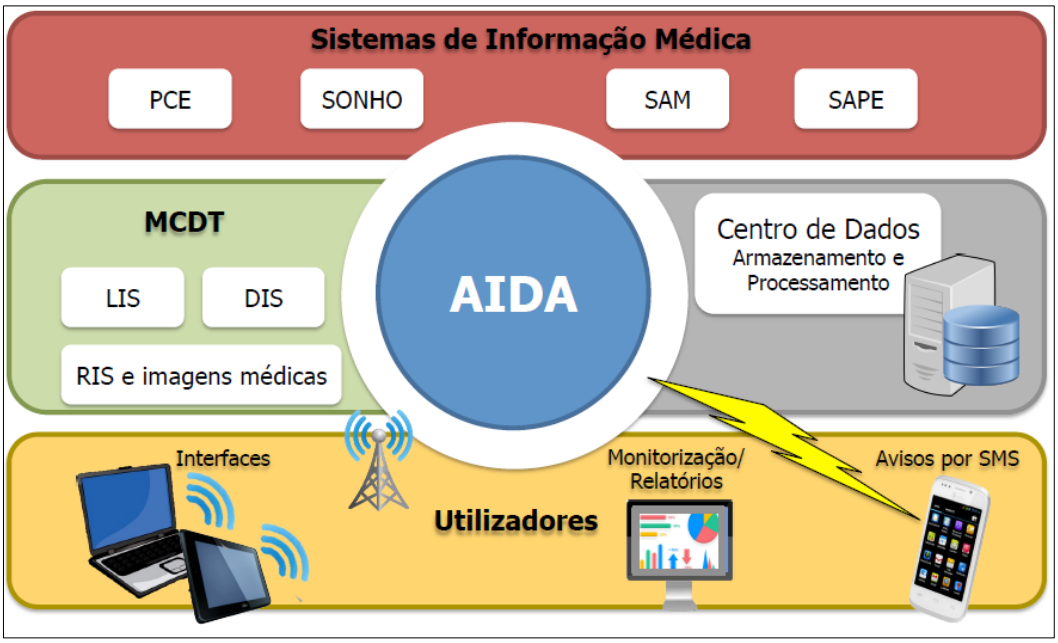


Figura 17 - Arquitetura da AIDA retirado de (Marins, 2013)

A Figura 17 ilustra a arquitetura da AIDA. De acordo com a arquitetura pode-se verificar que a AIDA possibilita a comunicação entre todos os sistemas de informação médica (PCE, SONHO, SAM, SAPE) todos os sistemas dos Meios Complementares de Diagnóstico e Terapêutica (MCDT) (LIS, DIS, RIS e imagens médicas) através de mensagens entre agentes inteligentes.

A plataforma AIDA constitui o elemento central no ambiente hospitalar pois é constituído por subsistemas que garantem a interoperabilidade e a comunicação entre os sistemas heterogéneos tais como (Cardoso, 2013; Duarte et al., 2010; Marins, 2013):

- **Processo Clinico Eletrónico** (PCE) - constitui um repositório de informação sobre a história clínica de um paciente que foi sujeito a cuidados de saúde, num formato que pode ser processado pelo computador, armazenado e transmitido a partir de um sistema seguro e acessível para vários utilizadores autorizados.

- **Sistema de Informação Administrativa, em Portugal** (SONHO) - representa, geri e arquiva as informações administrativas durante um episódio (recolha de todas as operações atribuídas a um determinado paciente desde o inicio até ao final do tratamento).
- **Sistema de Apoio Médico** (SAM) – representa, geri e arquiva a informação clínica durante o episódio.
- **Sistema de Apoio às Praticas de Enfermagem** (SAPE) - representar, gerir e arquivar informação sobre as práticas de enfermagem durante o episódio.
- **Meios Complementares de Diagnóstico e Terapêutica** (MCDT) – é constituída por sistemas de Informação de todos os serviços e sistemas do hospital, em particular os Laboratórios e os serviços de imagiologia e radiologia.

3.5. Metodologias de Desenvolvimento

Quando se fala nas metodologias de desenvolvimento de aplicações móveis, é preciso adotar uma abordagem metodológica para o desenvolvimento das aplicações. Atualmente existem duas abordagens diferentes para o desenvolvimento de aplicações móveis:

- Abordagem Nativa;
- Abordagem Multiplataforma.

A abordagem nativa consiste em desenvolver uma aplicação para uma determinada plataforma móvel, enquanto que a abordagem multiplataforma consiste em desenvolver uma aplicação para ser executada em plataformas diferentes, ou seja, plataformas heterogéneas. A abordagem multiplataforma encontra-se dividido em quatro categorias diferentes: WEB, híbrida, interpretada e compilação-cruzada.

3.5.1. Abordagem Nativa

As aplicações desenvolvidas segundo esta abordagem são projetadas para serem executadas numa plataforma específica. Como por exemplo, no caso do *Android* o desenvolvimento das suas aplicações é baseado no Java e o IDE utilizado é *Android* SDK (atualmente o *Android Studio*). No caso do *iOS* o desenvolvimento das suas aplicações é baseado no *Objective-C* e o IDE utilizado é *Xcode* IDE (Heitkötter, Hanschke, & Majchrzak, 2013; Scandurra & Rosario, sem data). Para ter uma melhor perceção sobre a abordagem nativa, a Tabela 9 apresenta as diferentes plataformas, linguagem, formatos e distribuição das aplicações nativas para cada sistema operativo móvel.

Tabela 9 - Desenvolvimento de Aplicações Nativas retirado de (IBM, 2012)

	Android	**Apple iOS**	**BlackBerry**	**Windows Phone**
Linguagem	Java	Objective-C, C, C++	Java (J2ME)	C#, VB, .NET entre outros
IDE	Android SDK Android Studio	Xcode	BB Java Eclipse Plug-in	Visual Studio, Windows Phone Tools
Formato	.apk	.app	.cod	.xap
App Stores	Google Play	Apple App Store	BlackBerry App World	Windows Phone Marketplace

3.5.2. Abordagem Multiplataforma

A abordagem multiplataforma consiste em desenvolver uma aplicação para ser executada em diferentes plataformas a partir de um único código fonte. Existem quatros categorias de desenvolvimento multiplataforma que são:

- Abordagem WEB;
- Compilação-Cruzada;
- Híbrida;
- Interpretada.

a) Abordagem WEB

A abordagem WEB também conhecida como aplicações WEB, são aplicações que são projetadas para serem executadas nos *browsers* dos dispositivos móveis, de modo a que os conteúdos são baixados da internet. Estes tipos de aplicações WEB são desenvolvidos com base nas tecnologias WEB tais como *Hyper Text Markup Language* (HTML), *Cascading Style Sheets* (CSS) e JavaScript e dadas as novas características emergentes do HTML5 e CSS3, possibilitam o desenvolvimento de aplicações cada vez mais ricas e complexas (Ciman & Gaggi, 2014; Delia, Galdamez, Thomas, Corbalan, & Pesado, 2015).

Segundo esta abordagem (WEB), os dispositivos móveis não têm a necessidade de armazenar quaisquer conteúdos que a aplicação WEB necessite, pois, os dados da aplicação são conduzidos pelo servidor. Uma vez que a aplicação é baseada num *browser*, este é independente da plataforma que o dispositivo utiliza (Rahul Raj & Seshu Babu Tolety, 2012; Xanthopoulos & Xinogalos, 2013). A Figura 18 ilustra o modelo de arquitetura das aplicações WEB. A arquitetura WEB é constituída por um *browser* que executa no dispositivo móvel do utilizador (cliente), que conecta a aplicação propriamente dita. Essa ligação é realizada através da internet, recorrendo a utilização dos *Web Services*. A aplicação por seu lado (servidor), processa a informação solicitada e responde ao utilizador (cliente) que através do mesmo meio de comunicação, ou seja, através dos *Web Services*.

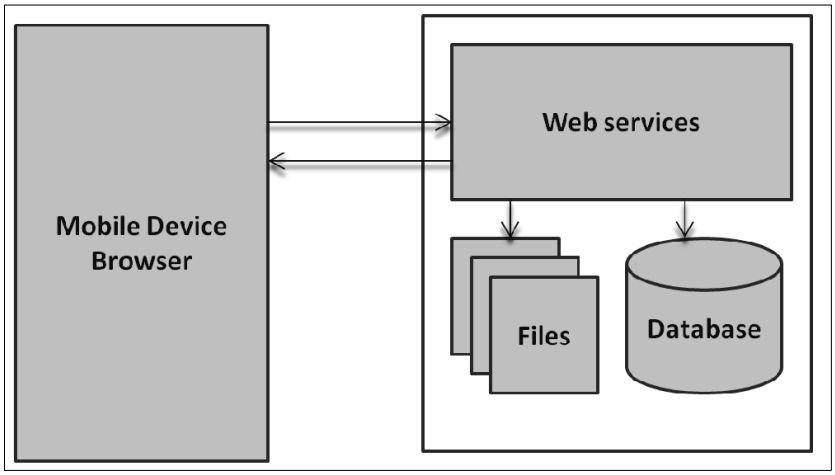


Figura 18 - Arquitetura das Aplicações WEB retirado de (Rahul Raj & Seshu Babu Tolety, 2012)

Nesta abordagem, algumas operações da aplicação são realizadas no lado do cliente, mas, no entanto, as operações críticas ou mais importantes são realizadas no servidor da aplicação. O cliente recebe a interface do utilizador e a lógica de validação dos dados do utilizador, enquanto que o servidor implementa a lógica do negócio (*business logic*) (Rahul Raj & Seshu Babu Tolety, 2012).

Uma das grandes desvantagens das aplicações WEB residem no fato de estas terem o acesso limitado ao *hardware* e aos dados dos dispositivos. Outra desvantagem é o tempo necessário para carregar e baixar o conteúdo da página WEB. Uma vez que estes tipos de aplicações não são instalados fisicamente no dispositivo móvel, existem situações, como por exemplo, quando os dispositivos se encontram no "modo Avião", estes tipos de aplicações WEB são inacessíveis para o utilizador (Heitkötter, Hanschke, et al., 2013; Thakare, Shirodkar, Parween, & Parween, 2014; Xanthopoulos & Xinogalos, 2013).

As aplicações baseadas na abordagem WEB apresentam inúmeras vantagens tais como (Rahul Raj & Seshu Babu Tolety, 2012):

- A aplicação não requer qualquer tipo de instalação no dispositivo móvel, visto que o mesmo é acedido por meio de uma *Uniform Resource Locator* (URL) no *browser* do dispositivo;
- Os dados da aplicação encontram-se todos no servidor da mesma, de modo que não é necessário efetuar qualquer tipo de atualização da aplicação no dispositivo móvel;
- Uma vez que os *browsers* são muitos padronizados, a interface do utilizador WEB é facilmente reutilizada por outras plataformas.

Esta abordagem ainda carece de alguns desafios que precisam de ser ultrapassadas no desenvolvimento destes tipos de aplicações, tais como (Rahul Raj & Seshu Babu Tolety, 2012):

- As aplicações não podem ser distribuídas nas lojas *online* de aplicações móveis, visto que são acedidos por meio de um URL. Os utilizadores dos dispositivos móveis procuram as aplicações que precisam nas lojas *online*. A inexistência destes tipos de aplicações nas lojas *online* pode provocar um impacto negativo na popularidade das mesmas. Embora, já existe a possibilidade de converter alguns sites para o formato "*reader*", sendo que estas já apresentam um formato móvel e podem ser aceites nas lojas *online*;

- Podem apresentar menor desempenho devido a problemas de conexão e atrasos na rede;
- Não tem o acesso a todos os recursos de *hardware* do dispositivo móvel tais como: Camera, GPS, Bluetooth, NFC, Acelerómetro, etc.;
- Contrariamente às aplicações desktop, as aplicações WEB tem de suportar diferentes resoluções de ecrã. Este é um dos fatores mais importante que deve ser considerado quando se pretende desenvolver uma aplicação móvel;
- O teste manual sobre a aparência da aplicação em resoluções de ecrã diferentes consome uma quantidade de tempo considerável;
- O programador da aplicação tem menos controlo sobre a forma como os diferentes *browsers* irão processar o conteúdo;
- As aplicações WEB são limitadas para impulsionar os gestos (*gestures*) oferecidos pela plataforma;
- Monitorizar uma aplicação WEB não é tão simples como uma aplicação nativa.

Geralmente é mais simples começar a implementar uma aplicação WEB móvel do que implementar uma aplicação nativa. Mas, no entanto, as aplicações WEB requerem a realização de testes em diferentes dispositivos para testar o comportamento dos mesmos em diferentes dispositivos.

b) Abordagem Compilação-Cruzada

A abordagem compilação-cruzada ou também designadas aplicações *generated,* o *cross-compiler* (multicompilador) faz a conversão do código fonte para binários nativos. O *cross-compiler* é o responsável por geral o código executável para uma determinada plataforma específica. O programador pode escrever código fonte em uma linguagem de programação comum e o *cross-compiler* faz a compilação do código fonte para o código nativo de uma plataforma específica. A aplicação final utiliza a linguagem nativa (gerada pelo *cross-compiler*), pelo que pode ser considerado uma aplicação nativa para todos os efeitos. Esta abordagem permite que o programador tenha o total acesso sobre todos os recursos nativos do dispositivo móvel e o desempenho pode ser considerado semelhante (senão mesmo igual) relativamente às aplicações nativas (Ciman & Gaggi, 2014).

Na verdade, alguns testes realizados têm apresentado que para as aplicações mais complexas, a solução nativa é melhor, desde que o código gerado dê piores desempenhos à aplicação resultante, em comparação com o código escrito pelo programador. Este tipo de abordagem depende muito da eficiência e fiabilidade do *cross-compiler*. Como exemplo de *frameworks* de desenvolvimento de aplicações segundo a abordagem compilação-cruzada têm-se: o **Applause** e o **Rhodes** (Rahul Raj & Seshu Babu Tolety, 2012). A Figura 19 ilustra a arquitetura das aplicações de compilação cruzada ou *generated*. Este tipo de abordagem, conforme se pode constatar na Figura 19, o código fonte da aplicação é escrito numa linguagem de programação, depois o código é processado através do c*ross compiler* que a partir do código inicial gera aplicações nativas para a plataforma pretendida. Aqui têm-se o especial destaque para o *cross compiler*, pois através dele é possível gerar a mesma aplicação para as diferentes plataformas móveis.

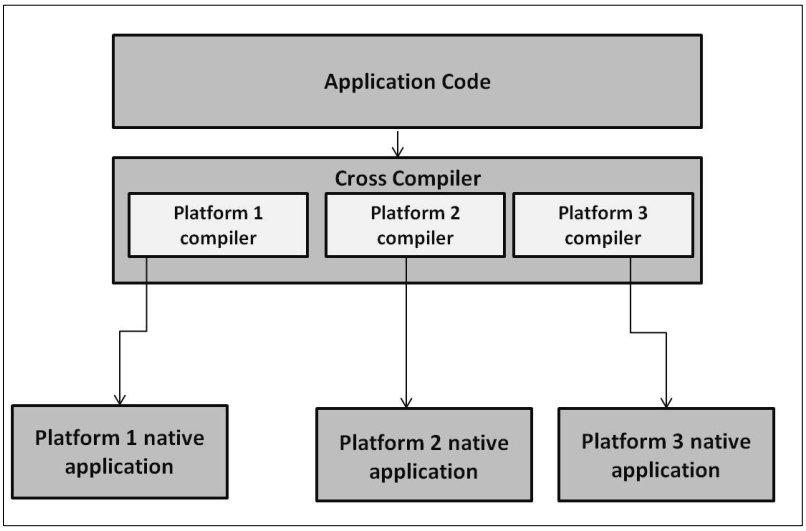


Figura 19 - Arquitetura das Aplicações de Compilação-Cruzada retirado de (Rahul Raj & Seshu Babu Tolety, 2012)

As aplicações baseadas na arquitetura compilação-cruzada apresentam as seguintes vantagens (Rahul Raj & Seshu Babu Tolety, 2012):

- As aplicações de compilação-cruzada oferecem todos os recursos que as aplicações nativas oferecem;
- O *hardware* e o *software* podem ser acedidos. Todos os componentes da interface do utilizador nativo podem ser utilizados;
- O desempenho é o principal destaque destas aplicações.

Os principais desafios que esta abordagem apresenta são (Rahul Raj & Seshu Babu Tolety, 2012):

- A principal desvantagem da compilação-cruzada reside no fato da interface do utilizador não poder ser reutilizada. Da mesma forma, os recursos específicos de uma plataforma, como o acesso à camera, serviços de localização, notificações locais, entre outros não podem ser reutilizados. Estas características são específicas das plataformas móveis e a forma de acede-los varia de plataforma para plataforma;
- Esta abordagem é mais apropriada para as aplicações simples, mas quando se trata de aplicações mais sofisticadas esta abordagem é superada pela abordagem nativa;
- Identificar e corrigir os problemas das fases da compilação-cruzada é difícil para o programador da aplicação.

c) Abordagem Híbrida

A abordagem híbrida é um meio termo entre a abordagem WEB e a abordagem nativa. Neste caso a aplicação funciona de duas formas diferentes, ou seja, as aplicações híbridas são desenvolvidas com recurso às tecnologias WEB e são executados nos dispositivos móveis como se fossem aplicações nativas. Por isso, esta abordagem é também conhecida como "WEB *to native wrapper*" (Charkaoui, Adraoui, & Benlahmar, 2014; Rahul Raj & Seshu Babu Tolety, 2012). Esta abordagem utiliza o motor *WEBKit* para exibir os controlos, botões e as animações. O motor *WEBKit* é o responsável pelo desenho e gestão dos objetos da interface do utilizador (Ciman & Gaggi, 2014).

O acesso aos recursos nativos do dispositivo é realizado pela aplicação híbrida a partir das APIs especializadas na camada de abstração. Esta abordagem tira vantagem tanto do mecanismo de navegação, como também dos recursos do dispositivo. Contrariamente das aplicações WEB, as aplicações híbridas precisam de ser baixadas e instaladas no dispositivo móvel como se fossem aplicações nativas, mas a sua performance é muitas vezes inferiores a estas (aplicações nativas), de modo que a sua execução requer da execução de um *browser* que faz parte da aplicação final (Ciman & Gaggi, 2014; Delia et al., 2015).

Quando comparado com as aplicações nativas, estes são inferiores, pois necessitam de mais tempo de processamento e não alcançam totalmente a aparência nativa (Thakare et al., 2014). Como por exemplo de uma *framework* mais popular para o desenvolvimento de aplicações

híbridas é o **PhoneGap** ou **Apache Cordova**. A implementação do código para desenvolver as aplicações híbridas pode ser feita utilizando várias tecnologias e plataformas de desenvolvimento, mas para conseguir uma interface parecida à nativa, ou seja, uma interface de utilizador simulada, é necessário recorrer a *frameworks* de desenvolvimento WEB específico, tais como o **jQuery**, **Sencha Touch**, entre outras *frameworks* de *User Interface* (UI) (ferramentas utilizadas na abordagem WEB). Geralmente estas ferramentas são designados de ***UI-frameworks*** (Rahul Raj & Seshu Babu Tolety, 2012; Xanthopoulos & Xinogalos, 2013). A Figura 20 ilustra a arquitetura de aplicações híbridas. Conforme se pode constatar na mesma, as aplicações híbridas são constituídas por três camadas distintas entre as quais o *Native Library* (Biblioteca Nativa), *JavaScript Abstraction Layer* (Camada de Abstração do JavaScript) e o *Hybrid Application* (Aplicação Híbrida). As camadas aplicação híbrida e a biblioteca nativa interagem com os *Web Services* que conectam com a base de dados da aplicação e apresentam as informações na interface do utilizador, enquanto que a camada de abstração JavaScript acede aos recursos do dispositivo.

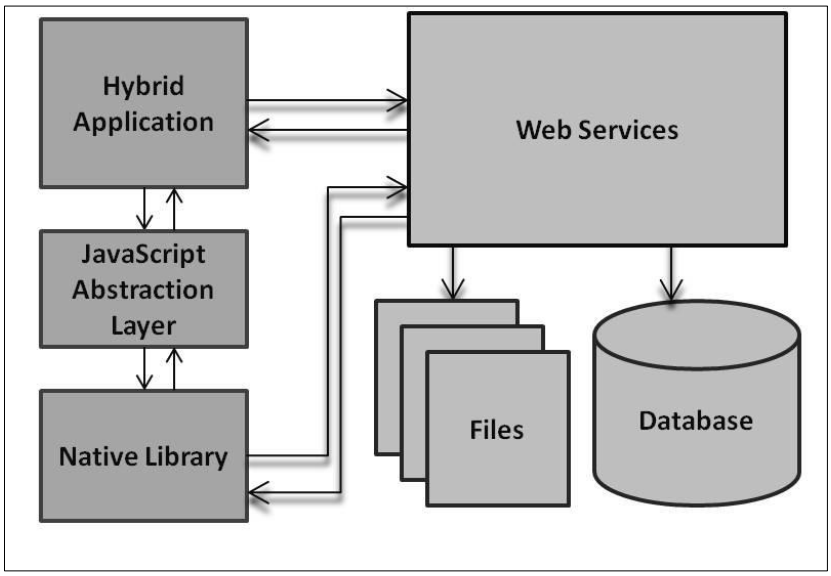


Figura 20 - Arquitetura das Aplicações Híbridas retirado de (Rahul Raj & Seshu Babu Tolety, 2012)

As aplicações baseadas nesta abordagem apresentam as seguintes vantagens (Rahul Raj & Seshu Babu Tolety, 2012; Thakare et al., 2014):

- As aplicações híbridas são distribuídas através das lojas *online* (*web Stores*);
- A principal vantagem da abordagem híbrida reside no fato da interface do utilizador poder ser reutilizada em diferentes plataformas, utilizando recursos da plataforma nativa;

- Uma vez que os recursos da plataforma nativa são disponibilizados através da camada de abstração de *hardware,* a aplicação pode utilizar os recursos do dispositivo;
- Os recursos de *hardware* e da plataforma do dispositivo podem ser acedidos com APIs de *frameworks* específicas.

Os principais desafios que esta abordagem apresenta são (Rahul Raj & Seshu Babu Tolety, 2012; Thakare et al., 2014):

- As aplicações híbridas apresentam desempenho inferior quando comparadas com as aplicações nativas, desde que a sua execução é efetuada num *browser*;
- Uma vez que as aplicações híbridas utilizam a linguagem de programação JavaScript na camada de abstração do *hardware,* são submetidos às vulnerabilidades de comunicação em espaço cruzado, tais como o *Cross-Site-Scripting* ou **XSS**, que consiste na injeção dos algoritmos maliciosos utilizando a linguagem de programação JavaScript, entre outros. Estas aplicações carecem de comportamentos específicos da plataforma JavaScript e incompatibilidades do *threading model* com o JavaScript;
- Mesmo que a interface do utilizador possa ser reutilizada em diferentes plataformas, este não terá a mesma aparência do que uma aplicação nativa.

d) Abordagem Interpretada

A abordagem interpretada ou simplesmente *Runtimes* o código da aplicação é implementado no dispositivo móvel e é interpretado seguidamente, ou seja, a aplicação é desenvolvida numa linguagem específica (Ex: JavaScript), mas depois o código da aplicação é interpretado por um interpretador que será futuramente convertida em aplicação nativa para as diferentes plataformas. Nesta abordagem, existe um interpretador que executa o código em tempo de execução (Delia et al., 2015; Thakare et al., 2014). Esta abordagem permite aceder aos recursos nativos que são disponibilizados por intermédio de uma camada de abstração. O compilador interpreta o código fonte, convertendo-o ao mesmo tempo em aplicações para diferentes plataformas, portanto, tem suporte a desenvolvimento de aplicações multiplataforma. O acesso aos recursos da API nativa é feito por meio da camada de abstração e utiliza os componentes da interface do utilizador nativo da plataforma específica para interagir com o utilizador.

A lógica da aplicação é implementada de forma independente, recorrendo a várias tecnologias tais como: Java, Ruby, XML, entre outros (Charkaoui et al., 2014; Thakare et al., 2014). Esta abordagem pode atingir um alto nível de código reutilizável, mas, no entanto, poderá reduzir um pouco o desempenho, devido ao processo de interpretação da mesma. Um exemplo de ambiente de desenvolvimento mais conhecido para desenvolver aplicações interpretadas é o **Appcelerator Titanium Mobile** (Ciman & Gaggi, 2014; Delia et al., 2015; Rahul Raj & Seshu Babu Tolety, 2012; Thakare et al., 2014). A Figura 21, ilustra a arquitetura das aplicações interpretadas. Conforme se pode ver na mesma, as aplicações interpretadas têm quatro camadas o *Application Code* (Código da Aplicação), o *Interpreter* (Interpretador), o *Abstraction Layer* (Camada de Abstração) e o *Native APIs* (APIs Nativas).

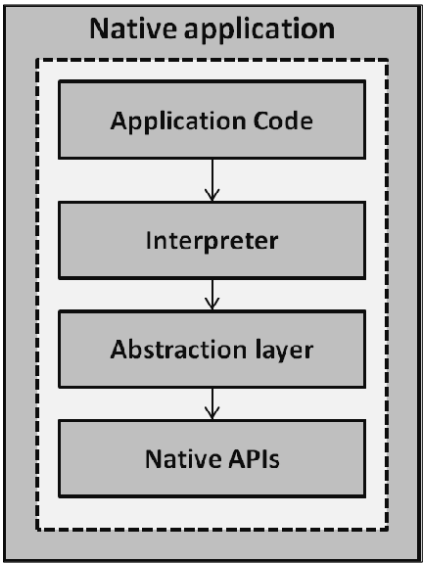


Figura 21 - Arquitetura das Aplicações Interpretadas retirado de (Rahul Raj & Seshu Babu Tolety, 2012)

As aplicações baseadas nesta abordagem apresentam as seguintes vantagens (Rahul Raj & Seshu Babu Tolety, 2012):

- As aplicações interpretadas têm a aparência de uma aplicação nativa. A lógica do negócio pode ser reutilizada em diferentes plataformas;
- As aplicações interpretadas são distribuídas através das lojas *online* (*web Stores*);
- O *hardware* do dispositivo e os recursos da plataforma estão envolvido com a API da *framework*.

De modo análogo os principais desafios apresentados por esta abordagem são (Rahul Raj & Seshu Babu Tolety, 2012):

- A reutilização da interface do utilizador depende da camada de abstração da *framework*;
- A principal desvantagem das aplicações interpretadas é o fato do desenvolvimento ser dependente de um conjunto de características facultadas pela *framework* escolhida;
- O desempenho das aplicações interpretadas pode sofrer degradações devido ao processo de interpretação do código em tempo de execução.

3.5.3. Critérios de Seleção da Abordagem Multiplataforma Adequada

O critério de seleção da abordagem multiplataforma a ser escolhido depende muito das exigências da aplicação, bem como das plataformas alvos, o tipo de aplicação, a forma como é realizada o acesso aos dados e ao *hardware*, da interface do utilizador, do seu aspeto, do seu desempenho e do meio de distribuição *online*. Por vezes é necessário ainda considerar o fator suporte das ferramentas e das *frameworks* ao selecionar uma abordagem multiplataforma (Rahul Raj & Seshu Babu Tolety, 2012). A Tabela 10 apresenta de forma concisa alguns dos pontos relevantes na escolha de uma das abordagens de desenvolvimento multiplataforma. Na referida tabela, é possível consultar qual é a *framework* adotada, a linguagem de programação utilizada e as plataformas suportadas por cada uma da abordagem multiplataformas.

Tabela 10 - Metodologias Multiplataforma e Frameworks adotados retirado de (Rahul Raj & Seshu Babu Tolety, 2012)

Abordagem Multiplataforma	**Frameworks adotados**	**Linguagens utilizadas**	**Plataformas suportadas**
WEB	Tecnologias Web	HTML, CSS, JavaScript	Android, iOS, Windows, BlackBerry
Híbrido	PhoneGap	JavaScript	Android, iOS, Windows, BlackBerry
Interpretada	Titanium, JMango	JavaScript	Android, iOS, BlackBerry
Compilação-Cruzada	Mono, Applause	C#	Android, iOS, Windows

Cada aplicação móvel tem as suas características específicas, o que implica que deverá ser aplicada uma abordagem multiplataforma para cada aplicação de acordo com as suas características. A metodologia deve ser escolhida com sabedoria, consoante o tipo de aplicação a ser desenvolvida. A execução nativa refere-se às abordagens compilação-cruzada e interpretada, uma vez que ambos são executados no espaço nativo. O desempenho das aplicações interpretadas é inferior quando comparadas com as aplicações de compilação-cruzada, devido à sua interpretação dinâmica (Rahul Raj & Seshu Babu Tolety, 2012).

Classificação das Aplicações

Normalmente as aplicações móveis são classificadas por (Rahul Raj & Seshu Babu Tolety, 2012):

- Aplicações de dados conduzidos pelo servidor;
- Aplicações baseadas em Sensor/IO;
- Aplicações autónomas/independentes;
- Aplicações cliente-servidor.

➢ Aplicações de dados conduzidas pelo servidor

As aplicações de dados conduzidos pelo servidor são aplicações do tipo cliente, na qual a lógica do negócio (*business logic*) encontra-se no servidor. A aplicação móvel permite a visualização de informações e a interação com o utilizador. Neste caso a abordagem WEB é a mais adequada para as aplicações deste género. No caso da abordagem WEB, a interface do utilizador das aplicações pode ser reutilizada. Pelo fato de a lógica do negócio da aplicação ser implementada no servidor, a aplicação móvel não precisa de se preocupar com quaisquer tipos de atualizações da aplicação. No caso de aplicação necessitar executar qualquer ação ao nível da plataforma, como por exemplo, agendar notificações locais acerca da recuperação dos dados, a abordagem híbrida seria a mais adequada (Rahul Raj & Seshu Babu Tolety, 2012).

