Mestrado em Engenharia Informática
Estágio
Relatório Final

Módulo de agendamento para a aplicação MedicineOne iPhone

Pedro de Mesquita Faustino
pfaustino@student.dei.uc.pt

Orientadores:
Prof. Doutor Jorge Henriques (DEI)
Mestre Edgar Lopes (MedicineOne)

Data: 1 de Julho de 2014

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Agradecimentos

À minha família, em especial aos meus pais e ao João Ruivo por me terem proporcionado a oportunidade de apostar na minha formação académica e por todo o apoio que me deram ao longo do seu percurso.

Ao Edgar Lopes, meu orientador na *MedicineOne*, e ao Bruno Coelho por toda a ajuda e suporte incansável que me deram ao longo do estágio.

Ao meu orientado do DEI, o Professor Doutor Jorge Henriques, que se mostrou sempre disponível para me orientar no decorrer do estágio.

À *MedicineOne* e a toda a sua equipa por me terem integrado plenamente na empresa e por todo o apoio prestado sempre que foi necessário.

Resumo

A chegada dos *Smartphones* veio revolucionar o conceito associado aos telemóveis, pois estes deixaram de ser meros aparelhos de comunicação, passando a oferecer um maior número de funcionalidades. Tal fato deu origem a um novo mercado para as *software houses*, que se dedicaram a desenvolver aplicações móveis para as mais variadas tarefas.

A *MedicineOne* é uma das empresas que atua neste mercado e que tem vindo a estender as funcionalidades das suas aplicações *desktop*, direcionadas para a gestão clínica, para as plataformas *iPhone* e *iPad*. A aplicação móvel "*MedicineOne*" permite que os profissionais de saúde sejam auxiliados na sua atividade, no que respeita ao acompanhamento dos utentes e à prescrição de medicação. No entanto, a *MedicineOne* deseja, além disso, estender a sua oferta e incorporar nesta aplicação um módulo de agendamento que possibilite ao profissional de saúde gerir a sua agenda profissional.

O objetivo deste estágio concretizou-se, então, no desenvolvimento do módulo de agendamento, o qual será, posteriormente, integrado na aplicação referida e cuja finalidade é a de permitir que o profissional de saúde navegue na sua agenda pelos formatos ano, mês e dia, possibilitando a consulta das marcações e a gestão das mesmas, através da sua desmarcação, transferência ou, até mesmo, da marcação de novas consultas. Para além disso, o módulo deverá, ainda, comunicar com os devidos serviços, a fim de trocar os dados a manipular, dados esses que respeitam a informação clínica, que apresenta necessidades de segurança, pelo que os objetivos deste estágio incluíam, também, a implementação de serviços e de mecanismos de segurança.

Palavras-chave

"agenda", "consultas", "iOS", "MedicineOne", "segurança", "confidencialidade", "integridade", "serviços", "profissional de saúde"

Índice

Lista de figuras

Lista de tabelas

Capítulo 1
Introdução

A evolução constante na área das tecnologias que se tem vindo a verificar ao longo dos anos conduziu à alteração dos modelos de negócio e dos processos de trabalho de diversos sectores. A área da saúde foi um dos sectores onde se verificaram bastantes modificações, causadas pelos avanços tecnológicos, sendo atualmente imprescindível, em certos ramos da saúde, o uso da tecnologia. Um estudo realizado pela *Economist Intelligence Unit* em 2013 refere que 92,5% dos profissionais da área da saúde se tornaram mais dependentes da tecnologia nos últimos três anos [1].

Uma das empresas que esteve atenta a esta tendência e que se especializou neste nicho de mercado foi a *MedicineOne*, onde decorreu o estágio curricular inserido no âmbito da disciplina de Estágio do Mestrado em Engenharia Informática, ministrado na Universidade de Coimbra, sob a orientação do Professor Doutor Jorge Henriques e do Mestre Edgar Lopes.

O presente capítulo apresenta o enquadramento do estágio, a definição dos seus objetivos e a estrutura do documento.

1.1. Enquadramento do estágio

A *MedicineOne* é uma empresa sediada em Coimbra, no Instituto Pedro Nunes, que se dedica, exclusivamente, ao desenvolvimento de *software* na área da saúde. Iniciada em 1989, altura em que a introdução de novas tecnologias no mercado da saúde em Portugal apresentava bastantes obstáculos, a *MedicineOne* conseguiu identificar, com sucesso, as necessidades que foram surgindo ao longo do tempo junto das comunidades de profissionais de saúde, levando à constituição de uma carteira de clientes que conta com nomes como a EDP - Sãvida - Medicina Apoiada S.A., TAP/UCS, Galp Energia, *Idealmed* SGPS, Hospitais Universidade de Coimbra, Serviço Nacional de Saúde, entre outos [2].

O sucesso da *MedicineOne* advém da sua constante preocupação em melhorar a sua eficiência e a qualidade dos seus produtos. Ao longo do tempo, foram estabelecidas parcerias estratégicas com as empresas CMP Medica Portugal, Oni *Communications*, IMS *Health* e *Microsoft Corporation* [2], possibilitando a troca constante de conhecimento, e foi alcançada a certificação da norma NP 4457 [3]. Mais recentemente, deu-se início à expansão para o mercado internacional, com a aposta no mercado brasileiro, onde existe uma enorme comunidade de profissionais de saúde.

Uma das aplicações presentes no Portefólio da *MedicineOne* respeita à ferramenta "*MedicineOne*", que está disponível nas plataformas *iPhone*, *iPad*, *Mac* e *Windows*. Esta é uma das aplicações portuguesas direcionadas para a área da saúde que mais êxito tem tido junto da comunidade portuguesa de profissionais deste sector. Com a crescente utilização de dispositivos móveis por parte desta comunidade, é importante destacar a versão para as plataformas móveis que fornece a possibilidade de estes acederem, a qualquer momento, ao seguinte conjunto de funcionalidades:

- Registar problemas de saúde diagnosticados ao utente;
- Registar alergias diagnosticadas ao utente;

- Prescrever, eletronicamente, medicamentos aos utentes;
- Consultar o histórico de medicamentos prescritos aos utentes;
- Alertar possíveis incompatibilidades entre medicamentes, ou conflitos com as alergias dos utentes;

Esta aplicação permite, então, que os profissionais de saúde tenham, a qualquer momento, informação centralizada sobre os problemas de saúde e alergias diagnosticadas aos seus utentes, melhorando, assim, o acompanhamento dos mesmos. Outra funcionalidade bastante importante, devido à enorme quantidade de medicamentos existentes no mercado farmacêutico, diz respeito à emissão de alertas aquando da prescrição de medicamentos, ou seja, caso exista algum conflito entre dois medicamentos que possa vir a prejudicar o utente, ou caso existam incompatibilidades entre um medicamento e as alergias diagnosticadas ao mesmo, o profissional de saúde é alertado para que possa tomar as devidas precauções [4].

Este conjunto de funcionalidades tem vindo a contribuir para o sucesso da aplicação da "*MedicineOne*", que conta já com 11.400 profissionais de saúde registados e que processa 60% das prescrições eletrónicas efetuadas na clínica privada em Portugal. A introdução desta aplicação no dia-a-dia da comunidade aderente conduziu a que 23% da população portuguesa já tenha aí o seu processo clínico eletrónico registado [4].

Sempre focada na inovação e na satisfação dos seus clientes, a *MedicineOne* identificou a necessidade de suprir o problema associado a imprevistos que podem levar ao cancelamento urgente de consultas. Quando ocorre um imprevisto deste tipo por parte do profissional de saúde, muitas das vezes o utente apenas é notificado após já se ter deslocado ao serviço de saúde. Por outro lado, se o utente não comparecer numa consulta, é frequente que esse bloco de tempo não seja aproveitado para prosseguir com as restantes marcações. Estes problemas afetam o funcionamento dos serviços de saúde na perspetiva de maximizar o tempo útil dos profissionais, causando extensas filas de espera e descurando a comunicação entre os utentes e estes. Para dar resposta a estas adversidades a *MedicineOne* propôs-se a desenvolver um módulo de agendamento para as versões móveis da "*MedicineOne*", que possibilita ao profissional de saúde uma melhor organização das suas consultas, e deu início ao desenvolvimento de uma aplicação para os utentes, o que permitirá posteriormente a comunicação entre ambas aplicações.

O âmbito deste estágio tem como um dos objetivos o desenvolvimento do módulo de agendamento, especificamente para a plataforma *iPhone*. Esta especificidade está associada ao facto de o *iPhone* apresentar um ecrã de reduzidas dimensões, o que representa um desafio, pois os componentes a desenvolver devem garantir uma navegação fluida e intuitiva.

De um modo geral, este módulo deve acrescentar as seguintes funcionalidades à aplicação móvel já existente:

- Navegar na agenda pelos formatos ano, mês e dia;
- Consultar marcações e as suas propriedades;
- Gerir estado das marcações;
- Marcar e desmarcar consultas;
- Transferir consultas;
- Pesquisar marcações;

1.2. Objetivos

Tal como foi referido anteriormente, o âmbito do estágio assentou no desenvolvimento de um módulo de agendamento que permita ao profissional de saúde gerir, diretamente, a partir da aplicação móvel "*MedicineOne*", a sua agenda de marcações de consultas. Para além de possibilitar a navegação sobre os diferentes tipos de visualização (anual, mensal e diária) e a gestão das consultas (marcação, pesquisa, consulta das propriedades, etc.), o módulo deve estabelecer uma comunicação com os servidores da *MedicineOne* que, por sua vez, irão disponibilizar os serviços necessários para trocar os dados a manipular. Uma vez que estes dados dizem respeito a informação clínica, que é de natureza sensível e que se encontra preservada por leis de proteção de informação, é necessário garantir a segurança dessa informação trocada durante essa comunicação.

Posto isto, os objetivos do estágio convergem nos seguintes pontos:

- Desenvolvimento de um módulo que:
    - Permita a navegação pelas vistas referentes ao ano, mês e dia;
    - Permita a consulta de marcações e das suas propriedades, a gestão do estado das consultas, a marcação, desmarcação, transferência e pesquisa de consultas;
    - Comunique com os servidores da *MedicineOne*;
    - Seja integrado com a aplicação *iPhone* da *MedicineOne*;
- Desenvolvimento dos serviços que permitam a comunicação de dados com o módulo;
- Implementação de mecanismos de segurança que garantam a proteção dos dados trocados entre os serviços e o módulo;

1.3. Estrutura do documento

No primeiro capítulo (Introdução) apresenta-se o enquadramento do estágio e definem-se os objetivos do mesmo.

O segundo capítulo é dedicado à análise de plataformas e tecnologias. Aqui apresenta-se uma análise a componentes de navegação que possam ser reutilizados, à linguagem de desenvolvimento do módulo e à plataforma adotada para desenvolver serviços, e a mecanismos de segurança que assegurem a confidencialidade e integridade das mensagens, e uma breve conclusão sobre as decisões tomadas.

O terceiro capítulo é dedicado ao planeamento e metodologia. É apresentado o planeamento traçado para o estágio, bem como os consequentes desvios a este, e a metodologia de desenvolvimento adotada.

O quarto capítulo é dedicado ao desenho da solução. É apresentada a análise dos requisitos funcionais e não funcionais, e arquitetura envolvente ao módulo.

O quinto capítulo é dedicado ao trabalho desenvolvido ao longo do estágio. Aqui apresentam-se as diversas decisões tomadas e os detalhes de implementação no que diz respeito aos objetivos definidos.

Finalmente, no sexto capítulo, retiram-se conclusões do trabalho desenvolvido e relata-se, sinteticamente, o trabalho futuro agendado.

Capítulo 2
Análise de plataformas e tecnologias

Uma das fases mais importantes na etapa inicial deste estágio refere-se à análise elaborada sobre componentes de navegação já existentes, que pudessem vir a ser reutilizados e adaptados ao módulo, ao estudo da linguagem de programação a utilizar para o desenvolvimento de aplicações para *iPhone* e da plataforma adotada para o desenvolvimento de serviços, e à análise de mecanismos de segurança que assegurem a proteção dos dados trocados.

Embora a maioria das funcionalidades do módulo a desenvolver sejam bastante específicas para o âmbito da aplicação em que este vai ser integrado, identificou-se a oportunidade de reutilizar e adaptar componentes desenvolvidos para *iOS*, que permitissem navegar entre diferentes vistas (anual, semanal e diária), à semelhança de um calendário, uma vez que esta funcionalidade é mais genérica que as restantes.

Devido à impossibilidade de integrar os componentes existentes na agenda nativa do *iPhone*, procedeu-se à análise de componentes *open-source* [5] com o mesmo tipo de finalidade, as quais serão apresentadas posteriormente.

Sendo que um dos critérios, a ter em ponderação, se refere à sua *interface*, dado que, com o lançamento da nova versão do Sistema Operativo *iOS*, no dia 18 de Setembro [6], o *iOS* 7, que até à data de 12 de Janeiro de 2014 se encontrava presente em 79% dos dispositivos móveis da *Apple* [7], definiu-se, então, como requisito, que a *interface* deste componente seja o mais semelhante possível à utilizada na aplicação nativa, a qual acompanha esta nova versão, a fim de permitir uma interação mais intuitiva e familiar para o utilizador.

A linguagem de programação recomendada pela *Apple*, como *standard*, para o desenvolvimento de aplicações para os seus Sistemas Operativos é *Objective-C* [8]. Tendo em conta esta recomendação e advindo o facto da aplicação "*MedicineOne*" ser baseada nesta mesma linguagem, o desenvolvimento do módulo de agendamento para *iPhone* seguirá o mesmo padrão, cumprindo, assim, as recomendações da *Apple* e simplificando a futura integração com a aplicação. Uma vez que o estagiário nunca tinha abordado esta linguagem de programação, a equipa de desenvolvimento da *MedicineOne* proporcionou-lhe uma fase de formação que lhe permitiu a aquisição dos principais conceitos que a englobam, e que serão apresentados posteriormente.

Para que o módulo de agendamento forneça as funcionalidades propostas revela-se necessário que este interaja com serviços existentes nos servidores da *MedicineOne*, permitindo, assim, que os dados a manipular sejam trocados e que as regras de negócio aplicadas pela *MedicineOne* sejam respeitadas. Embora o desenvolvimento destes serviços estejam sob a responsabilidade do estagiário, a *Framework* e a linguagem de programação a adotar devem seguir os padrões dos restantes serviços da *MedicineOne*, que assentam na *Framework Windows Communication Foundation* (WCF), que também será apresentada posteriormente, e que se encontram desenvolvidos na linguagem C#.

Atualmente, a *MedicineOne* assegura a proteção dos dados trocados durante a comunicação das suas aplicações com os seus serviços através da utilização do protocolo criptográfico *Secure Socket Layer* (SSL) sobre Hypertext Transfer Protocol (HTTP), denominado de *HyperText Transfer Protocol Secure* (HTTPS), que permite a criação de um canal de comunicação encriptado [9]. Contudo, o funcionamento do protocolo SSL apresenta algumas vulnerabilidades, que serão discutidas, pelo que a *MedicineOne* optou por estender as medidas de segurança a adotar, garantindo, assim, a confidencialidade e integridade dos dados trocados de forma independente, face aos mecanismos oferecidos por este. Posto isto, é da responsabilidade do

estagiário conduzir uma análise aos diversos mecanismos de segurança existentes que garantam a confidencialidade e integridade dos dados trocados entre o módulo de agendamento e os serviços, e apresentar uma solução que vá ao encontro dos requisitos citados.

2.1. Componentes de navegação na agenda

Tal como foi referido anteriormente, foi elaborada uma análise a componentes *open-source* que pudessem ser reutilizados e adaptados ao módulo de modo a simplificar a implementação da funcionalidade de navegação na agenda. Dois dos critérios a ter em conta durante esta análise é o suporte à navegação pelos formatos ano, mês e dia, e a semelhança da *interface* com a adotada na agenda nativa do *iOS* 7.

Posto isto, as seguintes subsecções apresentam a análise da agenda nativa, onde se realçam os aspetos de *interface* a ter em conta, e dos componentes *open-source*. Esta análise conduz a uma síntese acerca das decisões tomadas sobre o desenvolvimento desta funcionalidade.

2.1.1 Agenda nativa

O lançamento da nova versão do Sistema Operativo do *iPhone* fez-se acompanhar da reestruturação de várias aplicações nativas existentes nas versões anteriores. Uma das que sofreu grandes alterações diz respeito à agenda nativa, que a seguinte figura pretende ilustrar:

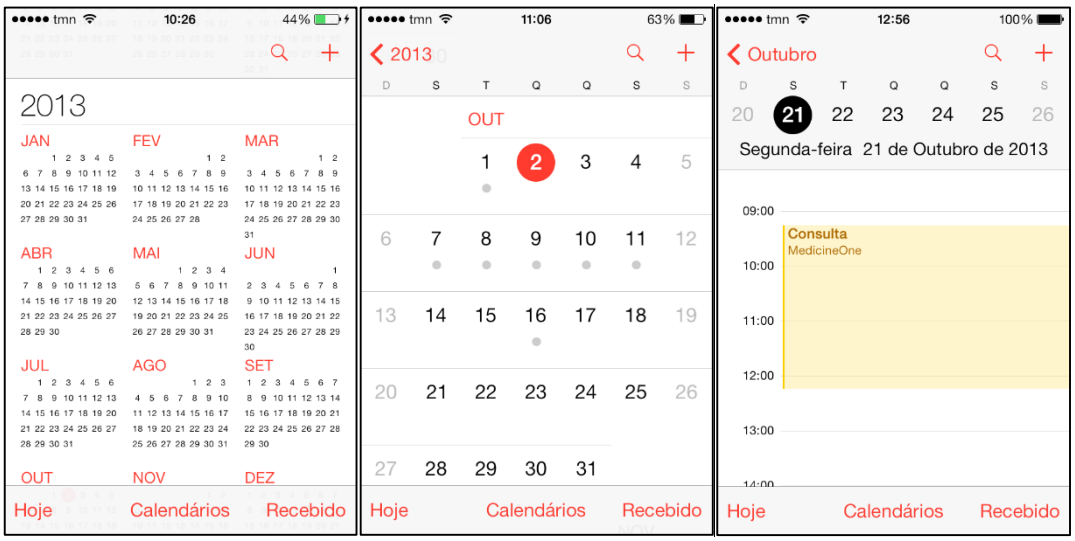

Figura 1 - *Interface* da agenda nativa

Ao analisar a Figura 1, podemos identificar os três formatos de navegação definidos como requisito para o módulo a desenvolver: vista anual (imagem à esquerda), mensal (imagem ao centro) e diária (imagem à direita).

Através da navegação na vista anual, o utilizador pode deslocar-se com rapidez entre datas referentes a diferentes anos, ao deslizar o seu dedo na vertical sobre o ecrã. Ao selecionar um mês de um determinado ano, o utilizador transita para a vista mensal da data correspondente, tal como se pode verificar, na Figura 1, em que este acede à vista referente ao mês de Outubro de 2013.

A navegação pelo formato mensal permite que o utilizador tenha uma visão geral das tarefas agendadas para os dias de um determinado mês, que podem ser identificadas através da

marcação com um círculo cinzento. À semelhança da vista anual, o utilizador pode navegar com rapidez entre diferentes meses e pode selecionar um determinado dia, ocorrendo, posteriormente, uma transição para a vista diária.

É importante referir que, em ambas as vistas, anual e mensal, o deslize vertical, que permite navegar por diferentes datas, ocorre de forma contínua, o que possibilita uma deslocação mais rápida sobre diferentes datas ao invés da rapidez oferecida por um deslize sequencial, que obriga o utilizador a transitar por todas as datas até chegar à desejada.

O formato de navegação diária encontra-se dividido em duas partes: na parte superior existe uma barra de navegação semanal e na parte inferior é apresentada a agenda diária. De modo a evitar o recurso à vista mensal, cada vez que o utilizador deseja aceder a uma data próxima à escolhida, este pode recorrer à barra de navegação semanal para transitar rapidamente entre datas de uma mesma semana, através do deslize horizontal do dedo sobre a agenda diária ou da seleção de uma das datas presentes na barra, ou entre diferentes semanas, através do deslize horizontal do dedo sobre a barra. Na parte inferior, isto é, na agenda diária, o utilizador pode analisar, em detalhe, e interagir com as tarefas agendadas para uma determinada data.

É importante referir que existem vários detalhes nesta aplicação, por vezes impercetíveis, que devem ser tomados em consideração para o desenvolvimento da funcionalidade de navegação no módulo a fim de tornar a sua utilização mais familiar para o utilizador, como por exemplo a utilização de diversas animações, os diferentes modos de interação com a aplicação, a presença do botão "Hoje" na parte inferior do ecrã, que permite a transição imediata para a data atual, o destaque dado à data atual nas diferentes vistas, entre outros.

Desta análise reforça-se a premissa de que as posteriores aplicações a analisar devem suportar a navegação pelos formatos anual, mensal e diária, sendo que esta ainda deve conter uma navegação semanal, e apresentar uma *interface* o mais semelhante possível à apresentada na Figura 1, embora este requisito não seja tão prioritário uma vez que a sua adaptação apresenta menos complexidade que o desenvolvimento dos diversos formatos de navegação.

2.1.2 TapkuLibrary

O *TapkuLibrary* é uma biblioteca *iOS* desenvolvida sobre *Cocoa Touch* e *UIKit*, que serão abordados mais à frente, cujo objetivo é fornecer componentes que podem ser integrados em aplicações [10].

Um dos componentes presentes nesta biblioteca está associado à funcionalidade de navegação desejada, tal como se pode verificar na seguinte figura:

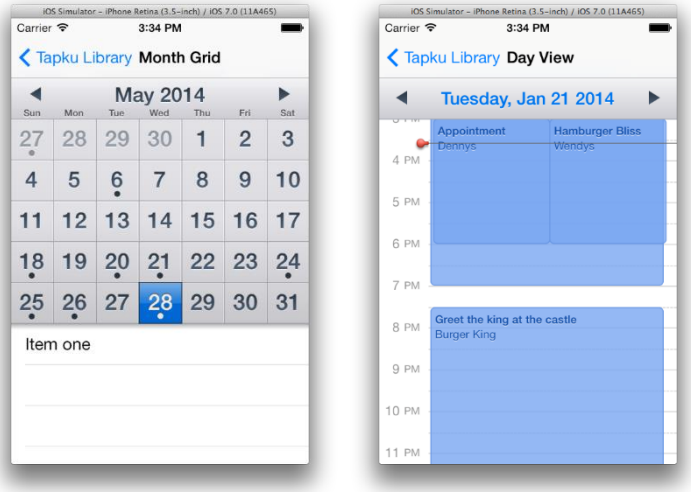

Figura 2 - *Tapku Library interface*

Analisando a Figura 2, é possível identificar que existem apenas dois formatos de navegação: mensal e diária. Na vista mensal o utilizador obtém uma visão geral das tarefas agendadas para o mês em questão e pode ainda averiguar tarefas marcadas para determinados dias, como é o caso do dia 28 de Maio de 2014 (imagem à esquerda), em que está agendado o "*Item one*". No que diz respeito ao modo de navegação, para além de não ser suportada a interação através do deslize do dedo, este requer que o utilizador transite, sequencialmente, por todos os meses, até chegar ao desejado, não sendo tão rápido e flexível como a solução que se procura.

Na vista diária o utilizador pode analisar em detalhe as diversas tarefas agendadas para um determinado dia, tal como desejado. Embora seja suportada a transição entre datas adjacentes, não é possível navegar, rapidamente, entre diferentes semanas, o que obriga o utilizador a transitar pelos vários dias, até atingir a semana desejada, ou a recorrer à vista mensal.

Embora ambas as vistas correspondam, em certa parte, aos requisitos da funcionalidade a implementar, para além de estas não se encontrarem interligadas, isto é, não existir uma transição da vista mensal para a diária, não há suporte para o formato anual, o que implica que, caso o utilizador deseje navegar para outro ano, este tenha de avançar por todos os meses de um ano até alcançar o desejado. Outro aspeto indesejável diz respeito à visualização das tarefas agendadas numa determinada data da vista mensal, pois um dos objetivos de suportar três formatos de navegação é manter a informação organizada e separada, pelo que este nível de detalhe deve ser mantido apenas na vista diária.

2.1.3 MBCalendarKit

O *MBCalendarKit* é um calendário desenvolvido sobre *UIKit* [11], cuja *interface* é a apresentada na Figura 3:

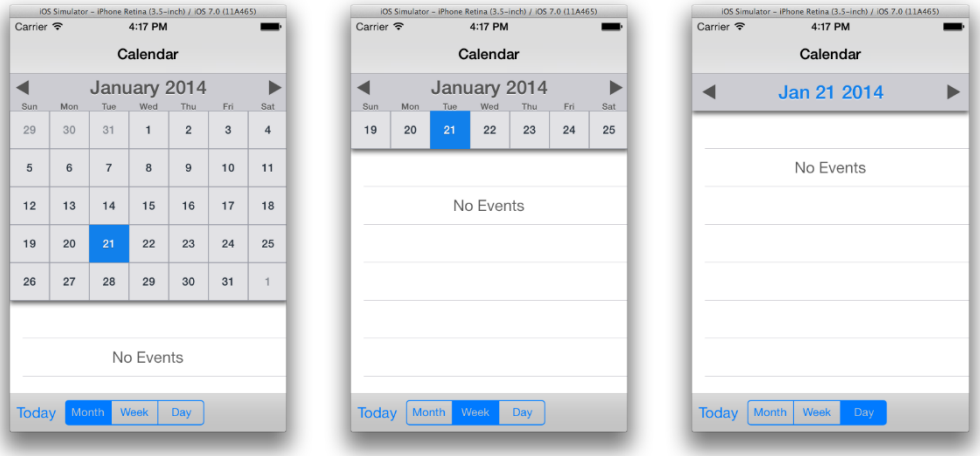

Figura 3 - *MBCalendarKit interface*

À semelhança do componente analisado anteriormente, este também apresenta apenas as vistas mensal (imagem à esquerda) e diária, com a diferença de incorporar dois tipos de vista diária: uma que contempla a navegação semanal (imagem ao centro), tal como desejado, e outra que apenas permite navegar entre dias adjacentes (imagem à direita).

Analisando a vista mensal, a primeira crítica a apontar é a impossibilidade de recorrer à navegação através do deslize do dedo sobre o ecrã, obrigando o utilizador a transitar ao longo dos meses de modo sequencial. Surge, novamente, nesta vista, uma área reservada para a exposição das tarefas agendadas numa determinada data que, tal como já tinha sido referido anteriormente, deve apenas ser apresentado na vista diária. Na impossibilidade de adicionar eventos, uma vez que este componente não contempla essa funcionalidade, é impossível averiguar se os dias com tarefas agendadas apresentam algum tipo de distinção relativamente aos restantes.

Embora a vista diária (imagem ao centro) não possibilite o deslize do dedo para transitar entre diferentes semanas, o modo de navegação aplicado possibilita este tipo de transição. Uma das críticas a apontar nesta vista é a inexistência de uma linha temporal que acompanhe a agenda diária, o que impossibilita ao utilizador saber a que horas tem agendadas as suas tarefas.

Um dos pontos positivos deste componente diz respeito à existência do botão "*Today*", na zona inferior do ecrã, que possibilita ao utilizador deslocar-se, rapidamente, para a data atual. Contudo, a existência dos botões "*Month*", "*Week*" e "*Day*" revela um problema associado à transição entre as diferentes vistas, pois esta é despoletada pela ação destes botões em vez de ser resultado da seleção de uma data.

2.1.4 ABCalendarPicker

ABCalendarPicker é um componente *User Interface* (UI) totalmente configurável baseado em calendários e suporta diferentes tipos de vistas que vêm acompanhadas de várias animações [12]. A seguinte figura pretende ilustrar os vários *layouts* que esta oferece:

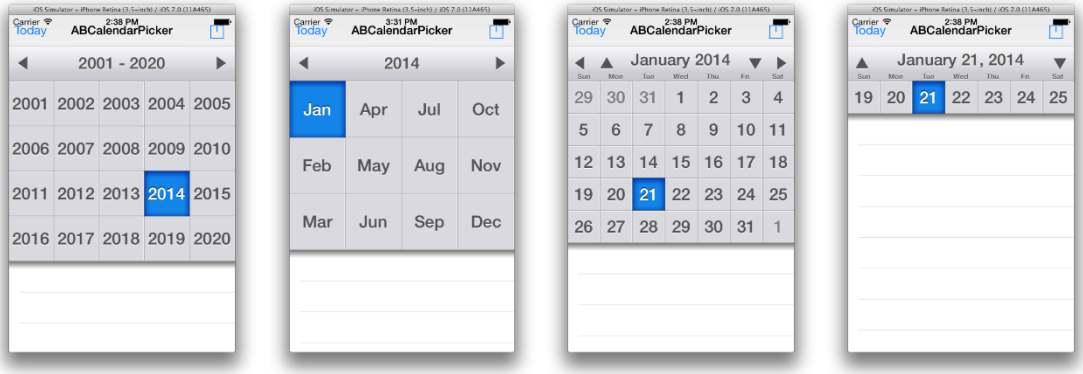

Figura 4 - *ABCalendarPicker interface*

Tal como se pode verificar na Figura 4, o *ABCalendarPicker* apresenta um leque mais variado de formatos de navegação que os componentes previamente analisados. A vista anual (primeira imagem) é suportada e permite, à semelhança da agenda nativa, um acesso rápido a diferentes anos. Contudo, a grande desvantagem assenta no facto de haver uma separação entre o ano e os meses correspondentes, pois tal como se pode verificar nas duas primeiras imagens, existe uma vista para selecionar o ano e outra para selecionar o mês correspondente.

No que diz respeito à vista mensal (terceira imagem), a navegação entre os diferentes meses ocorre de forma sequencial, isto é, o utilizador precisa de percorrer todos os meses até chegar ao desejado. Por outro lado, é oferecido um modo de navegação inexistente na aplicação nativa que consiste em permitir que o utilizador avance ou recue um ano em relação a um dado mês. Uma vez que este componente também não possui a funcionalidade de adicionar eventos, não é possível analisar se o utilizador tem acesso a uma visão geral acerca das tarefas agendadas para um determinado mês.

O formato de navegação diário (quarta imagem) apresenta uma *interface* bastante semelhante à agenda nativa, encontrando-se também dividido em duas partes. Embora não seja possível confirmar se os eventos são apresentados na parte inferior referente à agenda diária, pelas razões descritas anteriormente, a barra de navegação semanal apresenta o comportamento desejado, ainda que o modo de interação seja ligeiramente diferente.

À semelhança dos componentes apresentados nas subsecções anteriores, este também induz no erro de incorporar em distintas vistas uma área onde as tarefas agendadas poderão ser consultadas. As restantes críticas assentam na utilização de um modo de navegação sequencial, ao invés de contínuo, na separação dos meses de um determinado ano na vista anual e na impossibilidade do utilizador poder consultar as horas em que as tarefas estão agendadas na vista diária. Por outro lado, incorpora todas as vistas necessárias e possui alguns detalhes, como o uso de animações e a presença do botão "*Today*" (zona superior esquerda do ecrã).

2.1.5 Síntese

De acordo com o critério referente aos formatos de navegação a serem suportados pelo módulo a desenvolver, que dizem respeito ao anual, mensal e diário, apenas o componente *ABCalendarPicker* foi ao encontro dos requisitos impostos, dado que os restantes só apresentam o formato mensal e diário.

Comparando as *interfaces* dos diferentes componentes, é possível verificar algumas semelhanças. Contudo, quando estas são comparadas à *interface* da aplicação nativa, denota-se uma grande disparidade. A razão para esta divergência está associada ao facto do lançamento do *iOS* 7 ser relativamente recente, pelo que na altura em que foi efetuada a análise dos

componentes *open-source* não foi possível encontrar nenhum adaptado à nova versão do Sistema Operativo da *Apple*, pois estes eram, maioritariamente, baseados na interface da agenda nativa da versão do *iOS* 6.

Tendo em conta o facto da *interface* aplicada aos diversos componentes ser baseada na versão da agenda nativa para *iOS* 6, o que esteve na origem das críticas apontadas durante a sua análise, o estagiário propôs que o desenvolvimento da funcionalidade de navegação na agenda fosse feito de raiz. Esta proposta foi fundamentada pela existência de diferenças significativas entre as duas versões da agenda nativa, pela possibilidade de ser dada uma maior atenção aos detalhes que acompanham a nova versão e pela oferta de uma maior flexibilidade na condução do seu desenvolvimento. Com base nos argumentos apresentados, a equipa da *MedicineOne* aceitou a proposta.

2.2. Objective-C

Objective-C é a linguagem de programação definida como padrão pela *Apple* para o desenvolvimento de aplicações para os seus Sistemas Operativos, o *iOS* e o *Mac OSX*, e teve a sua origem na empresa *NeXT Software*, que foi fundada por um dos visionários da *Apple*, Steve Jobs. Embora seja baseada na linguagem C, esta oferece também recursos orientados a objetos [13]. O ambiente de desenvolvimento integrado fornecido pela *Apple* para a criação de aplicações para *Mac*, *iPhone* e *iPad* é o *Xcode*, que inclui uma ferramenta de análise de performance, um simulador *iOS* e vem acompanhado dos mais recentes *Software Development Kits* (SDK) para *iOS* e *OS X* [14]. Posto isto, a escolha da ferramenta de desenvolvimento assenta sobre o *Xcode*.

As seguintes subsecções apresentam os principais conceitos adquiridos ao longo do estudo da linguagem, que dizem respeito à *Framework Cocoa Touch* e a algumas considerações a ter em conta a nível de performance e de serviços de acesso à rede.

2.2.1 Cocoa Touch Framework

Cocoa Touch é a principal *Framework* utilizada no desenvolvimento de aplicações para *iOS*. Para além de estender *Frameworks* existentes em *MAC* e introduzir *Frameworks* exclusivas para *iOS*, esta também disponibiliza *Application Programming Interfaces* (API) que permitem um acesso direto ao sistema, como, por exemplo, ao *Hardware Global Positioning System* (GPS), para obter a localização do dispositivo, e tecnologias como *multitasking*, *gesture recognizers*, *storyboards*, entre outras [15] [16]. Algumas das *Frameworks* fornecidas pelo *Cocoa Touch* que merecem destaque são as seguintes:

- *Core Data*;
- *Core Animation*;
- *UIKit*.

Core Data é uma *Framework* de modelagem de dados, direcionada para aplicações orientadas a objetos, uma vez que promove a persistência dos objetos ao longo do seu ciclo de vida. Esta permite a criação do modelo de dados da aplicação e fornece a infraestrutura necessária para armazenar, aceder e partilhar os dados [17].

A *Framework Core Animation* fornece os meios necessários para criar animações que melhorem a experiência de interação do utilizador através da manipulação dos diferentes elementos visuais presentes na aplicação, como por exemplo aplicar uma rotação [18]. Para efeitos de animação existe ainda outra *Framework*, o *Core Graphics*, que é fornecida numa camada inferior

ao *Cocoa Touch*, uma vez que arquitetura do *iOS* é composta por diferentes camadas e o *Cocoa Touch* corresponde à de mais alto nível [19]. Esta *Framework* permite uma manipulação de menor nível fornecendo a API de acesso ao *Quartz 2D*, que é um mecanismo de desenho bidimensional que permite por exemplo a transformação das coordenadas de espaço de um elemento visual [20] [21].

O *UIKit* fornece a infraestrutura necessária para desenvolver a *interface* gráfica das aplicações [22]. A criação da *interface* tem como base as *Views*, que corresponde a uma área retangular no ecrã que gere os eventos de toque e o desenho dos elementos nessa mesma área. Estas podem variar entre simples botões, até elementos de maior complexidade, como *table views* ou *collection views* [23] [24].

2.2.2 Performance

Tendo em conta o ambiente operacional das aplicações *iOS*, torna-se necessário conduzir o seu desenvolvimento de acordo com as restrições apresentadas, de modo a que a performance da aplicação não seja afetada por estas limitações. Para ajudar os programadores a melhorarem aspetos de performance das suas aplicações, a *Apple* disponibilizou um guia onde são apresentados alguns aspetos a ter em consideração [25]:

- Usar a memória eficientemente;
- Reduzir o consumo de energia;
- Otimizar o código;

Aplicações que requeiram grandes quantidades de memória estão condenadas a apresentar uma degradação de performance, pois, de acordo com o modelo de memória virtual do *iOS*, cada aplicação está limitada ao uso da quantidade de memória disponível num determinado momento. Quando ocorrem problemas de memória, isto é, quando a memória disponível atinge valores muito baixos, o *iOS* envia uma notificação para as aplicações, pelo que se torna aconselhável que estas implementem mecanismos que permitam a libertação de memória. De modo a manter os consumos de memória reduzidos é aconselhável eliminar o seu vazamento, que pode ser detetado com recurso a uma ferramenta que será apresentada noutro ponto, deve ser feito uso da *Framework Core Data* para armazenar largas quantidades de dados, uma vez que esta oferece um modo eficiente de gerir grandes conjuntos de dados, sem que estes se encontrem todos em memória ao mesmo tempo, os recursos devem ser carregados à medida que forem sendo necessários, ao invés de serem todos carregados de uma vez, entre outros [25].

Devido às limitações que a bateria dos dispositivos móveis apresentam, é bastante importante reduzir os consumos de energia causados pela aplicação. Uma vez que o sistema de gestão da energia do *iOS* permite a suspensão de *Hardware* que não esteja a ser utilizado, é importante que sejam feitas algumas otimizações de modo a que a aplicação tire vantagens desta funcionalidade para reduzir os consumos. Um dos aspetos que o guia da *Apple* aborda é o uso das classes *NSRunLoop* e *NSTimer* para o agendamento da execução de operações, uma vez que estas permitem que o *Central Processing Unit* (CPU) seja suspenso fora dos períodos agendados. Devido ao impacto que a transmissão de dados através da rede pode ter a nível de consumos de energia é importante que o acesso a servidores externos seja apenas efetuado quando necessário e que os dados a transmitir apenas contenham a informação necessária [25].

O ambiente de desenvolvimento integrado fornecido pela *Apple*, o *Xcode*, fornece a ferramenta *Instruments*, que permite a otimização do código através da análise da performance de uma aplicação num dispositivo real. Esta ferramenta contém vários módulos internos que

permitem recolher dados sobre diferentes aspetos do comportamento dos processos de uma aplicação, o que possibilita a posterior análise do seu desempenho e de possíveis falhas que estejam a prejudicar a sua performance [25] [26].

2.2.3 Serviço de acesso à rede

Para implementar e suportar os serviços de acesso à rede, o *iOS* apresenta a camada *Core Services* que contém tecnologias e *Frameworks*, que permitem, entre outros, o acesso à rede, como é o caso da *CFNetwork Framework* [27]. Contudo, a equipa da *MedicineOne* recorre a uma biblioteca de terceiros quando deseja interagir com os seus servidores. A fim de utilizar o mesmo padrão de comunicação, de modo a evitar futuros problemas de integração, a biblioteca mencionada, denominada de *AFNetworking*, será adotada pelo estagiário durante o desenvolvimento da funcionalidade de comunicação entre o módulo e os servidores da *MedicineOne*.

A biblioteca *AFNetworking* foi construída sobre o *Foundation Uniform Resource Locator* (URL) *Loading System*, que é um conjunto de classes e protocolos fornecidos pela *Framework Foundation* [28] que permite aceder ao conteúdo referenciado por um URL [29], e fornece um conjunto de API que permitem estender as abstrações de rede presentes na *Framework Cocoa Touch* [30].

2.3. Windows Communication Foundation

WCF é uma *Framework* da *Microsoft* que tem como objetivo unificar as diversas tecnologias de programação distribuída fornecidas por esta, como o *Microsoft Message Queuing* (MSMQ) ou o *ASP.NET Web Services* (ASMX), e permitir o desenvolvimento de aplicações orientadas a serviços de forma agilizada. Recorrendo a esta *Framework* é possível desenvolver aplicações constituídas por diversos serviços que trocam dados com vários clientes, que, no presente cenário, correspondem às aplicações da *MedicineOne* [31] [32].

Os serviços desenvolvidos, com recurso à *Framework* WCF, encontram-se associados a um conjunto de conceitos, dos quais é importante destacar:

- *Endpoint*;
- *Address*;
- *Contract*;
- *Binding*.

A comunicação entre um cliente e um serviço WCF ocorre através dos *endpoints* expostos por este, que são constituídos por um *Address*, um *Binding* e um *Contract*. A relação destes três conceitos disponibiliza ao cliente as informações necessárias para que este possa aceder às funcionalidades oferecidas pelo serviço [33].

Um *Address* permite identificar, unicamente, um *endpoint* e fornece informações acerca do endereço onde o referido serviço se encontra, como por exemplo um *Uniform Resource Identifier* (URI), e do protocolo de transporte a utilizar para comunicar com este, sendo que o WCF fornece diversos protocolos, entre os quais HTTP e HTTPS [33].

Um *Contract* descreve as funcionalidades que um determinado *endpoint* expõe, sendo importante destacar os *Service Contracts*, que relatam as operações que os clientes podem efetuar nos serviços, e os *Data Contracts*, que definem os tipos de dados que são trocados com esse serviço [33].

Uma vez que existem diversos aspetos que caraterizam a comunicação com um serviço, como por exemplo os vários protocolos de transporte existentes, os diferentes modos de codificação da mensagem, os aspetos de segurança a ter em conta, entre outros, o WCF fornece *Bindings*, que não são mais que a combinação destes aspetos de forma a criar um padrão de comunicação, que podem ser adotados consoante o cenário em que o serviço vai atuar. Posto isto, um *binding* especifica como comunicar com um *endpoint*. [33].

É, também, importante referir que a arquitetura do WCF fornece diversos pontos de extensibilidade, isto é, permite personalizar o comportamento de um dado serviço caso as implementações fornecidas pela *Framework* não sejam suficientes para lidar com um dado cenário. A título de exemplo, pode ser necessário criar um ponto de extensibilidade sobre a serialização das mensagens de modo a lidar com um tipo de codificação diferente daqueles suportados pelo WCF [34].

2.4. Mecanismos de segurança

Tal como já foi referido, o mecanismo de segurança adotado pela *MedicineOne* para garantir a proteção dos dados, trocados durante a comunicação das suas aplicações com os seus serviços, é o HTTPS, que se baseia na extensão das capacidades de segurança do protocolo de criptografia SSL sobre comunicações HTTP.

Embora o protocolo SSL seja amplamente utilizado para a transmissão de informação sensível de modo seguro, são conhecidas algumas vulnerabilidades que podem colocar em causa a segurança dessa informação. A própria *Microsoft* recomenda que as medidas de segurança a adotar na *Framework* WCF, aquando do desenvolvimento de serviços desenhados para atuar num cenário de *Internet*, devem ser aplicadas a nível da mensagem, ao invés do nível de transporte, tal como ocorre com este protocolo. Esta recomendação baseia-se no fato da segurança, a nível de transporte, fornecer um canal seguro ponto-a-ponto, isto é, este apenas é estabelecido entre o cliente e o servidor que lhe é exposto, que pode corresponder a um servidor intermediário, tendo deste modo acesso ao conteúdo da mensagem. Por outro lado, a segurança a nível da mensagem oferece um canal seguro fim-a-fim, o que significa que apenas o destinatário final terá acesso ao conteúdo da mensagem [35].

Posto isto, as próximas subsecções visam apresentar o funcionamento do protocolo SSL, que permitirá compreender as razões de algumas das suas vulnerabilidades, as quais também serão abordadas.

Na última subsecção é levada a cabo uma análise aos diversos mecanismos de segurança que assegurem a confidencialidade e integridade das mensagens, da qual resultará a seleção da solução a adotar para atuar em complemento com o protocolo SSL.

2.4.1 Funcionamento do protocolo SSL

O protocolo SSL encontra-se associado aos conceitos de criptografia simétrica e assimétrica. A criptografia simétrica baseia-se na combinação de um algoritmo criptográfico com uma única chave para encriptar e desencriptar uma mensagem, o que implica que, tanto o seu recetor, quanto o seu emissor necessitam de conhecer, à partida, a chave a utilizar. Por outro lado, a criptografia assimétrica combina um algoritmo criptográfico com um par de chaves, uma privada e uma pública, sendo que uma mensagem encriptada com uma destas apenas pode ser desencriptada pela outra. Embora a criptografia assimétrica exija mais poder de processamento para encriptar e desencriptar, a distribuição de chaves resulta num processo mais simples, quando comparado com a criptografia simétrica, pois a chave pública pode ser disponibilizada livremente, enquanto a chave privada é mantida em segredo [36] [37].

A conjugação de ambos os mecanismos criptográficos permite ao SSL tirar partido das vantagens de cada um e mitigar as suas lacunas, sendo que, durante o estabelecimento de uma conexão segura, é usada a criptografia assimétrica e, após a criação de um canal de comunicação seguro, é realizada a troca de uma chave simétrica, que é usada durante a restante comunicação para encriptar e desencriptar os dados [37].