➢ Aplicações baseadas nos Sensores/IO

As aplicações baseadas nos sensores utilizam principalmente o *hardware* do dispositivo. Porém, existem algumas aplicações que processam os dados dos sensores localmente, enquanto que, outras dependem do servidor para o processamento destes dados. É uma abordagem que deverá ser selecionada sabiamente com base na área do processamento para estes tipos de aplicações.

As aplicações de compilação-cruzada seriam a melhor escolha, se os dados fossem processados no dispositivo móvel, devido ao fato de terem acesso ao *hardware* nativo e de proporcionar ótimo desempenho. As aplicações interpretadas poderiam ser a segunda opção, visto que oferecem um conjunto de recursos nativos a partir dos ambientes de desenvolvimento correspondente. Por outro lado, a nível do desempenho, as aplicações interpretadas são inferiores quando comparadas com às aplicações de compilação-cruzada. No caso de a aplicação necessitar de processamento intensivo e requer um servidor para o processamento dos dados, a abordagem híbrida poderá ser mais significativa (Rahul Raj & Seshu Babu Tolety, 2012).

- **Aplicações Autónomas**

As aplicações autónomas referem-se às aplicações em que os dados são produzidos e processados no próprio dispositivo móvel. O dispositivo móvel inclui as aplicações na qual os dados são provenientes do servidor e o processamento dos dados é do lado do cliente. A aplicação móvel processa localmente e apresenta a visualização das informações, o que possibilita de igual modo a interação com utilizadores. No caso das aplicações autónomas, a abordagem interpretada é a escolha mais indicada, uma vez podem ser aproveitados os recursos nativos da plataforma móvel (Rahul Raj & Seshu Babu Tolety, 2012).

- **Aplicações Cliente-Servidor**

As aplicações cliente-servidor, tanto o servidor como o cliente encontram-se envolvidos no processo de processamento dos dados. Neste caso, o cliente para além de permitir a visualização de informações que são encaminhados pelo servidor e a interação com o utilizador, também faz o processamento local dos dados de modo a ser apresentada futuramente. Os dados processados não são enviados para o servidor, visto que o cliente deve fazer a gestão e processamento desses dados, que futuramente serão disponibilizados aos utilizadores da aplicação. Nas aplicações cliente-servidor, a abordagem híbrida será a mais apropriada (Rahul Raj & Seshu Babu Tolety, 2012).

É importante a seleção de uma abordagem multiplataforma a ser seguido no desenvolvimento de uma determinada aplicação. A Tabela 11, ilustra a classificação de cada abordagem para os diferentes tipos de aplicações (Rahul Raj & Seshu Babu Tolety, 2012). As avaliações dos resultados são interpretadas da seguinte ordem:

1- Não preferencial;
2- Preferencial, mas não é a abordagem ideal;
3- Abordagem ideal.

Tabela 11 - Tipos de Aplicações e Abordagem Preferencial retirado de (Rahul Raj & Seshu Babu Tolety, 2012)

Tipo da aplicação	**WEB**	**Híbrida**	**Execução Nativa (Interpretada/ Compilação-Cruzada)**
Aplicação de dados conduzidos pelo servidor	3	2	1
Aplicação autónomas	1	2	3
Aplicações baseadas em Sensores/ IO (Dados processados no dispositivo)	1	2	3
Aplicações baseadas em Sensores/IO (Dados processados no servidor)	1	3	2
Aplicações Cliente Servidor	1	3	2

De acordo com a tabela apresentada, é visível em que tipos de aplicações uma abordagem pode ser preferencial em detrimento de outra abordagem.

3.5.4. Análise comparativa das abordagens de desenvolvimento multiplataforma

Esta subsecção apresenta a análise comparativa das abordagens de desenvolvimento multiplataforma das aplicações móveis, descritas anteriormente com base num conjunto de características, convergentes com um conjunto de critérios. Os critérios selecionados para a análise comparativa entre as abordagens de desenvolvimento multiplataforma são (Heitkötter, Hanschke, et al., 2013; Serrano, Hernantes, & Gallardo, 2013; Sommer & Krusche, 2013; Xanthopoulos & Xinogalos, 2013):

- **Mercado de distribuição** - avaliar a possibilidade e a facilidade de distribuição das aplicações nas lojas *online* (*Web Store*s) das plataformas dos dispositivos móveis, tais como a Google Play e o iTunes da Apple.
- **Tecnologias generalizadas** - avaliar se as aplicações podem ser criadas recorrendo às tecnologias generalizadas, tal como o JavaScript.
- **Acesso ao *hardware* e aos dados -** avaliar se as aplicações têm acesso total, não tem acesso ou tem acesso limitado ao *hardware* e aos dados do dispositivo móvel.
- **Interface do utilizador e a aparência** - avaliar se as aplicações têm suporte intrinsecamente aos componentes da interface do utilizador nativo e se a aparência é simulada através das ferramentas WEB, tais como o jQuery, Sencha Touch ou entre outras ferramentas;
- **Perceção do utilizador acerca do desemprenho** - avaliar se as aplicações têm baixo, médio ou alto desempenho de acordo com a perceção dos utilizadores finais (como o tempo de carregamento e a velocidade da execução) quando comparada com uma aplicação nativa. Trata-se de um critério de uma estimativa empírica aproximada com base na experiência prática e nas informações recolhidas na WEB. O desempenho global poderá ser afetado por diversos fatores.

As características acima apresentadas fazem parte de conjunto de características que podiam ser referenciadas, uma vez que existem muitas mais características que são importantes como estas, que também podiam ser referenciadas aqui para enriquecer ainda mais esta análise comparativa.

A Tabela 12, apresenta um modo sucinto esta análise comparativa das diferentes abordagens de desenvolvimento multiplataforma.

Tabela 12 - Análise comparativa das abordagens de desenvolvimento multiplataforma retirado de (Xanthopoulos & Xinogalos, 2013)

	WEB	**Híbrida**	**Interpretada**	**Compilação-Cruzada**
Mercado de Distribuição (*app Stores*)	Não	Sim, mas não garantido	Sim	Sim
Tecnologias Generalizadas	Sim	Sim	Sim	Não
Acesso ao *hardware* e aos Dados	Limitado	Limitado	Limitado	Acesso Total
Interface do utilizador e a aparência	Simulado	Simulado	Nativo	Nativo
Perceção do utilizador acerca do desempenho	Baixo	Médio	Médio	Alto

As aplicações WEB são aplicações baseadas no *browser* que não precisam de ser instaladas fisicamente nos dispositivos móveis. Estas requerem um tempo extra necessário para baixar o código fonte e os recursos da internet, o que faz com que estas aplicações tenham inevitavelmente baixo desempenho quando comparadas com às aplicações nativas. Por vezes podem tornar-se indisponíveis devido a falhas na rede. As *frameworks* como o jQuery Mobile são capazes de desenvolver aplicações com aparência nativa (Heitkötter, Hanschke, et al., 2013; Xanthopoulos & Xinogalos, 2013).

As aplicações híbridas simulam a aparência das aplicações nativas. As principais vantagens das aplicações híbridas são o fato de ter a capacidade de executar o código fonte em uma vasta gama de plataformas móveis e do seu desenvolvimento ser efetuado com base nas tecnologias de desenvolvimento WEB muito conhecidos tais como JavaScript. Não é necessário ter o conhecimento aprofundado sobre a plataforma de destino. Geralmente estas aplicações tem um desempenho médio, do ponto de vista dos utilizadores finais quando comparadas com as aplicações nativas, pois a interface do utilizador é baseada na WEB e não na utilização dos

componentes nativos otimizados. As APIs especializadas são intersetadas para fazer a interpretação do acesso ao *hardware* e a lógica do negócio. O acesso ao *hardware* e aos dados da aplicação é limitado, mas geralmente as aplicações são distribuídas através das lojas *online* (*Web Stores*) (Heitkötter, Hanschke, et al., 2013; Xanthopoulos & Xinogalos, 2013).

As aplicações interpretadas têm a interface do utilizador nativo e a lógica da aplicação é implementada independentemente. Do ponto de vista dos utilizadores finais, o desempenho geral das aplicações tem um desempenho médio, quando comparadas com as aplicações nativas. Embora a interface do utilizador é rápida, o tempo extra que é necessário para efetuar a interpretação da lógica da aplicação e a presença das APIs especializadas para o acesso aos recursos do *hardware*, faz com que haja uma ligeira perca no desempenho. A nível da fiabilidade, as APIs especializadas podem ser utilizadas para armazenar o estado e os dados das aplicações (como por exemplo o SQLite API), e geralmente a distribuição das aplicações é realizada sem nenhuma dificuldade nas lojas *online* (*Web Stores*) (Heitkötter, Hanschke, et al., 2013; Xanthopoulos & Xinogalos, 2013).

As aplicações de compilação-cruzada podem atingir alto desempenho tal como as aplicações nativas, pelo fato de o ambiente de desenvolvimento do *software* desta categoria têm suporte às principais plataformas móveis, como *Android*, *iOS* e *Windows Phone*. Estes têm total acesso ao *hardware* e aos dados do dispositivo móvel, e ainda tem a interface do utilizador nativo (Heitkötter, Hanschke, et al., 2013; Xanthopoulos & Xinogalos, 2013).

3.5.5. Análise comparativa do desenvolvimento de aplicações nativas e multiplataforma

Tanto como a abordagem nativa ou a abordagem multiplataforma, ambas apresentam vantagens e desvantagens. Para efetuar uma análise comparativa mais aprofundada, a Tabela 13 apresenta de forma detalhada o resultado da comparação entre as duas abordagens com base nos seguintes critérios (Scandurra & Rosario, sem data; Serrano et al., 2013):

- **Experiência e interface do utilizador** - as aplicações nativas apresentam uma interface mais fluida e responsiva do que as aplicações multiplataforma, mais concretamente no caso das animações e dos gestos. Isto acontece porque quando se desenvolve uma aplicação nativa numa determinada plataforma, têm-se o acesso completo ao API de

todos os dispositivos. As soluções multiplataformas já estão a oferecer APIs nativos para utilização, mas estes referem sempre a um subconjunto limitado dos recursos do dispositivo e mesmo assim, muitas vezes é necessário esperar até que sejam disponibilizados de modo a serem utilizados.

- **Desempenho** - uma das principais vantagens de utilizar a abordagem de desenvolvimento nativo, é o fato de que as aplicações executarem mais suavemente em qualquer dispositivo móvel que utiliza o mesmo sistema operativo. Enquanto que, a abordagem multiplataforma pode às vezes ser mais lento do que quando é utilizado abordagens nativas para desenvolver uma aplicação. Tal diferença pode ser facilmente notada durante a representação gráfica e animações.

- **Recursos específicos do dispositivo** - as abordagens nativas oferecem aos programadores das aplicações a vantagem de ter o acesso a todas as funcionalidades da plataforma. Por outro lado, as abordagens multiplataformas oferecem vantagens para um número limitado de funcionalidades. Como por exemplo, as aplicações multiplataformas, o suporte aos gráficos *high-end* e 3D são muitas vezes limitadas.

- **Distribuição via *app store*** - podem existir requisitos rigorosos para a admissão das aplicações nas lojas *online* (*Web Store*) públicas. Por exemplo, no caso da Apple Inc., requerem que os programadores submetam aplicações para efeito de testes e verificação da compatibilidade com os dispositivos móveis antes de serem disponibilizados nas lojas *online*. Neste sentido, as aplicações desenvolvidas com ferramentas nativas são geralmente aceites de forma mais fácil nas lojas *online* do que as aplicações multiplataformas, devido ao facto que as aplicações nativas foram desenvolvidas por programadores com vasto conhecimento da plataforma de destino.

- **Custo de desenvolvimento multiplataforma** - os programadores nativos, quando pretenderem direcionar às suas aplicações para múltiplas plataformas móveis, deverão adotar boas práticas da arquitetura da aplicação, de modo que sejam utilizados o modelo de dados que otimiza o esforço do desenvolvimento para múltiplas plataformas. No caso do desenvolvimento de aplicações nativas, requerem que os programadores tenham

conhecimento das plataformas de destino. Já no caso de desenvolvimento de aplicações multiplataformas, o custo de desenvolvimento é reduzido, ou seja, menor do que custo de desenvolvimento das aplicações nativas, pois não é necessário que os programadores tenham conhecimento da plataforma de destino. Talvez esta seja a maior vantagem, porque o esquema de compilações múltiplas faz com que as empresas ou marcas obtenham as aplicações para outras plataformas sem terem a necessidade de investir em novas equipas ou programadores específicos para determinado ecossistema.

- **Suporte a desenvolvedores** - os programadores geralmente preferem mais o desenvolvimento de aplicações nativas, pelo fato de ser mais fácil obter ajudas a partir dos fórums *online,* para obter respostas rápidas de outras pessoas que já desenvolveram estas aplicações há mais tempo. O desenvolvimento multiplataforma é uma técnica mais recente, pelo que é um bocado mais difícil obter ajudas. Mas, no entanto, já existem *frameworks* de desenvolvimento de aplicações multiplataformas que oferecem uma boa documentação a APIs, o que torna mais fácil a obtenção de ajudas.

- **Segurança** - as aplicações multiplataformas apresentam maiores riscos do que as aplicações nativas. A razão está diretamente ligada com os riscos do HTML5. A título de exemplo, para citar algumas das vulnerabilidades da segurança do URL, devido aos dados armazenados na cache do *browser* do dispositivo móvel. Contrariamente, os sistemas operativos com o *iOS* e o *Android* oferecem serviços de segurança dos dados integrados, tais como a criptografia dos dados. As técnicas de ataque como a manipulação de *cookies* e *SQL injection* para obter o acesso aos dados sensíveis no servidor *back-end,* bem como para os dispositivos móveis em si não são possíveis em aplicações nativas bem construídas.

- **Acesso atempadamente às novas inovações das plataformas móveis** - as *frameworks* multiplataformas podem não suportar todas as funcionalidades do sistema operativo ou do dispositivo móvel. Quando uma nova funcionalidade ou um novo recurso for adicionado no sistema operativo, esta não fica imediatamente disponível para a *framework* multiplataforma que está a ser utilizado. Para isso, é necessário um tempo de espera, de forma a receber as atualizações para suportar as novas funcionalidades.

- **Reutilização do código** - normalmente o código das aplicações multiplataformas são consideradas reutilizáveis. Em vez de ter que desenvolver uma ação específica ou uma sequência específica de ações para cada plataforma em particular, o programador escreve apenas um código uma única vez e depois pode reutilizá-la em outras plataformas ou mesmo outros projetos. Porém isso nem sempre é verdade, pois algumas *frameworks* multiplataformas utiliza muitas vezes o seu próprio subconjunto do JavaScript, o que significa que para mudar de plataforma o código utilizado anteriormente não poderá ser reutilizado sem algumas alterações substanciais.

- **Desafios do *design* -** as aplicações desenvolvidas segundo a abordagem nativa, tem o *design* simplificado devido ao suporte e aos serviços disponibilizados pelo sistema operativo. Por exemplo, o sistema operativo pode notificar as aplicações acerca dos eventos, tais como a chegada de mensagens e também sobre o nível da energia. No caso de aplicações desenvolvidas segundo a abordagem multiplataforma, os programadores têm de adicionar estes recursos de forma explícita. Além disso, nas *frameworks* multiplataformas, os programadores têm de projetar o *design* como é que cada recurso que podem precisar tem de ser implementada para cada plataforma de destino. Como por exemplo, projetar o *design* de uma aplicação para *iOS* é diferente de projetar o *design* uma aplicação para *Android*, pois as convenções da interface do utilizador e a experiência do utilizador são diferentes, e o *touch point* e os menus funcionam de maneira diferentes.

- **Disponibilidade do conhecimento de programação** - está amplamente reconhecido que existem mais programadores WEB do que nativos. Como a grande maioria das *frameworks* multiplataformas são baseadas em HTML e CSS, assim torna mais fácil a obtenção de conhecimentos necessários para os desenvolvedores WEB. As habilidades dos programadores nativos, geralmente são mais custosos e são mais difíceis de os obter.

Tabela 13 - Abordagem Nativa vs Multiplataforma retirado de (IBM, 2012; Scandurra & Rosario, sem data)

	Nativo	**Multiplataforma**
Experiencia do utilizador e interface do utilizador	Melhor	Pior
Desempenho	Melhor	Pior
Recursos específicos do dispositivo	Melhor	Pior
Distribuição via *app Store* (*Web Store*)	Melhor	Pior
Custo de desenvolvimento multiplataforma	Melhor	Pior
Suporte a desenvolvedores	Melhor	Pior
Segurança	Melhor	Pior
Acesso atempadamente às novas inovações das plataformas móveis	Melhor	Pior
Reutilização do código	Pior	Melhor
Desafios do Design	Pior	Melhor
Disponibilidade do conhecimento de programação	Pior	Melhor

A análise comparativa entre a abordagem nativa e a abordagem multiplataforma, presente na Figura 22, ilustra de forma resumida e concisa a comparação. O principal objetivo da abordagem multiplataforma é a redução de custos e o tempo de desenvolvimento, de forma a ser possível que um único código fonte desenvolvido pode gerar aplicações para diferentes plataformas móveis. Enquanto que, a abordagem nativa apresenta melhor interfaces do utilizador (UI) e melhores desempenho.

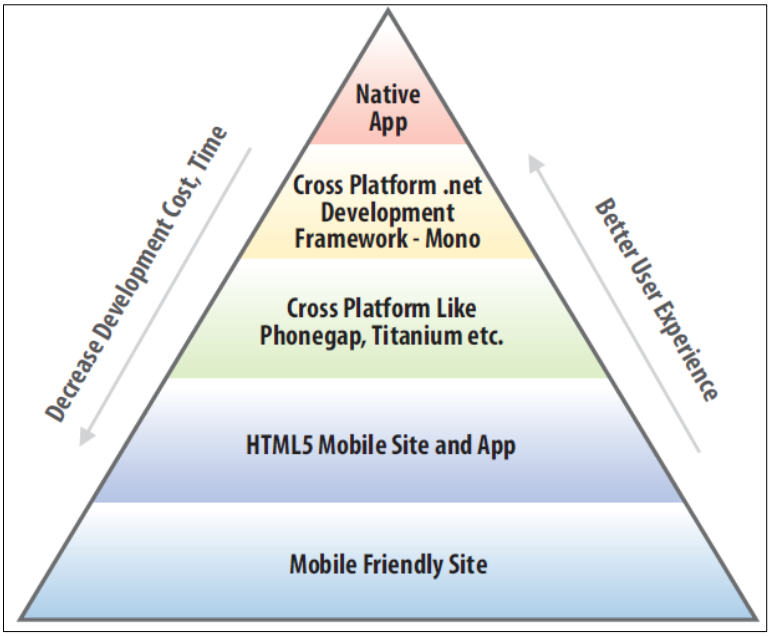


Figura 22 - Comparação entre abordagem nativa e abordagem multiplataforma retirado de (AllianceTek Inc, 2016)

3.6. Tecnologias de Desenvolvimento

Esta secção será apresenta a análise sobre as tecnologias (*frameworks*) de desenvolvimento de multiplataforma, que inicia com introdução do padrão ***Model-View-Controller*** (MVC) que é utilizado pela maioria das *frameworks* multiplataformas. Segue-se então a apresentação propriamente dita das *frameworks* estudadas. É ainda apresentada a análise dos critérios e requisitos de seleção de uma determinada *framework*. Para terminar, é apresentada a arquitetura geral do desenvolvimento de aplicações multiplataformas e uma breve contextualização sobre os *Web Services*.

3.6.1. Padrão Model-View-Controller

O padrão ***Model View Controller*** mais conhecido por **MVC**, muito utilizado no desenvolvimento de aplicações WEB, foi introduzido para gerir o aumento da complexidade destas aplicações e preencher a lacuna que entre o modelo mental do utilizador e o modelo digital existente no computador. O padrão **MVC** começou a ser desenvolvido em meados de 1970, por Trygve Reenskuag (Pop & Altar, 2014).

O padrão **MVC** foi criado com o intuído de reduzir o custo e melhorar qualidade do *software* no paradigma orientado a objetos. O padrão **MVC** melhora a modularidade ao encapsular detalhes voláteis de implementação através das interfaces estáveis que reduzem o esforço necessário para compreender e manter o *software* existente. Este padrão tem desempenhado um papel fundamental na maioria das *User Interface Framework* (*framework* de desenvolvimento de interface do utilizador). O padrão **MVC** promove a separação da lógica da aplicação e da apresentação, que consiste em decompor a aplicação em três componentes que comunicam entre si (*Model, View* e *Controller*) (Anzures-Garcia, Sanchez-Galvez, Hornos, & Paderewski-Rodriguez, 2016; Mahmoud & Maamar, 2006; Pop & Altar, 2014; Selfa, Carrillo, & Del Rocío Boone, 2006). Cada um dos componentes que constituem este padrão tem um conjunto de tarefas específicas a desempenhar. A Figura 23, ilustra a estrutura do padrão **MVC**, bem como a interação entre os componentes do mesmo.

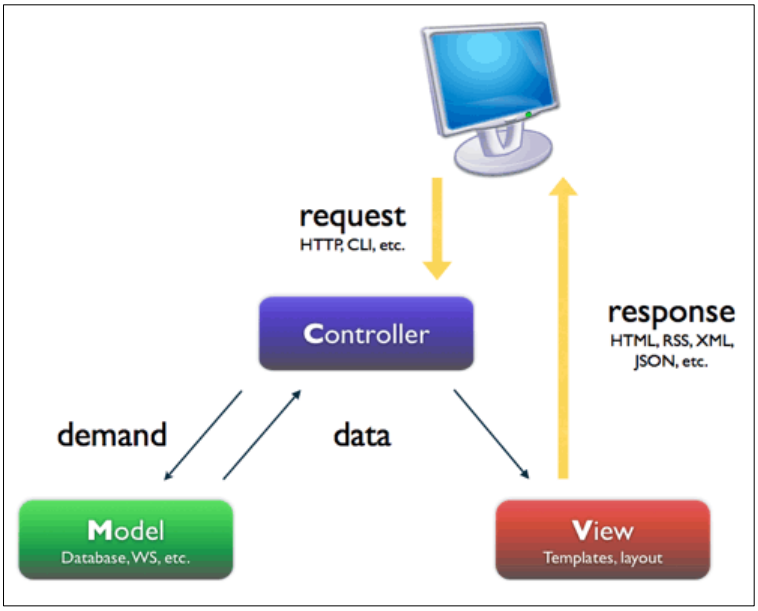


Figura 23 - Arquitetura e interações do MVC retirado de (Pop & Altar, 2014)

- ***Model*** **-** efetuar a gestão de todas as tarefas relacionadas com os dados. Esta camada é responsável pela lógica de negócio da aplicação. Ele responde aos pedidos de informações do seu estado (normalmente efetuado pela *View*) e responde às instruções para alterações do estado (normalmente efetuado pelo *Controller*);
- ***View*** **-** gere a saída gráfica e reflete o estado do *Model* e o *feedback* do utilizador**;**

- ***Controller*** **-** interpreta as entradas do utilizador (a partir do teclado, rato ou gestos), e seguidamente envia instruções ao *Model* ou *View* para executar determinadas ações baseadas nas entradas recebidas (Anzures-Garcia et al., 2016; Mahmoud & Maamar, 2006; Pop & Altar, 2014; Selfa et al., 2006).

Por outras palavras, o *model* representa os dados no sistema, o *view* é alguma forma de visualização do estado do *model* (é possível que um *model* tenha vários *views*), e o *controller* oferece as facilidades (a partir dos *event handlers*) na alteração do estado do *model* (Mahmoud & Maamar, 2006). É importante notar que tanto o *view* como o *controller* dependem do *model*, mas, no entanto, o *model* não depende nem do *view* e nem do *controller*. Este é um dos principais benefícios desta separação, pois possibilita desenvolver aplicações de forma distribuída por meio do *model, views* e *controller,* separando uma parte da outra, ou seja, este padrão tira a vantagem da possibilidade de modificar cada componente de forma totalmente independente. Esta separação ainda permite que o *model* seja construído e testado independentemente da apresentação visual da *view* (Masoud, Halabi, & Halabi, 2006). Outras vantagens trazidas pelo padrão **MVC** está na separação entre a apresentação e a lógica da aplicação, a fácil manutenção do sistema, a reutilização do código, a alteração na *view* sem afetar o *controller*, entre outras. Ainda outro benefício apresentado por este padrão é a fácil migração de programa herdados, devido ao fato das *views* serem separadas do *model* e do *controller.* Este padrão ainda fornece um ambiente que incorpora diferentes tecnologias em diferentes localizações e uma arquitetura com melhor suporte a escalabilidade (Palmieri, Singh, & Cicchetti, 2012).

3.6.2. Estudo das *Frameworks* Multiplataforma

Para o estudo das tecnologias de desenvolvimento multiplataforma foi escolhido um conjunto de *frameworks* de desenvolvimento diferentes que seguem a abordagem multiplataforma diferentes. Os critérios que foram utilizados para a escolha das ferramentas de desenvolvimento multiplataforma são:

- Suporte aos principais sistemas operativos com mais cota do mercado atualmente (Figura 4 e Figura 5);
- Licença gratuita – possibilita o desenvolvimento e a distribuição de aplicações multiplataformas gratuitamente, sem quaisquer custos adicionais.

As ferramentas de desenvolvimento escolhidos são: ***Appcelerator Titanium, PhoneGap, Rhodes, Xamarin, AngularJS (Mobile Angular UI), Bootstrap, jQuery Mobile UI e Sencha Touch***. Os quatros últimos são ferramentas de desenvolvimento WEB, que geralmente são designados por ***UI-frameworks***.

O estudo do mercado efetuado pela ***research2guidance*** (Tool, Tool, & Enterprise, 2014), concluiu que existem diversas ferramentas de desenvolvimento multiplataformas disponíveis, mas apenas alguns são mais utilizados. A Figura 24, ilustra as ações do mercado (*Market Shares*) dessas ferramentas. Como pode se constatar algumas das ferramentas escolhidas, a sua taxa de utilização cresceu comparativamente com o ano anterior (2013).

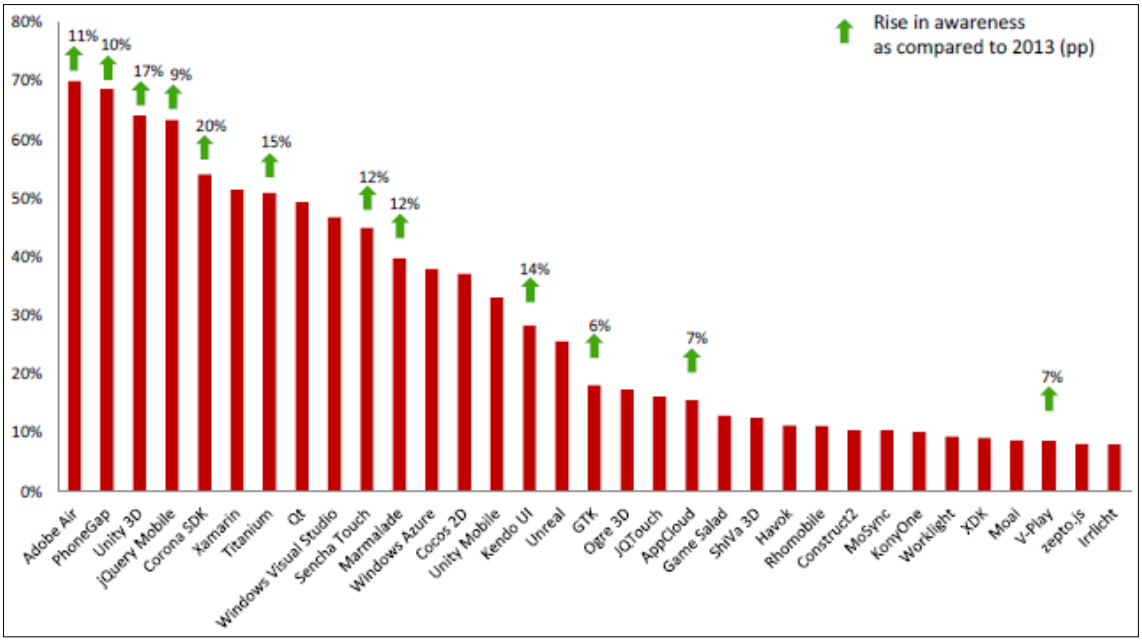


Figura 24 - Taxa de utilização das ferramentas de desenvolvimento multiplataforma em 2014 retirado de (Tool et al., 2014)

a) Appcelerator Titanium

Appcelerator Titanium é uma *framework open source* desenvolvida pela Appcelerator Inc. sob a licença do Apache 2.0, para o desenvolvimento de aplicações multiplataformas recorrendo à linguagem de programação JavaScript (Sommer & Krusche, 2013). Esta *framework* suporta as principais plataformas móveis como o *iOS*, *Android*, *BlackBerry* e o *Windows Phone*. O Titanium liga o JavaScript às bibliotecas nativas, compilando-o em *bytecode* e de seguida o SDK da

plataforma (*Android* ou *iOS*) constrói o *package* para a plataforma de destino (Ribeiro & da Silva, 2012).

O IDE oficial do Titanium é o Titanium Studio, baseado na plataforma do eclipse, compatível com os principais sistemas o Windows, Mac OS e Linux. O Titanium utiliza o padrão MVC para desenvolver aplicações móveis. Este padrão possibilita a criação de aplicações móveis distributivamente, a partir das definições de dados (*models*), da lógica do negócio (*controllers*) e das interfaces (*views*), oferecendo ao mesmo tempo um ponto de ligação entre estes elementos. No *model* a *framework* utiliza o JavaScript *backbone.js*, o *view* utiliza o XML e TSS e o *controller* utiliza o JavaScript simples (Ribeiro & da Silva, 2012).

A arquitetura do Titanium é composta por duas partes, sendo que a primeira as aplicações que encontram-se "agrupadas" juntamente com um interpretador JavaScript para executar o código da aplicação, e segunda a biblioteca do Titanium que oferece as APIs de acesso para aceder às funcionalidades e recursos do dispositivo, como sensores, acesso ao sistema de ficheiros, componentes da interface nativa, entre outras (Sommer & Krusche, 2013). A Figura 25 ilustra a arquitetura completa do Appcelerator Titanium.

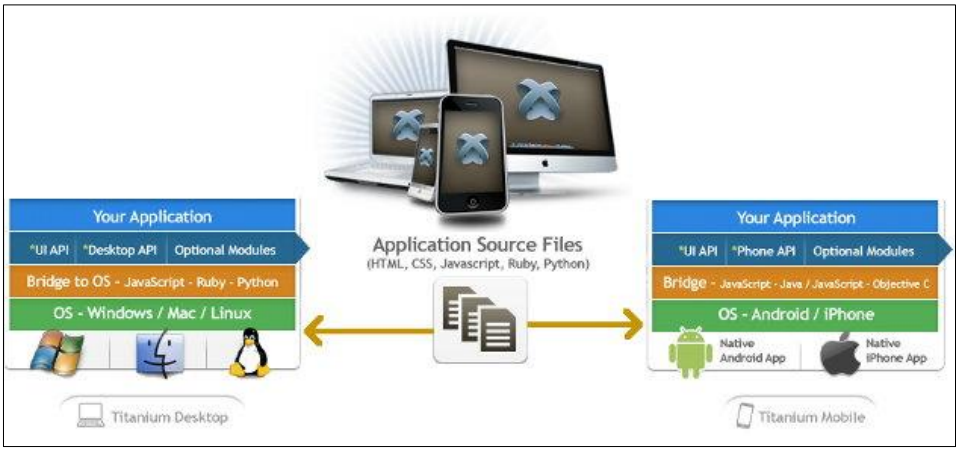


Figura 25 - Arquitetura do Appcelerator Titanium retirado de (Appcelerator Inc., 2016)

b) PhoneGap

O PhoneGap é uma ferramenta de desenvolvimento multiplataforma de aplicações móveis *open source* desenvolvido pela Adobe Systems Inc. sob a licença do Apache 2.0. Esta ferramenta de desenvolvimento multiplataforma possibilita aos programadores desenvolvam aplicações comerciais e *open source,* mas também lhes dá a possibilidade de usar qualquer outra combinação de licença. O ambiente de desenvolvimento multiplataforma desta ferramenta, permite desenvolver aplicações compatíveis para o *Android*, *Bada*, *BlackBerry*, *iOS*, *Symbian*, *WebOS* e *Windows Phone* (Adobe Systems Inc, 2016; Palmieri et al., 2012; Sommer & Krusche, 2013).

O PhoneGap é uma solução muito útil, para desenvolver aplicações móveis recorrendo a linguagens de programação WEB modernas, tais como HTML5, CSS3 e JavaScript em vez de utilizar linguagens de programação menos conhecidos como o *Objective-C*, ou outras linguagens. O PhoneGap tem o benefício de trazer muitas vantagens para os programadores qualificados, especialmente para atrair os programadores WEB (Palmieri et al., 2012; Ribeiro & da Silva, 2012). A Figura 26, ilustra o mecanismo de funcionamento do PhoneGap, e o motivo de a ferramenta é designado de "empacotador".

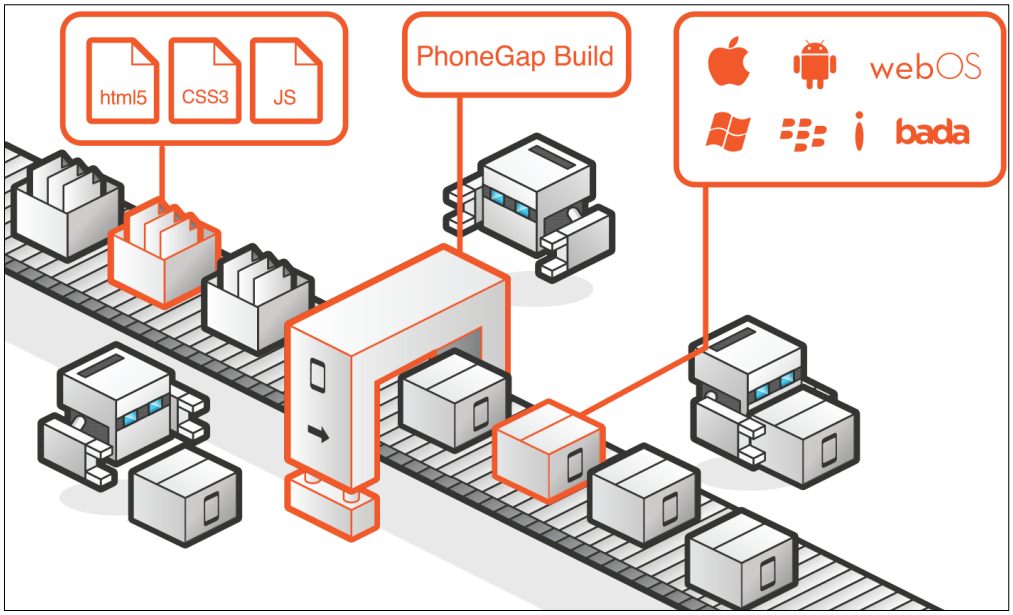


Figura 26 - Build PhoneGap Application retirado de (Adobe Systems Inc, 2016)

O PhoneGap é essencialmente um "empacotador" pois permite incluir aplicações escritas em linguagens de programação conhecidas, em aplicações nativas. As aplicações concebidas com recurso ao PhoneGap são aplicações híbridas, o que significa que não são puramente nativas e nem puramente WEB. O fato de não ser puramente nativo vem do fato do *layout* da aplicação ser realizada via *web-view* em vez da linguagem nativa do SO. Por outro lado, a falta de suporte HTML em algumas funções, faz com que a aplicação não seja puramente baseada na WEB (Palmieri et al., 2012; Ribeiro & da Silva, 2012).

O PhoneGap não tem uma IDE para o desenvolvimento de aplicações móveis, mas os programadores podem escrever o código fonte a partir de outras IDE's (Eclipse para *Android*, Xcode para *iOS*, entre outros) ou então a partir de Editor de Texto (*Sublime Text, Notepad++*, entre outros) e futuramente compilar o código fonte a partir do comando PhoneGap (***phonegap build***). Esta abordagem não permite que os programadores tenham um ambiente de desenvolvimento centralizado, pois o esforço necessário para compilar código fonte e produzir a aplicação executável é alta (Palmieri et al., 2012; Ribeiro & da Silva, 2012). O PhoneGap apresenta uma arquitetura constituída por três camadas: ***Web Application***, ***PhoneGap*** e ***OS and Native APIs.*** A Figura 27 ilustra as camadas da arquitetura do PhoneGap.

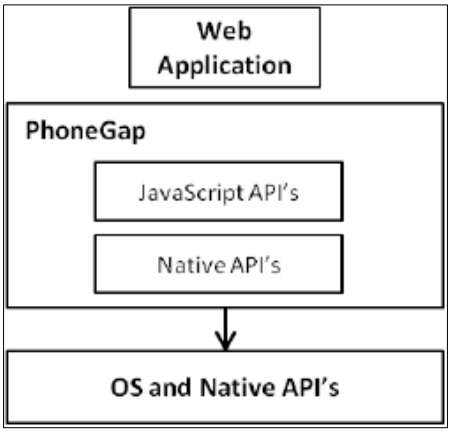


Figura 27 - Camadas da arquitetura do PhoneGap retirado de (Palmieri et al., 2012)

A camada exterior em cima (*Web Application*) da arquitetura do PhoneGap representa o código fonte da aplicação. A camada interior é composta por JavaScript e APIs nativas, sendo esta camada o responsável pela interface entre as camadas *Web Application* e o PhoneGap**.** Para além disso,

esta camada cuida da interface entre as APIs JavaScript que são utilizadas pelas aplicações e as APIs nativas de cada sistema operativo móvel. O PhoneGap oferece APIs JavaScript aos programadores para que estes tenham o acesso às funcionalidades e recursos dos dispositivos móveis, tais como: acelerómetro, código de barras. *Bluetooth*, calendário, camera, contatos, conexão, ficheiros, GPS, menu, NFC, entre outros. Para uma melhor perceção da arquitetura do PhoneGap, a IBM apresentou a arquitetura das camadas do PhoneGap detalhadamente, conforme ilustra a Figura 28 (Palmieri et al., 2012).

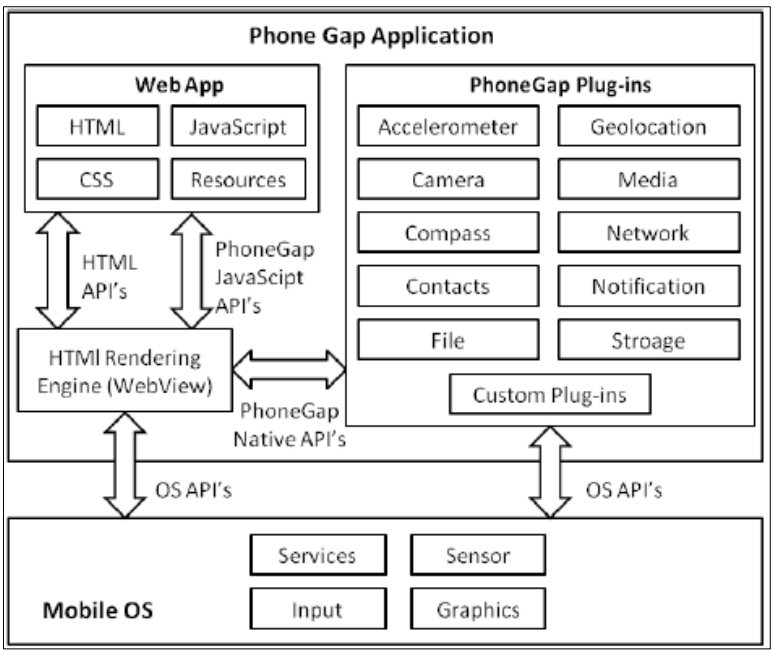


Figura 28 - Arquitetura completa do PhoneGap retirado de (Palmieri et al., 2012)

A figura anterior apresenta detalhadamente todas as interações entre as diversas partes que constituem a arquitetura do PhoneGap.

c) Rhodes

Rhodes é uma *framework* de desenvolvimento multiplataforma de aplicações móveis, desenvolvida pela RhoMobile que foi adquirido pela Motorola Solutions Inc. em Massachusetts Institute of Technology (MIT) em outubro de 2011. Foi desenvolvido para construir aplicações nativas para os principais sistemas operativos móveis (*Android*, *iOS*, *BlackBerry*, *Windows Phone*, *Symbian*). O principal objetivo da *framework* Rhodes é facultar um alto nível de produtividade e portabilidade no desenvolvimento de aplicações. Rhodes é uma *framework* de desenvolvimento *open source* baseado na linguagem de programação Ruby (Palmieri et al., 2012; Sommer & Krusche, 2013).

A suite RhoMobile tem uma IDE designado RhoStudio que é uma solução inovadora, dedicada aos programadores que pretendem desenvolver aplicações móveis recorrendo a uma IDE padrão. É uma solução que pode ser utilizada em diferentes sistemas operativos, tais como Linux, Mac OS e Windows. Alternativamente, o RhoMobile oferece a possibilidade de escrever o código fonte em qualquer outro IDE ou editor de texto, que tenha suporte ao HTML, CSS, JavaScript e Ruby. Os editores mais conhecidos são Eclipse, Visual Studio, NetBeans, IntelliJ e TextMate (Dalmasso, Datta, Bonnet, & Nikaein, 2013; Palmieri et al., 2012).

A *framework* Rhodes possibilita o desenvolvimento de aplicações móveis utilizando o padrão MVC. Na arquitetura, a lógica do negócio (*Controllers*) e as definições dos dados (*Models*) são escritas em Ruby e são separadas da interface visual (*Views*) que são escritas em HTML, CSS e JavaScript. O Rhodes oferece o acesso a recursos de dispositivos nativos através de um conjunto de APIs do Ruby (Ribeiro & da Silva, 2012). O Rhodes tem apenas três possibilidades para adicionar extensibilidade (*Plugins*) à *framework*, o primeiro pode ser adicionado às bibliotecas externas do Ruby ao Rhodes, o segundo pode ser adicionado criando extensões nativas para um determinado SDK de SO específico e o último, pode-se fazer uma extensão das *views* já existentes disponíveis no Rhodes (Palmieri et al., 2012).

Os componentes *Controllers,* HTML *Template* e *Source adapter* são as partes que os programadores das aplicações precisam de implementar para desenvolver as aplicações. Visto que os outros componentes são todos fornecidos pelo Rhodes, tais como o *Rhodes App Generator* que é uma IDE que pode ser o RhoStudio ou então outro editor, o *Ruby Executor* que executa o

código Ruby, o *Device Capabilities* que são as APIs, o *Rhom* que é a mini base de dados *Object Relational Mapper (ORM),* o *RhoSync Client* que é uma biblioteca para adicionar capacidade de sincronização de dados para as aplicações, e o *RhoSync* para simplificar o desenvolvimento da ligação para as aplicações empresariais de *back-end* (Palmieri et al., 2012). A Figura 29 ilustra a arquitetura do Rhodes.

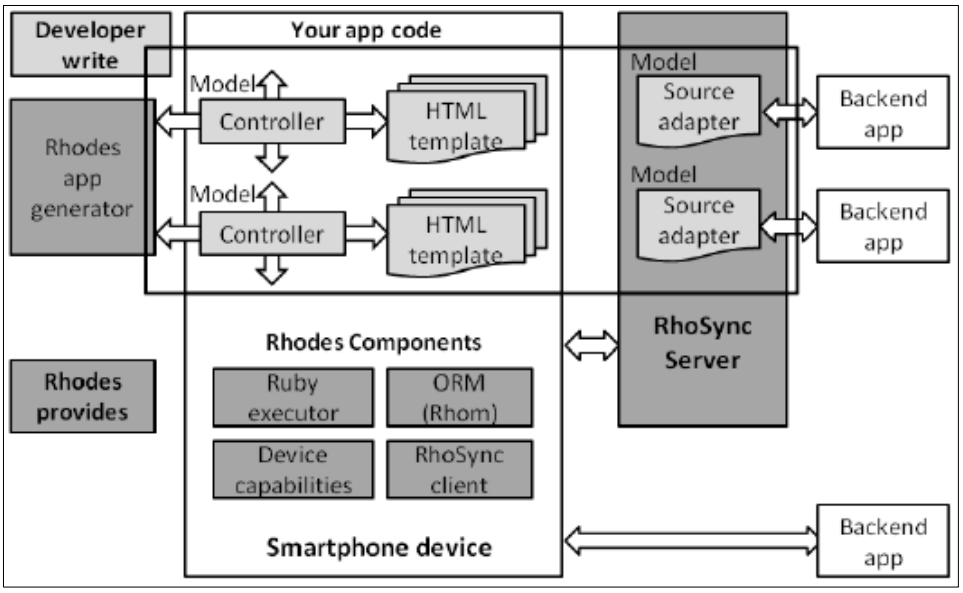


Figura 29 - Interface da arquitetura do Rhodes entre Smartphone e componentes do Rhodes retirado de (Palmieri et al., 2012)

d) Xamarin

O Xamarin é uma *framework* de desenvolvimento de aplicações móveis multiplataforma que fornece um ambiente unificado de desenvolvimento aos programadores. Os fatores que diferenciam esta *framework* das outras *frameworks* multiplataforma são: a manipulação dos recursos de *hardware*, o conceito de partilha de código fonte e ligações nativas. Estes fatores proporcionam bons resultados de desempenho e da experiência do utilizador nativo (Fh & Haberl, 2015).Esta *framework* apresenta uma abordagem descrita como:

"*building cross-platform native apps combines the essential characteristics of native apps native UI, native performance and native device access-with the efficiency and time-to-market advantage of code sharing*".

A citação afirma que o desenvolvimento de aplicações nativas recorrendo a ambientes multiplataformas combina as características essenciais das aplicações nativas tais como a interface de utilizador nativa, o desempenho nativo e o acesso aos recursos nativos dos dispositivos com eficiência e tira a vantagem na partilha de código fonte.

Geralmente, existem duas abordagens que podem ser utilizadas para o desenvolvimento de aplicações móveis com a ferramenta Xamarin. Estas abordagens são: *Xamarin.iOS* / *Xamarin.Android* e ainda por outro lado existe a possibilidade da utilização do *Xamarin.Forms* (Fh & Haberl, 2015).

A linguagem de desenvolvimento que o Xamarin utiliza é o C#, que é uma linguagem de programação multi-paradigama. O código C# é tradicionalmente imperativo, mas pode ser "aromatizado" com paradigmas de programação declarativa ou funcional. O C# é uma linguagem imperativa orientada a objetos fortemente tipificada que foi influenciada pelo C++ e Java (Fh & Haberl, 2015).

Inicialmente o foco do Xamarin, foi em três conjuntos básicos de bibliotecas .NET que são:

- ***Xamarin.Mac*** mais conhecido por *MonoMac*;
- ***Xamrin.iOS*** mais conhecido por *MonoTouch*;
- ***Xamarin.Android*** mais conhecido por Mono for *Android*.

Estas três bibliotecas formam a base da plataforma Xamarin. Basicamente eles são "empacotadores" do .NET do Mac nativo, *iOS* e APIs do *Android*. Estas bibliotecas permitem que os programadores utilizam o C# para desenvolver as suas aplicações móveis, e direcionar as APIs nativas dessas três plataformas com o benefício de ter acesso a todas as bibliotecas da classe da estrutura .NET (Fh & Haberl, 2015). A Figura 30, ilustra a arquitetura da *framework* Xamarin.

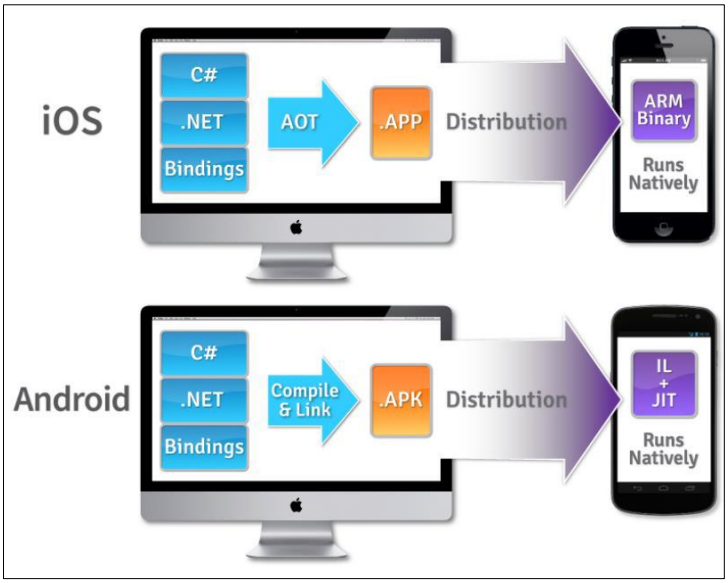


Figura 30 - Arquitetura do Xamarin retirado de (Fh & Haberl, 2015)

Para oferecer uma ligação direta e idêntica entre a plataforma do programador Xamarin e as APIs nativas, foi criada uma ligação nativa que permite aceder diretamente o *iOS* ou o SDK do *Android*. Isso significa que todos os objetos nativos podem ser utilizados com os mesmos nomes, as mesmas propriedades, atributos e métodos. O Xamarin não reimplementa o código em C# o que permite a *framework* lançar novas atualizações à mesma data para os principais sistemas operativos móveis (Fh & Haberl, 2015).

e) AngularJS (Mobile Angular UI)

AngularJS é uma *framework open source* mantida pela Google para o desenvolvimento de aplicações móveis e *desktop* na WEB. AngularJS é uma *framework* que utiliza o padrão MVC em JavaScript, cujo principal objetivo é a simplificação do desenvolvimento de aplicações WEB. É uma *framework* de desenvolvimento transversal a todas as plataformas e múltiplos dispositivos (Google, 2015; Tutorialspoint, 2016).

De certa forma, o AngularJS possibilita a utilização do padrão MVC no lado do cliente recorrendo às tecnologias *Ajax* para as suas requisições, e organiza as aplicações com o padrão MVC não apenas no lado do servidor, mas também no lado do cliente, ou seja, devido a alta complexidade de certas aplicações e do dinamismo das páginas HTML, o AngularJS faz com que parte da lógica seja "empurrada" para o cliente através do JavaScript (Google, 2015; Tutorialspoint, 2016).

Geralmente estes tipos de aplicações necessitam de uma maior organização do código no *front-end* e o AngularJS consegue realizar essa tarefa muito bem permitindo que o código seja divido em várias partes e separa as suas responsabilidades. Este ainda dispõe do mecanismo *Two-Way Data Binding* que incentiva a utilização do *Dependency Injection*. AngularJS é composto por Módulos, Diretivas, Controladores, Serviços e Rotas (Google, 2015; Tutorialspoint, 2016).

f) Bootstrap

O Bootstrap é a biblioteca HTML, CSS e JavaScript mais conhecida no mundo para o desenvolvimento de aplicações responsivos e aplicações móveis na WEB. O bootstrap tornou o desenvolvimento Web *front-end* mais rápido e mais fácil, pois ele foi desenvolvido pensando em pessoas com diferentes níveis (*skill level*) de desenvolvimento de aplicações, para todos os dispositivos, para todas as formas e para todos os tamanhos. Atualmente o bootstrap já vai na sua terceira versão, mas brevemente será lançado a quarta versão do bootstrap, que visa trazer novas melhorias e novas funcionalidades. Esta biblioteca é uma biblioteca essencialmente de desenvolvimento WEB, ela é compatível com os principais *browsers* (Chrome, Mozilla, Safari, Edge e IE) e com os principais sistemas operativos (*Android*, *iOS*, *Windows Phone*) (Bootstrap, 2016; Tutorialspoint, 2014).

g) jQuery Mobile UI

O jQuery Mobile UI é uma *framework* ***User Interface*** (***UI***), para o desenvolvimento de páginas Web responsivos para os dispositivos móveis, totalmente baseado no HTML5. Este para além de apresentar os componentes da UI, oferece as animações e o suporte ao JavaScript para *touch events.* Esta *framework* é compatível com os principais sistemas operativos móveis, tais como: *Android*, *iOS*, *BlackBerry* e *Windows Phone* (Heitkötter, Majchrzak, Ruland, & Weber, 2013). O jQuery Mobile UI inclui o sistema de *layouts* (*list, detail panes, overlays*) e um conjunto de controlos dos formulários e *widgets* da interface do utilizador *(toggles, sliders, tabs*) (Smutny, 2012).

h) Sencha Touch

O Sencha Touch é uma *framework* com suporte a MVC *open source* desenvolvido pela Sencha Inc. em junho de 2010. O Sencha Touch é uma *framework* ***User Interface (UI)*** baseada em HTML5, CSS3 e JavaScript para o desenvolvimento de aplicações multiplataformas (Sencha Inc., 2016). É uma *framework* de desenvolvimento de aplicações móveis de alto desempenho, compatível com *Android*, *iOS*, *BlackBerry*, *Windows Phone* e *Tizen* (Fh & Haberl, 2015; Sencha Inc., 2016; Smutny, 2012; Sommer & Krusche, 2013).

3.6.3. Critérios e Requisitos de Seleção

Esta subsecção apresenta as principais razões que estão na base da escolha das diferentes tecnologias de desenvolvimento, bem como os critérios que geralmente são utilizados para a escolha da tecnologia de desenvolvimento, mais adequada para atingir determinadas finalidades (Palmieri et al., 2012).

- **Sistema operativo móvel** suportado para perceber os respetivos efeitos no modelo do negócio;
- **Licenças das tecnologias** oferecidas para avaliar os termos e condições de utilização;
- **Linguagens de programação** oferecidas aos programadores para desenvolver as aplicações;
- **Disponibilidade dos APIs** oferecidas com o objetivo de obter uma ideia das diferentes partes do *hardware* acessíveis a partir do SO;
- **Acessibilidade para as APIs nativas** para comparar de que forma é possível acedê-los a partir de cada *framework*;
- **Desempenho (Utilização do CPU, da memória e consumo de energia)** quando comparado os resultados dos testes de desempenho de cada aplicação desenvolvida numa determinada *framework;*
- **Arquitetura** oferecida para o processo de desenvolvimento da aplicação;
- **Ambiente de desenvolvimento integrado** disponíveis para o desenvolvimento das aplicações.

Segundo o relatório apresentado pela ***research2guidance*** (Tool et al., 2014)**,** de um estudo que eles efetuaram, os programadores apontaram uma lista com os principais critérios pela ordem de importância que normalmente devem ser levado em consideração no momento da escolha de uma *framework* para o desenvolvimento de aplicações móveis multiplataforma. A Figura 31, ilustra então na ótica dos programadores, a lista destes critérios, bem como o grau de importância de cada um dos critérios pela ordem decrescente das mesmas (do mais relevante para o menos relevante).

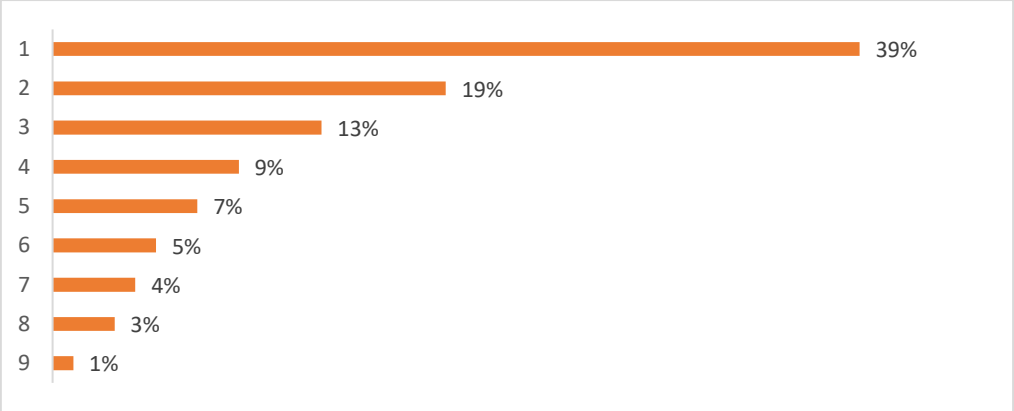


Figura 31 - Critérios a ser considerados na escolha da framework adaptado de (Tool et al., 2014)

Legenda:

1- Saída da ferramenta (tipo de aplicação);

2- Custo de ferramenta (Licença);

3- Qualidade da referência das aplicações;

4- Número de plataformas suportadas pela ferramenta;

5- Complexidade da ferramenta;

6- Ajuste para o foco da categoria indústria/aplicação;

7- Extensão das funcionalidades;

8- Qualidade dos serviços de apoio;

9- Extensão do suporte do fornecedor para distribuição e *marketing* de aplicativos.

a) APIs nativas disponíveis

As características das *frameworks,* tais como as linguagens de programação suportadas, o acesso às APIs nativas, o IDE e os *plugins* suportados são fatores que pesam no momento da escolha de uma *framework* em vez de outra *framework*. O acesso às APIs é um dos fatores mais importante, pois é a forma que possibilita ao programador ter o acesso aos recursos nativos do sistema operativo móvel para o qual pretende desenvolver a aplicação móvel. A Tabela 14, apresenta a comparação entre as diferentes *frameworks* analisadas, tendo em conta os aspetos anteriormente referenciados.

Tabela 14 - Comparação das características das frameworks adaptado de (Falk Markus, 2016; Palmieri et al., 2012; Ribeiro & da Silva, 2012; Serrano et al., 2013)

Framework	**Linguagem de programação**	**Acesso às APIs nativas**	**IDE**	***Plugin***
AngularJS	JavaScript	JavaScript	*	Sim
Appcelerator Titanium	JavaScript	JavaScript	Titanium Studio	Sim
PhoneGap	HTML, HTML5, CSS, CSS3, JavaScript	JavaScript	IDE nativo do Sistema Operativo (Eclipse, Xcode)	Sim
Rhodes	HTML, HTML5, CSS, JavaScript, Ruby	JavaScript	RhoStudio, RhoHub *	Sim
Xamarin	C#	C#	Xamarin for Visual Studio	Sim
jQuery Mobile	HTML, HTML5, CSS, CSS3, JavaScript	N. A	*	Sim
Sencha Touch	HTML, HTML5, CSS, CSS3, JavaScript	JavaScript*	Sencha Touch Architecture *	Sim

*outras IDEs alternativas como por exemplo, Eclipse, Visual Studio, NetBeans, IntelliJ, TextMate, Sublime, NotePad++, etc.

A Tabela 15, apresenta a comparação entres as *frameworks* analisadas e as APIs nativas dos SO. De acordo com a tabela, pode se ver quais são os APIs que cada uma das *frameworks* podem aceder.