Para finalizar, é importante referir o conceito de assinatura digital, associado à criptografia assimétrica, que resulta da aplicação de uma função de *hash* sobre uma mensagem de tamanho variável, que origina um código de tamanho fixo irreversível e único para a mesma mensagem, denominado de valor de *hash*, que posteriormente é encriptado com a chave privada do emissor e enviado em conjunto com a mensagem.

Com base na premissa de que uma mensagem desencriptada com a chave pública teve de ser previamente encriptada com a chave privada correspondente, ao desencriptar a assinatura digital com a chave pública do emissor, obtendo assim o valor de *hash* gerado por este, e ao aplicar a mesma função de *hash* sobre a mensagem, o recetor pode comparar ambos os valores de *hash* e, desse modo, garantir a autenticidade da sua origem, a integridade do seu conteúdo e o princípio de não-repudiação.

O SSL emprega este conceito em conjunto com a utilização de certificados digitais, que permitem associar uma chave pública à entidade que contém a chave privada correspondente, pois, de outro modo, qualquer indivíduo poderia distribuir a sua chave pública, fazendo-se passar por outra identidade. Para efetuar essa associação é necessário recorrer a entidades terceiras, denominadas de autoridades de certificação (AC), que emitem os certificados e que asseguram a identidade dos seus detentores [36] [38].

A Figura 5 pretende ilustrar, de forma simplificada, os passos necessários para estabelecer uma conexão segura, recorrendo ao protocolo SSL:

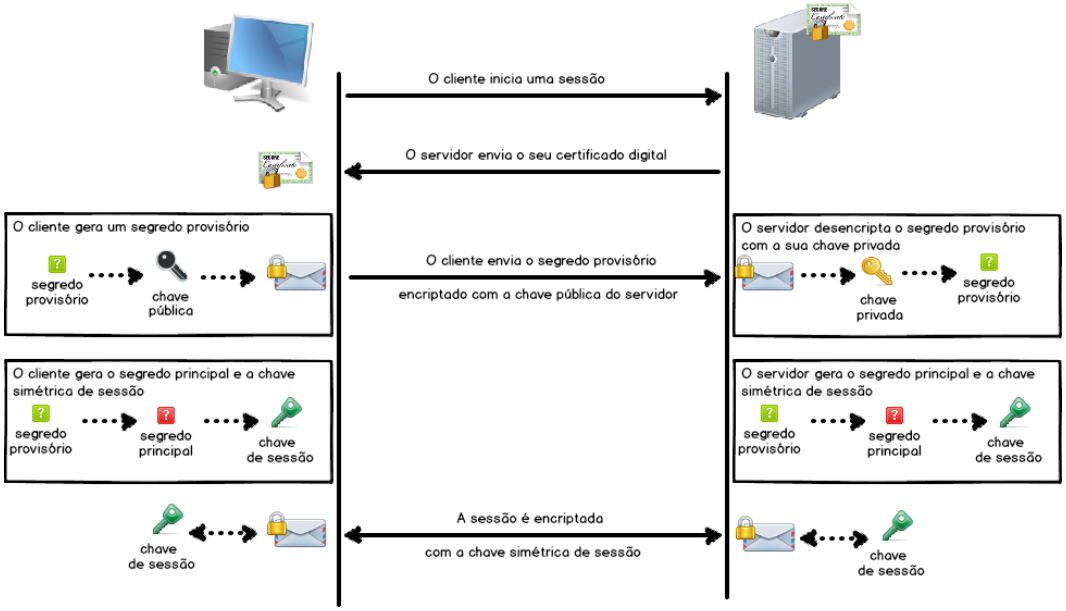

Figura 5 - Estabelecimento de uma conexão segura

De forma resumida, o estabelecimento de uma sessão SSL inicia-se por parte do cliente, que envia uma mensagem com diversos parâmetros para o servidor, sendo que este lhe responde com uma mensagem acompanhada do seu certificado digital, entre outros campos.

De seguida, o cliente autentica o servidor com base nas informações presentes no certificado digital, validando os seguintes itens:

- O certificado ainda não expirou;
- O certificado foi emitido por uma AC confiável;
- A chave pública da AC valida a assinatura digital do certificado;
- O nome de domínio presente no certificado corresponde ao nome de domínio do servidor.

Caso todos os pontos anteriores sejam válidos, o servidor é autenticado e, recorrendo aos dados trocados durante a fase inicial, o cliente gera um segredo provisório, que é encriptado com a chave pública fornecida no certificado, e este é enviado para o servidor, que por sua vez recorre à sua chave privada para o desencriptar.

Tendo em conta que tanto o cliente como o servidor conhecem o segredo provisório, ambos dão início a um conjunto de passos para gerar o segredo principal, que é utilizado para originar a chave de sessão. A partir deste momento, as mensagens trocadas durante a sessão estabelecida passam a ser encriptados e desencriptados com recurso a essa chave, que apenas ambos conhecem [39] [40].

2.4.2 Vulnerabilidades do protocolo SSL

Para além dos conceitos de criptografia e certificação digital, já abordados, subsiste outro fator, de natureza não informática, paralelamente ao funcionamento do SSL, que assenta na confiança depositada nas AC e na sua competência. Tal como foi referido, os clientes obtêm uma garantia de que estão a comunicar com o servidor associado ao recorrerem aos certificados digitais fornecidos. Contudo, para que este mecanismo funcione, é necessário que confiem no valor de verdade dos certificados digitais, isto é, têm de confiar nas AC que os emitem.

Devido ao elevado número de entidades que necessitam de certificados digitais, tornou-se impensável a existência de apenas uma AC que fizesse a gestão e monitorização de todos os certificados. Para solucionar este problema recorreu-se uma hierarquia de certificados, onde uma AC de raiz, situada no topo da mesma, emite certificados para outras de nível inferior, denominadas de AC subordinadas, que, por sua vez, podem emitir certificados para outros subordinados, ou para utilizações específicas. Deste modo, é assegurada uma cadeia de confiança em que qualquer certificado emitido por uma AC subordinada é considerado válido pelo cliente, desde que este confie na AC de raiz que emitiu o seu certificado [41].

Uma vez que um certificado digital emitido por uma AC é considerado válido por todos os clientes que confiam nela, se um atacante conseguir explorar uma vulnerabilidade que a comprometa, então esses clientes passam também a estar comprometidos.

Um dos ataques que demonstra o impacto deste cenário ocorreu no dia 29 e 30 de Janeiro de 2001, quando a *VeriSign Inc.*, uma das autoridades de certificação do topo da hierarquia, emitiu dois certificados digitais a um atacante que se fez passar por um funcionário da *Microsoft*, permitindo, assim, que este pudesse assinar código executável com esses certificados e, deste modo, convencer os utilizadores de que estariam a executar código proprietário da *Microsoft* [42]. Outro ataque, com contornos distintos, foi reportado no dia 3 de Dezembro de 2013 quando a *Google* descobriu que tinham sido emitidos certificados digitais para alguns dos seus domínios. Após uma fase de investigação, foi identificado que uma AC subordinada, cuja autoridade tinha sido herdada da *Agence nationale de la sécurité des systèmes d'information* (ANSSI) [43], tinha gerado esses certificados, que estavam a ser usados numa rede privada a fim de inspecionar o tráfego encriptado direcionado para esses domínios [44].

Tal como se pode verificar pelos exemplos apresentados, a segurança do protocolo SSL encontra-se, em certa parte, dependente da competência das AC e da confiança depositada nestas, pelo que o fato de existirem diversas AC ao longo da hierarquia pode contribuir para o aumento de número de ocorrências destes cenários.

É, ainda, importante referir que, para além das AC, qualquer utilizador poderá recorrer a ferramentas, como o *openssl* [45], para gerar os seus próprios certificados, que, caso os clientes optem por confiar nestes, têm tanta validade comos os gerados pelas AC. Isto significa que os atacantes têm a seu dispor uma brecha que pode ser explorada para levar a cabo ataques *man-in-the-middle* [46], pois, devido ao desconhecimento do funcionamento do protocolo SSL, os clientes muitas vezes confiam em certificados não fidedignos sem se aperceberem dos riscos associados.

Uma ferramenta que permite ilustrar o cenário descrito é o *Charles Web Debugging Proxy*, que atua como um *proxy* HTTP e que permite monitorizar todo o tráfego HTTP e HTTPS gerado entre uma máquina e a *Internet* [47]. De modo a capturar o tráfego HTTPS de forma desencriptada, o *Charles* atua como um *man-in-the-middle* e, cada vez que um cliente tenta estabelecer uma ligação segura a um servidor, este gera, automaticamente, um certificado digital para esse servidor, que é assinado com o seu próprio certificado de raiz, e envia-o para o cliente. Por outro lado, o *Charles* recebe o certificado do servidor e estabelece uma sessão SSL com este, pelo que, tanto o servidor, como o cliente, são levados a pensar que estão a comunicar diretamente um com o outro, quando na realidade ambos estão a comunicar com o *Charles* através de sessões SSL distintas [48].

Uma vez que o cliente encripta as mensagens, com recurso à chave pública fornecida pelo certificado digital gerado pelo *Charles*, o tráfego HTTPS recebido por este é desencriptado sem qualquer restrição. Contudo, é importante referir que para que o *Charles* atue como *man-in-the-middle* e capture o tráfego HTTPS, é necessário que o cliente adicione o certificado de raiz do *Charles* à lista de certificados confiáveis da sua máquina, permitindo, assim, que os certificados gerados para os servidores sejam válidos, pois, caso contrário, estes podem ser rejeitados, por não serem fidedignos.

A Figura 6 pretende ilustrar, de forma simplificada, o funcionamento do *Charles* como *man-in-the-middle*:

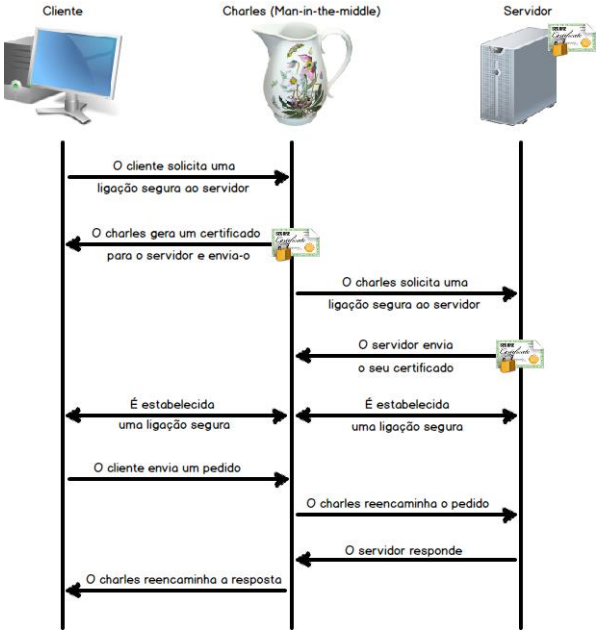

Figura 6 - Funcionamento do *Charles* como *man-in-the-middle*

Para comprovar que as medidas de segurança aplicadas atualmente pela *MedicineOne* não são suficientes para proteger as mensagens quando os mecanismos do SSL são contornados, foi instalada numa das máquinas da empresa a ferramenta *Charles Web Debugging Proxy* e procedeu-se à invocação de serviços existentes nos servidores da *MedicineOne*, cuja resposta dos mesmos foi capturada pelo *Charles*, configurado para atuar como *man-in-the-middle*.

A seguinte figura ilustra uma dessas respostas:


```
"GetPatientResult": {
    "Address": "",
    "BirthDate": "01-02-1972 00:00:00",
    "CCNumber": null,
    "CCValidityDate": null,
    "CellPhone": "93123654",
    "Email": "maria.doente@gmail.com",
    "FathersName": null,
    "HealthSystemList": [{
        "BeneficiaryNumber": "1234567890",
        "DocumentNumber": "",
        "ExpirationDate": "",
        "HealthSystemName": "SNS",
        "MigrantEntityCode": "",
        "MigrantEntityDescription": "",
        "NationalNumberMandatory": true,
        "PkEntityDocumentType": "",
        "PkHealthSystem": "935601",
        "PkMigrantEntityCountry": "",
        "PkRegister": "baf067a7-b128-42fb-b2ee-67cbfc37050d",
        "PreferentialEntity": "False"
    }, {
        "BeneficiaryNumber": "1234567890",
        "DocumentNumber": "",
        "ExpirationDate": "13-12-2015 00:00:00",
        "HealthSystemName": "ADSE",
        "MigrantEntityCode": "",
        "MigrantEntityDescription": "",
        "NationalNumberMandatory": false,
        "PkEntityDocumentType": "",
        "PkHealthSystem": "911001",
        "PkMigrantEntityCountry": "",
        "PkRegister": "848f4f9d-c2e3-441a-960a-7caa0640172d",
        "PreferentialEntity": "True"
    }],
    "MothersName": null,
    "Name": "Maria Isabel Doente",
    "NationalNumber": "193659786",
    "OS": false,
    "OSEndDate": null,
    "OSMotive": "",
    "Phone": "",
    "Photo": null,
    "Pk": "6d82e0b4-1e59-4633-b7b5-d90582cd6f9d",
    "PkCounty": "60300",
    "PkDistrict": "60000",
    "PkInstalation": "62f115e7-6dc6-42ad-8b7e-36bcfe07bf65",
    "PkNationality": 620,
    "PkOrganization": "e7415246-b622-4960-a308-e386c761b237",
    "PkParish": "60328",
    "PkSex": "1",
    "ProcessNumber": "969718",
    "RECM": false,
    "RECMEndDate": "15-04-2019 00:00:00",
    "RECMMotive": "4001",
    "TaxpayerIdNumber": null
}
}
```


Figura 7 - Resposta de um serviço capturada pela ferramenta *Charles*

Embora a *MedicineOne* recorra ao protocolo SSL para estabelecer canais de comunicação encriptados entre as suas aplicações e os seus serviços, observando a figura anterior é possível verificar que, através da utilização da ferramenta *Charles*, facilmente se consegue analisar o conteúdo de uma mensagem que, à partida, apenas deveria ser acessível para as aplicações e serviços da *MedicineOne*.

Tendo em conta a informação apresentada na resposta ilustrada na Figura 7, rapidamente se conclui que o impacto de um ataque deste tipo é bastante negativo para a empresa, pois o atacante pode ter acesso a diversos dados que se encontram protegidos por leis de proteção de informação.

Apesar de existirem outros ataques, de maior complexidade, que permitem explorar outras vulnerabilidades do protocolo SSL, os exemplos apresentados ao longo desta subsecção são suficientes para demonstrar que, embora o SSL forneça mecanismos de segurança bastante confiáveis e seja amplamente utilizado, se as premissas do mesmo não forem respeitadas, como por exemplo as ACs serem comprometidas ou os clientes confiarem em certificados não fidedignos, a segurança das mensagens é colocada em risco, e estas podem ser acedidas pelos atacantes, embora existam mecanismos que permitem mitigar estas vulnerabilidades.

Posto isto, a *MedicineOne* optou por expandir as suas medidas de segurança, de modo a garantir que a confidencialidade e integridade das mensagens trocadas entre as suas aplicações e os seus serviços são asseguradas por mecanismos criptográficos, independentes ao protocolo SSL.

2.4.3 Confidencialidade e integridade

Ao longo da presente secção tem vindo a ser referido que a *MedicineOne* pretende garantir a confidencialidade e integridade das mensagens trocadas entre as suas aplicações e os seus serviços, não tendo sido ainda, no entanto, exposta a definição destes termos.

A confidencialidade de uma mensagem refere-se à capacidade de impedir que entidades não autorizadas tenham acesso ao seu conteúdo, enquanto a integridade da mesma se refere à capacidade de garantir que o seu conteúdo não foi alterado por terceiros. Em conjunto com os termos autenticação, que diz respeito à capacidade de assegurar a identidade das entidades envolvidas numa troca de mensagens, e não-repúdio, que impede as entidades envolvidas numa troca de mensagens de negar a sua participação, estes definem os objetivos da criptografia [49].

Uma vez que diferentes mecanismos de segurança têm diferentes finalidades, isto é, nem todos procuram alcançar os objetivos descritos anteriormente na sua totalidade, é necessário analisar os aspetos de segurança que se pretendem alcançar e selecionar os mecanismos que vão ao encontro destes. No caso da *MedicineOne*, as necessidades de segurança passam pela garantia de confidencialidade e integridade das mensagens, pelo que devem ser analisados mecanismos de segurança que ofereçam esses serviços.

Tal como já foi referido previamente, existem dois mecanismos de segurança que possibilitam a encriptação de uma mensagem de modo a que apenas o seu emissor e o seu recetor tenham acesso a esta: a criptografia assimétrica, que recorre a um par de chaves, e a criptografia simétrica, que recorre apenas a uma chave.

Embora a criptografia assimétrica ofereça garantias de confidencialidade e integridade, o fato desta requerer um elevado poder de processamento durante o processo de encriptação e desencriptação, remete para que a sua utilização seja frequentemente direcionada para fins de autenticação, através das assinaturas digitais, ao invés de encriptação. Posto isto, e tendo em conta a capacidade de processamento do *iPhone*, a adoção deste mecanismo de segurança, como meio para garantir a confidencialidade e integridade das mensagens trocadas entre as aplicações da *MedicineOne* e os seus serviços, não é justificável.

Por outro lado, a criptografia simétrica requer menos poder de processamento, uma vez que a mesma chave é utilizada para encriptar e desencriptar as mensagens, pelo que este mecanismo é bastante utilizado para fornecer garantias de confidencialidade.

Tal como foi mencionado, a única desvantagem deste mecanismo está associada à distribuição da chave secreta, pois ambos, o emissor e o recetor, têm de a conhecer previamente. Contudo, tendo em conta a natureza deste projeto, esta desvantagem pode, facilmente, ser contornada, pois as aplicações da *MedicineOne* podem vir previamente munidas com a chave secreta a utilizar, permitindo, assim, que as mensagens trocadas entre estas e os servidores da *MedicineOne*, que também conhecem a chave, sejam encriptadas e desencriptadas por esta. Existem porém algumas desvantagens associadas à utilização deste mecanismo no presente cenário, pois caso a chave secreta seja comprometida não existe nenhuma solução eficaz para proceder à sua alteração nas aplicações instaladas nos *iPhones* dos utilizadores. Ainda assim, o mecanismo de encriptação a adotar para garantir a confidencialidade das mensagens recai sobre a criptografia simétrica.

Embora seja bastante comum concluir-se que a encriptação de uma mensagem é suficiente para garantir a sua integridade, pois se o atacante não possui a chave secreta para a desencriptar então não poderá alterar o seu conteúdo, esta falsa premissa pode conduzir a graves vulnerabilidades que colocam em causa a segurança das mensagens.

Apesar da encriptação de uma mensagem garantir que pessoas não autorizadas não têm acesso ao seu conteúdo, se um atacante capturar uma mensagem encriptada e adulterar a mesma, alterando os *bits* que a compõem, não existe forma do recetor detetar essas alterações durante a sua desencriptação. Ainda que este processo seja relativamente complexo e dependa dos métodos de encriptação aplicados, os atacantes podem tirar partido desta lacuna para alterarem o conteúdo das mensagens encriptadas [50].

Para além de poder alterar o conteúdo das mensagens encriptadas sem que o recetor o detete, o atacante pode, ainda, recorrer a este técnica para levar a cabo outros ataques que possibilitem a sua desencriptação.

Dois investigadores da Universidade *Ruhr-University Bochum*, na Alemanha, apresentaram um ataque levado a cabo sobre a encriptação *Extensible Markup Language* (XML), estandardizada pela *World Wide Web Consortium* (W3C) e utilizada por diversas entidades, como a *IBM* e a *Microsoft*, em que tiraram partido do fato das recomendações do processo de encriptação não obrigarem à utilização de mecanismos que garantam a integridade dos dados, pelo que os investigadores procederam à adulteração das mensagens encriptadas e à posterior análise das mensagens de erro retornadas pelo servidor, o que lhes permitiu obter informações suficientes para proceder à desencriptação das mesmas [51] [52].

Tal como foi aludido, diferentes mecanismos de segurança têm diferentes finalidades, pelo que, para garantir a integridade das mensagens, revela-se necessário recorrer a mecanismos distintos daqueles utilizados para assegurar a sua confidencialidade. Recordando o funcionamento das funções de *hash*, estas permitem a geração de um valor de *hash* de tamanho fixo, irreversível e único para uma dada mensagem. A criptografia assimétrica recorre a estas funções em conjunto com a encriptação do valor gerado por estas com recurso ao par de chaves para gerar uma assinatura digital para uma dada mensagem, o que permite garantir não só a integridade da mesma, mas também a sua autenticação e o princípio de não repúdio. Para além destes mecanismos, é ainda importante referir o conceito de *Message Authentication Code* (MAC), que também permite garantir a integridade dos dados.

À semelhança dos valores gerados pelas funções de *hash*, um MAC corresponde a um bloco de dados de tamanho fixo gerado a partir de um algoritmo, com a diferença de este ser gerado através da combinação de uma dada mensagem com uma chave secreta, que apenas o emissor e o recetor conhecem.

O processo de validação da integridade dos dados é semelhante ao aplicado na assinatura digital, pois o emissor gera um MAC através da combinação da mensagem com a chave secreta e envia este código em conjunto com esta. Por sua vez, o recetor recalcula o código MAC, com base na mensagem recebida e na chave secreta partilhada, e compara-o com o código recebido, garantindo assim que a mensagem original não sofreu modificações [50] [53].

Analisando as assinaturas digitais e os MAC, sob o mesmo ponto de vista adotado durante a comparação entre a criptografia assimétrica e simétrica, o primeiro mecanismo apresenta a desvantagem de recorrer a um par de chaves, enquanto o segundo requer apenas a utilização de uma chave secreta. Pelas mesmas razões apresentadas durante a escolha do mecanismo de encriptação, a integridade das mensagens trocadas entre as aplicações da *MedicineOne* e os seus servidores será garantida pela utilização de MAC.

Adotados os mecanismos que irão garantir a confidencialidade e integridade das mensagens, torna-se necessário analisar os diferentes modos de combinar as operações de encriptação e autenticação para gerar a mensagem cifrada, e que são os seguintes:

- *Encrypt and Authenticate* (E&A);
- *Authenticate then Encrypt* (AtE);
- *Encrypt then Authenticate* (EtA);

No primeiro caso, a mensagem cifrada resulta da encriptação do seu conteúdo e da posterior anexação do MAC gerado para este, sendo que o processo de desencriptação e verificação da integridade inicia-se com a desencriptação da mensagem, de modo a obter o seu conteúdo, e a posterior análise do MAC, sendo este o método adotado pelo protocolo SSH [54].

No método *Authenticate then Encrypt* a mensagem cifrada é gerada através da encriptação do seu conteúdo com conjunto com o MAC gerado para este, sendo que o protocolo SSL recorre a uma variante deste método. O processo inverso inicia-se com a desencriptação da mensagem, que permite obter o seu conteúdo e o MAC, e termina com a verificação da integridade através da comparação do MAC recebido e do gerado pelo recetor [54].