Tabela 15 - Comparação das APIs suportadas pelas frameworks adaptado de (Falk Markus, 2016; Palmieri et al., 2012; Ribeiro & da Silva, 2012; Serrano et al., 2013)

APIs nativas	**AngularJS**	**Bootstrap**	**Appcelerator Titanium**	**PhoneGap**	**Rhodes**	**Xamarin**	**jQuery Mobile**	**Sencha Touch**
Acelerómetro	-	-	✓	✓	✓	✓	-	✓
Armazenamento	-	-	✓	✓	✓	✓	-	-
Bluetooth	-	-	-	-	✓	✓	-	-
Calendário	-	-	✓	✓	✓	✓	-	-
Camera	-	-	✓	✓	✓	✓	-	-
Captura	-	-	✓	✓	✓	✓	-	-
Conexão	-	-	✓	✓	✓	✓	-	-
Contacto	-	-	✓	✓	✓	✓	-	-
Dispositivo	-	-	✓	✓	✓	✓	-	-
Eventos Nativos	-	-		✓	✓	✓	-	-
Ficheiro	-	-	✓	✓	✓	✓	-	-
Gestores Multitouch	-	-	✓	✓	✓	✓	-	✓
Geolocalização	-	-	✓	✓	✓	✓	-	✓
Menu	-	-	✓	-	-	-	-	-
Mensagem e Telefone	-	-	-	-	✓	✓	-	-
NFC	-	-	✓	✓	✓	✓	-	-
Notificação	-	-	✓	✓	✓	✓	-	-
Rotação de Ecrã	-	-	✓	✓	✓	✓	-	-
Vibração	-	-	✓	✓	✓	✓	-	-

OBS: Nem todas as plataformas móveis tem suporte aos APIs acima indicados suportados pelas *frameworks* de desenvolvimento. Algumas das *frameworks* acima apresentados não têm suporte

direto para as APIs nativas, pois são *frameworks* WEB essencialmente do UI, pelo que estes tiram partido das APIs nativas através da combinação com outras *frameworks* de desenvolvimento multiplataforma.

b) Compatibilidade com as plataformas móveis

As plataformas móveis são os sistemas operativos que vão executar as aplicações desenvolvidas pelas *frameworks*. A análise a seguir apresentada na Tabela 16, ilustra a compatibilidade de cada *framework* com os respetivos sistemas operativos móveis e sistemas operativos *desktop*.

Tabela 16 - Compatibilidade das frameworks com as plataformas móveis adaptado de (Falk Markus, 2016; Palmieri et al., 2012; Ribeiro & da Silva, 2012; Serrano et al., 2013)

Framework	Suporte OS Móvel	Suporte OS
AngularJS	Android, iOS, Windows Phone	Linux, Mac e Windows
Appcelerator Titanium	Android, BlackBerry, iOS, Windows Phone	Linux, Mac e Windows
Bootstrap	Android, BlackBerry, iOS, Windows Phone	Linux, Mac e Windows
PhoneGap	Android, BlackBerry, iOS, Symbian, WebOS, Windows Phone	Linux, Mac e Windows
Rhodes	Android, BlackBerry, iOS, Symbian, Windows Phone	Linux, Mac e Windows
Xamarin	Android, iOS, Watch OS, Windows Phone	Linux, Mac e Windows
jQuery Mobile	Android, BlackBerry, iOS, Windows Phone	Linux, Mac e Windows
Sencha Touch	Android, BlackBerry, iOS, Windows Phone	Linux, Mac e Windows

c) Licenças

A licença é um outro parâmetro útil para a comparação. Tal como apresentado na Tabela 17, as licenças disponíveis para estas *frameworks* são MIT, Apache 2.0, GNU *General Public License* 2 (GPL2) e comercial. As duas primeiras licenças são gratuitas, mas todas têm suporte *open-source*. Este parâmetro é muito importante para os programadores que desenvolvem aplicações móveis e que pretendem dar suporte à plataforma de desenvolvimento sem restrição comercial.

Tabela 17 - Comparação das características gerais das frameworks adaptado de (Falk Markus, 2016; Palmieri et al., 2012; Serrano et al., 2013)

Frameworks	**Licença**	***Open-Source***	**Abordagem**	**MVC**
AngularJS	MIT	Sim	WEB	Sim
Appcelerator Titanium	Apache 2.0, comercial	Sim	Interpretada	Sim
Bootstrap	MIT	Sim	WEB	Não
PhoneGap	Apache 2.0	Sim	Híbrida	Não
Rhodes	MIT, comercial	Sim	Compilação-Cruzada	Sim
Xamarin	Comercial *	Não	Compilação-Cruzada	Não
jQuery Mobile	GPL v3	Sim	WEB	Não
Sencha Touch	GPL/MIT	Sim	WEB	Sim

*versão universitário

A Tabela 18, apresenta de forma sucinta um resumo de todas as características anteriormente apresentadas que devem ser considerados quando pretende-se efetuar a escolha de uma solução. O objetivo desta tabela é construir um *ranking* comparativo das *frameworks* com o intuído de identificar qual delas aparentam ter a melhores classificações.

Tabela 18 - Ranking comparativo das frameworks

Frameworks	API's Nativos	Plataformas	Target	Interface Utilizador	Documentação	Total
AngularJS (Mobile UI)	-	4.5	4	4.5	4	17
Appcelerator Titanium	4.5	2	2	1	4	13.5
Bootstrap	-	4.5	4	4.5	4	17
PhoneGap	4.5	4	3	2	4	17.5
Rhodes	4.5	3	3	2	4	16.5
Xamarin	4.5	2	3	4	4	17.5
jQuery Mobile UI	-	4.5	4	4.5	4	17
Sencha Touch	1	4.5	4	4.5	4	18

Escala {1- Muito Pouco, 2- Pouco, 3- Razoável, 4- Bom, 5-Muito Bom}

d) Desempenho

O desempenho das aplicações é outro parâmetro importante a ser considerado quando se pretende desenvolver uma aplicação móvel. Por isso, torna-se necessário analisar quais *frameworks* e que combinações são capazes de proporcionar melhores resultados a nível do desempenho das aplicações móveis. A avaliação de desempenho das aplicações foi efetuada com base nos seguintes parâmetros consumo do CPU, consumo da memória e consumo de energia. As aplicações analisadas foram desenvolvidas com recurso às seguintes *frameworks*:

- PhoneGap;
- PhoneGap + jQuery Mobile;
- PhoneGap + Sencha Touch;
- Titanium.

As aplicações desenvolvidas para avaliar o desempenho das *frameworks* tem as seguintes características JSON (Dalmasso et al., 2013):

- A interface do utilizador tem botões, na qual cada um executa um pedido. Cada pedido é efetuado a um *Web Services* (disponível na internet) a partir de tecnologias AJAX, REST e SOAP.
- Analisar e apresentar os diferentes formatos de respostas: *Text*, XML.

- **Consumo da CPU**

Para medir a utilização do CPU, as aplicações foram desenvolvidas com o PhoneGap e outras *frameworks.* Os segmentos do código para medir a utilização do CPU foram adicionados ao *Android* ***Activity*** através do PhoneGap. Nenhuma aplicação foi desenvolvida com a *framework* Titanium, uma vez que não permite adicionar o ***Activity***, mas podia ser desenvolvido um *plugin* para este efeito, mas tal seria um processo muito demorado (Dalmasso et al., 2013). Foram seguidas duas abordagens diferentes para registar a utilização do CPU:

- A primeira abordagem, efetua a captura instantânea do CPU em cada estado do ciclo de vida da ***Activity*** (por exemplo: *onCreate, onStart, onPause, onStop* e *onDestroy*) das aplicações;
- A outra abordagem é ler um resultado "top" a cada segundo durante todo o ciclo de vida das aplicações. Seguidamente, efetuou-se o cálculo da média para cada estado do ciclo de do ***Activity.***

A Tabela 19, ilustra os resultados obtidos da realização do teste da utilização do CPU (Dalmasso et al., 2013).

Tabela 19 - Métricas da utilização do CPU retirado de (Dalmasso et al., 2013)

Frameworks utilizadas	**Utilização do CPU a partir da abordagem da captura instantânea**	**Utilização da CPU a partir da abordagem do comando "top"**
PhoneGap + HTML + CSS	81.927771%	Max: 10% Min: 0% Average: 2%
PhoneGap + jQuery + HTML	80.26316%	Max: 42% Min: 0% Average: 10%
PhoneGap + Sencha Touch	44.0%	Max: 32% Min: 0% Average: 8%

Os valores obtidos da abordagem de captura instantânea foram calculados quando a aplicação estava a efetuar muita computação, por isso apresenta valores muito altos. Também é de notar que estes valores podem variar muito de um milésimo de segundo para outro, uma vez que são instantâneos na utilização do CPU por cada curto período de tempo.

Os valores obtidos a partir do resultado "top" apresentam a variação da utilização do CPU. O valor mínimo é sempre zero, uma vez que a aplicação obtém a página solicitada e apresenta-o, a aplicação não utiliza nenhum tempo de espera do CPU para a próxima entrada do sistema. O valor médio é calculado utilizando o tempo total decorrido. A partir desta abordagem, é claro que a primeira aplicação utiliza muito menos CPU, mas a interface do utilizador não é muito sofisticada (não é parecido com a UI nativa). Quando é utilizado o Sencha Touch com o PhoneGap, a utilização do CPU é maior, mas, no entanto, a UI é significativamente melhor (UI nativa simulada, ou seja, tem a UI parecida à nativa) (Dalmasso et al., 2013).

➢ Consumo da Memória

As informações da utilização da memória são obtidas através das ferramentas DDMS do ADT.

- ***Proporcional Set Size* (PSS)** – é a quantidade da memória partilhada com outros processos. Esta memória não será libertada caso o processo terminar, mas é indicativo da quantidade que este processo está a contribuir para a memória total carregada;
- ***Unique Set Size* (USS)** – é o conjunto de páginas que são exclusivas de um processo. Esta é a quantidade de memória que será libertada caso o a aplicação for encerrada (Dalmasso et al., 2013).

Tabela 20 - Métrica da utilização da memória retirado de (Dalmasso et al., 2013)

Frameworks utilizadas	PSS	USS
PhoneGap	12091	6036
PhoneGap + jQuery + HTML	14730	9424
PhoneGap + Sencha Touch	24526	20164
Titanium	17500	8676

De acordo com os resultados obtidos na Tabela 20 acima apresentado, consta-se que a aplicação desenvolvida com a *framework* PhoneGap tem menor valor de PSS e USS. Isto acontece devido ao fato do PhoneGap ser projetado para não utilizar nenhum elemento de estilo ou ferramenta para melhorar a UI.

Contrariamente, a integração do PhoneGap com o jQuery Mobile ou com o Sencha Touch corresponde a um ambiente completo para uma melhor UI no desenvolvimento de aplicações. Assim, neste caso, a memória utilizada aumenta com a utilização de ficheiros HTML e JavaScript. O Titanium dispõe de um SDK completo e a utilização da memória é portanto, elevada (Dalmasso et al., 2013)

- **Consumo de Energia**

O consumo de energia das aplicações móveis tem recebido muita atenção dos investigadores atualmente. Para utilizar eficazmente a bateria do dispositivo móvel, as aplicações desenvolvidas com as *frameworks* multiplataformas devem ser eficientes a nível do consumo da energia, i. e. gastar menos energia possível. A medição do consumo de energia das aplicações foi obtida a partir do "***Power Tutor***". O ***Power Tutor*** é uma aplicação *Android* muito popular que relata o consumo de energia de cada aplicação individualmente num dispositivo móvel. A Tabela 21, a seguir ilustra os resultados obtidos. É de notar que os valores correspondem a média do consumo de energia das aplicações (Dalmasso et al., 2013) .

Tabela 21 - Métrica do consumo de energia retirado de (Dalmasso et al., 2013)

Framework utilizado	**Consumo de energia (mW)**
PhoneGap + HTML + CSS	107
PhoneGap + jQuery + HTML	168
PhoneGap + Sencha Touch	120

Novamente, o resultado indica que a primeira aplicação consome menos energia entre as três aplicações desenvolvidas. Isto reside no fato da primeira aplicação ter uma UI muito simples. A aplicação desenvolvida com PhoneGap e Sencha Touch funciona de forma eficiente, mesmo consumindo 120mW de energia (Dalmasso et al., 2013).

Tendo em consideração a análise anteriormente apresentada sobre o desempenho das aplicações, facilmente conclui-se que a utilização de apenas uma *framework* pode se obter resultados razoáveis a nível do desempenho, mas peca-se quanto à aparência (interface do utilizador) da aplicação. Por forma a tentar conseguir melhores resultados pode se utilizar a combinação das *frameworks* para que seja possível melhorar a interface do utilizador e conseguir bons resultados quanto ao desempenho geral das mesmas.

Para terminar esta subsecção sobre a avaliação dos critérios e requisitos de seleção, a ferramenta de desenvolvimento multiplataforma escolhida para a realização da dissertação é o **PhoneGap**. Mas pelo facto, desta ferramenta possuir uma fraca classificação ao nível da interface de

utilizador, sentiu-se a necessidade de efetuar uma combinação do PhoneGap com outras ferramentas de desenvolvimento WEB, de modo a obter melhor aparência da interface do utilizador. De acordo com as análises efetuadas ao longo desta subseção, os candidatos com melhores classificações para esta combinação são **AngularJS (Mobile UI)** e o **Sencha Touch**. Ambas proporcionam um bom resultado a nível da interface do utilizador, e são compatíveis com o PhoneGap. Neste a escolha da utilização de uma ou outra *framework* do UI, vai depender muito do resultado esperado e dos objetivos que se pretende atingir dentro do tempo disponível.

3.6.4. Arquitetura Geral do Desenvolvimento de Aplicações Multiplataformas

Desenvolver uma aplicação móvel para ser executado em múltiplas plataformas móveis tais como *Android*, *BlackBerry*, *iOS*, *Windows Phone*, *Symbian*, entre outros costuma ser uma tarefa muito difícil, tendo em conta o número de implementações tecnológicas de cada plataforma móvel (Dalmasso et al., 2013). A Figura 32, ilustra a arquitetura geral do desenvolvimento de aplicações multiplataformas.

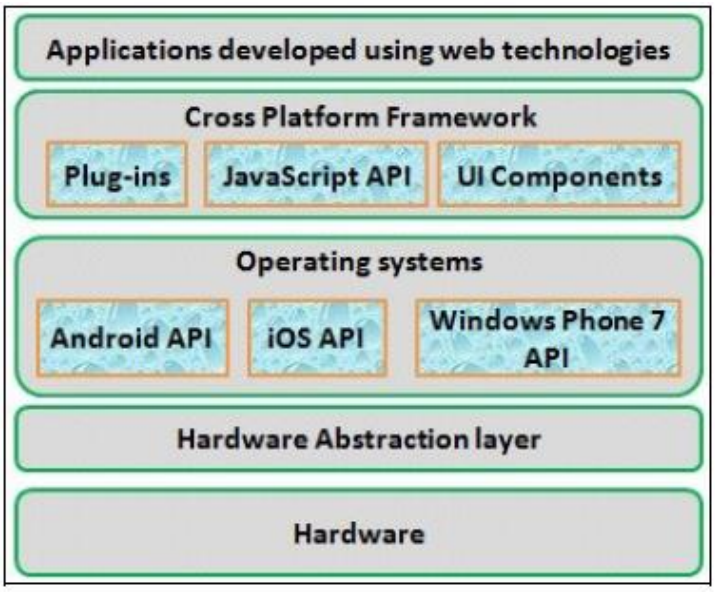


Figura 32 - Arquitetura geral do desenvolvimento de aplicações multiplataformas retirado de (Dalmasso et al., 2013)

O programador implementa a lógica de negócios ou as funcionalidades das aplicações através das tecnologias WEB. As *frameworks* multiplataformas permitem a implementação da interface do utilizador, o fácil acesso ao armazenamento e os recursos do dispositivo móvel (tais como: camera, contatos, sensores) que interagem com APIs JavaScript. Estas por sua vez, interagem com a API nativa das plataformas móveis (Dalmasso et al., 2013). As *frameworks* de desenvolvimento de aplicações multiplataformas vêm tornando-se cada vez mais sofisticados, de modo que também tem melhorado o desempenho das aplicações e ainda estão a efetuar esforços para conseguir uma UI muito semelhante às aplicações nativas.

3.6.5. Web Services

Os *Web Services* são aplicações WEB com a capacidade de interagir entre si, permitindo a automatização de tarefas que só podiam ser realizadas através da interação dos humanos, ou seja, é uma norma que define formas de interação aplicação-aplicação, recorrendo a formatos abertos. A utilização de protocolos normalizados, proporciona a capacidade de intercâmbio (troca) de informação entre aplicações em ambientes eminentemente heterogéneos (Araújo, 2005; Gottschalk, Graham, Kreger, & Snell, 2002).

Ao longo dos anos outras iniciativas têm foram desenvolvidas no âmbito de sistemas distribuídos, o que resultou em vários protocolos como *Distributed Component Model* (NCOM), *Common Object Request Broker* (CORBA) e *Java 2 Platform, Enterprise Edition* (J2EE). No entanto estas soluções são proprietárias, de difícil implementação e caras, o que, em ambientes heterogéneos como a Internet, as torna muito inadequadas. Os *Web Services*, por outro lado, aparecem como uma plataforma independente para suporte ao desenvolvimento de aplicações distribuídas sobre Internet, respondendo a todas estas questões (Araújo, 2005).

Sendo aplicações modulares, auto-descritivas, acessíveis através de um URL, independentes das plataformas de desenvolvimento e que permitem a interação entre aplicações sem intervenção humana, os *Web Services* apresentam .se como soluções para os atuais problemas de integração dos sistemas. Estas características devem-se em grande parte ao fato de se basearem em normas *standard*, entre as quais se destacam o **XML**, **SAOP**, **WSDL**, **UDDI, JSON** e **REST** (Araújo, 2005; Feij & Ramalho, 2004; Gottschalk et al., 2002; Ibm, 2005; Pautasso, 2009; Vinoski, 2008). Para

terminar, e não menos importante, é também apresentado uma breve descrição do que é um **API** (Fh & Haberl, 2015).

- ✓ **XML** (*eXtensible Markup Language*) – meta linguagem de anotação na qual estão todas as outras normas que servem de base aos *Web Services*;

- ✓ **SOAP** (*Simple Object Access Protocol*) – linguagem de anotação com a qual se pode descrever o protocolo de comunicação responsável pela troca de mensagens de e para os Web Services (uma mensagem SOAP é um documento XML);

- ✓ **WSDL** (*Web Service Definition Language*) – linguagem de anotação definida em XML e que tem como objetivo descrever a API de um Web Service;

- ✓ **UDDI** (*Universal Description, Discovery Language)* – linguagem de anotação definida em XML com a qual se cria a meta informação característica de um Web Service; vários registos UDDI são agrupados em repositórios que possuem uma interface/API de pesquisa para permitir a uma aplicação cliente pesquisar e localizar um serviço.

- ✓ **JSON** (*JavaScript Object Notation*) – utiliza os pares nome / valores, sendo muito similar ao XML. Os pares nomes / valores não precisar de estar numa ordem específica, para além disso, tal como acontece com o XML, o JSON faculta a resiliência às mudanças e evita a fragilidade dos formatos de gravação fixa;

- ✓ **REST** (*REpresentational State Transfer*) – representa um conjunto coordenado de restrições arquitetónicas aplicadas a componentes, conectores e elementos de dados que visa a minimizar a latência e comunicação da rede e ao mesmo tempo maximiza a independência e escalabilidade das implementações dos componentes. Esta é aplicado no desenvolvimento de Web Services substituindo o SOAP.

- ✓ **API** (*Application Programming Interface*) – representa um conjunto de regras (métodos) bem definidos e especificações que as aplicações devem seguir para efetuar a comunicação entre vários componentes dos *softwares*. É uma interface entre as

diferentes aplicações que facilita a interação, semelhante a forma como a interface do utilizador facilita a interação entre humanos e computadores.

3.7. Conclusão

Para terminar este capítulo, no qual foi efetuado um estudo do estado da arte (revisão de literatura) sobre alguns aspetos e conceitos importantes sobre os dispositivos móveis e o desenvolvimento multiplataforma de aplicações móveis, bem como também da utilização de dispositivos móveis na área da saúde. A realização deste estudo proporcionou uma base sólida sobre da utilização dos dispositivos móveis na área da saúde, bem como o atual estado do conceito *Bring Your Own Device* no geral e na área da saúde.

Acerca das metodologias de desenvolvimento das aplicações, a realização deste estudo, concluiu que a seleção de uma metodologia de desenvolvimento multiplataforma, depende de vários fatores, principalmente da exigência da aplicação, mas também das plataformas alvos, da aparência e da interface do utilizador, do tipo de aplicação de acesso aos dados e recursos (*hardware*), do desempenho e da distribuição do mercado. O desenvolvimento de aplicações móveis multiplataformas requer que seja selecionada uma *framework* de desenvolvimento multiplataforma, de acordo com o contexto de desenvolvimento da aplicação móvel.

Em suma, a seleção de uma determinada abordagem de desenvolvimento multiplataforma, deve ser realizada com base no tipo de aplicação móvel que se pretenda desenvolver, ou seja, deve-se ter em conta os vários fatores que estão relacionados com os critérios apresentados neste capítulo.

4. Requisitos e Arquitetura

4.1. Introdução

Este capítulo tem como propósito a identificação, o levantamento dos requisitos, bem como a apresentação da arquitetura final da solução. O desenvolvimento de *software* engloba uma serie de fases e atividades, com o intuito de se chegar ao objetivo principal: a entrega de um produto funcional, dentro do tempo e orçamento estipulado. Uma destas atividades é sem dúvida o levantamento de requisitos. O levantamento de requisitos consiste em descobrir, analisar, documentar e verificar os requisitos de um sistema e as suas restrições. Para esta atividade, podem ser aplicadas várias técnicas, desde entrevistas, inquéritos, cenários, *brainstorming*, *workshops*, *focus groups*, categorização de *stakeholders*, etnografia, prototipagem, entre outras técnicas. Nesta dissertação, para o levantamento de requisitos foram utilizados praticamente duas das técnicas anteriormente mencionadas: *workshop* e *brainstorming***.** Estas foram utilizadas em conjunto, ou seja, o *brainstorming* foi utilizado em *workshops* (reuniões) realizados para o levantamento e recolha de requisitos (identificação e definição de requisitos do sistema).

4.2. Requisitos da Solução

Os requisitos definidos foram divididos em duas categorias: requisitos funcionais e não funcionais. Os requisitos funcionais são aqueles que fazem parte da solução, ou seja, sem a implementação desses não haverá solução, enquanto os requisitos não funcionais garantem um bom funcionamento da solução. Esta subsecção irá apresentar e descrever esses requisitos que foram identificados.

4.2.1. Requisitos Funcionais

A Tabela 22 apresenta os requisitos funcionais identificados para a solução, bem como a descrição de cada um dos requisitos.

Tabela 22 - Requisitos Funcionais

Requisito	Descrição
Gestão de Consultas	A aplicação deverá permitir ao utilizador efetuar a marcação de consultas, ver o histórico de consultas marcadas e ver o histórico das consultas realizadas.
Gestão de Utentes	A aplicação deverá permitir ao utilizador gerir, os seus dados pessoais, tais como: contato, endereço, *email*, foto de perfil e password.
Gestão de Senhas	A aplicação deverá permitir ao utilizar efetuar o pedido de senhas, mesmo que este ainda não se encontre no hospital (desde que o utente esteja dentro do raio permitido para tirar senhas). A aplicação deverá permitir obter informações, sobre o número de senhas atendidas, o tempo médio de espera até o atendimento e o total de senhas por atender.
Gestão de Localização	A aplicação deverá identificar a localização do utente e calcular se este está no local correto, para futuramente efetuar o registo de presenças.

4.2.2. Requisitos não funcionais

A Tabela 23 apresenta os requisitos não funcionais identificados como importante para a solução, bem como uma breve descrição de cada requisito identificado. Os requisitos não funcionais foram divididos em duas categorias, requisitos internos e externos.

Tabela 23 - Requisitos não Funcionais

Requisito	Descrição
Nível interno (solução)	
Disponibilidade	A aplicação deverá ser resistente a falhas que possam impedir o seu normal funcionamento, de modo a que este esteja sempre disponível.

Desempenho	A aplicação deverá apresentar um tempo de resposta muito baixo.
Escalabilidade	A aplicação deverá ser de fácil manutenção, de modo a permitir a introdução de novas melhorias e atualizações. A aplicação deverá permitir uma fácil adaptação a novos requisitos.
Segurança	A aplicação deverá garantir na íntegra a segurança e a integridade dos dados dos utentes presentes na mesma. Para isso deverá existir meios de controlo de acesso na aplicação.
Usabilidade	A aplicação deverá apresentar uma interface do utilizador amigável, intuitiva e de fácil utilização, para garantir uma boa interação entre o utilizador e a aplicação. Todas as ações devem ser transparentes, de modo a que o utilizador compreenda todos os seus efeitos.
Nível externo	
Éticos e Legais	A aplicação não pode divulgar aos terceiros nenhuma informação pessoal e clínica sobre os utentes. A aplicação deve estar de acordo com as normas legais.
Interoperabilidade	A aplicação deverá ser capaz de interagir com outros sistemas heterogéneas para a troca de informações.

4.3. Arquitetura da Solução

A arquitetura da solução resume-se no modelo de aplicações Cliente-Servidor. Pelo que o modelo global contempla duas partes importantes da solução: Centro Hospitalar do Porto (CHP) Assistente Pessoal Hospitalar (Cliente) e a plataforma Agência para a Integração, Difusão e Arquivos de Informação Médica (AIDA) (Servidor). A Figura 33, ilustra o modelo de arquitetura presente na solução aprestada.

- **Aplicação Móvel (CHP Assistente Pessoal Hospitalar)** – é o responsável pelo processamento e apresentação dos dados provenientes do servidor e pela interação com o utente.

- **Servidor (AIDA)** – é a plataforma responsável para disponibilizar os serviços alojados, de modo a garantir o correto funcionamento da aplicação móvel.

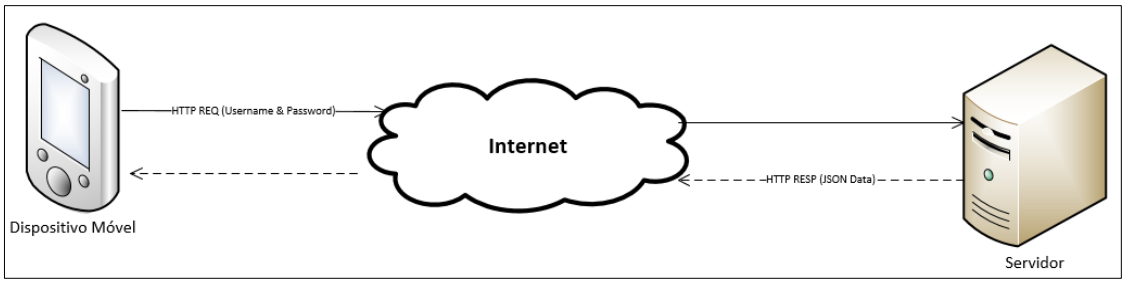


Figura 33 - Arquitetura da final da solução

A imagem ilustrada na Figura 33, retrata a presente arquitetura segue o paradigma cliente-servidor, no qual o cliente á aplicação móvel e o servidor é a plataforma AIDA. O protocolo de comunicação entre as duas partes é o protocolo HTTP, no qual o cliente realiza pedidos ao servidor por meio de uma URL e as respostas são devolvidas em formato *JavaScript Object Notation* (JSON).

A Figura 34, ilustra o diagrama de sequência das comunicações efetuadas entre o cliente (aplicação) e o servidor (AIDA). Como se pode ver na figura a comunicação inicia do lado do cliente, depois é direcionado para o servidor, que por sua vez retorna um objeto JSON com as informações solicitadas pelo cliente. Neste caso, trata-se do diagrama de sequência de autenticação do utente.

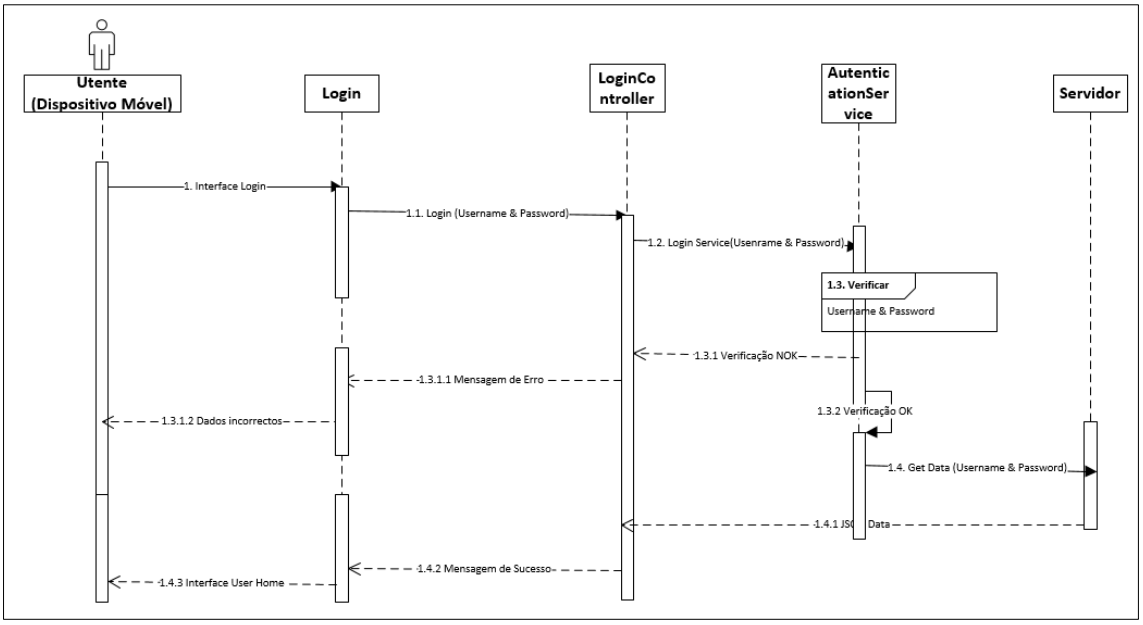


Figura 34 - Ilustração da comunicação entre as partes da solução

4.3.1. Componentes da Arquitetura

- **Aplicação**

A aplicação móvel constitui a parte principal da solução apresentada, pois é o componente que efetua a interação com o utilizador final, permitindo a interação com o sistema e usufruir das funcionalidades da aplicação. Nesta primeira fase, a aplicação desenvolvida é direcionada para os dispositivos móveis com o sistema operativo Android. A aplicação não possui uma base de dados local, pelo que o funcionamento do mesmo depende totalmente da conexão do dispositivo móvel à internet (Wi-Fi ou Redes móveis), visto que é necessários efetuar pedidos constantes ao servidor para que os utilizadores possam visualizar os dados. Todas as respostas enviadas pelo servidor são recebidas no formato de JSON e a aplicação quando às recebe do servidor, efetua o tratamento dos mesmos antes de os apresentar ao utilizador. Além disso, alguns dados são armazenados temporariamente nos *Cookies* e no *Local Storage.* Quando o utilizador terminar a sua sessão na aplicação, ou seja, efetuar o *Log Out*, esses dados armazenados temporariamente são todos eliminados. Para instalar a aplicação, o utilizador deverá aceitar um conjunto de permissões de modo que a aplicação possa ter acesso aos recursos nativos do dispositivo móvel.

- **Servidor**

O servidor também constitui uma parte importante da solução apresentada, pois neste encontra-se a base de dados onde estão armazenadas todas as informações. Este constitui-se numa interface para os colaboradores do centro hospitalar, ou seja, o servidor é o intermédio entre os colaboradores e os dados que são apresentados aos utentes na aplicação. Ainda, no servidor encontra-se um conjunto de *Web Services* REST que foram desenvolvidos para comunicar com a aplicação. A Tabela 24 apresenta algumas informações técnicas do servidor utilizado na solução final.

Tabela 24 - Informações do Servidor

Nome	Designação
Sistema Operativo	Linux
Arquitetura	X86_64
MySQL Version	5.5.57-cll
Versão do PHP	5.3.29
Apache Version	2.2.31
Server Load	4.11(24 cpus)
Memory	----

4.3.2. Funcionalidades da Solução

A Tabela 25 apresenta detalhadamente as funcionalidades desenvolvidas para a solução proposta, tanto as funcionalidades para o cliente (aplicação móvel) como para o servidor (AIDA).

Tabela 25 - Lista das funcionalidades da solução

Cliente	Servidor
Marcação de consultas; Visualização das consultas marcadas; Visualização do histórico de consultas Sincronização das consultas marcadas com o calendário do dispositivo móvel;	Criação de eventos de consultas; Visualização dos eventos de consultas; Criação de um portal para os médicos; Criação de um portal para o serviço de senhas.

Cliente	Servidor
Visualização dos dados pessoais; Atualização dos dados pessoais; Serviço de localização; Serviço de senhas.	

Para o desenvolvimento das funcionalidades no servidor, as linguagens de programação utlizadas foram: **PHP** e **JavaScript**. Os detalhes sobre as informações das funcionalidades e configuração do servidor encontram-se no **Anexo B – Funcionalidades e Configuração do Servidor**.

4.4. Conclusão

Neste capítulo foram apresentados os requisitos funcionais e não funcionais da solução e uma breve descrição dos mesmos. Por outro lado, também foi apresentado a arquitetura final da solução, também com uma breve explicação da forma com é realizada toda a interação entres as duas partes da arquitetura.

Em suma, este capítulo resume-se basicamente no levantamento de requisitos da solução efetuado no início da dissertação e que foi sendo aperfeiçoado ao longo do desenvolvimento da mesma, fazendo com que a solução final fosse de encontro com as necessidades apresentadas. De realçar que ao longo do desenvolvimento do sistema, alguns dos requisitos sofreram algumas alterações, de modo a garantir uma solução mais aperfeiçoada.

5. Implementação e Testes da Solução

5.1. Introdução

Este capítulo tem como propósito a descrição do processo de seleção das tecnologias de desenvolvimento da solução final, bem como a especificação da implementação e a apresentação dos resultados dos testes realizados. Este capítulo também apresenta as tecnologias que estão por de trás da solução apresentada e as razões que levaram à sua escolha.

5.2. Seleção da Tecnologia e Implementação da Solução

A seleção da tecnologia constituiu-se num dos grandes desafios desta dissertação, visto que, foi efetuado de acordo com os estudos realizados anteriormente sobre as metodologias de desenvolvimento (0) e seus critérios (0), bem como das tecnologias de desenvolvimento (3.6.2). Deste modo, as tecnologias selecionadas para o desenvolvimento da solução final foram: **PhoneGap + Angular Mobile UI.** Estas tecnologias são *frameworks* de desenvolvimento de aplicações móveis (abordagem multiplataforma), e foram escolhidas entre um conjunto de tecnologias analisadas.

Na primeira etapa, realizou-se a escolha da abordagem de desenvolvimento multiplataforma e logo a seguir a escolha da tecnologia de desenvolvimento correspondente à mesma. A escolha da abordagem multiplataforma rege-se principalmente pela exigência da aplicação, mas também pelas plataformas alvo, tipo de aplicação, acesso aos dados e recursos nativos (*hardware*), interface do utilizador e aparência, desempenho e pelo mercado de distribuição, segundo os critérios base de seleção de abordagens multiplataformas.

Os principais critérios escolhidos e levados em consideração para a escolha da abordagem de desenvolvimento mais adequada, foram os seguintes: a exigência e o tipo de aplicação. Cada aplicação móvel tem as suas particularidades e características únicas, e a escolha da abordagem a ser seguida deverá assentar-se em grande parte nessas mesmas particularidades e características.

No que diz respeito ao tipo de aplicações, existem quatro (4) tipos de aplicações:

- Aplicações de dados conduzidos pelo servidor;
- Aplicações baseadas em sensores/IO;
- Aplicações autónomas/independentes;
- Aplicações cliente-servidor.

Destes tipos de aplicações acima mencionados, a aplicação a desenvolver espelha-se no tipo cliente-servidor, pelo facto de que tanto o cliente como o servidor encontram-se envolvidos no processamento de dados na solução apresentada. Do lado cliente, para além da constante interação com o utilizador e de possibilitar a visualização das informações provenientes do servidor, também realiza o processamento de dados localmente que possam ser apresentados posteriormente ao utilizador. Estes dados são processados localmente e como tal não necessitam de ser enviados para o servidor. Assim, a aplicação cliente será o responsável pela gerência e processamento destes mesmos dados.

A abordagem de desenvolvimento que melhor se adequa para as aplicações do tipo cliente-servidor é a abordagem híbrida, como se pode constatar na Tabela 11. O PhoneGap é atualmente considerado uma *framework* de referência no mercado de desenvolvimento de aplicações híbridas devido a sua modularidade, daí a escolha recai sobre a mesma.

Este pode ser conjugado com algumas ***UI-Frameworks*** (Ex. AngularJS Mobile UI, Bootstrap, jQuery Mobile, Sencha Touch) para se conseguir uma melhor interface do utilizador. Dos *UI-Framework* estudados e apresentados no ponto 3.6.2 nas alíneas e), f), g) e 0, o escolhido foi o **AngularJS (Mobile UI)**. A escolha desta *framework* deve-se ao facto de ser uma *framework* recente, muito popular, com grande capacidade de expansão e utilizado por uma enorme comunidade de programadores, em parte devido a sua grande modularidade. Outros fatores que tiveram na base dessa escolha, reside no facto de poder ser conjugado com o PhoneGap para produzir bons resultados a nível da interface do utilizador (interface simulada).

No que diz respeito ao desempenho produzido por aplicações desenvolvidas com recurso a estas duas tecnologias, não foi possível encontrar nenhum estudo que efetuasse essa análise sobre o desempenho de aplicações desenvolvidas com os mesmos. Por outro lado, teve se a iniciativa de

desenvolver a aplicação e no final realizar um conjunto de testes de desempenho para tentar elaborar uma comparação simples com os resultados obtidos das outras aplicações que foram desenvolvidas com o PhoneGap e outras *Ui-Frameworks*. A análise comparativa do desempenho produzido por aplicações desenvolvida com recurso ao PhoneGap e outras *UI-Frameworks* pode ser consulta nas seguintes tabelas (Tabela 19 - utilização do CPU, Tabela 20 - utilização da memória e Tabela 21 - utilização da energia). O AngularJS Mobile UI oferece uma vasta gama de componentes *Graphic User Interface* (GUI) otimizados para *input touch* para aplicações WEB móveis. Esses componentes vão desde botões, a elementos de formulários, listas, *sliders*, ícones, separadores, menus, etc.

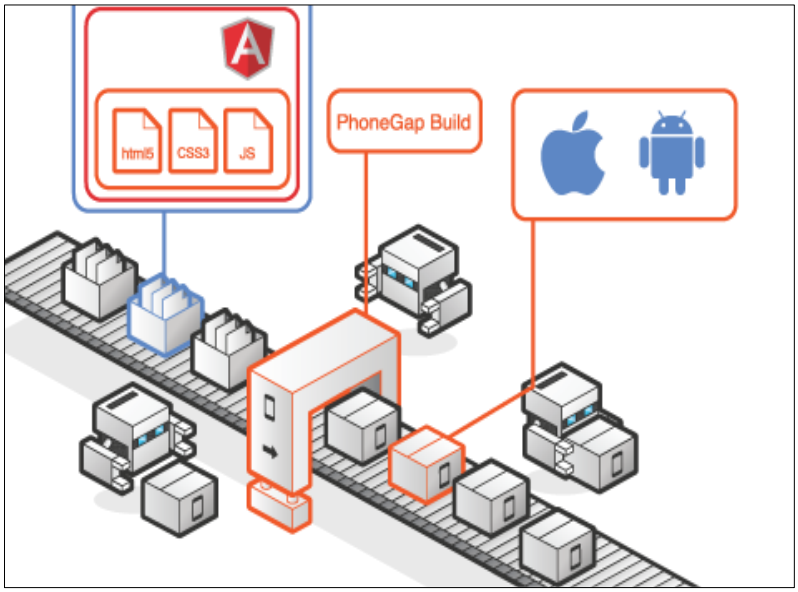


Figura 35 - PhoneGap + AngularJS Mobile UI adaptado (Nuls, 2015)

A Figura 35 ilustra a combinação AngularJS UI + PhoneGap. De acordo com a figura, pode-se constatar que tudo o que for código HTML, CSS e JavaScript é feito em AngularJS UI enquanto o PhoneGap atua como o "empacotador", ou seja, efetua a *build* do código fonte escrito do lado do AngularJS UI.

5.3. Implementação da solução CHP Assistente Pessoal Hospitalar

Esta subsecção destina-se a apresentação da implementação da solução desenvolvida ao nível dos componentes da arquitetura da mesma. Será apresentada a implementação da aplicação móvel e a descrição dos URL's que fazem a conexão entre o cliente e o servidor (pedido de dados).

✓ Aplicação Móvel

Para o desenvolvimento do CHP – Assistente Pessoal Hospitalar as *frameworks* de desenvolvimento escolhidas foram:

- **PhoneGap** que é um "empacotador" que possibilita incluir aplicações escritas em linguagens de programação conhecidas (Ex: JavaScript) em aplicações nativas;
- **AngularJS (Mobile UI)** que é uma *framework* de alto desempenho para o desenvolvimento dei interfaces de utilizador, que permite desenvolver aplicações multiplataformas utilizando JavaScript.

O AngularJS UI é uma *framework Modal-View-Controller* (MVC) que rapidamente começou a ganhar novas perspetivas devido a sua modularidade, que passou também a ser conhecido como uma *framework* **MVW** (*Modal-View-Whatever*). Esta modularidade do AngularJS é uma abordagem que facilita o desenvolvimento, a estruturação e organização do código fonte da aplicação, de modo que exista uma separação entre a interface e a camada lógica do *software*. Com a existência desta separação entre estas duas camadas, as alterações efetuadas na interface da aplicação não afetam a manipulação dos dados, e enquanto que as manipulações dos dados podem ser reorganizadas sem alterar a interface da aplicação. O AngularJS (*Mobile UI*) consiste no desenvolvimento de um ou mais *Modals, Views, Controllers, Services, Directives e Modules*, como se pode ver na Figura 36 apresentada a seguir.

As *Views* têm duas funções, sendo que a primeira consiste na representação dos dados dos *Models* e a segunda consiste na obtenção dos dados de entrada introduzidos pelos utilizadores para os *Controllers*. Os *Controllers* pela sua vez, transformam estas entradas de dados em alterações no comportamento da aplicação. Os *Controllers* efetuam a medição da entrada, convertendo-a em comandos para os *Models* ou para as *Views*, i.e., os *Controllers* são as camadas

de ligação entre os *Models* e as *Views*. Os *Models* são as camadas que possuem toda a lógica da aplicação, e são os responsáveis pelas regras de negócio, lógica e funções.

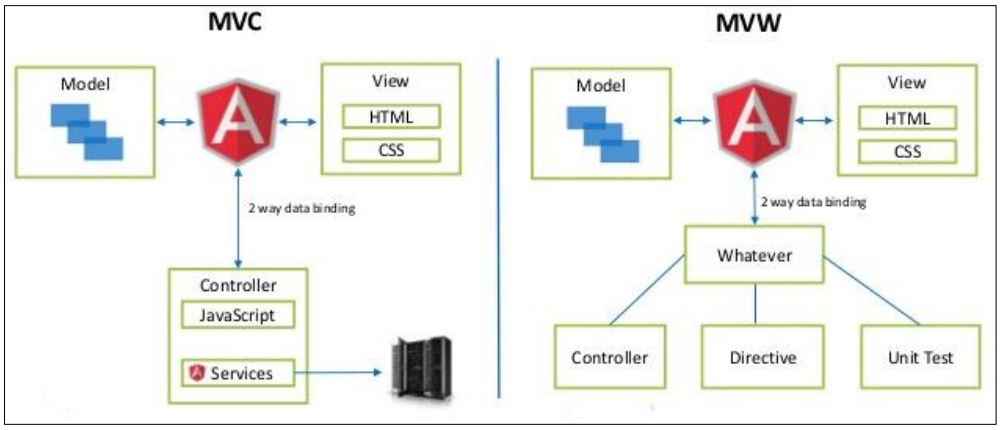


Figura 36 - Padrão MVC e MVW do AngularJS retirado de (Edureka, sem data)

O desenvolvimento de aplicações com as *frameworks* que utilizam o padrão **MVC** traz muitas vantagens tais como a separação das diferentes partes da aplicação (lógica de entrada, lógica de negócio e UI). Em caso de projetos de maiores dimensões, este padrão facilita o seu desenvolvimento e a sua manutenção. A aplicação desenvolvida é constituída por 21 *Views,* 9 *Controllers,* 10 *Services* e 1 *Directives*. A Figura 37 ilustra a estrutura interna da aplicação.

Figura 37 - Estrutura interna da solução

As *Views* encontram -se na diretoria "*templates*", com exceção à *View* inicial que encontra -se na diretoria "html" cujo é designada por *index.html*. Os *Controllers, Modals, Services, Directives* encontram-se todos na diretoria do "js", pois são todos escritos na linguagem JavaScript. O ficheiro app.js é o ficheiro por onde a aplicação começa a ser executado.

a) Configuração de *Plugins*

O PhoneGap dispõe de um conjunto de *plugins* que permitem o acesso aos recursos nativos dos dispositivos móveis (quer a nível do *hardware* do dispositivo, quer a nível das aplicações nativas). Esses *plugins* podem ser implementados de várias formas diferentes e mesmo a sua implementação pode variar de plataforma para plataforma. Para usufruir desses recursos nativos, efetuou-se a importação dos *plugins*. A importação dos *plugins* procedeu-se através da linha de comandos (*command line*) do PhoneGap. O comando utilizado para adicionar os *plugins* é:

- ***phonegap plugin add <id_plugin>***

Quando executado este comando o ficheiro **config.xml** é automaticamente atualizado, pelo que não tem a necessidade de adicionar nada ao ficheiro.

A Tabela 26, apresenta os recursos nativos (*calendar, dialogs, geolocation, network information, vibration e whitelist*) que foram utlizados na aplicação.

Tabela 26 - Acesso aos recursos nativos

Plugin	Recurso Nativo	Descrição
Calendar	READ_CALENDAR	Permite ler o calendário do dispositivo móvel.
	WRITE_CALENDAR	Permite escrever no calendário do dispositivo móvel.
Dialogs	N/A	Permite apresentar notificações recorrendo a recursos nativos do dispositivo móvel.
Geolocation	ACCESS_COARSE_LOCATION	Permite utilizar o NETWORK_PROVIDER apenas para determinar a posição.
	ACCESS_FINE_LOCATION	Permite utilizar o serviço de GPS_PROVIDER e NETWORK_PROVIDER.
Network Information	INTERNET	Permite aceder à internet.
	ACCESS_NETWORK_STATE	Permite aceder ao estado da rede Wi-Fi.
	ACCESS_WIFI_STATE	Permite saber o estado da interface Wi-Fi do dispositivo móvel.
Vibration	VIBRATE	Permite a vibração do dispositivo móvel.
Whitelist	N/A	Permite a implementação de políticas *whitelist* para a navegação na aplicação *webview.*

A importação dos *plugins* necessários de modo a usufruir dos recursos nativos do dispositivo móvel é especificado na Tabela 27.

Tabela 27 - Importação dos plugins

Plugin	Comando utilizado
Calendar	*phonegap plugin add cordova-plugin-calendar*
Dialogs	*phonegap plugin add cordova-plugin-dialogs*
Geolocation	*phonegap plugin add cordova-plugin-geolocation*
Network Information	*phonegap plugin add cordova-plugin-network-information*
Vibration	*phonegap plugin add cordova-plugin-vibration*
Whitelist	*phonegap plugin add cordova-plugin-whitelist*

A aplicação só executa as suas tarefas apenas quando o dispositivo móvel tiver uma ligação à internet (Dados Móveis ou Wi-Fi). A aplicação foi desenvolvida para que quando o dispositivo móvel não estiver ligado a internet a mesma envia uma notificação ao utilizador, informando que não tem ligação à internet. Assim, procedeu-se a importação do *plugin Network Information*, que verifica do estado de ligação do dispositivo móvel à internet.

A possibilidade de sincronizar os eventos da aplicação com o calendário do dispositivo móvel acabou por ter um impacto extra na solução, uma vez que todas as informações dos eventos das consultas marcadas na aplicação são automaticamente criadas e guardadas no calendário do dispositivo móvel. Para usufruir deste recurso, procedeu-se a importação do *plugin calendar* para obter o acesso ao calendário do dispositivo móvel. Com este *plugin* através da função *window.plugins.calendar.createEventWithOptions (title, eventLocation, notes, startDate, endDate, calOptions, success, error)*, foi possível criar novos eventos e adicionar ao calendário do dispositivo móvel. Trata -se de uma funcionalidade, cuja importância é de certa forma é indiscutível, pois possibilita que os utentes sejam notificados sobre as suas marcações de consultas, evitando assim que os mesmos esqueçam desse compromisso.

```
var calOptions = window.plugins.calendar.getCalendarOptions();
calOptions.firstReminderMinutes = 120;
calOptions.calendarId = 1;
window.plugins.calendar.createEventWithOptions(title,eventLocation,notes,startDate,endDate,calOptions,success,error);
```

Figura 38 - Sincronização da consulta com o calendário e notificação da consulta

A Figura 38 ilustra o código que foi utilizado para sincronizar a consulta marcada pelo utente com o calendário do dispositivo móvel e ainda apresenta a notificação que o utente recebe na data da realização da consulta anteriormente marcada.

A aplicação tem a necessidade de enviar notificações ao utente, para realizar este requisito, procedeu-se a importação do *plugin dialogs*, para criar mensagens de notificações para o utente. As mensagens de notificações utilizadas destes *plugin* são: *alert*, *confirm*, *beep* e *vibrate*.

A necessidade de reconhecer a posição exata do utente através da aplicação acabou por ser um dos requisitos fundamentais da aplicação. Para efetuar a identificar a posição do utente, procedeu-se a importação do *plugin geolocation*. Com recurso a este *plugin*, é possível identificar as coordenadas geográficas (latitude e longitude) do utente. Para obter estas informações, foi utilizado a função *navigator. geolocation.getCurrentPosition()*.

O *plugin vibrate* foi utlizado para vibrar o dispositivo móvel, sempre que haja novas notificações e quando o mesmo não tiver ou perder a conexão com a internet. A vibração do dispositivo móvel é possível com recurso à função *navigator.vibrate().*

Existe um modelo que é utilizado para controlar o acesso aos domínios externos. Para esse efeito, o *plugin whitelist* foi importado para gerir e controlar os acessos, que por norma a política de segurança é restringir todo o acesso à rede. Com este *plugin* é possível permitir o acesso a domínios e subdomínios de rede específicos.

Tal como, muitas aplicações móveis, o CHP – Assistente Pessoal Hospitalar necessita de algumas permissões de acesso que o utilizador deverá aceitar para a aplicação. No momento de instalação da aplicação é apresentado ao utilizador as permissões que deverá aceitar para autorizar que a aplicação aceda aos recursos nativos do sistema operativo móvel. A Figura 39 ilustra as permissões que o utilizador deverá aceitar.

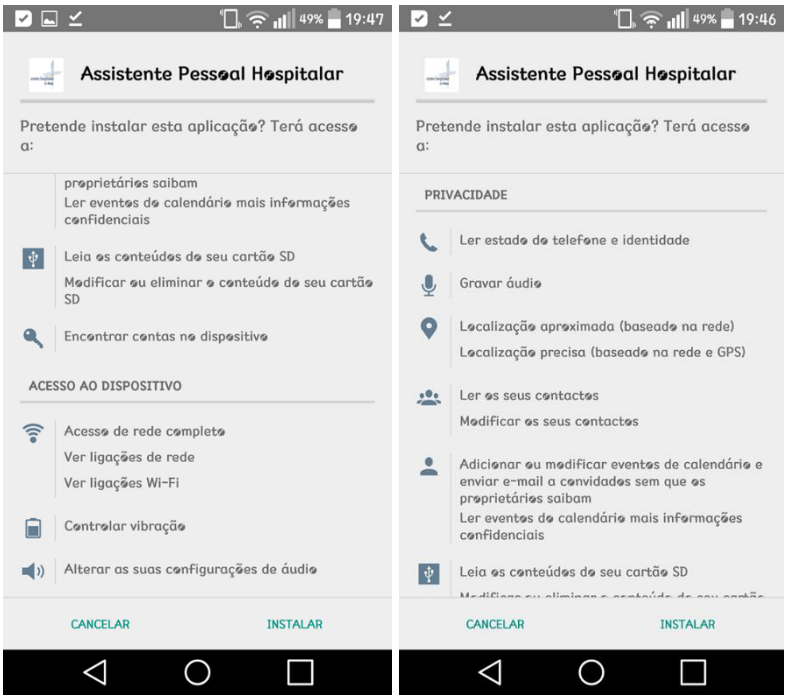


Figura 39 – Permissões

b) Protocolos de Comunicação

Os protocolos de comunicação consistem na especificação de todos os serviços de interface de dados entre o cliente (CHP – Assistente Pessoal Hospitalar) e o servidor da Agência para a Integração, Difusão e Arquivos de Informação Médica (AIDA). Devido ao número de serviços utilizados na aplicação, apenas será apresentado alguns dos serviços da mesma nas tabelas seguintes.

Tabela 28 - Autenticação do Utente

Métodos	**POST**
URL	https://webitcloud.net/projects/avirtual/?username=**username**& password=**password**
Path	/login
Parâmetros de Entrada	*Username* ou Número de Utente e *Password* do utente
Parâmetros de Saída	JSON *Array Object*, com os dados pessoais do utente
Descrição	Autenticação do Utente
Dados de Retorno	Caso de sucesso entra na área pessoal do utente, e em caso de insucesso é enviada uma mensagem de erro.

A autenticação do utente é realizada a partir da URL disponibilizada na Tabela 28. Para tal, recebe como parâmetros de entrada o *Username* ou o número de utente e a *password* do utente. Posto isso, o servidor faz uma verificação da existência do utente na base de dados com base nos dados disponibilizados no parâmetro de entrada. Em caso de a verificação for positiva, o servidor responde com JSON *Array Object* com o conjunto de informações referentes a esse utente. E em caso de verificação negativa, o servidor responde na mesma com JSON *Array Object,* mas com uma mensagem de erro.

Tabela 29 - Marcação de Consulta

Métodos	**POST**
URL	https://webitcloud.net/projects/avirtual/marcar-consulta.php
Path	/marcacao
Parâmetros de Entrada	*Object*
Parâmetros de Saída	N/A
Descrição	Marcação de consultas
Dados de Retorno	Caso de sucesso é marcada a consulta e é adicionado ao calendário do dispositivo a data e o local da consulta marcada. Em caso de insucesso, é apresentado uma mensagem de erro.

A marcação de consulta é efetuada a partir da URL acima apresentada na Tabela 29. Para tal recebe como parâmetro de entrada um único objeto, sendo que este objeto contém um conjunto de informações facultadas pelo utente a quando do pedido de marcação de consulta. Tais informações são: tipo de consulta (aberta ou especializada), em caso de consulta especializada o nome da especialização da consulta, e o nome do médico disponível para a consulta de acordo com as informações anteriormente introduzidas e por último a data da realização da consulta. Do lado do servidor é efetuado uma validação desses dados antes de serem guardados na base de dados. Em caso de a verificação for positiva é criado um novo registo na base de dados com a respetiva informação.

Tabela 30 - Lista tipos de consultas

Métodos	**GET**
URL	https://webitcloud.net/projects/avirtual/lista-tipos-consulta.php
Path	/marcacao
Parâmetros de Entrada	N/A
Parâmetros de Saída	JSON *Array Object*
Descrição	Listar os tipos de consultas
Dados de Retorno	Em caso de sucesso é retornado um JSON *Array Object* com os tipos de consultas disponíveis para o utente, e em caso de insucesso é retornado na mesma um JSON *Array Object,* mas com mensagem de erro.

A partir da URL indicado na Tabela 30, é possível dar a conhecer os tipos de consultas disponíveis para o utente. Este método não precisa de nenhum parâmetro de entrada. O parâmetro de saída deste método é JSON *Array Object*. O servidor recebe o pedido e faz uma pesquisa na base de dados e retorna os dados para o utente. Paralelamente, os métodos **GetEspecialidades()** e **GetMedicos()** seguem a mesma analogia apresentada na tabela acima. Por outro lado, o parâmetro de saída é um JSON *Array Object.* Os dados de retorno é um JOSN *Array Object* com as respetivas informações.

Tabela 31 - Lista marcação consulta utente

Métodos	**POST**
URL	https://webitcloud.net/projects/avirtual/lista-marcacao-consulta-utente.php?id_utente=**id**
Path	/consulta
Parâmetros de Entrada	Identificação do utente
Parâmetros de Saída	JSON *Array Object*
Descrição	Listar as consultas marcadas do utente
Dados de Retorno	Em caso de sucesso, é retornado um JSON *Array Object* com todas as consultas marcadas do utente, e em caso de insucesso é retornado um JSON *Array Object* com mensagem de erro.

A partir da URL indicada na Tabela 31, é possível obter todas as consultas marcadas de um determinado utente. Este método tem como parâmetro de entrada o **id** (identificação do utente) e tem como parâmetro de saída o JOSN *Array Object*. De modo análogo, tem-se os métodos **GetMedico(id)**, **GetEspecialiade(id)**, **GetTipoConsulta(id)** e **GetDetalhesConsulta(id)** seguem a mesma analogia apresenta na tabela acima e retorna os respetivos

c) Funcionalidades *Core* da Solução CHP – Assistente Pessoal

As funcionalidades *core* da solução foram agrupadas nas seguintes categorias:

- **i.** Autenticação
- **ii.** Consultas
- **iii.** Perfil
- **iv.** Localização
- **v.** Senhas

Autenticação - Para usufruir das funcionalidades *core* da solução é preciso efetuar a autenticação do utilizador na aplicação. Quando a aplicação é executada, é verificada o estado da conexão do dispositivo móvel à internet e caso este não estiver conectado, é apresentado uma notificação de falta de conexão. Para autenticar na aplicação, o utilizador deverá preencher os campos obrigatórios (*username* ou numero de utente e *password*), ou seja, os seus credenciais. Após a

submissão desses dados, a aplicação solícita um pedido de autenticação do utilizador ao servidor como especificado na Tabela 28 anteriormente apresentado. Caso a autenticação for efetuada com sucesso, a aplicação direciona o utilizador para a interface *UserHome*.

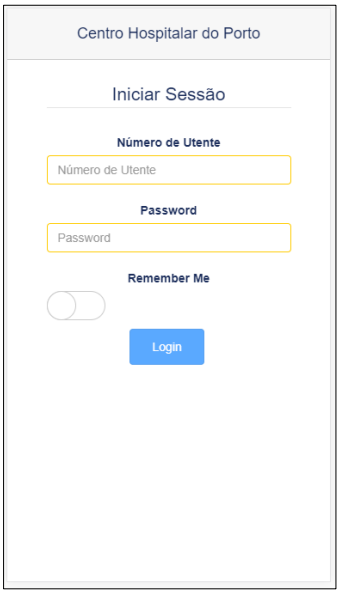
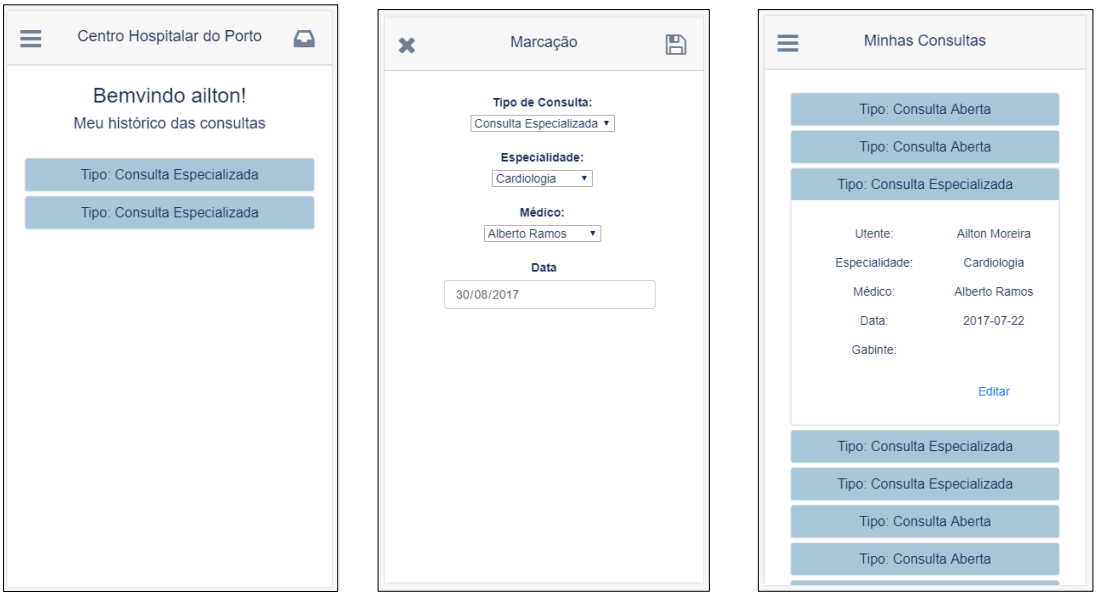


Figura 40 - Interface de autenticação

A Figura 40 ilustra a interface de autenticação, cujo apresenta apenas dois campos obrigatórios um para o *username* ou número de utente e o outro para a *password*. O botão *Login* é ativado quando os campos obrigatórios estiverem preenchidos. Existe ainda o botão *Remember Me* que possibilita que a aplicação guarde localmente o *username* ou o número de utente do utilizador.