Para finalizar, o último método baseia-se na encriptação do conteúdo da mensagem e na posterior anexação do MAC gerado para o conteúdo encriptado, sendo que o processo de desencriptação e verificação se inicia com a validação do MAC e a posterior desencriptação da mensagem cifrada [54].

A Figura 8 pretende ilustrar os três métodos descritos anteriormente, considerando que M corresponde ao conteúdo da mensagem a codificar e C diz respeito ao texto cifrado resultante do processo de encriptação:

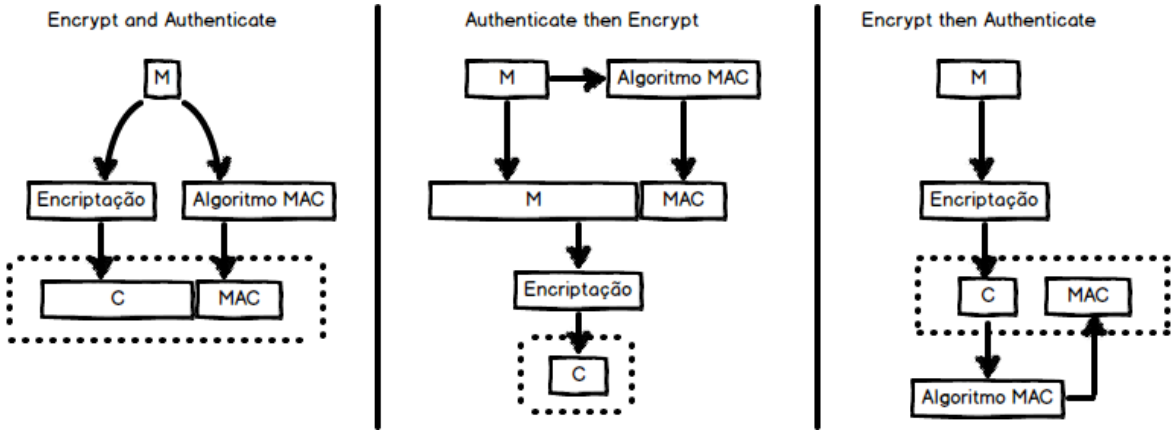

Figura 8 - Combinação dos processos de encriptação e autenticação para gerar uma mensagem cifrada

Destes três métodos, o único que garante a integridade das mensagens, antes destas serem desencriptadas, isto é, a integridade das mensagens cifradas, é o último, pois os restantes apenas garantem a integridade do conteúdo após a sua desencriptação.

Ao garantir a integridade das mensagens cifradas é possível detetar aquelas que tenham sido modificadas, antes de proceder à sua desencriptação, e pode-se evitar a obtenção de informações sobre o conteúdo das mesmas através do seu MAC, uma vez que este é gerado sobre o conteúdo encriptado, que para um mesmo valor apresenta resultados diferentes.

Posto isto, o mecanismo mais seguro diz respeito ao *Encrypt then Authenticate*, pois os restantes possibilitam aos atacantes a condução de ataques que lhes permitam decifrar o conteúdo das mensagens ou proceder à sua alteração de modo indetetável [55].

Definidos os mecanismos a adotar para garantir a confidencialidade e integridade das mensagens trocadas e após a análise dos diferentes métodos de combinação do processo de encriptação e autenticação de modo a gerar uma mensagem cifrada, procedeu-se à pesquisa de bibliotecas que fossem ao encontro das decisões tomadas e que pudessem ser implementadas tanto do lado das aplicações da *MedicineOne* como do lado dos seus serviços, por questões de interoperabilidade.

Tendo em conta estes dois requisitos, a biblioteca *RNCryptor* [56] apresenta as características necessárias, que são resumidas nos seguintes pontos:

- Encontra-se implementada nas linguagens *Objetive-C* e C#;

- Utiliza o mecanismo de encriptação *Advanced Encryption Standard* (AES) com uma chave de 256 *bits*, que é um dos algoritmos de criptografia simétrica mais populares [57];

- Utiliza o algoritmo *Hash-based Message Authentication Code* (HMAC) combinado com a função *Secure Hash Algorithm*-256 (SHA-256). HMAC refere-se a um dos tipos de algoritmos usados para gerar um MAC, que se baseia na utilização de uma função de hash subjacente sobre uma mensagem e uma chave secreta;

- Utiliza o método *Encrypt then Authenticate*.

2.5. Conclusão

Como resultado da análise às plataformas e tecnologias envolventes ao módulo de agendamento, o estagiário adquiriu conhecimentos essenciais e tomou decisões importantes para o futuro trabalho a desenvolver.

Da análise crítica elaborada aos diversos componentes de navegação, constatou-se que o esforço de adaptação e reutilização dos mesmos seria bastante superior ao desenvolvimento de uma solução de raiz, o que conduziu a optar-se pela segunda escolha.

O estudo realizado sobre a linguagem de programação *Objective-C* fomentou o conhecimento das principais *Frameworks* fornecidas pela camada de mais alto nível do *iOS*, que englobam o *Core Data*, *Core Animation* e *UIKit*, e originou a elaboração de um guia que permite melhorar a performance do módulo a desenvolver. Neste ponto, também foi realizada uma análise prévia à biblioteca de acesso à rede utilizada pela equipa da *MedicineOne*.

Do estudo elaborado sobre a *Framework* WCF, que permite desenvolver aplicações orientadas a serviços, resultou a aquisição de conhecimentos associados aos conceitos *Endpoint*, através do qual ocorre a comunicação entre um cliente e um serviço, *Address*, que define onde o serviço se encontra, *Contract*, que descreve as funcionalidades expostas por um *endpoint*, e *Binding*, que especifica como comunicar com um *endpoint*.

No que respeita à análise dos mecanismos de segurança a adotar, para garantir a proteção das mensagens, o estagiário optou por recorrer à biblioteca *RNCryptor*, que permite garantir a confidencialidade e integridade das mensagens com recurso aos princípios de criptografia simétrica, MAC e *Encrypt then Authenticate*, que foram mencionados pelo estagiário como sendo os mecanismos adequados para perfazer os requisitos apresentados.

Capítulo 3
Planeamento e metodologia

O presente capítulo apresenta o planeamento definido na fase inicial do estágio relativamente ao plano de trabalhos a desenvolver durante o primeiro e o segundo semestre em que este decorreu, expõe os desvios que surgiram ao longo destes semestres, relativamente a essa planificação, e aborda a metodologia adotada durante o processo de desenvolvimento do módulo de agendamento.

3.1. Planeamento do estágio

O planeamento do estágio encontra-se dividido pelos dois semestre em que este decorreu, sendo que o seguinte diagrama de *Gantt* ilustra o plano de trabalhos agendado para o primeiro semestre:

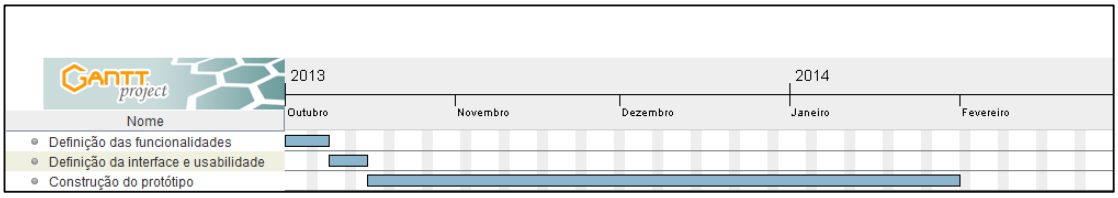

Figura 9 - Planeamento do 1º semestre

A definição das funcionalidades corresponde à elaboração de um documento, por parte da equipa da *MedicineOne*, onde são descritas aquelas que devem ser implementadas no módulo de agendamento. Esta tarefa deve ser acompanhada de perto pelo estagiário, de modo a que este conheça as funcionalidades em detalhe.

A definição da *interface* e usabilidade diz respeito à elaboração dos *mockups*, que servem de base para os primeiros testes de usabilidade, e à definição dos fluxos de utilização.

A construção do protótipo corresponde à tarefa mais longa deste semestre e assenta no desenvolvimento do protótipo funcional do componente de navegação do módulo de agendamento, que deve incorporar o material gráfico desenvolvido pela equipa de *design* da *MedicineOne*.

No que diz respeito ao planeamento do segundo semestre, a figura que se segue ilustra as tarefas a realizar:

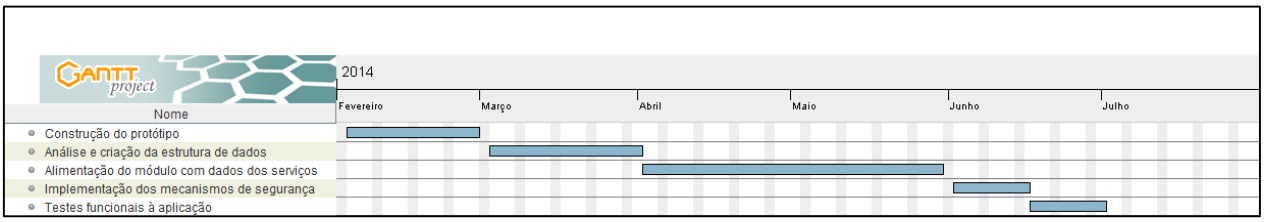

Figura 10 - Planeamento do 2º semestre

A primeira tarefa corresponde à conclusão da construção do protótipo do componente de navegação, iniciada no primeiro semestre.

A análise e criação da estrutura de dados respeita à identificação e criação das tabelas do modelo de dados, com recurso à *Framework Core Data*, de modo a armazenar os dados trocados

durante a comunicação entre o módulo e os serviços, e ao desenvolvimento das vistas que recorrem aos dados armazenados por estas.

A tarefa seguinte corresponde à criação dos serviços e à alimentação do módulo com os dados provenientes destes, através do seu carregamento no modelo de dados.

A terceira tarefa respeita à implementação dos mecanismos de segurança no módulo de agendamento e nos serviços, garantindo, deste modo, que as mensagens trocadas entre ambos se encontram protegidas.

A última tarefa do estágio diz respeito à condução de uma série de testes funcionais sobre o módulo, por parte da equipa da *MedicineOne*, que têm o intuito de verificar se as funcionalidades cumprem os requisitos especificados.

3.1.1 Desvios do planeamento

Ao longo do estágio nem sempre foi possível respeitar a planificação descrita na secção anterior. Embora as tarefas planeadas para o plano de trabalhos do primeiro semestre tenham sido realizadas sem quaisquer desvios, no segundo semestre surgiram algumas alterações relativamente à ordem de trabalhos e aos prazos estabelecidos, que serão apresentadas de seguida.

Durante a fase de análise e criação da estrutura de dados, o estagiário identificou a vantagem em proceder à criação das tabelas em paralelo com o desenvolvimento dos serviços, uma vez que a organização destas depende da estrutura dos dados retornados por estes. Posto isto, após um período de desenvolvimento recorrendo a tabelas fictícias, o estagiário procedeu à criação dos serviços e, em paralelo, identificou e criou as tabelas necessárias de acordo com os dados retornados por estes. Esta decisão permitiu agilizar a execução de ambas tarefas, pelo que ao invés da sua duração conjunta ocupar um período de três meses, ambas foram concluídas em apenas um mês e meio.

Aquando da criação dos serviços, surgiram alguns imprevistos que levaram ao condicionamento do desenvolvimento de certas funcionalidades do módulo de agendamento. Uma vez que os serviços a desenvolver recorrem a outros já criados pela equipa da *MedicineOne*, que implementam as suas regras de negócio, o estagiário deparou-se com a inexistência de serviços que dessem suporte a certas funcionalidades do módulo de agendamento.

Face a essa situação, foi necessário colocar o desenvolvimento de algumas funcionalidades em segundo plano, sendo que o estagiário deu início à análise e implementação dos mecanismos de segurança de modo a dar continuidade ao seu trabalho, até que fosse encontrada uma solução por parte da equipa da *MedicineOne*.

Até à data da conclusão da implementação dos mecanismos de segurança, a equipa da *MedicineOne* ainda não tinha encontrado qualquer solução para o problema apresentado, pelo que o estagiário optou por concluir o desenvolvimento das funcionalidades pendentes, que ficaram assim preparadas para interagir com os futuros serviços.

Posto isto, na fase final do estágio a equipa da *MedicineOne* conduziu os testes funcionais sobre o módulo tendo em conta a inexistência de alguns serviços.

3.2. Metodologia de desenvolvimento

A metodologia de desenvolvimento adotada pela equipa da *MedicineOne* e, consequentemente, pelo estagiário recaiu sobre o *Kanban*. Esta metodologia considera que o processo de desenvolvimento de *software* pode ser, metaforicamente, caracterizado como um tubo que recebe por uma das suas entradas pedidos de funcionalidades e emite pela outra o *software* com estas incorporadas, sendo que, no seu interior, decorrem os processos de análise de requisitos, desenvolvimento de código e testes [58].

Um dos princípios do *Kanban* é de que as equipas, para serem eficientes, não devem estar sobrecarregadas, uma vez que isto prejudica a qualidade do serviço que produzem [59]. Logo, o funcionamento do *Kanban* assenta na utilização de um quadro onde são representadas, por diferentes colunas, as várias fases de desenvolvimento de *software*, como, por exemplo, a fase de implementação, ou de testes, sendo que, em cada uma, são colocados cartões que representam itens de trabalho que vão transitando pelas diversas fases de desenvolvimento. De modo a garantir que as equipas não ficam sobrecarregadas, no topo de cada coluna é colocado um número que representa o limite de itens de trabalho permitidos em cada fase [58].

A seguinte figura ilustra um exemplo da utilização do *Kanban*:

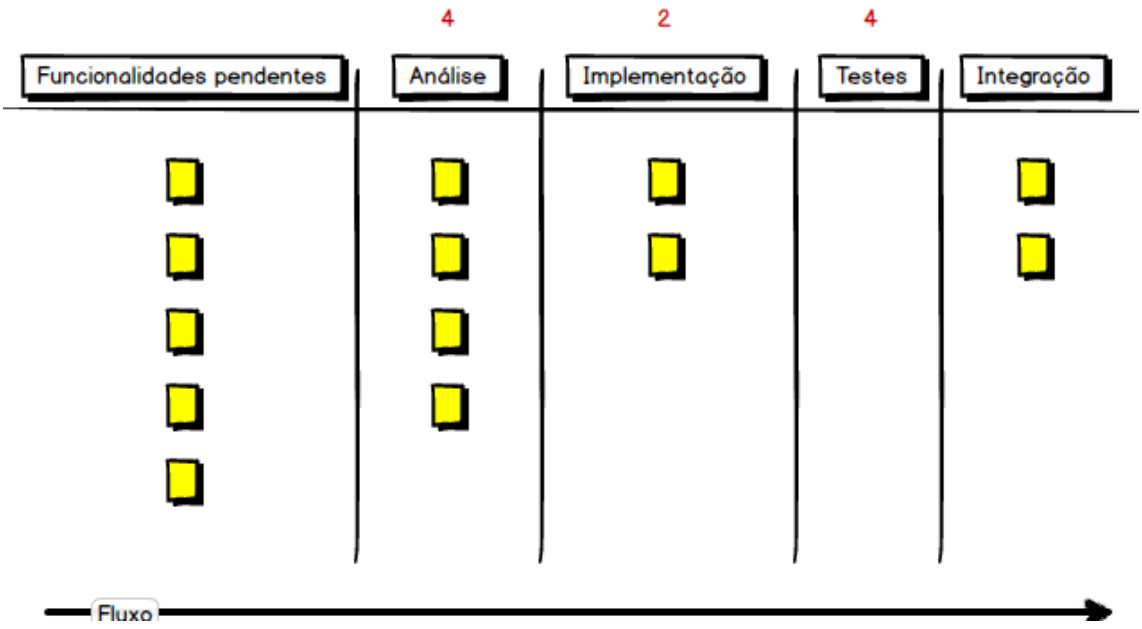

Figura 11 - Exemplo da aplicação da metodologia *Kanban*

Como se pode verificar na Figura 11, as fases de desenvolvimento em que o *Kanban* foi aplicado foram divididas pelas colunas apresentadas.

A cada uma destas fases foi associado o número de cartões máximos permitidos, que está representado no topo de cada coluna, através do número vermelho, sendo que cada fase apenas pode aceitar novos itens se o seu limite ainda não tiver sido atingido.

Por exemplo, a fase de análise não pode aceitar novos itens das funcionalidades pendentes até que a fase de implementação envie um dos seus cartões para a fase de testes e aceite um dos itens da fase de análise. De acordo com a metáfora já apresentada, a coluna das funcionalidades pendentes pode ser considerada uma das entradas do tubo, enquanto a coluna correspondente à integração, representa a outra.

Recorrendo a esta metodologia de desenvolvimento, não só é possível evitar a sobrecarga das equipas, como também se pode detetar, facilmente, qual das fases de desenvolvimento poderá estar a causar efeitos de *bottleneck*, que no exemplo da Figura 11 pode ser apontado à fase de implementação, possibilitando que sejam efetuadas alterações nas equipas que permitam contrariar este efeito [58].

A aplicação desta metodologia, no desenvolvimento do módulo de agendamento, revelou-se bastante semelhante à apresentada no exemplo da Figura 11. Embora tenha sido elaborada uma análise prévia às funcionalidades a implementar, o estagiário inicia o desenvolvimento de uma destas com a sua análise detalhada.

Concluída essa análise, prossegue a fase de implementação da funcionalidade, que quando finalizada dá início à fase de testes. Após a execução de uma série de testes realizados pelo estagiário, a funcionalidade é dada como concluída e este procede ao desenvolvimento de outra. No que diz respeito aos limites dos itens em cada fase, uma vez que o estagiário é o único membro a desenvolver o módulo de agendamento, existe o limite de um cartão por cada fase.

Capítulo 4
Desenho da solução

Após o estudo realizado sobre as diferentes plataformas e tecnologias envolventes ao módulo de agendamento, que foram abordadas no capítulo 2, o estagiário procedeu à fase de planeamento da solução a adotar para o seu desenvolvimento.

Esta fase resultou da análise da documentação fornecida pela *MedicineOne*, onde são abordadas as diversas funcionalidades que o módulo de agendamento deve suportar, e de um conjunto de reuniões efetuadas entre o estagiário e a equipa de desenvolvimento, que possibilitaram a troca de ideias.

Assim sendo, as seguintes secções abordam a especificação dos requisitos do módulo, que possibilitará uma melhor compreensão do seu funcionamento e do seu comportamento quando o utilizador interage com este, e a definição da arquitetura do ambiente em que o módulo será inserido, onde serão abordados os diversos componentes que contribuem para o funcionamento do módulo de agendamento.

4.1. Análise dos requisitos

Através da análise da documentação fornecida e da troca de opiniões com os membros da equipa da *MedicineOne*, o estagiário identificou os diversos requisitos que ditam o funcionamento do módulo de agendamento. A especificação destes requisitos conduziu à definição das diversas funcionalidades a implementar e forneceu uma melhor compreensão sobre o comportamento que cada uma deve apresentar perante os diversos cenários de utilização do módulo.

Da análise, previamente referida, resultaram dois conjuntos de requisitos: os funcionais e os não funcionais. Os primeiros estão relacionados com o modo de funcionamento do módulo, como por exemplo as imposições sobre a forma como deve ser feita a navegação na agenda, ou a gestão das consultas, enquanto os segundos respeitam a restrições sobre o modo de atuação do módulo como, por exemplo, a existência de necessidades de segurança, associadas à comunicação com os servidores da *MedicineOne*, motivada pela troca de dados sensíveis.

As seguintes subsecções abordam os requisitos funcionais, que são ilustrados através de um diagrama de casos de uso, e também os requisitos não funcionais.

4.1.1 Requisitos funcionais

Enquanto que os requisitos, que possibilitam ao profissional de saúde gerir as suas consultas, foram identificados através da análise da documentação fornecida, os requisitos associados à navegação na agenda foram extraídos da análise detalhada do funcionamento da agenda do *iPhone*, pois, tal como foi já foi exposto, a navegação deve ser baseada nesta, para fins de usabilidade.

A identificação dos vários requisitos que compõem o módulo de agendamento permitiu a elaboração de um diagrama de casos de uso, que ilustra os diferentes cenários de utilização que os utilizadores do mundo real podem realizar, ao interagir com este. A descrição detalhada desses cenários permite uma melhor compreensão sobre o modo de funcionamento de cada funcionalidade.

O diagrama que ilustra os diferentes casos de uso do módulo pode ser consultado na seguinte figura:

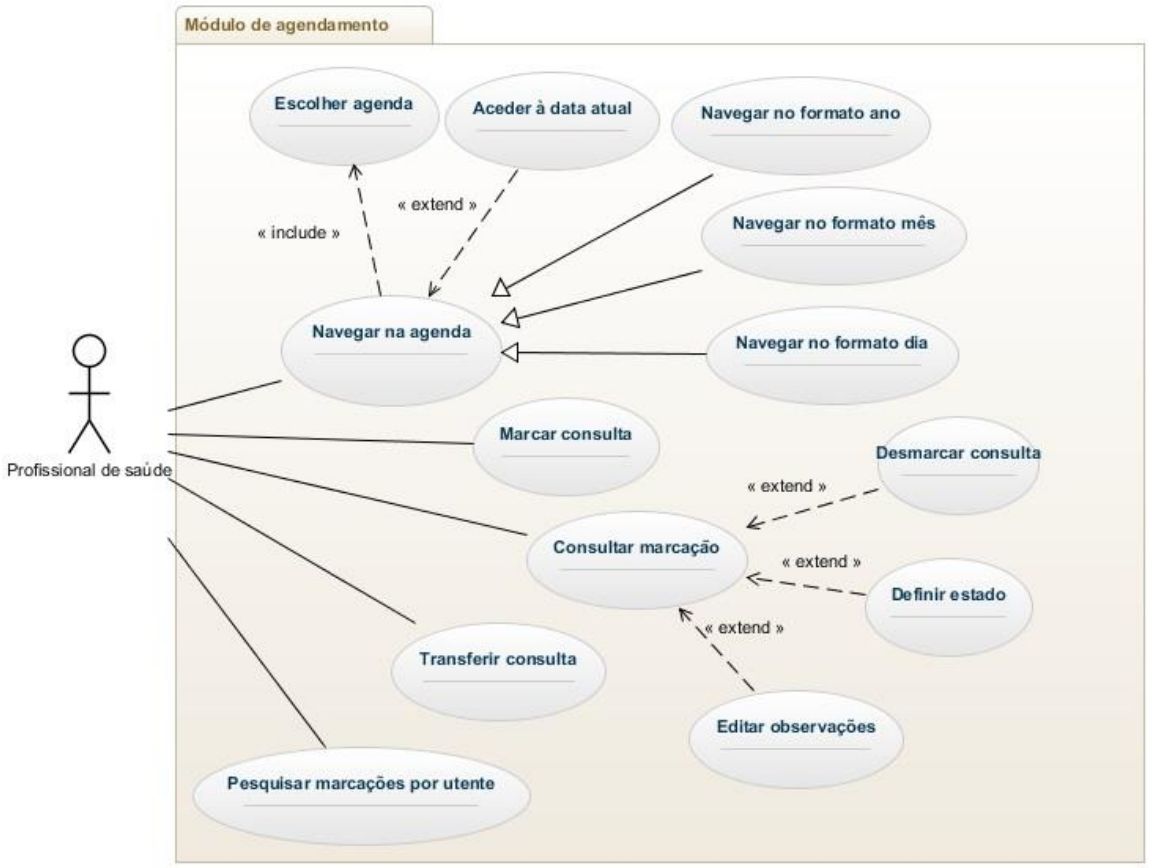

Figura 12 - Diagrama de casos de uso do módulo de agendamento

Analisando a Figura 12, verifica-se que apenas existe um ator que interage com o módulo, o profissional de saúde, uma vez que a aplicação "*MedicineOne*" foi desenvolvida com o intuito de ser utilizada por este. Posto isto, o profissional de saúde pode interagir com o módulo para fins de navegação na agenda, de modo a consultar as marcações para uma determinada data, e de gestão das suas consultas, onde poderá, por exemplo, marcar ou transferir consultas.