Consultas - Após a autenticação do utilizador ser efetuada com sucesso, a aplicação entra na área pessoal do utente. A primeira interface a ser apresentado é a interface *UserHome*, conforme se pode confirmar na Figura 41 (imagem à esquerda **a**). Na referida interface é apresentada as informações sobre o histórico das consultas realizadas pelo utente. Do lado superior esquerdo tem-se o botão menu bar, que tem o menu da aplicação e do lado direito tem-se o botão de notificações.

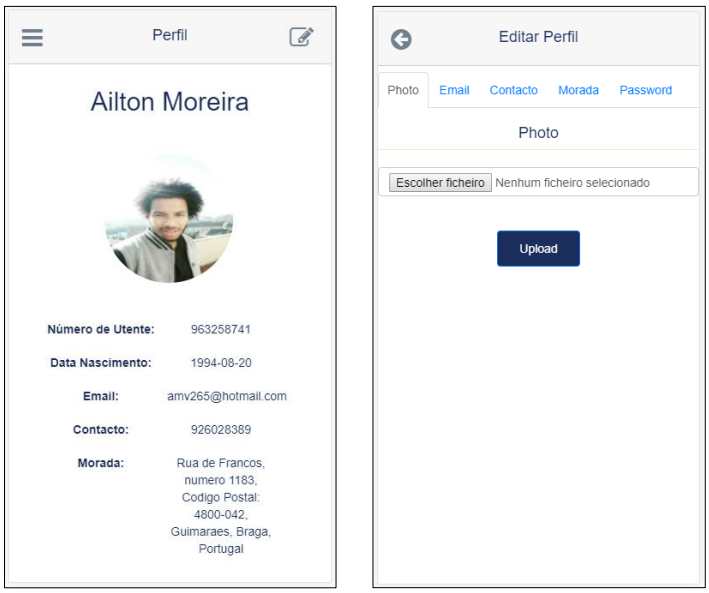


a) b) c)

Figura 41 - Área autenticada da consulta

Ainda na mesma figura (Figura 41), na imagem ao centro (**b**) é possível ver o processo de marcação de consultas. O mesmo trata-se de um processo simples, tendo o utilizador apenas de escolher que tipo de consulta pretende marcar (consulta aberta ou consulta especializada). Caso o utilizador pretender uma consulta especializada, terá de escolher a especialidade pretendida e seguidamente o médico disponível para a consulta da especialidade escolhida. Caso o utilizador pretenda marcar uma consulta aberta, apenas terá de escolher o médico disponível para este tipo de consulta. Depois disso, o próximo passo consiste em escolher a data da realização da consulta. Para o cancelamento da marcação da consulta, encontra-se no canto superior esquerdo o botão "cancelar", e no canto superior direito encontra-se o botão "guardar", que consiste em guardar a consulta (este ultimo só funciona quando todos os dados do formulário estiverem preenchidos, ou seja, todos os campos do formulário são de preenchimento obrigatório). A imagem a direita na Figura 41 (**c**) é a interface que permite a visualização das consultas marcadas pelo utilizador. Ao carregar sobre o tipo de consulta, este expande-se e apresenta os detalhes sobre a referida consulta.

Perfil - A imagem a esquerda da Figura 42 (**a**) consistiu-se na interface de perfil do utilizador, onde é possível ver as informações básicas sobre o utilizador, tais como nome, número de utente, data nascimento, endereço de morada, correio eletrónico, contacto e fotografia. No canto superior direito desta interface encontra-se o botão "editar perfil", que direciona para a interface de editar perfil (imagem à direita **b**).



a) b)

Figura 42 - Perfil do utilizador

Na Figura 42 a imagem a direita (**b**) é a interface de editar perfil, que é composto por cinco (5) "*tabs*", sendo que cada uma contém uma interface referente a um dado do utilizador que é permitido editar. Estes dados editáveis são: o email, contato, fotografia de perfil, morada e *password*. Assim, caso o utilizador pretender editar o seu contacto deverá ir diretamente ao *tab* contacto e assim por diante. Ainda, no canto superior esquerdo da referida imagem encontra-se o botão "voltar", que permite ao utilizador voltar para a interface de visualização de dados pessoais.

Localização - A Figura 43 ilustra a interface de serviço de localização. A imagem à esquerda (**a**) corresponde à interface com as informações sobre o local, a data e as coordenadas geográficas (latitude e longitude) do evento. Ainda na mesma interface, encontra-se presentes dois botões,

sendo um deles aquele que permite apresentar o mapa do local onde o utilizador se encontra e o outro permite efetuar o registo de presença. Esta interface é atualizada a cada 60 segundos, com o intuito de determinar a posição exata do utilizador. Quando é pressionado o botão de efetuar o registo de presenças, é apresentado um *modal pop up* com a interface de efetuar o registo de presenças, conforme ilustrada na imagem à direta (**b**) da Figura 43. O registo de presenças é efetuado recorrendo às coordenadas geográficas (latitude e longitude) do utilizador e efetua uma série de cálculos para determinar se o utilizador encontra-se na localização correta. As informações das coordenadas geográficas foram obtidas através do navegador do browser do próprio dispositivo móvel, pois o seu grau de acuidade é maior do que o *do Global Positioning System* (GPS) principalmente quando é utilizado dentro dos edifícios.

a) b)

Figura 43 - Serviço de localização

Senhas - A Figura 44 ilustra a interface do serviço de senhas. Na imagem à esquerda (**a**) é possível ver a interface inicial, onde o utilizador pode escolher o serviço que pretende usufruir e tirar a respetiva senha. Para tirar a senha do serviço pretendido, o utente apenas precisa de carregar no botão "**+**" (mais). A imagem ao centro (**b**) ilustra a interface de tirar a senha do utilizador, onde apresentado um *modal pop up* com a vez do utilizador e o respetivo serviço. O serviço de senhas também utiliza as informações sobre as coordenadas geográficas (latitude e longitude) do

utilizador para determinar se o utilizador encontra dentro da área permitida para tirar a senha. As informações das coordenadas geográficas foram obtidas através do *plugin Geolocation* do PhoneGap.

Figura 44 - Serviço de senhas

A imagem mais à direita (**c**) ilustra a interface da vez do utilizador. Que quando chega a sua vez este é notificado. Mas antes disso a aplicação também notifica o utilizador que a sua vez está próximo (esta notificação depende do número das senhas de tolerâncias). Na interface de notificação da vez do utilizador, é apresentado um *modal pop up* com o serviço escolhido pelo utilizador e a vez do utilizador.

d) Processamento de Dados

O servidor envia as respostas dos pedidos do cliente no formato JSON. O cliente (aplicação) por sua vez têm de efetuar o tratamento dos dados recebidos do servidor antes de os apresentar e disponibilizar ao utilizador. O próprio Angular UI já possui mecanismos internos para tratar os dados nesse formato, não necessitando de criar novos meios e tratamento, pelo que é preciso apenas extrair os referidos dados do JSON e apresentá-los ao utilizador.

5.4. Testes

Este subcapítulo destina-se à realização de testes na solução implementada, de modo a certificar que a mesma funciona da forma como é esperada. Assim sendo, a solução será submetida a testes de funcionalidades, desempenho, compatibilidade, interoperabilidade e usabilidade. A aplicação de testes tem como principal objetivo a identificação das principais falhas do sistema para que possam ser corrigidos antes da entrega final.

5.4.1. Testes das Funcionalidades

O objetivo do teste das funcionalidades como o próprio nome já diz é testar as funcionalidades da solução para apurar se o mesmo funciona de acordo com os requisitos funcionais especificados no ponto 4.2 Requisitos da Solução Para proceder a avaliação do mesmo foi criado uma nomenclatura:

A – Funciona; **B** – Funciona com algumas limitações; **C** – Não funciona;

Tabela 32 - Testes de funcionalidades

Requisitos	**Resultado**
1. Autenticação	A
2. Marcação de consultas	A
3. Visualização de consultas marcadas	A
4. Visualização de histórico de consultas realizadas	A
5. Visualização de dados pessoais	A
6. Atualizar dados pessoais (email, morada, contacto)	A
7. Sincronização das datas das consultas com o calendário do dispositivo móvel	A
8. Serviço de localização	A
9. Serviço de senhas	A

Segundo a análise presente na Tabela 32, pode se concluir que os requisitos propostos foram alcançados e testados com sucesso. Outro objetivo deste teste é tentar encontrar e identificar os erros ou falhas no sistema para uma futura correção. As funcionalidades testadas na solução

apresentada vão de encontro aos problemas identificados e os objetivos definidos para este projeto. Alguns dos problemas enfrentados pelos utentes naquela unidade hospitalar, tais como a demora no processo de marcação de consultas, o congestionamento de pessoas no atendimento, e o desconhecimento de determinados lugares da unidade hospitalar foram todos solucionados com a criação do artefacto.

5.4.2. Testes de Desempenho

O objetivo do teste de desempenho é analisar a aplicação e encontrar o seu limite durante a execução de tarefas no que diz respeito a capacidade de processamento de dados e tempo resposta aos pedidos realizados. Como tal, o teste foi efetuado em dispositivos móveis diferentes e com características diferentes, de modo a verificar o comportamento da aplicação nos mesmos. Os dispositivos utilizados para este e os próximos testes foram os apresentados na Tabela 34.

Tabela 33 - Teste de desempenho

Pedidos	**Tempo de resposta**				
	ASUS	**LG**	**Huawei**	**OnePlus**	**Vodafone**
Execução	7.50S	8.16S	5.12S	6.44S	6.38S
Autenticação	3.70S	5.32S	3.30S	3.79S	3.14S
Marcação	2.91S	3.56S	3.09S	3.23S	2.86S
Consultas	2.29S	2.59S	2.06S	2.27S	2.18S
Perfil	2.80S	3.27S	2.87S	3.02S	3.08S
Localização	4.24S	4.24S	4.45S	4.33S	4.37S
Senhas	4S	4.64S	3.85	4.16S	4.39S

OBS: Estes valores já incluem o tempo de resposta de acesso à rede e tempo de carregamento dos dados.

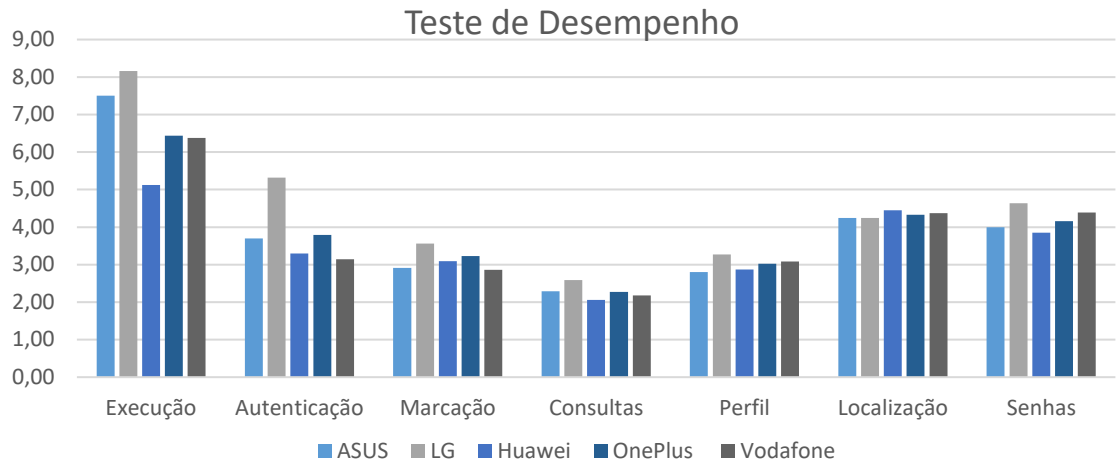


Gráfico 1 - Teste de desempenho

A nível geral a aplicação executa como era esperado, mas, no entanto, notou-se uma ligeira lentidão, causado pelo fato de ser uma aplicação híbrida e, como foi apresentado anteriormente, as aplicações híbridas apresentam resultados inferiores a nível do desempenho quando comparados com as aplicações nativas. A Tabela 33 e o Gráfico 1 apresentam os dados, no qual pode-se analisar melhor esta lentidão. De acordo com os dados obtidos, pode-se verificar que a aplicação apresenta maior lentidão ao ser executado, visto que o tempo de execução em alguns destes dispositivos móveis foram mais do que oito (8) segundos só para iniciar a aplicação. É de realçar que, para cada interface carregada é acrescido de dois (2) segundos de *time out*, de modo a garantir que todos os dados são devidamente carregados antes da interface ser apresentada na sua totalidade.

5.4.3. Testes de Compatibilidade

O objetivo de teste de compatibilidade é analisar a compatibilidade das aplicações nos diferentes ambientes e plataformas. O mesmo pretende analisar o comportamento da aplicação nos diferentes dispositivos e versões dos sistemas operativos móveis. A solução do CHP Assistente Pessoal Hospitalar é destinada aos dispositivos móveis com o SO *Android*. Este teste será realizado em diferentes dispositivos móveis com o SO *Android* e as diferentes versões do SO *Android*. A Tabela 34, apresenta as informações sobre os diferentes dispositivos móveis utilizadores para a realização do teste de compatibilidade.

Tabela 34 - Teste de compatibilidade

	ASUS	LG	Huawei	OnePlus x	Vodafone
Android	7.0 - Nougat	5.0 - Lollipop	7.0 - Nougat	6.0.1 - Marshmallow	6.0.1 - Marshmallow
Chipset	MediaTek MT6737	Qualcomm MSM8974AC Snapdragon 801	HiSilicon Kirin 650	Qualcomm Snapdragon 801	Qualcomm MSM8939 Snapdragon 615
GPU	Mali-T720MP2	Adreno 330	Mali-T830MP2	Adreno 330	Adreno 405
Memória	32GB	32GB	16GB	16GB	16GB
Modelo	ASUS _X0081D	LG-D855	VNS-L31	E1003	Smart ultra 6
Processador	Quad-core 1.3 GHz Cortex-A53	Quad-core 2.5 GHz Krait 400	Octa-core (4x2.0 GHz Cortex-A53 & 4x1.7 GHz Cortex-A53)	Quad-core 2.3 GHz Krait 400	Octa-core (4x1.5 GHz Cortex-A53 & 4x1.1 GHz Cortex-A53)
RAM	3GB	3GB	3GB	3GB	2GB

Nos dispositivos móveis que foram utlizados para a realização do teste de compatibilidade da aplicação, não foram detetados problemas de incompatibilidade.

5.4.4. Testes de Interoperabilidade

A interoperabilidade é a capacidade de dois ou mais sistemas heterogéneos comunicarem entre si, para trocar e utilizar informações provenientes de ambas as partes. A realização do teste de interoperabilidade tem o propósito de analisar o estado da comunicação entre os constituintes da solução (a aplicação e o servidor da plataforma AIDA). A Tabela 35, apresenta os dados obtidos na realização durante a realização do teste. O protocolo utilizado para a comunicação entre as partes do sistema é o protocolo HTTP. O cliente comunica com o servidor através de uma URL e o

servidor envia a resposta em formato JSON. A avaliação dos resultados é utiliza a seguinte nomenclatura:

A- Funciona; **B** – Não funciona.

Tabela 35 - Teste de interoperabilidade

Pedido	Descrição	Protocolo	Resultado
Login	Autenticação	HTTP	A
Marcação	Marcação de consultas	HTTP	A
Consultas	Obter e apresentar os dados das consultas	HTTP	A
Perfil	Obter e apresentar os dados do utilizador	HTTP	A
Localização	Obter e sincronizar os pedidos sobre o serviço de localização.	HTTP	A
Senhas	Obter e sincronizar os pedidos sobre o serviço de senhas.	HTTP	A

5.4.5. Testes de Usabilidade

O objetivo do teste de usabilidade é a obtenção do *feedback* dos utilizadores no que diz respeito a utilização da aplicação, ou seja, este teste foca na experiência do utilizador nomeadamente na possibilidade de realizar determinadas ações na aplicação. Os testes deste género são muitos comuns e são utilizados para avaliar a qualidade de um determinado produto. Para este teste a solução final é disponibilizado a um conjunto de potenciais utilizadores da mesma para o efeito de teste. Cada participante terá de utilizar a aplicação para executar determinadas ações. Para realizar a avaliação da usabilidade da aplicação é disponibilizado a cada participante um questionário onde poderá responder o seu grau de concordância com as ações da aplicação.

Os questionários de testes de usabilidade (no qual é apresentado um conjunto de ações/tarefas da aplicação e o participante terá que avaliar cada uma das ações/tarefas segundo um grau de dificuldade) e avaliação da usabilidade (no qual é apresentado um conjunto de afirmações e o participante terá de escolher uma resposta para cada afirmação apresentada de acordo com o nível de concordância) podem ser consultado na íntegra no **Anexo A - Questionários dos Testes**.

Para realização deste teste foram escolhidos um total de 20 potenciais utilizadores para testar a aplicação e depois para responder aos questionários. Como já foi referenciado anteriormente, para o teste da usabilidade da aplicação efetuou-se dois questionários um sobre a usabilidade da aplicação e o outro sobre a avaliação da usabilidade da mesma. Cada questionário é composto por dez (10) perguntas múltiplas escolha no qual o participante terá de escolher uma resposta. Para efetuar esta avaliação, nos questionários utilizou a **escala de Likert** para as respostas múltiplas escolha numa escala de 1 a 5 (Brown, 2010).

O questionário sobre o teste de usabilidade da aplicação, as respostas seguem a seguinte nomenclatura segundo a **escala de Likert**:

1 – Muito Difícil; **2** – Difícil, **3** – Indiferente, **4** – Fácil; **5** – Muito Fácil.

O questionário sobre a avaliação da usabilidade da aplicação as respostas seguem a seguinte nomenclatura segundo a **escala de Likert**:

1 – Discordo Totalmente; **2** - Discordo; **3** – Indiferente; **4** – Concordo; **5** – Concordo Totalmente.

As respostas obtidas desses questionários serão apresentadas e discutidas no capítulo seguinte ponto 6.3 - Avaliação dos Resultados, onde é efetuada uma análise completa sobre a aplicação e a avaliação dos resultados obtidos.

5.5. Conclusão

Neste capítulo foi apresentado a implementação da solução proposta para esta dissertação, que iniciou pela especificação da tecnologia escolhida para o desenvolvimento da solução final, as decisões de implementação da solução de CHP – Assistente Pessoal Hospitalar, e bem como um conjunto de testes que foram realizados à solução. A seleção da abordagem de desenvolvimento foi efetuada com base nos estudos realizados na primeira fase da dissertação. Com a abordagem de desenvolvimento escolhida, procedeu-se então a escolha das tecnologias a serem utilizadas para o desenvolvimento da solução. Essas tecnologias têm de seguir a mesma analogia de desenvolvimento que a abordagem escolhida, que neste caso é a abordagem de desenvolvimento multiplataforma.

Como foi apresentado, a tecnologia eleita para o desenvolvimento da solução foi a combinação do **Angular Mobile UI** + **PhoneGap**, na qual o Angular UI foca se essencialmente no desenvolvimento da interface do utilizador, enquanto o PhoneGap foca se no acesso aos recursos nativos dos dispositivos móveis e no processo de gerar a aplicação híbrida.

Ainda, foi apresentada a implementação da solução desenvolvida na qual foram expostos os seguintes tópicos:

- O acesso aos recursos nativos dos dispositivos móveis;
- O protocolo de comunicação utilizado entre o cliente e o servidor,
- As principais funcionalidades da solução desenvolvida;
- O processamento dos dados provenientes do servidor.

O objetivo da realização dos testes é a identificação das possíveis falhas do sistema para as corrigir e ter um *feedback* por parte dos potenciais utilizadores da aplicação. Estes testes tiveram um impacto positivo na solução desenvolvida, a partir dos mesmos dos mesmos foi possível comprovar que todos os objetivos inicialmente propostos para a solução foram alcançados, e as funcionalidades funcionam de acordo como eram esperados. A realização dos testes de interoperabilidade da aplicação e compatibilidade da aplicação com os diferentes SO móveis tem o objetivo de analisar o comportamento da aplicação quando é executada em diferentes

dispositivos móveis. Globalmente, acerca do desempenho produzido pela aplicação os resultados obtidos vão de encontro ao que era esperado em aplicações híbridas.

6. Monitorização e Avaliação dos Resultados

6.1. Introdução

A inclusão deste capítulo como próprio nome aponta, tem como propósito efetuar a monitorização da aplicação e a avaliação dos resultados obtidos durante a realização dos testes da mesma. Este capítulo pretende reportar todos os resultados e *feedback* dos utilizadores obtidos durante a fase testes.

6.2. Monotorização da Aplicação

A nível global a aplicação executa da forma como era expectável. A nível do desempenho produzido, constou-se que o mesmo não necessita de grandes recursos a nível do *hardware*, enquanto que a nível do *software* este executa sem qualquer problema em dispositivos móveis com sistema operativo igual ou superior ao *Android* 5.0. Como já era sabido, constatou-se que o desempenho da aplicação não é igual ao desempenho de uma aplicação nativa, como já foi apresentado e discutido anteriormente.

Para proceder a monotorização, foi instalado a aplicação *Power Tutor* para medir o desempenho da aplicação. Com aplicação *Power Tutor* instalada, procedeu-se a execução da aplicação CHP – Assistente Pessoal Hospitalar, na qual foi efetuada um conjunto de tarefas, desde autenticação, marcação de consulta, visualização das consultas marcadas, visualização e atualização dos dados pessoais, execução do serviço de localização e do serviço de senhas. Após a execução destas tarefas, voltou-se para a aplicação *Power Tutor* para efetuar a recolha dos dados.

Tabela 36 - Monitorização de desempenho da solução

Características	Valores Médios
Consumo do CPU	7.9%
Consumo de Energia	78 mW
Consumo da Memória	23 MB

Segundo os dados apresentados na Tabela 36, o consumo médio de energia pela aplicação é de 78 mW, enquanto o consumo médio da utilização do CPU ronda os 7.9%, e o consumo médio da memória ronda os 23 MB segundo as estatísticas de memória do dispositivo móvel.

6.3. Avaliação dos Resultados

Este subcapítulo apresenta os resultados obtidos dos questionários realizados, e uma breve análise desses resultados. Como foi referido anteriormente, na realização dos testes foram selecionados um conjunto de 20 potenciais utilizadores da aplicação para efetuarem o teste da mesma. Esses potenciais utilizadores são jovens com idade compreendida entre os 22 e 26 anos e todos fazem parte da comunidade académica.

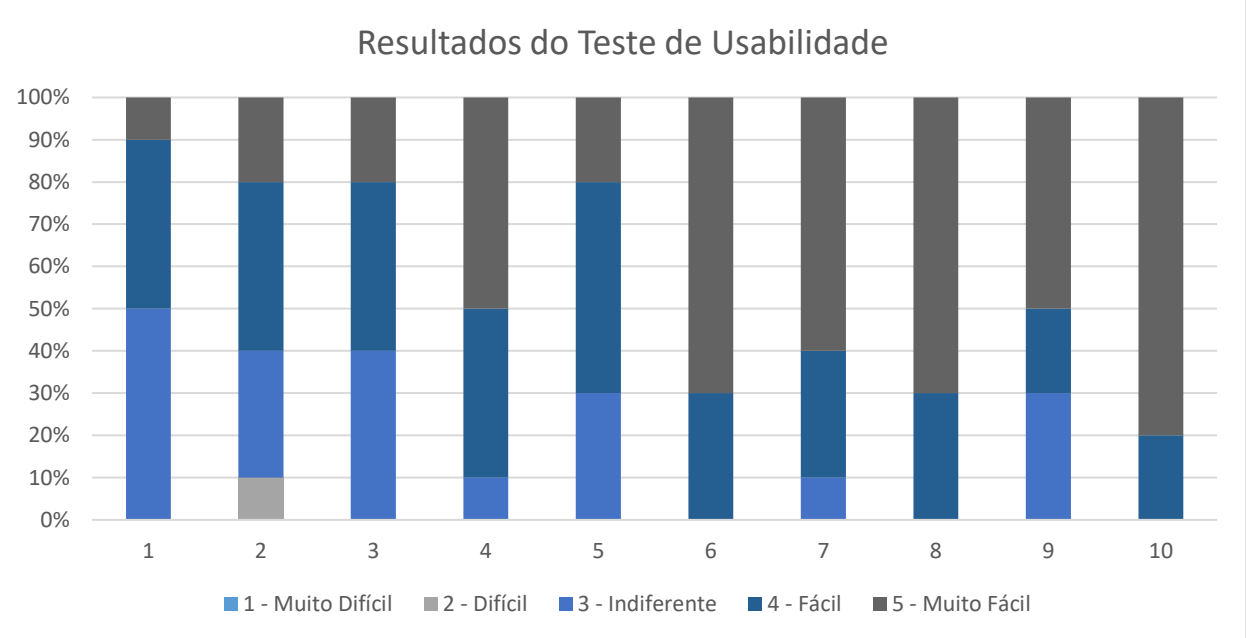


Gráfico 2 - Resultados agregados do teste de usabilidade

Legenda:

1 - Como classifica o processo de autenticação da aplicação?

2 - Como classifica o processo de marcação de consultas na aplicação?

3 - Como classifica a da visualização das consultas marcadas?

4 - Como classifica a funcionalidade de serviço de localização?

5 - Como classifica o processo de registo de presenças?

6 - Como classifica a funcionalidade de serviço de senhas?

7 - Como classifica o processo de tirar as senhas?

8 - Como classifica a visualização e o facto de poder editar os dados pessoais?

9 - Como classifica a sincronização com o calendário do dispositivo móvel?

10 - No geral, como classifica a aparência da aplicação?

De acordo com o Gráfico 2, pode se verificar que praticamente todas as funcionalidades da solução apresentada são fáceis ou muito fáceis de serem realizadas, sendo apenas em alguns casos que é considerada indiferente (neutra). Segundo os resultados apresentados no gráfico, a tarefa 1 é considerada muito fácil por 10% dos participantes, enquanto que, 40% dos participantes consideram esta tarefa fácil e os outros 50% consideram indiferente (talvez por ser uma forma de autenticação muito utilizada atualmente).

Já a tarefa 2 foi classificada difícil por 10% dos participantes, enquanto que, 30% dos participantes classificaram esta tarefa indiferente (neutra), i.e., não é fácil nem é difícil. Ainda na mesma tarefa, 40% dos participantes classificaram a mesma como sendo fácil, enquanto que, os restantes 20% classificaram a tarefa muito fácil.

A tarefa 3 foi classificada indiferente (neutra) por 40% dos participantes, enquanto que, os outros 40% dos participantes classificaram a mesma de fácil e os restantes 20% dos participantes classificaram a mesma tarefa de ser muito fácil.

A tarefa 4 foi classificada por 10% dos participantes como uma tarefa indiferente (neutra), enquanto que, 40% dos participantes classificaram a mesma tarefa como uma tarefa fácil e por outro lado a metade dos participantes (50%) classificaram esta tarefa como uma tarefa muito fácil.

A tarefa 5 foi classificada por 30% dos participantes como uma tarefa indiferente (neutra), enquanto que, 50% dos participantes classificaram a mesma tarefa de ser fácil e os restantes 20% a classificaram como uma tarefa muito fácil.

A tarefa 6 foi classifica por 30% dos participantes como uma tarefa fácil, enquanto que, os outros 70% classificaram esta tarefa como uma tarefa muito fácil.

Já a tarefa 7 foi classificada por 10% dos participantes como uma tarefa indiferente (neutra), enquanto que, 30% dos participantes classificaram a mesma tarefa como uma tarefa fácil e os restantes 60% dos participantes classificaram a referida tarefa como uma tarefa muito fácil.

A tarefa 8 foi classificada fácil por 30% dos participantes, enquanto que, os outros 70% dos participantes classificaram a mesma tarefa como uma tarefa muito fácil.

A tarefa 9 foi classificada como indiferente (neutra) por 30% dos participantes, enquanto que, 20% dos participantes classificaram a mesma tarefa de fácil e a outra metade dos participantes (50%) classificaram a mesma tarefa como uma tarefa muito fácil.

Por último, a tarefa 10 foi classificada por 10% dos participantes como uma tarefa fácil, enquanto que, os outros 90% dos participantes classificaram a tarefa como uma tarefa muito fácil.