De seguida, é apresentada uma tabela, com uma breve descrição dos casos de uso (CU) apresentados, podendo ser consultado o anexo A, para uma análise mais detalhada:

Casos de uso	Descrição
CU01 – Navegar na agenda	O módulo deve permitir que o utilizador navegue na agenda de modo hierárquico pelos formatos ano, mês e dia. A navegação deve ser o mais semelhante possível à apresentada na agenda nativa do *iPhone* de modo a que a sua utilização seja mais intuitiva para o utilizador. A seleção de uma determinada data numa destas vistas deve remeter para a vista seguinte, à exceção da vista diária que deve ser posicionada na data selecionada.
CU02 – Escolher agenda	O módulo deve suportar o acesso às diferentes agendas de um utilizador e permitir que este selecione qual deseja consultar. As agendas podem ser filtradas com recurso à barra de pesquisa.

CU03 – Navegar no formato ano	A navegação no formato ano deve apresentar os diversos meses de um dado ano e possibilitar a navegação por diferentes anos através do deslize vertical do dedo no ecrã.
CU04 – Navegar no formato mês	A navegação no formato mensal deve apresentar os diversos dias de um dado mês, distinguindo os que têm consultas agendadas dos que não têm, e possibilitar a navegação por diferentes meses através do deslize vertical do dedo no ecrã.
CU05 – Navegar no formato dia	A navegação no formato diário deve apresentar a agenda diária, onde estão representadas as consultas agendadas ao longo do dia com a informação relativa ao nome do utente, a sua idade, os atos clínicos agendados e a hora da consulta, e possibilitar a navegação por diferentes dias através da seleção de uma das datas presentes na barra de navegação semanal ou do deslize horizontal do dedo sobre a agenda diária. Caso o utilizador deseje navegar por diferentes semanas deve deslizar o dedo horizontalmente sobre a barra de navegação.
CU06 – Aceder à data atual	O módulo deve permitir que em qualquer uma das vistas de navegação o utilizador possa ser posicionado rapidamente sobre a data atual ao clicar no botão "Hoje".
CU07 – Marca consulta	O módulo deve suportar a adição de novas consultas. Para tal, o utilizador deve recorrer à barra de pesquisa para filtrar o utente a selecionar e deve indicar a data e hora de início e de fim da marcação, o seu autor, o método utilizado, e, opcionalmente, um ato clínico associado.
CU08 – Consultar marcação	O módulo deve permitir que o utilizador consulte as propriedades de uma dada marcação, que dizem respeito à sua data e hora de início e de fim, ao seu estado, que indica o progresso do utente na unidade hospitalar, aos atos clínicos agendados, à localização do utente e às observações adicionais.
CU09 – Desmarcar consulta	O módulo deve permitir que o utilizador desmarque consultas, caso seja necessário. Para tal, este deve aceder aos detalhes de uma marcação e clicar no botão "Desmarcar".
CU10 – Definir estado	O módulo deve permitir que o utilizador altere o estado da marcação conforme o progresso do utente na unidade hospitalar. Para tal, este deve aceder aos detalhes de uma marcação, clicar no botão "Editar" e selecionar o estado desejado.
CU11 – Editar observações	O módulo deve permitir que o utilizador edite as observações associadas a uma marcação. Para tal, este deve aceder aos detalhes de uma marcação, clicar no botão "Editar" e modificar o campo associado às observações.
CU12 – Transferir consultas	O módulo deve permitir que utilizador possa transferir consultas entre diferentes *slots* temporários. Para tal, este deve clicar, sem largar, sobre uma consulta e arrastá-la para o *slot* desejado,

	largando-a de seguida. Ao arrastar a consulta para fora do ecrã, o módulo deverá transitar para o dia seguinte ou anterior.
CU13 – Pesquisar marcações por utentes	O módulo deve permitir que o utilizador pesquise as marcações associadas a um utente. Para tal, este deve recorrer à barra de pesquisa para filtrar o utente, sendo de seguida apresentadas as consultas associadas a este. Por predefinição apenas são apresentadas as marcações futuras, mas caso o utilizador clique no botão "Ver o passado" também devem ser apresentadas as consultas passadas. Ao selecionar uma das consultas apresentadas, o utilizador deve ser remetido para a vista diária correspondente.

Tabela 1 - Descrição dos casos de uso do módulo de agendamento

4.1.2 Requisitos não funcionais

Para além dos requisitos funcionais, que definem como é que o módulo deve agir de acordo com os diferentes cenários de utilização, também foram identificados diversos requisitos não funcionais, que limitam o seu modo de atuação.

Em concreto, foram identificados os que agora se enunciam:

- **Requisito de integração:** O módulo deve ser integrado na aplicação "*MedicineOne*";
- **Requisito de usabilidade:** A componente de navegação deve apresentar uma *interface* semelhante à agenda nativa;
- **Requisito de performance:** O módulo deve assegurar que a interação com a componente de navegação não apresenta quebras de performance;
- **Requisito de interoperabilidade:** O módulo deve comunicar com os servidores da *MedicineOne*;
- **Requisito de segurança:** Os dados trocados durante as comunicações entre o módulo e os serviços devem ser protegidos;

A integração do módulo de agendamento com a aplicação "*MedicineOne*" terá implicações sobre a *interface* a desenvolver, uma vez que esta deve ser enquadrada com a da aplicação. Ainda no que diz respeito à *interface*, por motivos de usabilidade, a componente de navegação deve ser o mais semelhante possível à apresentada na agenda nativa, pois isto permitirá tirar partido da familiaridade de utilização que o utilizador com ela mantém.

A nível de performance, é necessário garantir que, devido à enorme quantidade de dados com que o módulo terá de lidar, a experiência de interação com a funcionalidade de navegação não seja comprometida, pelo facto dos dados serem apresentados dinamicamente.

Revela-se, também, relevante ter em conta a política de acesso aos servidores para a transmissão dos dados, pois um acesso constante aos mesmos condicionará a experiência de interação com a funcionalidade de navegação, para além do impacto que isso poderá causar a nível de consumos de energia, tal como foi referido no guia de performance elaborado na subsecção 2.2.2.

Para que o módulo possibilite ao utilizador a interação com as diversas funcionalidades que este disponibiliza, como, por exemplo, a seleção entre diferentes agendas, ou a visualização de consultas agendadas, é necessário que este estabeleça uma comunicação com os servidores da *MedicineOne*, a fim de trocar dados com os serviços alojados nestes.

Uma vez que os dados trocados entre o módulo de agendamento e os serviços dizem respeito a aspetos associados com a agenda profissional do utilizador e com dados clínicos de natureza sensível, para evitar que sejam acedidos indevidamente, estes devem ser protegidos através de mecanismos de segurança e as comunicações devem ser estabelecidas via HTTPS.

4.2. Arquitetura

O módulo de agendamento, que será integrado na aplicação "*MedicineOne*", vai atuar num cenário onde existem outros elementos, como é o caso dos servidores, e respetivos serviços, da *MedicineOne*, com os quais este irá interagir. Esta interação traduz-se no estabelecimento de uma conexão segura, via HTTPS, que permite a troca de mensagens através da *Internet*.

A seguinte figura ilustra a arquitetura do cenário envolvente ao módulo de agendamento, onde se podem verificar todos os elementos que vão interagir, direta ou indiretamente, com este:

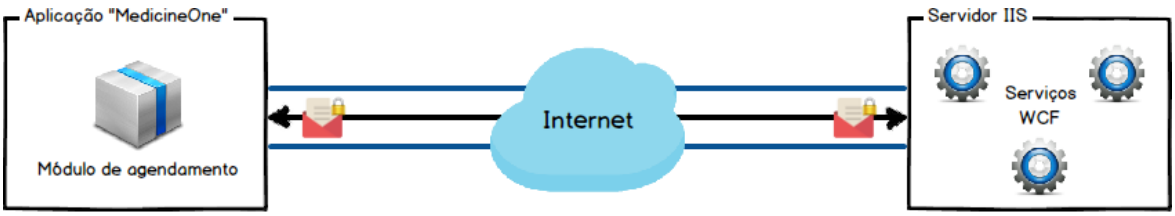

Figura 13 - Arquitetura de alto nível

Analisando a Figura 13, é possível verificar que o módulo de agendamento irá comunicar, diretamente, com um servidor *Internet Information Services* (IIS) da *MedicineOne*, através de um canal de comunicação encriptado, de modo a trocar mensagens, também estas encriptadas, com os serviços disponibilizados por este.

Para além do módulo comunicar com os serviços a fim de receber os dados a manipular, este também necessitará de enviar as alterações efetuadas pelo profissional de saúde, pois existem outras aplicações da *MedicineOne* que partilham o acesso a estes dados.

Os serviços disponibilizados assentam sobre a arquitetura *Representational State Transfer* (REST), que se distingue dos restantes mecanismos, como o *Simple Object Access Protocol* (SOAP), pela sua simplicidade e por ser baseada num modelo orientado a recursos.

Os serviços *RESTful* caraterizam-se por serem *stateless*, ou seja, cada solicitação enviada pelo cliente contém todas as informações necessárias para invocar o serviço, não sendo necessário armazenar qualquer informação no servidor, por utilizarem métodos do protocolo HTTP, como o *GET*, *POST*, *PUT* e *DELETE*, que mapeiam as funções *Create*, *Read*, *Update* e *Delete* (CRUD) para que os recursos possam ser manipulados, por recorrerem a URI, que identificam os recursos de modo intuitivo, e por utilizarem os formatos XML ou *JavaScript Object Notation* (JSON) para representar os dados transferidos [60].

Como foi descrito na secção 2.3, os serviços expõem *endpoints* para que os clientes possam comunicar com estes. Uma vez que os serviços da *MedicineOne* podem ser acedidos por aplicações de diferentes plataformas, cujas necessidades são distintas, é comum que estes exponham *endpoints* destinados a cada uma.

Posto isto, o módulo de agendamento irá comunicar com um dado serviço através do seu *endpoint* direcionado para a plataforma *iOS*, recorrendo ao formato JSON para representar os dados trocados nas mensagens.

A Figura 14 ilustra a comunicação do módulo de agendamento com um serviço WCF genérico:

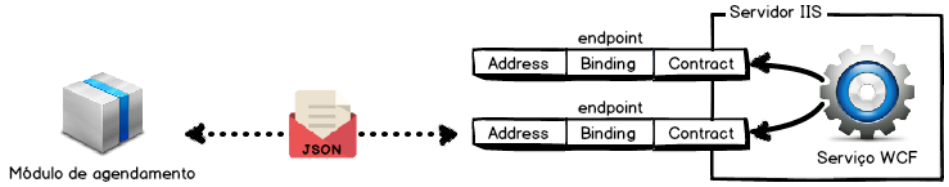

Figura 14 - Comunicação entre o módulo de agendamento e um serviço WCF

De modo a dar suporte às funcionalidades do módulo de agendamento, será desenvolvido um serviço WCF que disponibilize um conjunto de operações cujos objetivos passam por retornar dados, efetuar validações, invocar outros serviços, entre outros.

De forma resumida, as suas operações devem efetuar as seguintes tarefas:

- Retornar a lista de agendas de um utilizador;
- Retornar a agenda associada a um utilizador;
- Retornar as datas que contêm consultas agendadas;
- Retornar a lista de consultas agendadas entre duas datas;
- Retornar os detalhes de uma consulta;
- Retornar a lista de marcações de um dado utente;
- Verificar se existem marcações num determinado período de tempo;
- Validar se é possível agendar uma consulta num dado período de tempo;
- Retornar os atos clínicos associados a um período de tempo;

No que respeita aos mecanismos de segurança, para além da encriptação do canal de comunicação estabelecido entre o módulo de agendamento e o servidor da *MedicineOne*, através do uso de HTTPS, as mensagens trocadas com os serviços também devem encontrar-se protegidas, com recurso à biblioteca *RNCryptor*. O processo referente à encriptação e autenticação das mensagens, por parte do módulo de agendamento, e a consequente validação e desencriptação, por parte dos serviços, será o seguinte:

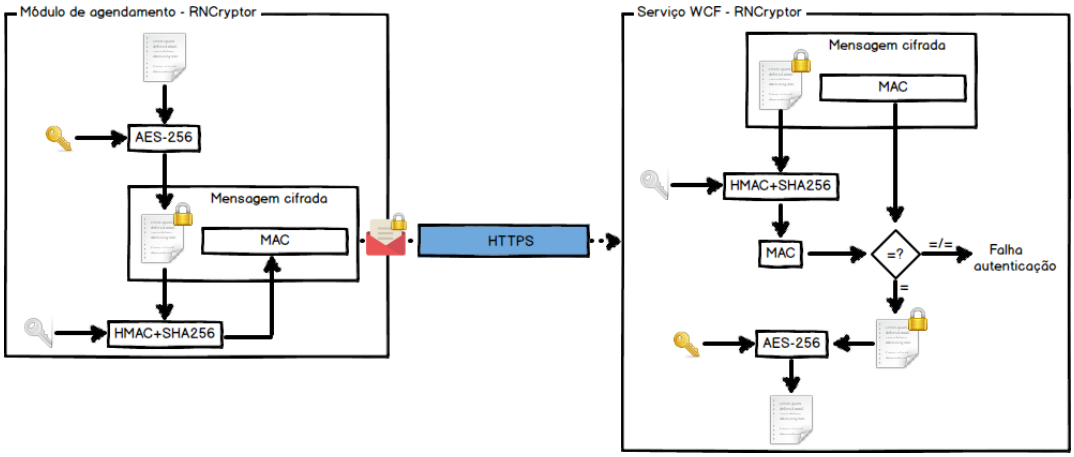

Figura 15 - Funcionamento da biblioteca *RNCryptor* no módulo de agendamento e nos serviços

O processo ilustrado na Figura 15 também se aplica na ordem inversa, ou seja, quando os serviços retornarem uma mensagem ao módulo de agendamento, estes devem efetuar a sua encriptação e autenticação, enquanto o módulo será responsável pelo processo de validação e desencriptação.

É importante mencionar que as chaves utilizadas para encriptar e autenticar uma mensagem são geradas aleatoriamente pela biblioteca *RNCryptor*, com base numa palavra-chave. Para tal, a biblioteca recorre à função *Password-Based Key Derivation Function 2* (PBKDF2), que permite derivar uma chave através da combinação de um valor *salt*, que corresponde a um conjunto de dados aleatórios, de uma função pseudoaleatória como, por exemplo, uma função criptográfica, e de uma palavra-chave, sendo que este processo é repetido por um determinado número de iterações, o que possibilita aumentar a segurança das chaves geradas [61] [62].

Para que o recetor da mensagem consiga derivar as mesmas chaves, através da função PBKDF2, em conjunto com os dados cifrados e o MAC, é enviado um cabeçalho que contém informação relativa ao *salt* utilizado pelo emissor para gerar a chave de encriptação e a chave de autenticação, entre outros campos [62].

Posto isto, quando o recetor recebe uma mensagem cifrada, procede à separação dos elementos contidos nesta, isto é, os campos do cabeçalho, os dados cifrados e o MAC, gera as chaves de encriptação e autenticação com recurso à função PBKDF2 e realiza as restantes operações ilustradas na Figura 15.

Por uma medida de segurança complementar, as mensagens enviadas do módulo de agendamento para os serviços deverão conter um campo adicional, no seu cabeçalho, que irá corresponder a um MAC gerado a partir da combinação de alguns dos campos que o compõem, garantindo, assim, a integridade dos mesmos.

No que diz respeito ao módulo de agendamento, a sua arquitetura será baseada no modelo *Model-View-Controller* (MVC), que possibilita a separação entre o modelo de dados e a representação visual, sendo que estes comunicam através de controladores, tal como ilustra a Figura 16:

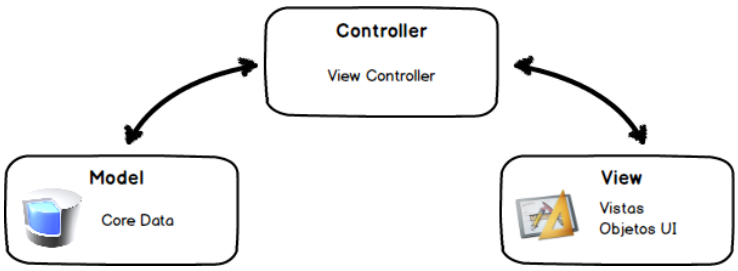

Figura 16 - Arquitetura *Model-View-Controller* adaptada ao módulo de agendamento

O *model* reporta-se à estrutura de dados do módulo, que será desenvolvida com recurso à *Framework Core Data* e que possibilitará o armazenamento, e posterior acesso, dos dados retornados pelos serviços. É importante mencionar que a estrutura de dados do módulo será constituída por dois tipos de tabelas: as dinâmicas e as locais.

As tabelas dinâmicas vão lidar com conjuntos de dados que estão constantemente a sofrer alterações, pelo que estas serão frequentemente carregadas com dados provenientes dos serviços, independentemente de estes terem, ou não, sofrido alterações.

Por outro lado, as tabelas locais vão armazenar dados que raramente sofrem alterações, pelo que, através de um sistema de controlo de versões existente em algumas das tabelas da base de dados, determinados serviços asseguram que estas apenas são carregadas quando os seus dados sofrem alterações, o que possibilitará uma redução dos dados transferidos, pois deste modo as tabelas locais serão carregadas com menos frequência que as dinâmicas.

A *view* corresponde aos elementos visuais que irão compor o conteúdo gráfico do módulo, como, por exemplo, *table views*, ou objetos, como botões. Uma vez que o módulo de agendamento vai apresentar uma *interface* gráfica particular, irá recorrer-se ao mecanismo de herança, de forma a criar subclasses dos componentes visuais fornecidos pela *framework UIKit*, permitindo, assim, que os mesmos sejam adaptados às orientações gráficas em vigor e promovendo a sua reutilização ao longo da *interface* do módulo.

O *controller*, que consente a comunicação entre o *model* e a *view*, será baseado na classe *view controller*, fornecida pelo *iOS* e que gere a apresentação do conteúdo no ecrã. Ao agir como mediador, entre o modelo de dados e a representação visual, o controlador notificará os elementos visuais, quando ocorrerem alterações no modelo de dados, e reportará a este as alterações, originadas pela interação do utilizador com os elementos visuais [63] [64].

Revela-se importante destacar que, no módulo de agendamento, deverão existir diversos *view controllers*, sendo que estes vão gerir a apresentação do conteúdo referente à navegação nos diferentes formatos, às agendas associadas ao utilizador, aos detalhes de uma consulta, à pesquisa das marcações de um utente e aos respetivos resultados, à pesquisa de um utente e aos detalhes de uma marcação de uma consulta.

Capítulo 5
Implementação

O presente capítulo apresenta as decisões tomadas e os detalhes referentes à fase de implementação do módulo de agendamento, dos serviços que lhe dão suporte e dos mecanismos de segurança que asseguram a proteção das mensagens trocadas entre estes.

5.1. Módulo de agendamento

Antes de se dar início à fase de implementação do módulo de agendamento, procedeu-se à elaboração dos *mockups* do mesmo, que podem ser consultados no anexo B, sendo importante mencionar que estes refletem a simplicidade requerida às funcionalidades do módulo, que devem ser realizadas através de um pequeno conjunto de passos. Tendo em conta as dimensões do ecrã do *iPhone*, também foi dada importância à apresentação da informação de forma concisa e de fácil identificação.

Desta tarefa resultou a definição dos fluxos de utilização das várias funcionalidades a desenvolver, o que permitiu ao estagiário conceber uma abordagem de desenvolvimento sobre as diversas classes a implementar, bem como os seus atributos e métodos.

Recorrendo ao princípio de herança, em que diversas classes, denominadas de subclasses, partilham os atributos e métodos herdados de outra, denominada de superclasse, foi criada a classe *ViewControllerAgendaBase*, a partir da qual descendem todas as classes referentes aos *view controllers* que gerem a apresentação do conteúdo do módulo de agendamento.

É nesta classe que se encontram, por exemplo, os atributos referentes à data atual ou à data que assegura o sincronismo entre as vistas de navegação, ou o método que notifica estas vistas quando a data atual é alterada, permitindo, assim, que estas procedam às devidas alterações.

No que respeita aos controladores criados a partir do *ViewControllerAgendaBase*, estes podem ser agrupados de acordo com as funcionalidades a que dão suporte.

Por um lado, os controladores *ViewControllerYear*, *ViewControllerMonth* e *ViewControllerWeek*, que dão suporte às funcionalidades de navegação na agenda, são responsáveis pela gestão da apresentação do conteúdo referente às vistas de navegação no formato ano, mês e dia.

Os restantes controladores lidam com o conteúdo visual referente às funcionalidades de gestão das consultas, como, por exemplo, a pesquisa de marcações por utente, ou a visualização dos detalhes de uma consulta, sendo considerados os seguintes:

- *ViewControllerAgendas*;
- *ViewControllerDetalhesConsulta*;
- *ViewControllerPesquisarMarcacoes*;
- *ViewControllerPesquisarMarcacoesResultados*;
- *ViewControllerSelecionarUtente*;
- *ViewControllerDetalhesMarcacao*.

Posto isto, as seguintes subsecções abordam a estrutura de navegação adotada, a implementação do modelo de dados, os mecanismos que gerem o acesso aos serviços, os

desafios que foram surgindo ao longo do desenvolvimento do módulo e os testes realizados com recurso à ferramenta *Instruments* de forma a garantir a otimização do código.

5.1.1 Estrutura de navegação

A navegação, ao longo do módulo de agendamento, através dos diferentes *view controllers*, foi implementada seguindo o seguinte esquema:

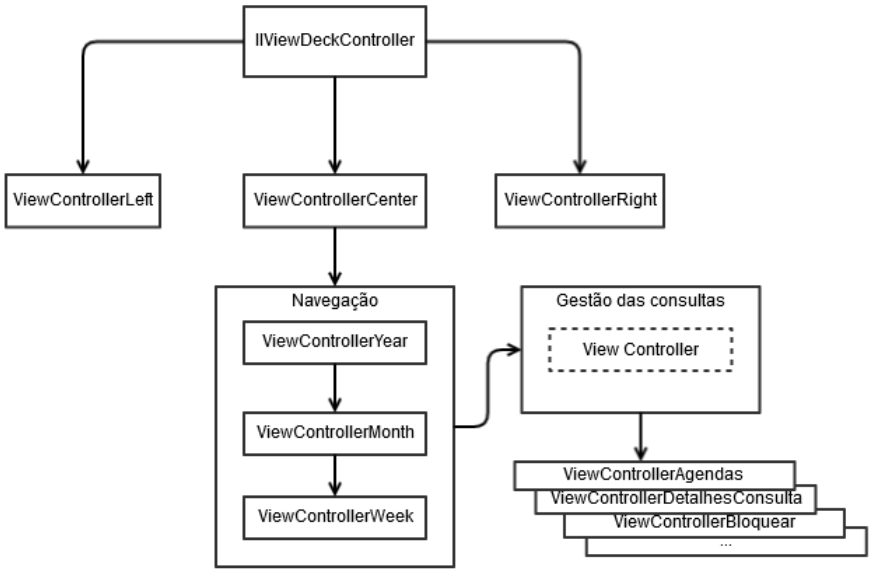

Figura 17 - Estrutura de navegação do módulo de agendamento

Na base do sistema de navegação, adotado pela aplicação "*MedicineOne*", encontra-se a classe *IIViewDeckController*, que gere três *view controllers*, tal como se pode verificar na Figura 17.

O controlador *ViewControllerLeft* é responsável pela apresentação da vista referente ao menu existente na aplicação "*MedicineOne*", onde são listados os diversos módulos pelos quais o utilizador pode navegar. Este menu torna-se visível quando o utilizador desliza o dedo, horizontalmente, para a direita, sobre o ecrã do *iPhone*.

O *ViewControllerRight* é usado quando se pretende apresentar uma vista que ocupa o lado direito do ecrã, e que contenha uma lista de itens, dos quais o utilizador deverá selecionar o desejado. Os *view controllers*, referentes às funcionalidades de gestão das consultas, recorrem a este controlador com alguma frequência, permitindo, por exemplo, que o utilizador selecione o estado referente ao progresso do utente na unidade hospitalar, através da lista de estados que lhe é apresentada.

O *ViewControllerCenter* tem a particularidade de agir como um *Navigation Controller*, que é um controlador específico, o qual permite gerir a navegação do conteúdo de modo hierárquico. Este tipo de controlador possui uma estrutura em forma de pilha, sendo que os diversos *view controllers*, pelos quais o utilizador navega, são adicionados e removidos a essa pilha.

Assim sendo, quando o utilizador acede ao módulo de agendamento, é-lhe apresentado o conteúdo gerido pelo *ViewControllerWeek*, sendo adicionados à pilha todos os controladores associados à funcionalidade navegação, pela ordem ilustrada na Figura 17. Tal ocorre para que seja possível animar as transições entre as diferentes vistas de navegação, à semelhança do que acontece na agenda nativa do *iOS7*.

A navegação para um dos controladores responsáveis pelas funcionalidades de gestão das consultas resulta na sua adição à pilha do *ViewControllerCenter*, que contém, pelo menos, um dos controladores referentes à navegação.

5.1.2 Modelo de dados

O modelo de dados do módulo de agendamento foi implementado com recurso à criação de três tabelas, que podem ser consultadas, em maior detalhe, no anexo C, sendo que estas armazenam a informação associada à agenda, as datas com consultas agendadas e os detalhes das mesmas, respetivamente, as quais são carregadas com dados cada vez que o módulo de agendamento interage com os serviços, e da utilização da tabela "Utilizador", desenvolvida no âmbito de outros módulos da aplicação "*MedicineOne*".

A tabela que armazena a informação respeitante à agenda encontra-se relacionada com as restantes, através de uma ligação de um-para-muitos, exceto com a tabela "Utilizador", em que a relação entre ambas ocorre na ordem inversa.

Na vista de navegação mensal não existe a necessidade de apresentar qualquer detalhe sobre as consultas, uma vez que esta vista recorre apenas a um círculo cinzento para identificar a existência de consultas numa dada data, tal como se pode verificar na Figura 18. Assim, de modo a seguir o guia de performance elaborado na 2.2.2, que refere que deve ser evitada a transmissão de dados que contenham informação desnecessária, a tabela que armazena as datas com consultas agendadas permite armazenar a informação estritamente necessária para esta vista.

Figura 18 - Identificação das consultas na vista de navegação mensal

O modelo de dados também é composto por tabelas locais, que se encontram sincronizadas com tabelas da base de dados da *MedicineOne*, por meio de serviços específicos, que visam armazer os estados usados para descrever o progresso do utente na unidade hospitalar e os motivos que podem ser associados a uma desistência, uma falta, uma transferência e uma desmarcação de uma consulta.

5.1.3 Acesso aos serviços

De modo a gerir num só local todos os acessos aos serviços que auxiliam o módulo de agendamento, foi implementada a classe *AgendaManagement*. Esta classe contém os métodos responsáveis pelo tratamento dos dados a enviar para o serviços, sendo que estes são armazenados num *NSDictionary*, em que cada dado é associado a um identificador, que de seguida é convertido para o formato JSON e reencaminhado para o método responsável pela invocação dos serviços. É também, nesta classe, que se encontra o método que gere o carregamento dos dados provenientes dos serviços nas respetivas tabelas do modelo de dados.

Ao receber os dados transmitidos pela rede, a biblioteca *AFNetworking* armazena-os num *NSDictionary*, permitindo, assim, que estes possam ser manipulados com maior facilidade. De modo a efetuar o carregamento dos dados armazenados no *NSDictionary* no modelo de dados, torna-se necessário efetuar o seu mapeamento para os campos das respetivas tabelas, pelo que, de modo a agilizar este processo, o estagiário recorreu à implementação de *Server Objects*.

Um *Server Object* corresponde a uma classe que é constituída por variáveis com os nomes idênticos aos dos atributos de uma dada tabela e por um método que efetua o mapeamento entre os dados de um *NSDictionary*, através do seu identificador, e as suas variáveis.

Então, quando o método que gere o carregamento dos dados recebe o *NSDictionary* fornecido pela biblioteca *AFNetworking*, este recorre ao *Server Object* correspondente, que efetua o mapeamento entre os dados e os atributos da tabela de forma automática. Após o processo de mapeamento, o *Server Object* é carregado diretamente na respetiva tabela, permitindo, deste modo, a redução da quantidade de código necessária para processar as diversas respostas dos serviços.

5.1.4 Desafios

Um dos desafios que surgiu durante a implementação dos controladores referentes às funcionalidades de navegação, mais especificamente do *ViewControllerYear* e do *ViewControllerMonth*, respeita à apresentação das datas, à medida que se navega pela *collection view*, que é o objeto usado por estas para representar as diversas datas que compõem as suas vistas.

Em ambas as vistas, geridas por estes controladores, os dias são representados por células dispostas numa grelha, sendo que a sua posição se encontra associada ao dia da semana a que estes correspondem. Ao navegar, por exemplo, por diferentes anos, é comum que um dado dia de um determinado mês corresponda a diferentes dias da semana, o que significa que a disposição das células da *collection view* terá de ser alterada, tal como ilustra a seguinte figura:

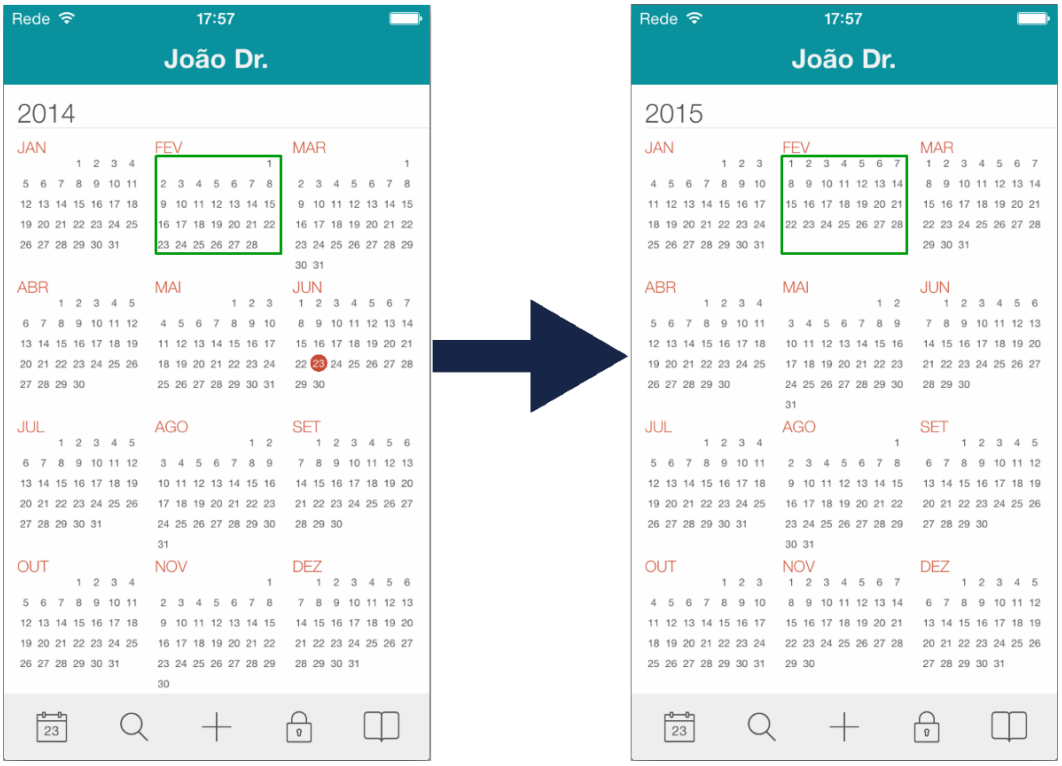

Figura 19 - Desafio da implementação das classes *ViewControllerYear* e *ViewControllerMonth*

Como se pode verificar na Figura 19, os dias que compõem o mês de Fevereiro, assinalados com um quadrado verde, apresentam uma disposição diferente de um ano para o outro, o que se aplica também aos restantes meses. Uma vez que cada ano tem, pelo menos, 365 dias, que vão sendo apresentados à medida que o utilizador navega pela *collection view*, é necessário

otimizar o processo de posicionamento das células, de modo a que a performance das vistas de navegação anual e mensal não sejam comprometidas.

Posto isto, recorreu-se a um mecanismo fornecido pelas *collection views*, que permite que as suas células sejam reutilizadas, sendo apenas instanciadas aquando da criação da *collection view*. Deste modo, é possível que estas sejam reutilizadas e deslocadas para a posição correta de acordo com o dia da semana correspondente, evitando a criação de diversas células à medida que se navega na *collection view*, o que iria causar a degradação da performance destas vistas.

Outro desafio que surgiu durante a implementação dos controladores referentes à navegação, incluído também o *ViewControllerWeek*, refere-se ao fato de ser necessário definir, *a priori*, o número de células a apresentar na *collection view*.

Embora fosse possível, por exemplo, recorrer a um *array* de tamanho elevado, que permitisse armazenar um conjunto limitado de datas, tal solução levaria a que o utilizador atingisse o limite da *collection view*, uma vez que a navegação sobre esta se traduz em percorrer os índices do *array*, de modo a apresentar os elementos armazenados neste.

Contudo, as *collection views* fornecem um método que permite recarregar os dados usados para a construir, pelo que a combinação deste método com um *array* mutável, que permite a adição e remoção de elementos de forma dinâmica, permitiu ao estagiário desenvolver uma solução adequada ao problema apresentado.

Quando o utilizador se aproxima dos limites da *collection view*, os elementos do *array* mutável são removidos e é gerado um novo conjunto de datas, que são adicionadas a este, sendo de seguida invocado o método referido anteriormente. Após a atualização da *collection view*, esta é imediatamente reposicionada sobre as células referentes à posição central do *array*, o que oferece ao utilizador a sensação de que a *collection view* não apresenta quaisquer limites de navegação.

A seguinte figura ilustra o funcionamento da solução desenvolvida, aplicada sobre a vista de navegação diária:

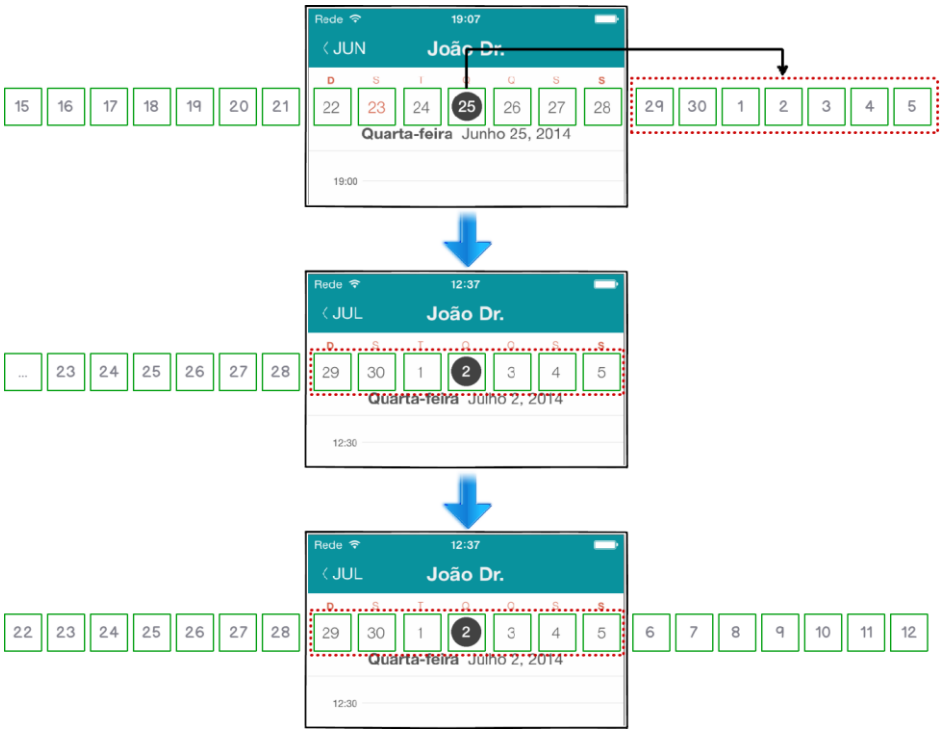

Figura 20 - Solução para o limite do número de células a apresentar numa *collection view*

Analisando a imagem superior da Figura 20, o *array* mutável, utilizado na vista de navegação diária, é constituído por 21 posições, que são carregadas com várias datas de acordo com aquela selecionada pelo utilizador, que neste caso se refere ao dia 25. Deste modo, é possível armazenar um número suficiente de dias para representar uma janela temporal de três semanas, sendo apresentada, na barra de navegação semanal, a semana correspondente à data selecionada pelo utilizador.

Uma vez que esta vista permite ao utilizador deslocar-se por diferentes semanas, através do deslize horizontal do dedo sobre a barra de navegação semanal, caso este navegue para a semana seguinte, como ilustra a imagem superior da Figura 20, a *collection view* acompanha este movimento e apresenta as células referentes à última semana armazenada no *array*, como se pode verificar na imagem ao centro.

Atingido o limite do *array*, os seus elementos são removidos e, com base na nova data selecionada, que neste caso corresponde ao dia 2, são efetuados os cálculos necessários para determinar as datas a carregar no *array*, tendo em conta a janela temporal de três semanas.

Após o preenchimento do *array* com as novas datas, é invocado o método da *collection view* que permite recarregar os elementos armazenados neste e, de modo a surtir o efeito de navegação infinita, a *collection view* é novamente centrada nas células referentes à semana associada à data selecionada, tal como ilustra a imagem inferior da Figura 20.

Um dos desafios que surgiu associado ao acesso aos serviços encontra-se relacionado com a vista de navegação mensal. Uma vez que a navegação sobre esta vista requer que o módulo de agendamento comunique, frequentemente, com os serviços, a fim de receber as datas que têm consultas agendadas, foi necessário definir uma janela temporal, relativamente às datas entre as quais os dados devem ser solicitados, de modo a que o utilizador possa navegar sobre esta sem que ocorram, constantemente, acessos aos serviços.

Decorrente desta situação, o estagiário considerou janelas temporais de três, seis e doze meses, que se aplicam aos meses anteriores e posteriores relativamente ao acedido pelo utilizador, e realizou um conjunto de dez testes, que consistiram no acesso à vista de navegação mensal, para cada uma das janelas consideradas. De modo a realizar estes testes sobre o pior cenário possível, a base de dados da *MedicineOne* foi carregada com diversas consultas, garantindo, assim, que cada janela temporal apresenta uma consulta por dia.

Durante a execução dos testes, o estagiário recolheu dados relativamente ao número de consultas transferidas, à duração média dos pedidos, desde que estes são enviados pelo módulo, até que são recibos, e ao tamanho das respostas retornadas pelos serviços, tendo sido obtidos os seguintes resultados:

Janela temporal	Número de consultas transferidas	Duração média do pedido	Tamanho da resposta
3 meses	214	709 ms	6,25 Kb
6 meses	396	724 ms	11,48 Kb
12 meses	731	823 ms	21,07 Kb

Tabela 2 - Resultados dos testes realizados às diferentes janelas temporais

Analisando a Tabela 2, conclui-se que as diferenças entre as diversas janelas temporais não são significativas ao ponto não ser exequível a adoção de uma destas, pois os tempos médios de duração dos pedidos variam apenas na ordem dos milissegundos e o tamanho máximo que as

respostas podem atingir é da ordem dos 21,07 Kb, que é um valor bastante reduzido. Posto isto, o estagiário optou por adotar a janela temporal de doze meses.

Esta decisão permite tirar vantagens do fato do *iOS* desativar o *Wi-Fi* e as redes celulares quando não são detetadas transferência de dados através da rede, uma vez que, recorrendo à janela temporal de 12 meses, o utilizador pode navegar pela vista de navegação mensal sem ter de aceder aos serviços com tanta frequência, o que apresenta benefícios do ponto de vista dos consumos de energia [25].

5.1.5 Otimização

Ao longo da fase de implementação do módulo de agendamento foram sendo executados diversos testes de performance com recurso à ferramenta *Instruments*, disponibilizada pelo ambiente de desenvolvimento *Xcode*, de modo a assegurar a qualidade do código desenvolvido.

Recorrendo aos diversos módulos internos que a ferramenta apresenta, procedeu-se à recolha de dados sobre aspetos relacionados com a utilização de memória por parte do módulo de agendamento, que foram posteriormente analisados de forma a detetar possíveis falhas que colocassem em causa a sua performance.

Os dados obtidos permitiram analisar o crescimento da memória, através da monitorização dos objetos que vão sendo alocados, à medida que se vai interagindo com o módulo. Através desta análise é possível verificar se o aumento da memória não ocorre de forma contínua quando são executadas repetidamente as mesmas operações, como, por exemplo, aceder à vista de navegação diária e retroceder, pois caso contrário significa que o módulo abandona memória e que irá, eventualmente, atingir os limites da quantidade de memória disponível [65].

Para além da análise do crescimento da memória, os dados recolhidos também permitiram identificar fugas de memória, que correspondem a blocos de memória que, depois de serem alocados e utilizados, não podem ser novamente referenciados e acedidos [65].

5.2. Serviços

Embora já existam serviços, desenvolvidos pela equipa da *MedicineOne*, que implementam as regras de negócio referentes ao agendamento, dando assim suporte à aplicação "*MedicineOne*" para a plataforma *Windows*, o estagiário necessitou de desenvolver um serviço específico que fosse de encontro às necessidades da plataforma sobre a qual o módulo de agendamento se encontra desenvolvido.

Assim sendo, o estagiário implementou um serviço composto por um conjunto de operações, que servem diferentes propósitos, que se baseiam na invocação dos serviços já desenvolvidos, de modo a obter o conjunto de dados retornados por estes.

Uma vez que estes dados correspondem a data sets, que para além de conterem informação excessiva relativamente às necessidades do módulo, não são serializáveis para o formato JSON, o que impede a sua transmissão direta para o módulo, o estagiário recorreu à implementação de classes, estas serializáveis, que são constituídas por diversas variáveis, que representam a informação a transferir para o módulo.

Deste modo, após as operações invocarem os respetivos serviços, é efetuada uma filtragem sobre os data sets retornados e os campos necessários são associados às variáveis do objeto da classe correspondente, que de seguida é enviado para o módulo de agendamento.

Para expor este serviço aos clientes, foi implementado um *endpoint*, identificado a partir do *Address* "*Rest*", cujo *binding* definido foi o *webHttpBinding*, que é direcionado para a comunicação

com serviços REST [66], e o *Contract*, que descreve as operações que este *endpoint* disponibiliza, foi definido com recurso a uma *interface*, que define, por exemplo, quais as operações que podem ser acedidas a partir de determinados URL, entre outros.

5.3. Mecanismos de segurança

Tal como foi referido na subsecção 2.4.3, a confidencialidade e integridade das mensagens trocadas entre os serviços e o módulo de agendamento é assegurada com recurso à biblioteca *RNCryptor*. Posto isto, a implementação dos mecanismos de segurança consistiu na integração desta biblioteca nos serviços e no módulo.

Contudo, uma vez que a comunicação entre ambos é baseada na transferência de dados no formato JSON, que, ao serem recebidos pelo módulo e pelos serviços, são serializados para as respetivas estruturas de dados, foi necessário efetuar as devidas alterações de modo a que as mensagens cifradas fossem desencriptadas antes de se proceder à sua serialização, pois estas não apresentam a estrutura do formato JSON.

No caso do módulo de agendamento, procedeu-se à alteração do método da biblioteca *AFNetworking* que lida com a serialização dos dados recebidos, permitindo, assim, que este efetue também a desencriptação dos mesmos, com recurso aos métodos fornecidos pela classe *RNCryptor*.