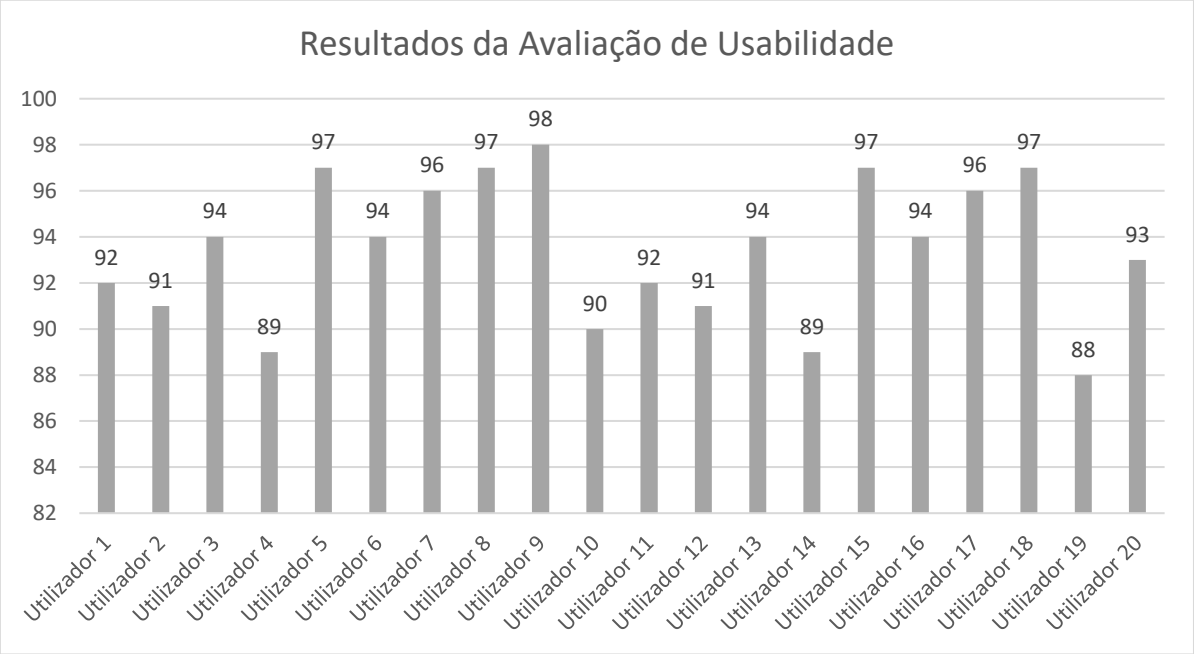


Gráfico 3 - Resultados agregados da avaliação de usabilidade

Média das pontuações das avaliações: **93.45 pontos.**

Os sistemas com a pontuações abaixo do limiar de 60 pontos são classificados por sistemas com baixas experiências e satisfação do utilizador, enquanto os, sistemas com pontuações acima dos 80 pontos são classificados por sistemas com altas experiências e satisfação do utilizador.

Assim, a uma primeira vista de olhos no Gráfico 3, pode se ver logo que a solução apresentada representa um sistema com boas experiências do utilizador e com alto índice de satisfação dos utilizadores pois a menor classificação atribuída ao sistema está acima dos 80 pontos que representa o limite mínimo para que um sistema seja classificado como sistema com altas experiências do utilizador.

Calculando a média das pontuações das avaliações realizadas pelos vinte (20) utilizadores selecionados (**93.45 pontos**), conclui-se que a solução desenvolvida vai de encontro com as expetativas dos utilizadores, ou seja, os utilizadores ficaram satisfeitos com a experiência do utilizador da aplicação, bem como a simplicidade e a facilidade na realização das tarefas na aplicação. Como já foi referido anteriormente, os questionários utilizados para a realização dos testes de usabilidade podem ser consultados na íntegra no **Anexo A - Questionários dos Testes**.

6.4. Conclusão

Conclui-se que a solução apresentada funciona como era esperado e que vai de encontro com as expetativas inicialmente previstas. No que diz respeito à monitorização da aplicação, a mesma executou como era esperado e realizou todas as tarefas ao qual foi submetida, consumindo o mínimo de recurso possível, como se pode ver na Tabela 36. A análise da avaliação dos resultados dos questionários realizados demonstrou que de uma forma geral a solução foi bem aceite pelos participantes pelo que os mesmos classificaram a solução como uma solução fácil ou muito fácil de utilizar. Ainda sobre a avaliação da solução a média das classificações atribuídas pelos participantes do questionário é de **93.45 pontos**, o que significa que a solução apresentada teve um impacto positivo no meio dos participantes dos questionários.

7. Conclusão

7.1. Síntese do Trabalho

Com a conclusão da implementação e análise da solução, agora é momento de realizar o balanço de todo o trabalho desenvolvido e efetuar a retrospetiva das ideias adquiridas ao longo da realização desta dissertação. Como é visível, atualmente o sector das tecnologias de informação é uma das áreas que está em constante evolução e crescimento, pelo que frequentemente surgem novidades a nível das tecnologias. Uma das áreas de tecnologias de informação que se tem beneficiado e crescido muito com esta evolução é o mercado de desenvolvimento de aplicações móveis. Para efetuar o balanço de todo trabalho realizado, dividiu-se a análise por partes para facilitar a sua interpretação e compreensão. Assim iniciou-se o balanço pelo mapeamento dos objetivos da dissertação com a metodologia *Design Science Research* (DSR), seguido do mapeamento dos problemas identificados com as funcionalidades desenvolvidas na solução, e por último foi elaborado uma matriz de o mapeamento dos objetivos, dos problemas e das funcionalidades para uma análise final.

7.1.1. Mapeamento dos Objetivos

O principal objetivo inerente ao desenvolvimento desta dissertação é encontrar uma solução para colmatar os problemas no Centro Hospitalar do Porto (CHP). Com o intuído de cumprir o objetivo da dissertação recorreu-se à metodologia DSR para encontrar uma solução plausível para o problema. Após a análise do problema e de acordo com a metodologia DSR, a solução encontrada foi a criação de um artefacto, uma aplicação *mobile* do género assistente pessoal hospitalar de modo a melhorar a qualidade dos serviços prestados aos utentes no CHP.

Tabela 37 - Mapeamento dos objetivos do projeto com o DSR

	Objetivos da dissertação	***Design Science Research***
O - 1	Evitar longas filas de espera no atendimento para marcação de consultas, evitar o congestionamento de pessoas com senhas à espera de serem atendidas, e facilitar a estadia dos utentes quando dirigem aos CHP.	Identificação do Problema e Motivação
O - 2	Desenvolvimento de um artefacto do género assistente pessoal hospitalar, de modo a melhorar a qualidade dos serviços prestados aos utentes no CHP. Desenvolvimento do estado da arte.	Definição dos Objetivos da Solução
O - 3	Levantamento dos requisitos, *design* e desenvolvimento de uma aplicação mobile (protótipo funcional) para a solução apresentada.	*Design* e Desenvolvimento
O - 4	Implementação da solução desenvolvida, realização dos testes e monitorização da mesma.	Avaliação
O - 5	Apresentação e comunicação dos resultados obtidos com o desenvolvimento da solução.	Comunicação

O mapeamento presente na Tabela 37 pretende estabelecer a relação entre os objetivos da dissertação identificadas e as diferentes fases da metodologia DSR. Na mesma tabela pode-se verificar que em cada fase da metodologia é identificado um conjunto de objetivos para serem cumpridas.

7.1.2. Mapeamento dos Problemas

Devido aos vários problemas identificados durante a realização desta dissertação, sentiu-se a necessidade de criar um mapeamento entre os problemas identificados e as funcionalidades desenvolvidas para solucionar esses problemas, e, por conseguinte, verificar até que ponto essas funcionalidades são capazes de dar respostas aos problemas identificados. A Tabela 38 apresenta esse mapeamento entre os problemas e as funcionalidades.

Tabela 38 - Mapeamento dos problemas com as funcionalidades

	Problemas	Funcionalidade	
P - 1	Segurança dos dados pessoais dos utentes.	Autenticação	**F- 1**
P - 2	Longas filas de espera para marcação de consultas.	Marcação de consultas	**F- 2**
P - 3	Os utentes não tinham acesso às informações das consultas marcadas.	Visualização de consultas marcadas	**F- 3**
P - 4	Os utentes não tinham acesso ao histórico das consultas realizadas.	Visualização de histórico de consultas realizadas	**F- 4**
P - 5	Os utentes não tinham acesso aos seus dados pessoais.	Visualização de dados pessoais	**F- 5**
P - 6	Os utentes não podiam atualizar os seus dados pessoais.	Atualizar dados pessoais (email, morada, contacto)	**F- 6**
P - 7	Não existia nenhuma forma automática de adicionar e sincronizar as consultas com o calendário do dispositivo móvel dos utentes.	Sincronização das datas das consultas com o calendário do dispositivo móvel	**F- 7**
P - 8	Falta de meios para efetuar registo de presenças dentro da unidade hospitalar.	Serviço de localização	**F- 8**
P - 9	Muitas pessoas na fila de espera para serem atendidas.	Serviço de senhas	**F- 9**

Para cada um dos problemas identificados foi desenvolvido uma funcionalidade para solucionar o mesmo, de modo a satisfazer ainda mais os utentes do CHP. Foi ainda realizado um conjunto de testes para certificar que essas funcionalidades conseguem dar respostas aos problemas apresentados com a total segurança e integridade dos dados.

7.1.3. Consideração Final

As tabelas (Tabela 37 e Tabela 38) apresentam o mapeamento entre os objetivos da dissertação e a metodologia DSR e o mapeamento entre os problemas identificados com as funcionalidades da solução apresentada respetivamente. Esses mapeamentos foram criados com o objetivo de efetuar uma comparação direta entre os objetivos da dissertação e a fase da metodologia DSR e também entre os problemas identificados e as funcionalidades desenvolvidas de modo a tornar a sua compreensão dos problemas identificados e as soluções encontradas para esses problemas.
No primeiro mapeamento presente na primeira tabela (Tabela 37), a primeira fase consiste na identificação do problema e motivação para a realização desta dissertação. Os problemas identificados que levaram a realização desta dissertação são: as longas filas de espera no atendimento que os utentes estavam sujeitos, e o congestionamento de pessoas no CHP. A motivação que levou a realização da dissertação é o desenvolvimento de uma solução (criação de um artefacto) que pudesse colmatar os problemas identificados no CHP, de modo a melhor a qualidade dos serviços prestados aos utentes e ainda facilitar a estadia dos utentes naquela unidade hospitalar.

Ainda no mesmo mapeamento, mas na segunda fase, procedeu-se então à definição dos objetivos da solução. Com o problema já identificado, seguiu-se com a definição dos objetivos da dissertação. O objetivo consiste em encontrar de uma solução do género do assistente pessoal hospitalar (artefacto), com o intuito de introduzir melhorias na qualidade dos serviços prestados aos utentes no CHP.

Com os objetivos da dissertação definidos, o passo seguinte consistiu no desenvolvimento da revisão de literatura, que representa a base teórica de suporte para o trabalho prático desenvolvido. No desenvolvimento da revisão de literatura foram abordados vários temas que dizem respeito ao desenvolvimento de aplicações móveis. Inicialmente, foi realizado um pequeno estudo sobre os dispositivos e plataformas móveis, seguido de um estudo sobre a abordagem *Bring Your Own Device* (BYOD) ou então "traga o seu próprio dispositivo" em português, no qual foi abordado este tópico de uma forma geral e de uma forma particular direcionada para o BYOD no *Healthcare*. Também foi apresentado os casos de estudos de utilização de dispositivos móveis no *Healthcare.* Realizou-se também uma análise comparativa sobre as metodologias de desenvolvimento de aplicações móveis, com destaque a abordagem de desenvolvimento

multiplataforma. Outro estudo realizado foi sobre os critérios de seleção das abordagens de desenvolvimento multiplataforma, bem como uma análise comparativa entre as abordagens de desenvolvimento multiplataforma e análise comparativa entre o desenvolvimento de aplicações nativas e aplicações multiplataformas.

Efetuou-se ainda um estudo das tecnologias de desenvolvimento multiplataforma, na qual foram estudados um conjunto de oito (8) tecnologias diferentes. Com base nesse conjunto de tecnologias e procedeu-se a escolha das tecnologias para o desenvolvimento desta dissertação. O que se concluiu da realização da revisão de literatura, é que a escolha de uma abordagem de desenvolvimento multiplataforma depende principalmente da exigência do tipo de aplicação, bem como também das plataformas alvo, do tipo de aplicação, o acesso aos dados e recursos nativos (*hardware*), da aparência, da interface do utilizador, do desempenho da própria aplicação, do mercado de distribuição.

A seleção de uma abordagem de desenvolvimento multiplataforma em vez de outra abordagem depende principalmente do tipo de aplicação e das suas necessidades, pelo que cada caso é um caso, o que resulta por vezes na escolha de uma abordagem multiplataforma. Isto porque este pode trazer mais benefícios para uma determinada aplicação e não trazer muitos benefícios para outras aplicações. Para este projeto, a abordagem de desenvolvimento multiplataforma escolhida foi a abordagem híbrida, devido ao tipo de aplicação que era necessário desenvolver. A aplicação é do tipo cliente-servidor, pois tanto o cliente como o servidor estão envolvidos no processo de execução e processamento de dados utlizados na aplicação.

A estratégia de desenvolvimento de aplicações híbridas é recente e apresenta ainda algumas limitações, mas trata-se de uma estratégia de desenvolvimento de aplicações que apresenta um enorme potencial no mercado de desenvolvimento de aplicações móveis. Mas, no entanto, trata-se de uma estratégia de desenvolvimento que ainda tem um longo percurso pela frente para que se torne uma estratégia sólida e confiável dentro e fora da comunidade informática. Fatores como a melhoria progressiva da capacidade dos dispositivos móveis, a evolução do HTML5 e do CSS3 são os pilares que podem tornar esta estratégia ainda mais popular dentro e fora desta comunidade.

Com a da revisão de literatura terminada e antes de começar fase de desenvolvimento procedeu-se a realização do levantamento de requisitos e a especificação da solução e ao *design* da solução, correspondente a fase de *design* da metodologia DSR. Efetuou-se então o processo de levantamento e especificação dos requisitos e da arquitetura do sistema. Neste caso, os requisitos foram agrupados em duas categorias, sendo elas: requisitos funcionais e requisitos não funcionais. Ainda, foi apresentado e especificado toda a arquitetura da solução. Ainda nesta fase foram apresentados os primeiros *designs mockups* da solução final.

Depois, efetuou-se então a seleção da tecnologia de desenvolvimento e o consequentemente o desenvolvimento e implementação da solução, que corresponde a fase de desenvolvimento da metodologia DSR. A tecnologia eleita para o desenvolvimento da aplicação foi o **Angular (Mobile UI)** + **PhoneGap**.

Com o desenvolvimento da solução terminado, procedeu-se a fase de implementação e testes da solução, cujo principal objetivo era a identificação de erros e bugs, para que possam ser ajustados e a consequente avaliação dos resultados obtidos. Trata-se de uma fase importante pois é a fase onde a solução é testada todas as suas funcionalidades e é analisado o comportamento da mesma. Neste ponto é importante fazer referência a Tabela 38, pois retrata o mapeamento entre os diversos problemas identificados e as funcionalidades que foram desenvolvidas para colmatar esses problemas.

A última fase a ser cumprida é a comunicação dos resultados alcançados, com vista a efetuar um balanço final de tudo o que foi proposto no início da dissertação e tudo que foi possível realizar na mesma. Nesta fase é comunicada todos os resultados obtidos sobre a implementação e testes da solução e também os resultados obtidos a nível da realização da dissertação em si. Resumindo e concluindo apesar dos desafios encontrados durante a sua realização, foi possível cumprir com sucesso todos os objetivos inicialmente definidos para a realização desta dissertação.

É de realçar que durante a realização desta dissertação foi possível escrever dois artigos científicos. A primeira foca essencialmente nos benefícios do *Bring Your Own Device in Healthcare* e a segunda foca essencialmente no estudo e na análise dos resultados obtidos com o desenvolvimento do artefacto e no impacto que este teve no CHP.

Estes artigos acabaram por ser outra forma de comunicação dos resultados obtidos com a realização desta dissertação. Estes foram realizados no âmbito da avaliação e comunicação dos resultados obtidos que por sua vez corresponde as duas últimas fases da metodologia DSR que correspondem aos objetivos 4 (O - 4) e 5 (O - 5) da Tabela 37. As informações sobre os artigos podem ser consultadas no **Anexo C – Publicações**.

Tabela 39 - Matriz cruzamento Funcionalidades, Objetivos e Problemas

	Objetivos					
Problemas	**O - 1**	**O - 2**	**O - 3**	**O - 4**	**O - 5**	**Funcionalidades**
P – 1	F	F, P	F, P	F		**F – 1**
P – 2	P	F, P	F, P	F		**F – 2**
P – 3		F, P	F, P	F		**F – 3**
P – 4		F, P	F, P	F		**F – 4**
P – 5		F, P	F, P	F		**F – 5**
P – 6		F, P	F, P	F		**F – 6**
P – 7		F, P	F, P	F		**F – 7**
P – 8		F, P	F, P	F		**F – 8**
P – 9		F, P	F, P	F		**F – 9**

OBS: F: Funcionalidade, O: Objetivo, P: Problema

A matriz presente na Tabela 39 apresenta o cruzamento entre as funcionalidades da solução desenvolvida, os objetivos da dissertação e os problemas que motivaram o desenvolvimento desta dissertação. Para terminar, com a solução desenvolvida, o processo de marcação de consulta passa a ser mais rápido e muito mais eficiente, a possibilidade de tirar senhas sem estar presente no local é uma funcionalidade que foi desenvolvida com o intuído de evitar o congestionamento de utentes na fila de espera, ou seja, os utentes não precisam de estar sempre presente no local a espera da sua vez para serem atendidos. A funcionalidade de registo de presença (serviço de localização) é uma funcionalidade que foi implementada para ajudar os utentes a se localizarem em determinados lugares dentro do centro hospitalar e ainda efetuar o registo de presenças quando estiverem no lugar correto.

7.2. Trabalhos Futuros

Este subcapítulo destina aos trabalhos futuros que podem ser realizados no âmbito da realização deste projeto de dissertação. O principal trabalho futuro após a finalização desta dissertação é a instalação e o teste da solução no Centro Hospitalar do Porto (CHP), seguido da monotorização da mesma.

Depois de estar implementada a solução no CHP pode efetuar uma serie de melhorias e aperfeiçoamento na solução apresentada para melhorar cada vez mais a qualidade dos serviços prestados aos utentes. Nesta fase os objetivos definidos para este projeto foram cumpridos com sucesso, mas ainda há margem para melhorar alguns aspetos que podem ser explorados e aperfeiçoados com o intuito de cativar os utentes e futuros utilizadores da aplicação.

Algumas das sugestões de melhoramento que podem ser exploradas e aperfeiçoada com o intuito de expandir este projeto são:

- A interface gráfica podia ser melhorada, para a tornar ainda mais simples e atrativa para os utentes;
- A introdução de novas línguas na aplicação, uma vez que não são apenas utentes que falam a língua portuguesa que frequentam o centro hospitalar. Assim, podia se então adicionar novas línguas tais como: o Inglês e Francês.
- Gerar a mesma aplicação para outras plataformas móveis com elevada cota do mercado tais como o *iOS* e o *Windows Phone*, para que todos os utentes possam utilizar a aplicação. Nesta primeira fase, a aplicação foi desenvolvida apenas para dispositivos móveis com *Android*, mas consoante a sua popularidade, poderá ser distribuída a mesma aplicação para outras plataformas móveis.
- Desenvolvimento e incorporação de novas funcionalidades na aplicação para poder abranger mais leque de atividades.
- Desenvolver a mesma aplicação recorrendo a outras tecnologias (*framework*) de desenvolvimento de aplicações híbridas que permite melhores índices de desempenho.
- Implementar mais níveis de segurança na própria aplicação e no servidor, bem como no meio de comunicação entre ambas as partes, para tornar a solução assim mais segura, mais robusta, pois os dados partilhados entre a aplicação e o servidor são confidenciais e

precisam de encriptados durante o processo de troca de dados e mesmo no próprio dispositivo móvel estes precisam de ser encriptados.

- A incorporação da planta do edifício do centro hospitalar em 3D seria uma grande vantagem para aplicação, no caso do serviço de localização e para os seus utentes que passariam a usufruir de um serviço de melhor qualidade ainda.
- A incorporação de realidade virtual na aplicação para ajudar os utentes a identificar em tempo real determinados locais no centro hospitalar.

8. Referências

Adobe Systems Inc. (2016). PhoneGap. Obtido 7 de Novembro de 2016, de http://phonegap.com/

AllianceTek Inc. (2016). Comparison Chart for Mobile App Development Methods | Article | AllianceTek Inc. USA. Obtido de http://www.alliancetek.com/article-comparison-chart.html

Anzures-Garcia, M., Sanchez-Galvez, L. A., Hornos, M. J., & Paderewski-Rodriguez, P. (2016). MVC Design Pattern Based-Development of Groupware. Em *2016 4th International Conference in Software Engineering Research and Innovation (CONISOFT)* (pp. 71–80). IEEE. https://doi.org/10.1109/CONISOFT.2016.20

Appcelerator Inc. (2016). Mobile App Development Platform & MBaaS | Appcelerator. Obtido 7 de Novembro de 2016, de http://www.appcelerator.com/

Apple Inc. (2014). About the iOS Technologies. Obtido 13 de Outubro de 2016, de https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html

Araújo, M. A. (2005). Web services na informação geográfica. Obtido de http://repositorium.sdum.uminho.pt/handle/1822/4570

Arjun Goud, M., Vijaya, - M, Rao, B., & Rao, -Prof V Purnachandra. (2013). MOBILE DEVICES OVERVIEW. *International Journal of Computer Science and Informatics*, (PRINT), 2231–5292.

Bello Garba, A., Armarego, J., Murray, D., & Kenworthy, W. (2015). Review of the Information Security and Privacy Challenges in Bring Your Own Device (BYOD) Environments. *Journal of Information Privacy and Security*, *11*, 38–54. https://doi.org/10.1080/15536548.2015.1010985

BlackBerry OS. (2016). BlackBerry OS 10 – BlackBerry 10.3 OS Software Features - Global. Obtido 14 de Outubro de 2016, de http://global.blackberry.com/en/software/smartphones/blackberry-10-os.html

Bootstrap. (2016). Getting started · Bootstrap. Obtido 21 de Dezembro de 2016, de http://getbootstrap.com/getting-started/

Brown, S. (2010). Likert Scale Examples for Surveys. *Iowa State University*, 1–4. https://doi.org/10.1002/9780470479216.corpsy0508

Brzeziński, J., Kobusińska, A., Kobusiński, J., Stroiński, A., & Szałkowski, K. (2013). Mobile Platform for Executing Medical Business Processes and Data Collecting (pp. 149–159). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-37899-7_13

Cardoso, L. (2013). Desenvolvimento de uma Plataforma baseada em Agentes para a Interoperabilidade. Obtido de http://hdl.handle.net/1822/27770

Chao Wang, Wei Duan, Jianzhang Ma, & Chenhui Wang. (2011). The research of Android System architecture and application programming. Em *Proceedings of 2011 International Conference on Computer Science and Network Technology* (pp.

785–790). IEEE. https://doi.org/10.1109/ICCSNT.2011.6182081

Charkaoui, S., Adraoui, Z., & Benlahmar, E. H. (2014). Cross-platform mobile development approaches. Em *2014 Third IEEE International Colloquium in Information Science and Technology (CIST)* (pp. 188–191). IEEE. https://doi.org/10.1109/CIST.2014.7016616

Ciman, M., & Gaggi, O. (2014). Evaluating impact of cross-platform frameworks in energy consumption of mobile applications. Em *WEBIST 2014 - Proceedings of the 10th International Conference on Web Information Systems and Technologies* (Vol. 1, pp. 423–431). Obtido de http://www.scopus.com/inward/record.url?eid=2-s2.0-84902383531&partnerID=40&md5=e6ed54c3e9bc375baf038d2de81a9ccb

Core Communique. (2016). myCOL - A unique healthcare app launched to simplify hospitalization. Obtido 15 de Dezembro de 2016, de http://corecommunique.com/mycol-a-unique-healthcare-app-launched-to-simplify-hospitalization/

Dalmasso, I., Datta, S. K., Bonnet, C., & Nikaein, N. (2013). Survey, comparison and evaluation of cross platform mobile application development tools. Em *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)* (pp. 323–328). IEEE. https://doi.org/10.1109/IWCMC.2013.6583580

Delia, L., Galdamez, N., Thomas, P., Corbalan, L., & Pesado, P. (2015). Multi-Platform Mobile Application Development Analysis, 0–5.

Dhingra, M. (2016). Legal Issues in Secure Implementation of Bring Your Own Device (BYOD). *Procedia Computer Science*, *78*(December 2015), 179–184. https://doi.org/10.1016/j.procs.2016.02.030

Duarte, J., Salazar, M., Quintas, C., Santos, M., Neves, J., Abelha, A., & Machado, J. (2010). Data Quality Evaluation of Electronic Health Records in the Hospital Admission Process. *Computer and Information Science (ICIS), 2010 IEEE/ACIS 9th International Conference on*, 201–206. https://doi.org/10.1109/ICIS.2010.97

Edureka. (sem data). AngularJS for Absolute Beginners : Medialoot. Obtido 14 de Setembro de 2017, de https://www.slideshare.net/EdurekaIN/angularjs-for-beginners-55737689

Falk Markus, B. S. (2016). Mobile Frameworks Comparison Chart. Obtido 13 de Novembro de 2016, de http://mobile-frameworks-comparison-chart.com/

Feij, C. J., & Ramalho, C. (2004). Web Services : Metodologias de Desenvolvimento, 1–15.

Fh, D. I., & Haberl, N. (2015). University of Applied Sciences Cross Platform Development Possibilities and drawbacks of the Xamarin platform Table of Content Table of Content, (August).

Google. (2015). AngularJS — Superheroic JavaScript MVW Framework. Obtido 12 de Janeiro de 2017, de https://angularjs.org/

Gottschalk, K., Graham, S., Kreger, H., & Snell, J. (2002). Introduction to Web services architecture. *IBM System Journal*, *41*(2), 170–177. https://doi.org/10.1147/sj.412.0170

Gregor, S., & Hevner, A. R. (2013). P OSITIONING AND P RESENTING D ESIGN S CIENCE Types of Knowledge in Design Science Research. *MIS Quarterly*, *37*(2), 337–355. https://doi.org/10.2753/MIS0742-1222240302

Gronli, T.-M., Hansen, J., Ghinea, G., & Younas, M. (2014). Mobile Application Platform Heterogeneity: Android vs Windows Phone vs iOS vs Firefox OS. Em *2014 IEEE 28th International Conference on Advanced Information Networking and Applications* (pp. 635–641). IEEE. https://doi.org/10.1109/AINA.2014.78

Hammershoj, A., Sapuppo, A., & Tadayoni, R. (2010). Challenges for mobile application development. Em *2010 14th International Conference on Intelligence in Next Generation Networks* (pp. 1–8). IEEE. https://doi.org/10.1109/ICIN.2010.5640893

Hee-Yeon Cho, Choon-Sung Nam, & Dong-Ryeol Shin. (2010). A compariosn of open and closed mobile platforms. Em *2010 International Conference on Electronics and Information Engineering* (pp. V2-141-V2-143). IEEE. https://doi.org/10.1109/ICEIE.2010.5559730

Heitkötter, H., Hanschke, S., & Majchrzak, T. A. (2013). Evaluating Cross-Platform Development Approaches for Mobile Applications (pp. 120–138). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-36608-6_8

Heitkötter, H., Majchrzak, T., Ruland, B., & Weber, T. (2013). Evaluating Frameworks for Creating Mobile Web Apps. Em *WEBIST 2013 - 9th International Conference on Web Information Systems and Technologies* (pp. 209–221). https://doi.org/10.5220/0004356702090221

Hernandez, I. M. T., Viveros, A. M., & Rubio, E. H. (2013). Analysis for the design of open applications on mobile devices. Em *CONIELECOMP 2013, 23rd International Conference on Electronics, Communications and Computing* (pp. 126–131). IEEE. https://doi.org/10.1109/CONIELECOMP.2013.6525772

Hevner, A., & Chatterjee, S. (2010). *Design Research in Information Systems* (Vol. 22). Boston, MA: Springer US. https://doi.org/10.1007/978-1-4419-5653-8

Ibm. (2005). Web Services Overview. *IBM Corporation*. Obtido de http://publib.boulder.ibm.com/infocenter/iadthelp/v6r0/index.jsp?topic=/com.ibm.etools.webservice.doc/concepts/cws.html

IBM. (2012). Native, web or hybrid mobile-app development. *Thought Leadership White Paper*, 0–7.

IDM International Data Corporation. (2016). IDC: Smartphone OS Market Share 2016, 2015. Obtido 17 de Outubro de 2016, de http://www.idc.com/prodserv/smartphone-os-market-share.jsp

Kaur, P., & Sharma, S. (2014). Google Android a mobile platform: A review. Em *2014 Recent Advances in Engineering and Computational Sciences (RAECS)* (pp. 1–5). IEEE. https://doi.org/10.1109/RAECS.2014.6799598

Kestle, R., & Self, R. (2013). IS Practices for SME Success Series. *IS Practices for SME Success Series*, *1*(1), 1–148. https://doi.org/10.1093/itnow/bws010

Liu, J., & Yu, J. (2011). Research on Development of Android Applications. Em *2011 4th International Conference on Intelligent Networks and Intelligent Systems* (pp. 69–72). IEEE. https://doi.org/10.1109/ICINIS.2011.40

Lund, D., & Dunbrack, L. (2015). The Healthcare Industry: Embracing BYOD for Success IDC OPINION.