No que respeita às alterações levadas a cabo nos serviços, foi necessário recorrer aos pontos de extensibilidade, que a arquitetura WCF fornece, de forma a implementar um *encoder*, onde é feita a desencriptação das mensagens, com recurso à biblioteca *RNCryptor*, antes de estas serem processadas pelo *encoder* do WCF, que efetua a sua serialização.

A Figura 21 ilustra o ponto de extensibilidade implementado na arquitetura do WCF, que é apresentada de forma simplificada:

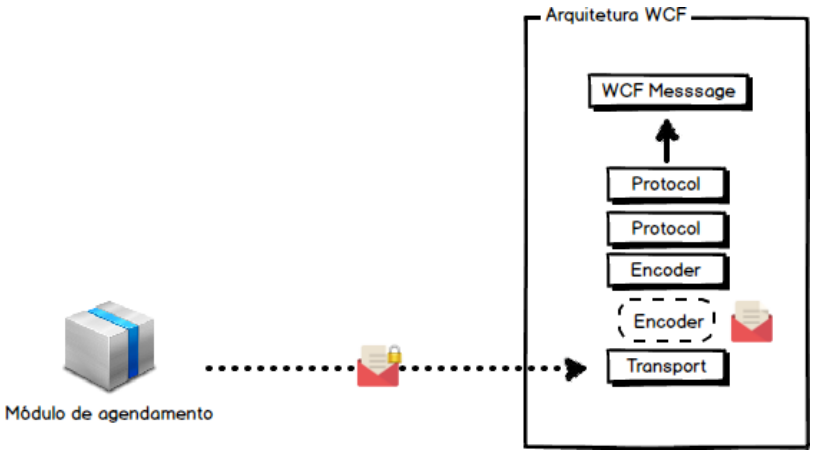

Figura 21 - Ponto de extensibilidade implementado na arquitetura WCF

Analisando a Figura 21, quando a mensagem é recebida pelo canal de comunicação do WCF, esta é encaminhada para o *encoder* implementado pelo estagiário, onde é feita a sua desencriptação. De seguida, a mensagem, já desencriptada, é enviada para o *encoder* de raiz, que efetua a sua serialização para a devida estrutura de dados, produzindo, deste modo, uma mensagem WCF, que prossegue pelas restantes camadas da arquitetura, onde podem ser trabalhados outros aspetos das mensagem [67].

O *encoder* implementado também é utilizado para encriptar as mensagens a enviar, que são serializadas pelo *encoder* do WCF, para o formato JSON, e encaminhadas para este, que através dos métodos fornecidos pela biblioteca *RNCryptor* procede à sua encriptação e autenticação.

Tal com foi referido na secção 4.2, por uma medida de segurança adicional, as mensagens enviadas a partir do módulo de agendamento contêm, no seu cabeçalho, um MAC gerado a partir da combinação de alguns dos campos que o compõem.

Os campos utilizados dizem respeito ao método HTTP usado, ao URL invocado, ao formato do conteúdo da mensagem e a um *token*, que é utilizado pela *MedicineOne* para identificar, unicamente, cada mensagem, evitando assim que os atacantes possam reenviar uma mensagem que já tenha sido processada pelos serviços, sobre os quais é aplicado o algoritmo HMAC+SHA-256.

A escolha destes campos baseia-se no guia que a Amazon disponibiliza aos seus clientes para que estes assinem e autentiquem as mensagens enviadas para os seus serviços REST, sendo que, neste caso, a adição do MAC ao cabeçalho da mensagem permite que os serviços validem a integridade destes campos, evitando, assim, que os atacantes os possam alterar de modo a obterem alguma vantagem que lhes permita conduzir ataques, uma vez que a utilização da biblioteca RNCryptor apenas assegura a confidencialidade e integridade do corpo da mensagem [68].

Capítulo 6
Conclusões e trabalho futuro

O presente estágio consistiu no desenvolvimento de um módulo, a ser integrado na aplicação "*MedicineOne*", que permitisse aos profissionais de saúde gerir a sua agenda de trabalho, através do seu *iPhone*. Para tal, o módulo deveria incorporar um sistema de navegação pelos formatos ano, mês e dia, e um sistema de gestão de consultas, que permitisse ao profissional de saúde preparar-se previamente para uma consulta ou proceder à desmarcação, transferência ou marcação destas de modo rápido e eficaz.

Para além do desenvolvimento do módulo de agendamento, os objetivos do estágio passaram ainda pela implementação de serviços, que possibilitassem a troca de dados com este, e de mecanismos de segurança, que assegurassem a proteção das mensagens trocadas entre ambos.

Ao longo da fase de implementação do módulo surgiram diversos desafios derivados da linguagem de programação adotada e da plataforma para a qual este foi desenvolvido. O fato do estagiário não ter conhecimentos sobre a linguagem *Objective-C* tornou esta fase não só desafiante, mas também gratificante, pois proporcionou a aquisição de novos conhecimentos sobre uma linguagem que permite desenvolver aplicações para uma plataforma que tem vindo a crescer bastante ao longo dos últimos anos. No que respeita a esta plataforma, foi necessário ter em conta as reduzidas dimensões do seu ecrã e as limitações de processamento que esta apresenta, pelo que o estagiário foi desafiado a desenvolver uma *interface* gráfica bastante interativa sem que isso comprometesse a performance do módulo.

No que respeita ao desenvolvimento do serviço que suporta as funcionalidades do módulo de agendamento, embora o estagiário já tivesse tido contacto com a implementação destes componentes recorrendo a outras tecnologias, o estágio proporcionou-lhe a oportunidade de trabalhar com a *Framework* WCF, que por ser proprietária da *Microsoft* é bastante utilizada no meio empresarial. Pela particularidade do serviço implementado servir uma aplicação móvel, o estagiário teve de tomar decisões relativamente à quantidade de dados a transferir, de forma a evitar o risco do módulo de agendamento apresentar consumos de tráfego de dados e de bateria excessivos.

A implementação dos mecanismos de segurança foi precedida por uma extensa análise ao funcionamento do protocolo SSL, onde foram abordados diversos conceitos relacionados com a área de segurança em sistemas de comunicação. Embora a maioria destes conceitos já fossem do conhecimento do estagiário, derivado da sua formação académica, o estágio tornou possível a sua aplicação a nível prático. Por outro lado, o estudo das vulnerabilidades do protocolo SSL, que até ao momento eram desconhecidas por parte do estagiário, levou à transferência do conhecimento adquirido para futuras situações da sua vida profissional, e até mesmo pessoal, tendo este concluído que até os mecanismos de segurança mais utilizados apresentam as suas falhas. No que respeita ao estudo da confidencialidade e integridade das mensagens, este não só permitiu ao estagiário conhecer os mecanismos que asseguram ambos, bem como desmitificar a ideia pré-concebida do senso comum de que a confidencialidade assegura integridade.

Todo o ambiente profissional envolvente ao estágio revelou-se bastante gratificante para o estagiário, não só do ponto de vista profissional, mas também pessoal. Concluído o estágio, pode-se constatar que todos os objetivos traçados foram alcançados com sucesso, o que pode ser confirmado através da análise do documento de testes elaborado pela equipa da *MedicineOne*, onde foram validadas as diversas funcionalidades implementadas, e pelas figuras que ilustram as diversas funcionalidades implementadas, que podem ser consultados, respetivamente, no anexo D e E.

Trabalho futuro

Uma vez que até à data de conclusão do estágio não foi encontrada nenhuma solução, por parte da equipa da *MedicineOne*, referente à inexistência dos serviços que implementam as regras de negócio referentes à desmarcação, edição e marcação de uma consulta, bem como à alteração do estado referente ao progresso do utente na unidade hospitalar, ficou definido para trabalho futuro a implementação desses serviços e as consequentes adaptações nas funcionalidades referentes, de modo a interagirem com estes, sendo que apenas após a conclusão deste trabalho é que o módulo será disponibilizado no mercado em conjunto com a aplicação "*MedicineOne*".

A equipa da *MedicineOne* optou ainda por retirar da presente versão do módulo de agendamento a funcionalidade de transferir consultas, sendo que esta será implementada em futuras versões.

Referências

1. Smart Systems, Smarter Doctors Humans and machines in healthcare. [Online]. Disponível em: http://ricoh.emailsrvc.net/track/dl/1394/Wave_2_Healthcare_final_WEB.pdf. [Acedido em 8 Janeiro 2014].

2. MedicineOne [Online]. Disponível em: http://www.medicineone.net/Empresa/MedicineOne/tabid/83/Default.aspx. [Acedido em 8 Janeiro 2014].

3. APCER - NP 4457 [Online]. Disponível em: http://www.apcer.pt/index.php?option=com_content&view=article&id=141%253Anp-4457&catid=10&Itemid=60&lang=pt. [Acedido em 8 Janeiro 2014].

4. Aplicação "MedicineOne" [Online]. Disponível em: http://mobile.medicineone.net/howitworksiphone/index. [Acedido em 8 Janeiro 2014].

5. Open Source Initiative [Online]. Disponível em: http://opensource.org/. [Acedido em 16 Janeiro 2014].

6. iOS 7 arrives on Apple devices September 18th [Online]. Disponível em: http://www.engadget.com/2013/09/10/ios-7-arrives-september-18th/. [Acedido em 16 Janeiro 2014].

7. App store distribution - Support [Online]. Disponível em: https://developer.apple.com/support/appstore/. [Acedido em 16 Janeiro 2014].

8. Programming with Objective-C [Online]. Disponível em: https://developer.apple.com/library/mac/documentation/cocoa/conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html#//apple_ref/doc/uid/TP40011210-CH1-SW1. [Acedido em 16 Janeiro 2014].

9. HTTPS (HTTP over SSL or HTTP Secure) [Online]. Disponível em: http://searchsoftwarequality.techtarget.com/definition/HTTPS. [Acedido em 27 05 2014].

10. tapkulibrary [Online]. Disponível em: https://github.com/devinross/tapkulibrary. [Acedido em 16 Janeiro 2014].

11. MBCalendarKit [Online]. Disponível em: https://github.com/MosheBerman/MBCalendarKit. [Acedido em 21 Janeiro 2014].

12. ABCalendarPicker [Online]. Disponível em: https://github.com/k06a/ABCalendarPicker. [Acedido em 21 Janeiro 2014].

13. FONSECA, N. AND REIS, C. AND SILVA, C. AND MARCELINO, L. AND CARREIRA, V. 2013. Desenvolvimento em iOS iPhone, iPad e iPod Touch – Curso Completo. 2ª Edição, FCA – Editora informática, Lisboa.

14. Xcode [Online]. Disponível em: https://developer.apple.com/xcode/. [Acedido em 22 Janeiro 2014].

15. iOS Technology Overview - Cocoa Touch Layer [Online]. Disponível em: https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iph

oneostechoverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html#//apple_ref/doc/uid/TP40007898-CH3-SW1. [Acedido em 22 Janeiro 2014].

16. Cocoa Touch Frameworks [Online]. Disponível em: https://developer.apple.com/technologies/ios/cocoa-touch.html. [Acedido em 16 Janeiro 2014].

17. Data Management in iOS [Online]. Disponível em: https://developer.apple.com/technologies/ios/data-management.html. [Acedido em 22 Janeiro 2014].

18. Graphics and animations in iOS [Online]. Disponível em: https://developer.apple.com/technologies/ios/graphics-and-animation.html. [Acedido em 22 Janeiro 2014].

19. iOS Technology Overview - About the iOS Technologies [Online]. Disponível em: https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneostechoverview/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007898-CH1-SW1. [Acedido em 23 Janeiro 2014].

20. iOS Technology Overview - Media Layer [Online]. Disponível em: https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneostechoverview/MediaLayer/MediaLayer.html#//apple_ref/doc/uid/TP40007898-CH9-SW9. [Acedido em 22 Janeiro 2014].

21. Quartz 2D Programming Guide [Online]. Disponível em: https://developer.apple.com/library/ios/documentation/GraphicsImaging/Conceptual/drawingwithquartz2d/Introduction/Introduction.html#//apple_ref/doc/uid/TP30001066. [Acedido em 22 Janeiro 2014].

22. UIKit Framework Reference [Online]. Disponível em: https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIKit_Framework/_index.html#//apple_ref/doc/uid/TP40006955. [Acedido em 22 Janeiro 2014].

23. View Programming Guide for iOS [Online]. Disponível em: https://developer.apple.com/library/ios/documentation/WindowsViews/Conceptual/ViewPG_iPhoneOS/Introduction/Introduction.html#//apple_ref/doc/uid/TP40009503-CH1-SW2. [Acedido em 23 Janeiro 2014].

24. UIKit User Interface Catalog [Online]. Disponível em: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/UIKitUICatalog/index.html#//apple_ref/doc/uid/TP40012857. [Acedido em 23 Janeiro 2014].

25. iOS App Programming Guide - Performance Tuning [Online]. Disponível em: https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/PerformanceTuning/PerformanceTuning.html#//apple_ref/doc/uid/TP40007072-CH8-SW8. [Acedido em 23 Janeiro 2014].

26. Instruments User Guide [Online]. Disponível em: https://developer.apple.com/library/ios/documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/InstrumentsQuickStart/InstrumentsQuickStart.html#//apple_ref/doc/uid/TP40004652-CH19-SW1. [Acedido em 23 Janeiro 2014].

27. iOS Technology Overview - Core Services Layer [Online]. Disponível em: https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iph

oneostechoverview/CoreServicesLayer/CoreServicesLayer.html#//apple_ref/doc/uid/TP40007898-CH10-SW5. [Acedido em 23 Janeiro 2014].

28. Foundation Framework Reference [Online]. Disponível em: https://developer.apple.com/library/mac/DOCUMENTATION/Cocoa/Reference/Foundation/ObjC_classic/_index.html. [Acedido em 23 Janeiro 2014].

29. URL Loading System Programming Guide [Online]. Disponível em: https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/URLLoadingSystem/URLLoadingSystem.html. [Acedido em 23 Janeiro 2014].

30. AFNetworking [Online]. Disponível em: https://github.com/AFNetworking/AFNetworking. [Acedido em 23 Janeiro 2014].

31. What Is Windows Communication Foundation [Online]. Disponível em: http://msdn.microsoft.com/en-us/library/ms731082%28v=vs.110%29.aspx. [Acedido em 16 06 2014].

32. What's the Difference between WCF and Web Services? [Online]. Disponível em: http://www.codeproject.com/Articles/139787/What-s-the-Difference-between-WCF-and-Web-Services. [Acedido em 16 06 2014].

33. LOWY, J. 2010. Programming WCF Services - Mastering WCF and the Azure AppFabric Service Bus. 3ª Edição, O'Reilly Media, Sebastopol, CA.

34. Introduction to Extensibility [Online]. Disponível em: http://msdn.microsoft.com/en-us/library/ms789051%28v=vs.110%29.aspx. [Acedido em 16 06 2014].

35. Chapter 7: Message and Transport Security [Online]. Disponível em: http://msdn.microsoft.com/en-us/library/ff648863.aspx. [Acedido em 28 05 2014].

36. Certificação digital [Online]. Disponível em: http://www.helviojunior.com.br/it/security/certificacao-digital/. [Acedido em 27 05 2014].

37. Transport Layer Security (TLS) e Secure Sockets Layer (SSL) [Online]. Disponível em: http://www.helviojunior.com.br/it/security/transport-layer-security-tls-e-secure-sockets-layer-ssl/. [Acedido em 27 05 2014].

38. Criptografia e Certificação Digital [Online]. Disponível em: http://www.training.com.br/lpmaia/pub_seg_cripto.htm. [Acedido em 06 06 2014].

39. Descrição de handshake por Secure Sockets Layer (SSL) [Online]. Disponível em: http://support.microsoft.com/kb/257591/pt. [Acedido em 30 05 2014].

40. Descrição do processo de autenticação de servidor durante o Handshake SSL [Online]. Disponível em: http://support.microsoft.com/kb/257587/pt. [Acedido em 06 06 2014].

41. Tipos de Autoridades de Certificação [Online]. Disponível em: http://technet.microsoft.com/pt-pt/library/cc732368.aspx. [Acedido em 02 06 2014].

42. Unauthentic "Microsoft Corporation" Certificates [Online]. Disponível em: https://www.cert.org/historical/advisories/CA-2001-04.cfm. [Acedido em 30 05 2014].

43. Agence nationale de la sécurité des systèmes d'information [Online]. Disponível em: http://www.ssi.gouv.fr/. [Acedido em 02 06 2014].

44. Further improving digital certificate security [Online]. Disponível em: http://googleonlinesecurity.blogspot.com.au/2013/12/further-improving-digital-certificate.html. [Acedido em 30 05 2014].

45. OpenSSL Project [Online]. Disponível em: http://www.openssl.org/. [Acedido em 02 06 2014].

46. Man in the Middle Attack [Online]. Disponível em: http://www.veracode.com/security/man-middle-attack. [Acedido em 03 06 2014].

47. About Charles [Online]. Disponível em: http://www.charlesproxy.com/overview/about-charles/. [Acedido em 03 06 2014].

48. SSL Proxying [Online]. Disponível em: http://www.charlesproxy.com/documentation/proxying/ssl-proxying/. [Acedido em 03 06 2014].

49. Security Functions of Cryptography [Online]. Disponível em: http://technet.microsoft.com/en-us/library/cc961634.aspx. [Acedido em 04 06 2014].

50. Top 3 Encryption Myths [Online]. Disponível em: http://redmondmag.com/articles/2005/02/01/top-3-encryption-myths.aspx. [Acedido em 06 06 2014].

51. XML Encryption Syntax and Processing [Online]. Disponível em: http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/. [Acedido em 12 06 2014].

52. XML Encryption Flaw Leaves Web Services Vulnerable [Online]. Disponível em: http://www.darkreading.com/vulnerabilities-and-threats/xml-encryption-flaw-leaves-web-services-vulnerable/d/d-id/1100912?. [Acedido em 12 06 2014].

53. What Is a Message Authentication Code? [Online]. Disponível em: http://www.wisegeek.com/what-is-a-message-authentication-code.htm. [Acedido em 13 06 2014].

54. Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm [Online]. Disponível em: http://cseweb.ucsd.edu/~mihir/papers/oem.pdf. [Acedido em 16 06 2014].

55. Should we MAC-then-encrypt or encrypt-then-MAC? [Online]. Disponível em: http://crypto.stackexchange.com/questions/202/should-we-mac-then-encrypt-or-encrypt-then-mac. [Acedido em 16 06 2014].

56. RNCryptor [Online]. Disponível em: https://github.com/RNCryptor/. [Acedido em 16 06 2014].

57. Cryptographic Right Answers [Online]. Disponível em: http://www.daemonology.net/blog/2009-06-11-cryptographic-right-answers.html. [Acedido em 16 06 2014].

58. What is Kanban? [Online]. Disponível em: http://www.kanbanblog.com/explained/. [Acedido em 24 Janeiro 2014].

59. Demystifying Kanban [Online]. Disponível em: http://www.netobjectives.com/files/resources/articles/Demystifying-Kanban.pdf. [Acedido em 24 Janeiro 2014].

60. RESTful Web services: The basics [Online]. Disponível em: http://www.ibm.com/developerworks/webservices/library/ws-restful/. [Acedido em 20 06 2014].

61. PKCS #5: Password-Based Cryptography Specification [Online]. Disponível em: http://www.ietf.org/rfc/rfc2898.txt. [Acedido em 22 06 2014].

62. Specification for RNCryptor data format version 3 [Online]. Disponível em: https://github.com/RNCryptor/RNCryptor-Spec/blob/master/RNCryptor-Spec-v3.md. [Acedido em 22 06 2014].

63. iOS App Programming Guide - Core App Objects [Online]. Disponível em: https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/AppArchitecture/AppArchitecture.html#//apple_ref/doc/uid/TP40007072-CH3-SW2. [Acedido em 26 Janeiro 2014].

64. Model-View-Controller [Online]. Disponível em: https://developer.apple.com/library/ios/documentation/general/conceptual/devpedia-cocoacore/MVC.html. [Acedido em 21 06 2014].

65. Locating Memory Issues in Your App [Online]. Disponível em: https://developer.apple.com/library/ios/documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/MemoryManagementforYouriOSApp/MemoryManagementforYouriOSApp.html#//apple_ref/doc/uid/TP40004652-CH11-SW1. [Acedido em 26 06 2014].

66. WebHttpBinding Class [Online]. Disponível em: http://msdn.microsoft.com/en-us/library/system.servicemodel.webhttpbinding.aspx. [Acedido em 25 06 2014].

67. Customizing and Extending the BizTalk WCF Adapters [Online]. Disponível em: http://blogs.msdn.com/b/paolos/archive/2009/11/17/customizing-and-extending-the-biztalk-wcf-adapters.aspx. [Acedido em 25 06 2014].

68. Signing and Authenticating REST Requests [Online]. Disponível em: http://docs.aws.amazon.com/AmazonS3/latest/dev/RESTAuthentication.html. [Acedido em 25 06 2014].

69. NSDictionary Class Reference [Online]. Disponível em: https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSDictionary_Class/Reference/Reference.html. [Acedido em 27 Janeiro 2014].

Anexo A – Descrição dos casos de uso

CU01 – Navegar na agenda	
Pré-condição	O ator tem de selecionar a agenda a que deseja aceder.
Fluxo básico	1. O ator acede à componente de navegação; 2. O módulo exibe a vista de navegação diária referente à data atual, que apresenta a barra de navegação semanal, que permite alterar rapidamente o dia selecionado, e a agenda diária, onde estão representadas as consultas agendas para essa data, que apresentam a hora de início, o nome e idade do utente e os atos clínicos marcados; 3. O ator clica no botão de retroceder; 4. O módulo exibe a vista de navegação mensal correspondente, que apresenta os diferentes dias desse mês, distinguindo os que têm consultas agendadas através de um círculo cinzento; 5. O ator clica no botão de retroceder; 6. O módulo exibe a vista de navegação anual, que apresenta os diferentes meses de um determinado ano;
Fluxo Alternativo	No passo 3 caso o ator deseje retornar à vista de navegação diária: 1. O ator seleciona um determinado dia. Retornar ao passo 2; No passo 6 caso o ator deseje retornar à vista de navegação mensal: 1. O ator seleciona um determinado mês. Retornar ao passo 4;
Pós-condição	O módulo deve apresentar a vista de navegação selecionada

Tabela 3 - Descrição do caso de uso "Navegar na agenda"

CU02 – Escolher agenda	
Pré-condição	O módulo tem de comunicar com os servidores da MedicineOne para transferir as agendas do ator. O ator tem de aceder a uma das vistas de navegação.
Fluxo	1. O ator clica no ícone associado às agendas; 2. O módulo exibe uma nova vista onde é apresentada uma barra de pesquisa e as diferentes agendas que o ator possui; 3. O ator seleciona a agenda desejada, podendo recorrer à barra de pesquisa para filtrar as agendas apresentadas; 4. O módulo apresentava a vista de navegação onde o ator se encontrava;
Fluxo Alternativo	Não existe
Pós-condição	O módulo altera a agenda a utilizar para a selecionada.