Mahmoud, Q., & Maamar, Z. (2006). Applying the MVC Design Pattern to Multi-Agent Systems. Em *2006 Canadian Conference on Electrical and Computer Engineering* (pp. 2420–2423). IEEE. https://doi.org/10.1109/CCECE.2006.277427

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems*, *15*(4), 251–266. https://doi.org/10.1016/0167-9236(94)00041-2

Marins, F. de A. (2013). *Monitorização e prevenção em plataformas de interoperabilidade hospitalar*. Minho University. Obtido de http://hdl.handle.net/1822/27773

Masoud, F. A., Halabi, D. H., & Halabi, D. H. (2006). ASP.NET and JSP frameworks in model view controller implementation. Em *Proceedings - 2006 International Conference on Information and Communication Technologies: From Theory to Applications, ICTTA 2006* (Vol. 2, pp. 3593–3598). IEEE. https://doi.org/10.1109/ICTTA.2006.1684998

Moyer, J. E. (2013). Managing Mobile Devices in Hospitals: A Literature Review of BYOD Policies and Usage. *Journal of Hospital Librarianship*, *13*, 197–208. https://doi.org/10.1080/15323269.2013.798768

myCol. (2016). myCOL - Personalized Hospital Assistance | Health Insurance Claim Form. Obtido 15 de Dezembro de 2016, de http://www.mycol.in/

Nitish Kirtiraj Shah, B. (sem data). Developing a decision support framework for planning and implementing Bring Your Own Device programmes in organizations, 2013–2015.

Nosrati, M., Hojat, R. K., & Hasanvand, A. (2012). Mobile Computing: Principles, Devices and Operating Systems. *World Applied Programming*, *2*(27), 399–408.

Nosrati, M., Karimi, R., & Hasanvand, H. (2012). Mobile Computing: Principles, Devices and Operating Systems. *World Applied Programming*, *2*(7), 399–408. https://doi.org/10.1002/wcm.1203/abstract

Nuls, A. M. P. Les. (2015). AngularJS: Créer une application mobile avec Cordova/PhoneGap, AngularJS et Bootstrap – Partie 1 – Application Mobile Pour Les Nuls. Obtido 14 de Setembro de 2017, de https://www.application-mobile-pour-les-nuls.fr/apache-cordova/angularjs-phonegap-creer-une-application-mobile-native

Ocano, S. G., Ramamurthy, B., & Wang, Y. (2015). Remote mobile screen (RMS): An

approach for secure BYOD environments. Em *2015 International Conference on Computing, Networking and Communications (ICNC)* (pp. 52–56). IEEE. https://doi.org/10.1109/ICCNC.2015.7069314

Oliver, E., & Earl. (2009). A survey of platforms for mobile networks research. *ACM SIGMOBILE Mobile Computing and Communications Review*, *12*(4), 56. https://doi.org/10.1145/1508285.1508292

Palmieri, M., Singh, I., & Cicchetti, A. (2012). Comparison of cross-platform mobile development tools. Em *2012 16th International Conference on Intelligence in Next Generation Networks, ICIN 2012* (pp. 179–186). IEEE. https://doi.org/10.1109/ICIN.2012.6376023

Pautasso, C. (2009). RESTful Web service composition with BPEL for REST. *Data & Knowledge Engineering*, *68*(9), 851–866. https://doi.org/10.1016/j.datak.2009.02.016

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Pop, D.-P., & Altar, A. (2014). Designing an MVC Model for Rapid Web Application Development. *Procedia Engineering*, *69*, 1172–1179. https://doi.org/10.1016/j.proeng.2014.03.106

Rahul Raj, C. ., & Seshu Babu Tolety. (2012). A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. Em *2012 Annual IEEE India Conference (INDICON)* (pp. 625–629). IEEE. https://doi.org/10.1109/INDCON.2012.6420693

Ribeiro, A., & da Silva, A. R. (2012). Survey on Cross-Platforms and Languages for Mobile Apps. Em *2012 Eighth International Conference on the Quality of Information and Communications Technology* (pp. 255–260). IEEE. https://doi.org/10.1109/QUATIC.2012.56

Scandurra, P., & Rosario, M. (sem data). Native versus Cross-platform frameworks for mobile application development, 4.

Scarfo, A. (2012). New security perspectives around BYOD. Em *Proceedings - 2012 7th International Conference on Broadband, Wireless Computing, Communication and Applications, BWCCA 2012* (pp. 446–451). IEEE. https://doi.org/10.1109/BWCCA.2012.79

Scrum Guides. (2016). Scrum Guide | Scrum Guides. Obtido 19 de Outubro de 2016, de http://www.scrumguides.org/download.html

Selfa, D. M., Carrillo, M., & Del Rocío Boone, M. (2006). A database and web application based on MVC architecture. Em *Proceedings of the 16th IEEE International Conference on Electronics, Communications and Computers, CONIELECOMP 2006* (Vol. 2006, p. 48). IEEE. https://doi.org/10.1109/CONIELECOMP.2006.6

Sencha Inc. (2016). Cross-platform Mobile Web App Development Framework for HTML5 and JS | Sencha. Obtido de https://www.sencha.com/products/touch/#overview

Serrano, N., Hernantes, J., & Gallardo, G. (2013). Mobile Web Apps. *IEEE Software*, *30*(5), 22–27. https://doi.org/10.1109/MS.2013.111

Smutny, P. (2012). Mobile development tools and cross-platform solutions. Em *Proceedings of the 13th International Carpathian Control Conference (ICCC)* (pp. 653–656). IEEE. https://doi.org/10.1109/CarpathianCC.2012.6228727

Sommer, A., & Krusche, S. (2013). Evaluation of cross-platform frameworks for mobile applications. *Proceedings of the 1st European Workshop on Mobile Engineering*, 363–376. Obtido de http://subs.emis.de/LNI/Proceedings/Proceedings215/363.pdf

TechTarget. (2016). What is MDM, MAM, and MIM? (And what's the difference?). Obtido 24 de Novembro de 2016, de http://www.brianmadden.com/opinion/What-is-MDM-MAM-and-MIM-And-whats-the-difference

Thakare, B. S., Shirodkar, D., Parween, N., & Parween, S. (2014). State of Art Approaches to Build Cross Platform Mobile Application. *International Journal of Computer Applications*, *107*(20), 975–8887.

Tool, C. T., Tool, C., & Enterprise, L. (2014). Cross-Platform Tool Benchmarking Find the right tool for your app project July 2014, (July).

Trusted Reviews. (2016). Siri vs Google Now vs Cortana: Which is best? Obtido 15 de Dezembro de 2016, de http://www.trustedreviews.com/opinions/siri-vs-google-now-vs-cortana

Turban, E., & Sharda, R. (2010). Decision support and business intelligence systems. *portal.acm.org*. Obtido de http://portal.acm.org/citation.cfm?id=1840968

Tutorialspoint. (2014). Bootstrap - Overview. Obtido 21 de Dezembro de 2016, de https://www.tutorialspoint.com/bootstrap/bootstrap_overview.htm

Tutorialspoint. (2016). Angularjs Tutorial. Obtido 14 de Novembro de 2016, de https://www.tutorialspoint.com/angularjs/

Vinoski, S. (2008). RESTful web services. *IEEE Internet Computing*, *12*(6), 46--50. https://doi.org/10.1109/MIC.2008.130

Walls, J. G., Widmeyer, G. R., & El Sawy, O. A. (1992). Building an Information System Design Theory for Vigilant EIS. *Information Systems Research*, *3*(1), 36–59. https://doi.org/10.1287/isre.3.1.36

Wang, Y., Wei, J., & Vangury, K. (2014). Bring your own device security issues and challenges. *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, 80–85. https://doi.org/10.1109/CCNC.2014.6866552

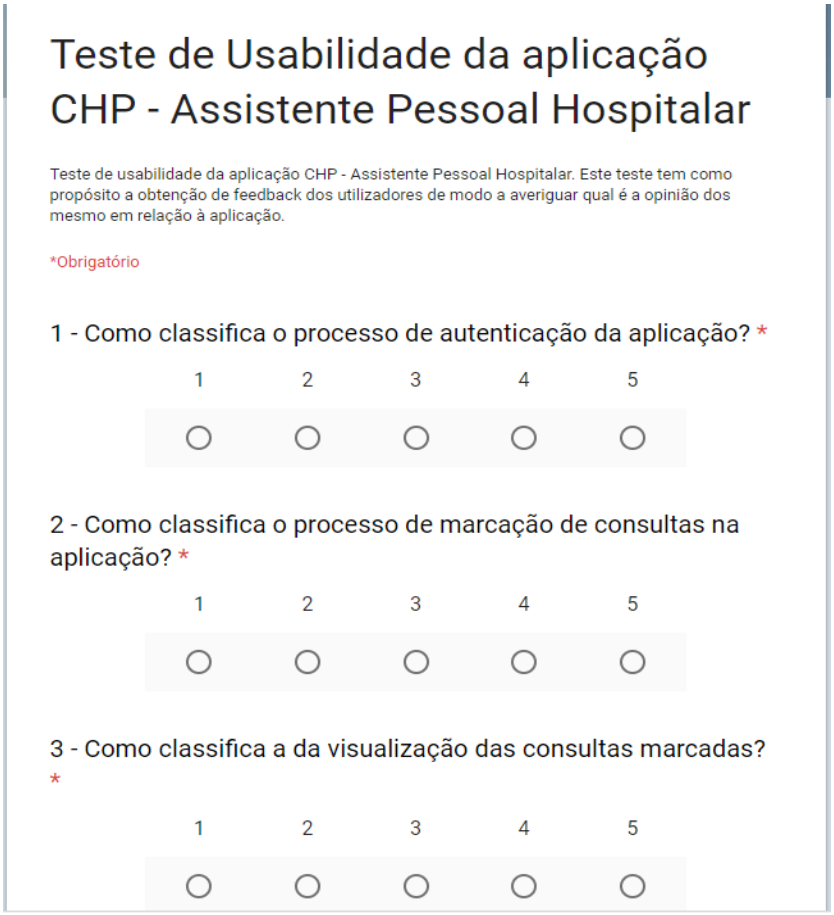
Xanthopoulos, S., & Xinogalos, S. (2013). A comparative analysis of cross-platform development approaches for mobile applications. Em *Proceedings of the 6th*

Balkan Conference in Informatics on - BCI '13 (p. 213). New York, New York, USA: ACM Press. https://doi.org/10.1145/2490257.2490292

9. Anexos

Anexo A - Questionários dos Testes

i. Testes de Usabilidade



Teste de Usabilidade da aplicação CHP - Assistente Pessoal Hospitalar

Teste de usabilidade da aplicação CHP - Assistente Pessoal Hospitalar. Este teste tem como propósito a obtenção de feedback dos utilizadores de modo a averiguar qual é a opinião dos mesmo em relação à aplicação.

*Obrigatório

1 - Como classifica o processo de autenticação da aplicação? *

1 2 3 4 5

2 - Como classifica o processo de marcação de consultas na aplicação? *

1 2 3 4 5

3 - Como classifica a da visualização das consultas marcadas? *

1 2 3 4 5

4 - Como classifica a funcionalidade de serviço de localização? *

1	2	3	4	5
○	○	○	○	○

5 - Como classifica o processo de registo de presenças? *

1	2	3	4	5
○	○	○	○	○

6 - Como classifica a funcionalidade de serviço de senhas? *

1	2	3	4	5
○	○	○	○	○

7 - Como classifica o processo de tirar as senhas? *

1	2	3	4	5
○	○	○	○	○

8 - Como classifica a visualização e o facto de poder editar os dados pessoais? *

1	2	3	4	5
○	○	○	○	○

9 - Como classifica a sincronização com o calendário do dispositivo móvel? *

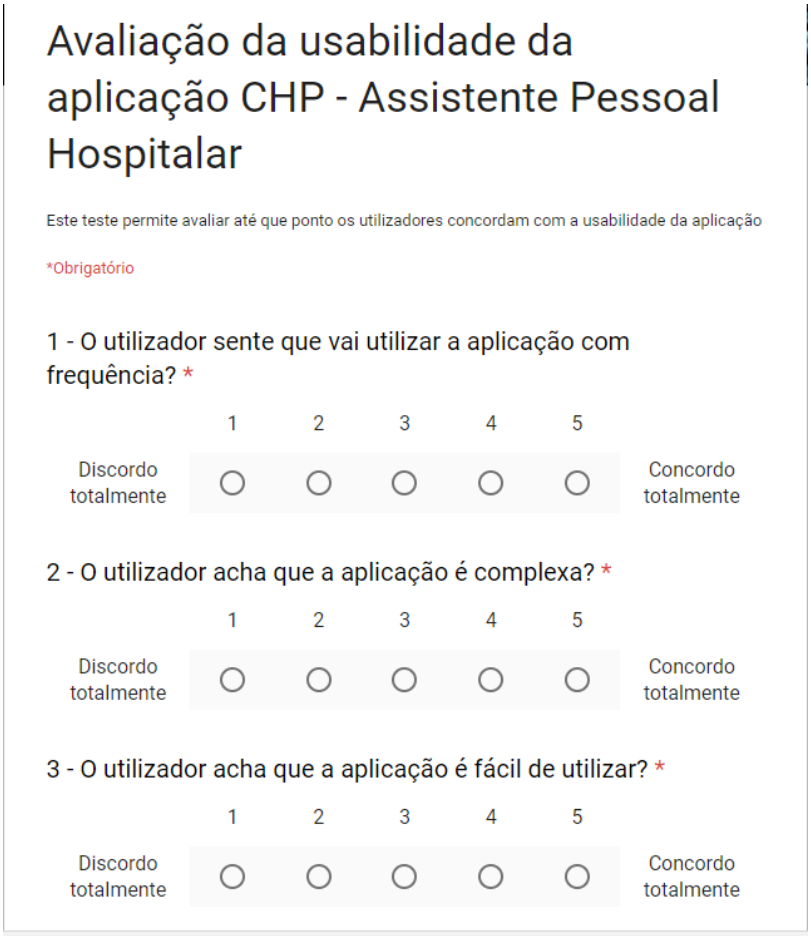
1	2	3	4	5
○	○	○	○	○

10 - No geral, como classifica a aparência da aplicação? *

1	2	3	4	5
○	○	○	○	○

SUBMETER

ii. Avaliação de Usabilidade



Avaliação da usabilidade da aplicação CHP - Assistente Pessoal Hospitalar

Este teste permite avaliar até que ponto os utilizadores concordam com a usabilidade da aplicação

*Obrigatório

1 - O utilizador sente que vai utilizar a aplicação com frequência? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

2 - O utilizador acha que a aplicação é complexa? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

3 - O utilizador acha que a aplicação é fácil de utilizar? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

4 - O utilizador acha que vai precisar de ajuda para utilizar a aplicação? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

5 - O utilizador acha que as funcionalidades da aplicação foram bem integradas? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

6 - O utilizador acha que a aplicação inconsistente? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

7 - O utilizador acha que a maioria das pessoas conseguem usar a aplicação e tirar bom proveito da mesma? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

8 - O utilizador em algum momento sentiu que a aplicação é complicada de se utilizar? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

9 - O utilizador recomendaria a aplicação às outras pessoas? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

10 - O utilizador faz uma apreciação positiva depois de ter utilizado a aplicação? *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

SUBMETER

Anexo B – Funcionalidades e Configuração do Servidor

Serviço de Senhas

Figura B 1 - Interface de Gestão de Senhas

A Figura B 1 ilustra a interface do serviço de senhas no servidor. Na mesma é apresentada os serviços disponíveis, bem como a fila de cada serviço, o número de senhas, o número de senhas em atendimento e o tempo médio de espera de cada serviço. Ainda se tem o botão "próxima senhas" que serve para chamar a próxima senha.

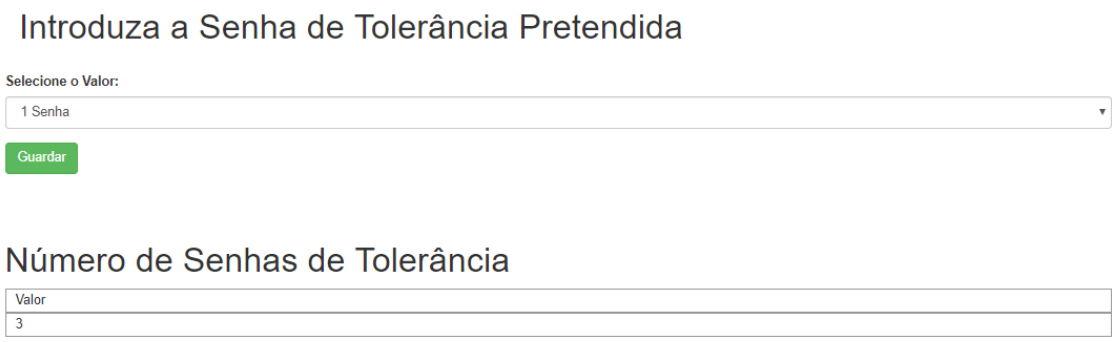


Figura B 2 - Interface de senhas de tolerância

A Figura B 2 ilustra a interface das senhas de tolerância. A mesma apresenta uma *drop down list* com os números das senhas de tolerância. Ainda tem o botão "guardar" para guardar as alterações efetuadas. Na parte inferior, tem-se a informação com o valor atual da senha de tolerância.

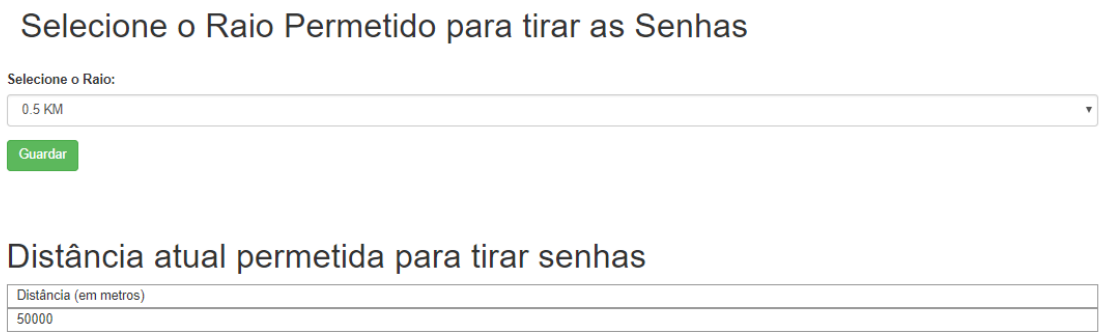


Figura B 3 - Interface do raio permitido para tirar senhas

A Figura B 3 ilustra a interface do raio permitido para tirar senhas. A mesma apresenta uma *drop down list* com os valores da distancia permitida para tirar senhas. Ainda tem o botão "guardar" para guardar as alterações efetuadas. Na parte inferior, tem-se a informação com o valor atual da distância permitida.

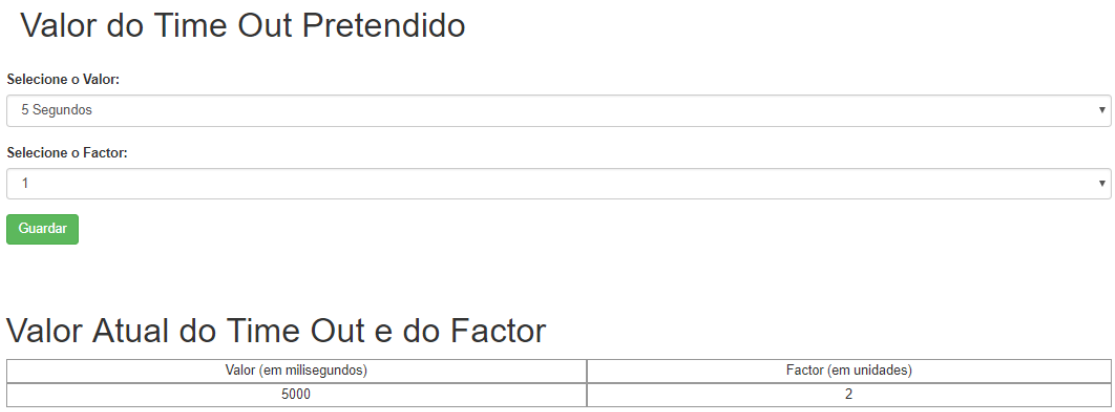


Figura B 4 - Interface de time out

A Figura B 4 ilustra a interface de *time out* e do fator da mesma. A mesma apresenta uma *drop down list* com os valores de *time out* pretendida e outra para o fator pretendido. Ainda tem o botão "guardar" para guardar as alterações efetuadas. Na parte inferior, tem-se as informações referentes aos valores atuais do *time out* e do fator.

Serviço de Localização

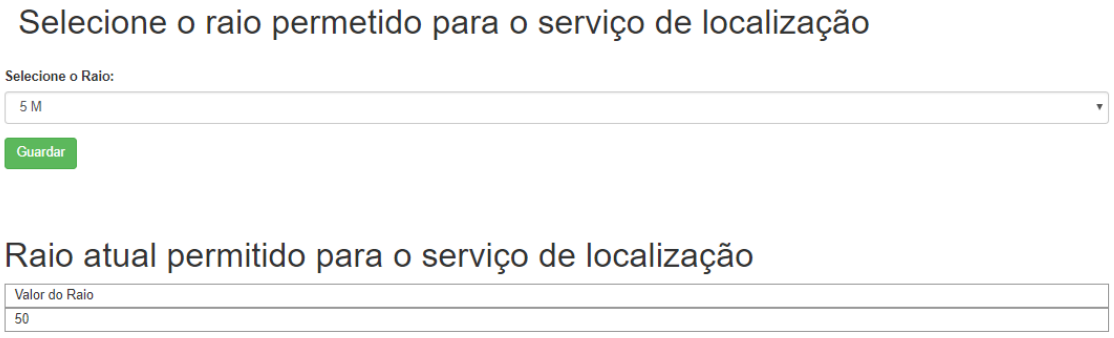


Figura B 5 - Interface de raio permitido para o registo de localização

A Figura B 5 ilustra a interface do raio permitido para o registo de presenças no serviço de localização. A mesma apresenta uma *drop down list* com os valores do raio efetuar o registo de presenças. Ainda tem o botão "guardar" para guardar as alterações efetuadas. Na parte inferior, tem-se a informação com o valor atual do raio permitido.

Figura B 6 - Interface de criar eventos

A Figura B 6 ilustra a interface de criar eventos no serviço de localização. Para criar o evento é necessário preencher o formulário acima apresentado. Para tal tem-se de preencher o formulário com o nome do evento, o local do evento, o código postal do local do evento, as coordenadas geográficas (latitude e longitude) do evento e data e hora da realização do evento. Ainda se tem o botão "adicionar" que serve para criar/registar novos eventos. Em caso de não se conhecer as coordenadas geográficas do local do evento, o botão "minha localização" serve para identificar a localização atual pretendida para a realização do evento.

A Figura B 7 ilustra a interface de visualização dos eventos criados e dos detalhes de cada um dos eventos.

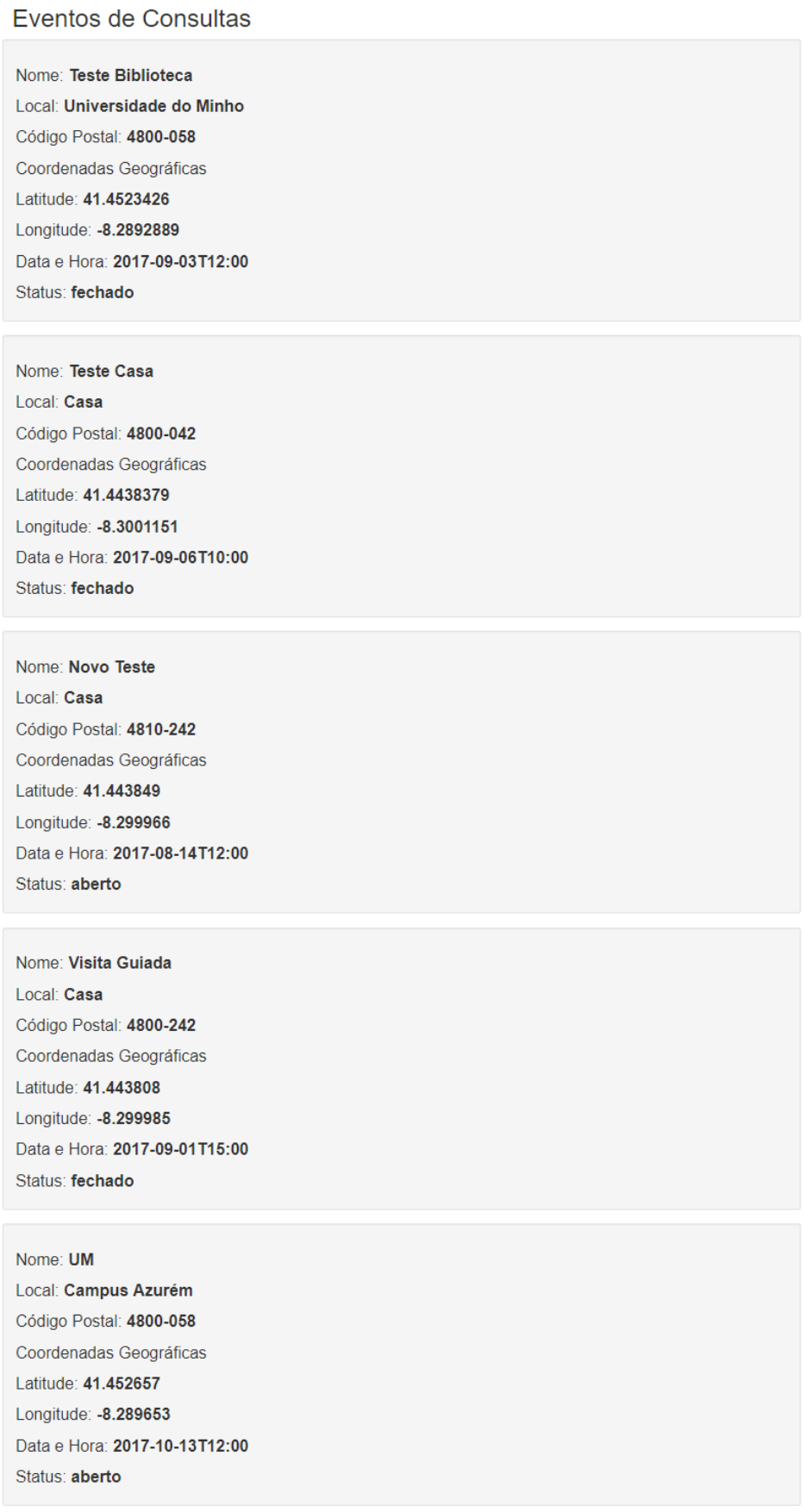


Figura B 7 - Interface de visualização dos eventos

Anexo C – Publicações

C.1 – Benefits of Bring Your Own Device in Healthcare

Autores: Ailton Moreira, Filipe Portela, Manuel Filipe Santos

Livro: *Next-Generation Mobile and Pervasive Healthcare Solutions. pp. 32-45. ISBN: 1522528512. IGI-Global. (2017).*

***Abstract*:** *Bring Your Own Device (BYOD) has become a very popular topic in information technology because this approach allows the employees to bring their personal devices into the organisation and they want to use them to access the organisation information. This trend has some benefits. Both for the organisation and for employees. This paper aims to identify those benefits as well the advantages and disadvantages of BYOD usage in the organisation. Also, it is present a SWOT analysis of BYOD usage. It is introduced an approach to BYOD in healthcare also. Utilising personal devices at work is beneficial to organisational employees because they are in some way satisfied, and they have more freedom and choice to use their devices. This liberty and choice can easily lead the employees to be more productive and flexible. The organisation who embraces BYOD policies noticed that their employees are happier, more productive, and more collaborative.*

***Keywords*:** *Bring Your Own Device (BYOD), BYOD Benefits, BYOD SWOT Analysis, Mobile Devices in Healthcare*

Relação com a dissertação: A escrita deste capitulo do livro permitiu aprofundar ainda mais os conhecimentos do BYOD em geral, bem como os seus benefícios, vantagens e desvantagens e os principais obstáculos que existem a quando da sua implementação. Na área da saúde, foi possível aprofundar também o conhecimento sobre a sua prática e os benéficos que a mesma tem. Ainda foi realizado uma análise SWOT do BYOD na área da saúde.

Estado: Publicado.

C.2 – IoT: A Case Study of BYOD in Healthcare

Autores: Ailton Moreira, Filipe Portela, Manuel Filipe Santos

Revista: *Special Issue on "Challenges and Applications of Internet of Things, Bid Data and Cloud"*

***Abstract*:** *Currently, we live in a world where mobile devices are increasingly common in our day-to-day activities. More employees are using their own mobile devices in the performance of their professional activities. This practice has been shown to be beneficial to the organisation as well as to the employee. But, at the same time this practice raises some questions about data security, how processes are executed, governance, and sometimes even the culture of the organisation itself. This article presents the analysis of the results of implementation and use of mobile devices (BYOD) in healthcare. To start the challenges and obstacles are analyzed that exist in the implementation of BYOD in healthcare. Then, follows the analysis of the results with the development of the proposed solution to the Centro Hospitalar do Porto. This analysis aims to present the benefits of using BYOD in healthcare and the obstacles that must be overcome when employees begin to use mobile devices in the development of their professional activities. According to the results obtained, globally the application was well accepted by the users since it has functionalities with the capacity to fill the various problems felt by the users. The same was rated by users as a user-friendly application. Also, it is suggested some recommendations that healthcare organizations should follow when implementing a BYOD strategy.*

***Keywords*:** *BYOD*, BYOD in *Healthcare*, *Healthcare*, *Mobile Device in Healthcare*

Relação com a dissertação: A escrita deste artigo permitiu a partilha dos resultados obtidos com mais comunidades científicas. Ainda este artigo apresenta o impacto que o BYOD pode ter na área da saúde (*healthcare*) tanto do ponto de vista dos utentes como também para os profissionais de saúde. Os resultados obtidos com a utilização do BYOD no *healthcare* são muito satisfatórios.

Estado: Em submissão.