Tabela 4 - Descrição do caso de uso "Escolher agenda"

CU03 – Navegar no formato ano	
Pré-condição	O ator tem de aceder à vista de navegação anual.
Fluxo básico	1. O ator desliza o dedo verticalmente no ecrã;
Fluxo Alternativo	Não existe
Pós-condição	O módulo deve acompanhar o movimento do dedo e apresentar os meses dos diferentes anos.

Tabela 5 - Descrição do caso de uso "Navegar no formato ano"

CU04 – Navegar no formato mês	
Pré-condição	O ator tem de aceder à vista de navegação mensal.
Fluxo básico	1. O ator desliza o dedo verticalmente no ecrã;
Fluxo Alternativo	Não existe
Pós-condição	O módulo deve acompanhar o movimento do dedo e apresentar os dias dos diferentes meses.

Tabela 6 - Descrição do caso de uso "Navegar no formato mês"

CU05 – Navegar no formato dia	
Pré-condição	O ator tem de aceder à vista de navegação diária.
Fluxo básico	1. O ator desliza o dedo horizontalmente na agenda diária ou seleciona uma das datas presentes na barra de navegação semanal;
Fluxo Alternativo	No passo 1 caso o ator deseje navegar por diferentes semanas: 1. O ator desliza o dedo horizontalmente sobre a barra de navegação semanal;
Pós-condição	O módulo deve apresentar as consultas agendadas para a data selecionada.

Tabela 7 - Descrição do caso de uso "Navegar no formato dia"

CU06 – Aceder à data atual	
Pré-condição	O ator tem de aceder a uma das vistas de navegação.
Fluxo	1. O ator clica no botão "Hoje";
Fluxo Alternativo	Não existe
Pós-condição	O módulo deve posicionar a vista de navegação em que o ator se encontra na data atual.

Tabela 8 - Aceder à data atual

CU07 – Marcar consulta	
Pré-condição	O ator tem de aceder a uma das vistas de navegação.
Fluxo	1. O ator clica no ícone associado à marcação de consultas; 2. O módulo exibe uma nova vista que contém uma barra de pesquisa; 3. O ator digita o nome do utente a pesquisar e seleciona o desejado; 4. O módulo exibe uma nova vista onde são apresentados os campos "Inicio", "Fim", "Iniciativa", "Marcada por", e "Ato". 5. O ator preenche cada um dos campos apresentados; 6. O ator clica no botão "Guardar";
Fluxo Alternativo	No passo 1 caso o ator deseje marcar uma consulta na vista de navegação diária:

	1. O ator seleciona, sem largar, o bloco de tempo onde deseja marcar a consulta. Retornar ao passo 2; No passo 4 caso o ator deseje cancelar a marcação da consulta: 1. O ator clica no botão de retroceder;
Pós-condição	O módulo deve remeter para a vista de navegação diária e focar a consulta marcada. O módulo deve enviar as alterações efetuadas para os servidores da *MedicineOne*.

Tabela 9 - Marcar consulta

CU08 – Consultar marcação	
Pré-condição	O ator tem de aceder à vista de navegação diária.
Fluxo	1. O ator clica na consulta desejada; 2. O módulo exibe uma nova vista onde apresenta a data, hora de início e de fim da consulta, o seu estado e um resumo dos atos clínicos agendados, das observações associadas à consulta e da localização do utente.
Fluxo Alternativo	Não existe
Pós-condição	O módulo deve apresentar os detalhes da consulta selecionada.

Tabela 10 - Descrição do caso de uso "Consultar marcação"

CU09 – Desmarcar	
Pré-condição	O ator tem de aceder aos detalhes de uma consulta.
Fluxo	1. O ator clica no botão "Desmarcar"; 2. O módulo exibe uma mensagem de notificação; 3. O ator confirma a ação; 4. O módulo remete para a vista de navegação diária;
Fluxo Alternativo	No passo 3 caso o ator deseje anular a desmarcação: 1. O ator cancela a ação;
Pós-condição	O módulo deve remover a consulta da agenda. O módulo deve enviar as alterações efetuadas para os servidores da MedicineOne.

Tabela 11 - Descrição do caso de uso "Desmarcar"

CU10 – Definir estado	
Pré-condição	O ator tem de aceder aos detalhes de uma consulta.
Fluxo	1. O ator clica no ícone de Editar; 2. O módulo desbloqueia o campo referente ao estado; 3. O ator clica no campo referente ao estado; 4. O módulo apresenta uma lista com os estados disponíveis; 5. O ator seleciona o estado desejado; 6. O ator clica no botão "Guardar";
Fluxo Alternativo	No passo 6 caso o ator deseje anular as alterações: 1. O ator clica no botão "Cancelar";
Pós-condição	O módulo deve apresentar o novo estado da consulta. O módulo deve enviar as alterações efetuadas para os servidores da MedicineOne.

Tabela 12 - Descrição do caso de uso "Definir estado"

CU11 – Editar observações	
Pré-condição	O ator tem de aceder aos detalhes de uma consulta.
Fluxo	1. O ator clica no ícone de Editar; 2. O módulo torna o texto editável; 3. O ator altera as observações; 4. O ator clica no botão "Guardar";
Fluxo Alternativo	No passo 4 caso o ator deseje anula as alterações: 1. O ator clica no botão "Cancelar";
Pós-condição	O módulo deve apresentar as novas observações da consulta. O módulo deve enviar as alterações efetuadas para os servidores da MedicineOne.

Tabela 13 - Descrição do caso de uso "Editar observações"

CU12 – Transferir consulta	
Pré-condição	O ator tem de aceder à vista de navegação diária.
Fluxo	1. O ator clica, sem largar, a consulta que deseja transferir; 2. O módulo ativa o modo de transferência que permite a deslocação da consulta; 3. O ator desloca a consulta para um dos slots disponíveis. Caso esta seja deslocada para fora do ecrã, a vista de navegação diária deve navegar para a data posterior ou anterior. Esta fase termina quando o ator larga a consulta; 4. O módulo exibe uma mensagem de notificação; 5. O ator confirma a ação;
Fluxo Alternativo	No passo 5 caso o ator deseje anular a transferência: 1. O ator cancela a ação;
Pós-condição	O módulo deve apresentar a consulta no novo slot. O módulo deve enviar as alterações efetuadas para os servidores da MedicineOne.

Tabela 14 - Descrição do caso de uso "Transferir consulta"

CU13 – Pesquisar marcações por utente	
Pré-condição	O ator tem de aceder a uma das vistas de navegação.
Fluxo	1. O ator clica no ícone de pesquisa; 2. O módulo exibe uma nova vista que contém uma barra de pesquisa; 3. O ator digita o nome do utente a pesquisar e seleciona o desejado; 4. O módulo exibe uma nova vista que lista as diversas consultas que o utente tem agendadas. Por predefinição apenas são apresentadas as futuras marcações; 5. O ator seleciona a consulta desejada;
Fluxo Alternativo	Na fase 4 caso o ator deseje visualizar as consultas passadas: 1. O ator clica na opção "Ver o passado". Retorna à fase 5;
Pós-condição	O módulo deve remeter para a vista de navegação diária e focar a consulta selecionada.

Tabela 15 - Descrição do caso de uso "Pesquisar marcações por utente"

Anexo B – Mockups do módulo de agendamento

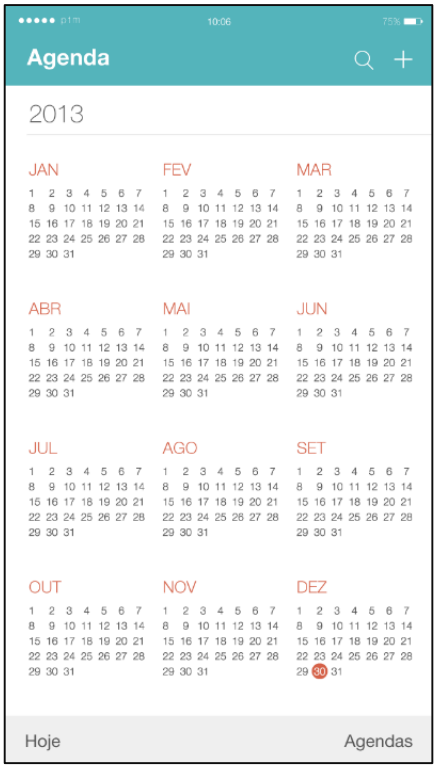

Figura 22 - Mockup da vista de navegação anual

Figura 23 - Mockup da vista de navegação mensal

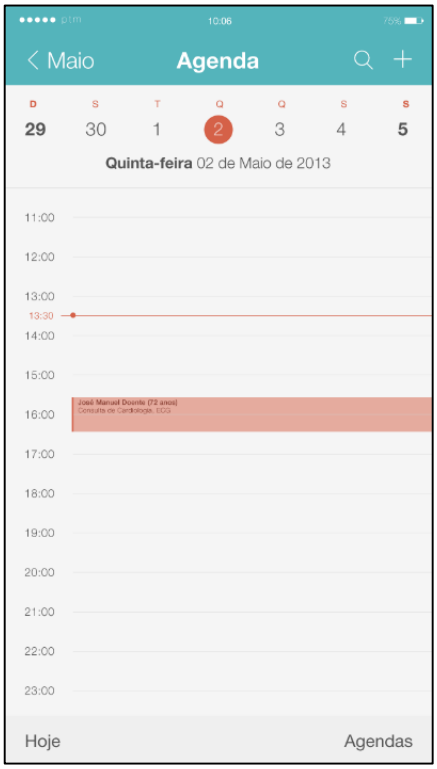

Figura 24 - Mockup da vista de navegação diária

Figura 25 - Mockup da vista das propriedades de uma consulta

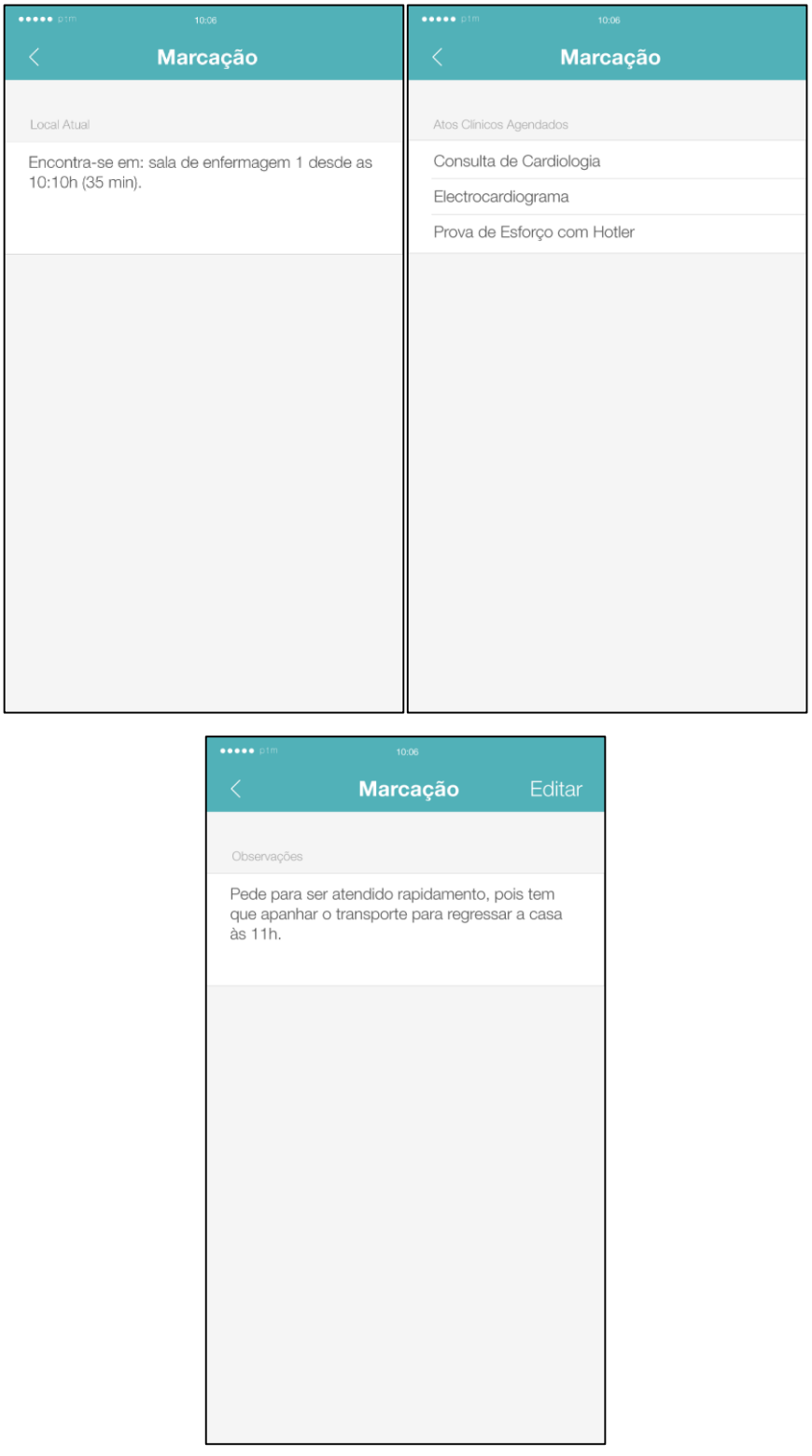

Figura 26 - Mockups da vista de detalhe das propriedades de uma marcação

Figura 27 - Mockup da vista de marcação de uma consulta

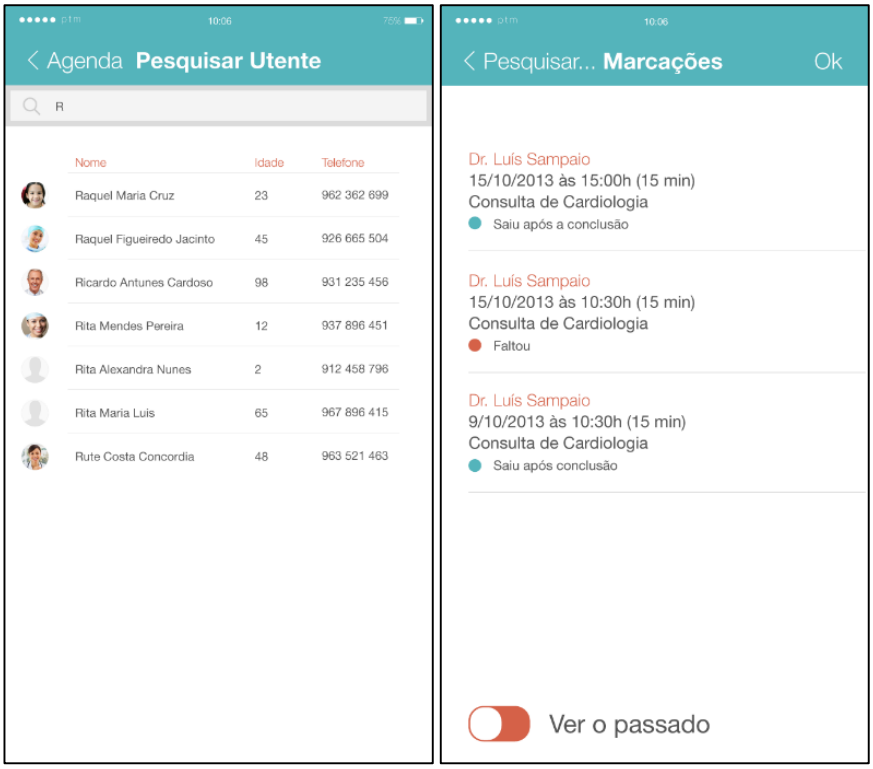

Figura 28 - Mockup da vista de pesquisa de marcações por utente

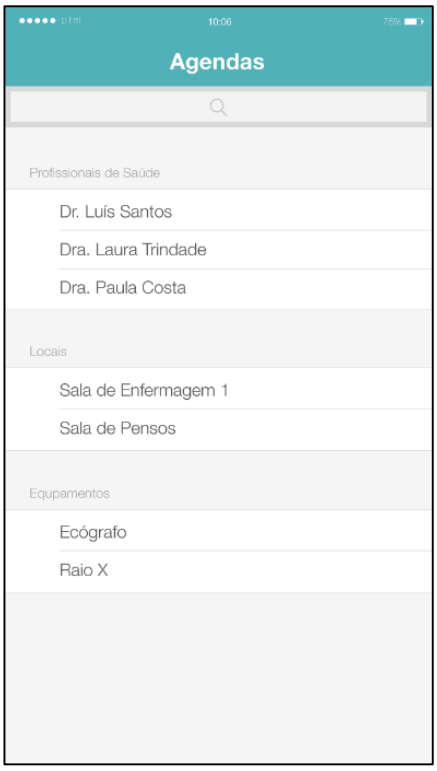

Figura 29 - Mockup da vista de seleção da agenda

Anexo C – Modelo de dados

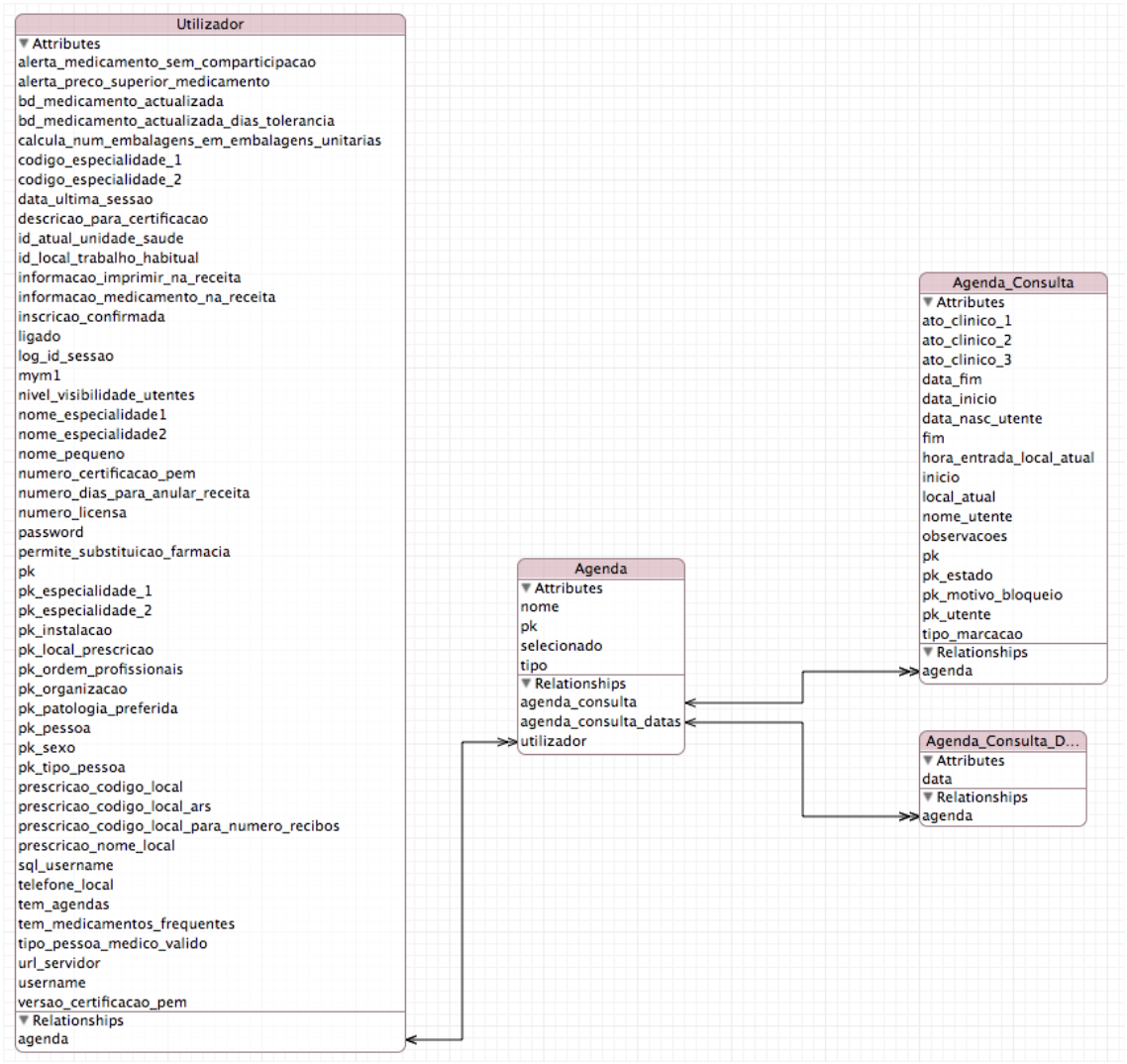

Figura 30 - Tabelas dinâmicas

Agenda		
Atributos	**Tipo**	**Descrição**
nome	NSString	Nome do profissional de saúde proprietário da agenda
pk	NSString	Identificador único da agenda
selecionado	NSNumber	Indicação se a agenda se encontra selecionada
tipo	NSNumber	Identifica o tipo de agenda (Profissional de saúde, Equipamento e Local)
agenda_consulta	NSSet	Conjunto de relações com a tabela Agenda_Consulta
agenda_consulta_datas	NSSet	Conjunto de relações com a tabela Agenda_Consulta_Datas
utilizador	Utilizador	Relação com a tabela Utilizador

Tabela 16 - Descrição da tabela Agenda

Agenda_Consulta_Datas		
Atributos	**Tipo**	**Descrição**
data	NSDate	Data em que uma consulta se encontra agendada
agenda	Agenda	Relação com a tabela Agenda

Tabela 17 - Descrição da tabela Agenda_Consulta_Datas

Agenda_Consulta		
Atributos	**Tipo**	**Descrição**
ato_clinico_1	NSString	Nome do ato clínico 1
ato_clinico_2	NSString	Nome do ato clínico 2
ato_clinico_3	NSString	Nome do ato clínico 3
data_fim	NSDate	Data de fim da consulta
data_inicio	NSDate	Data de início da consulta
data_nasc_utente	NSDate	Data de nascimento do utente
Fim	NSDate	Data e hora de fim da consulta
hora_entrada_local_atual	NSDate	Hora de entrada do utente no local onde se encontra
Inicio	NSDate	Data e hora de início da consulta
local_atual	NSString	Local onde o utente se encontra
nome_utente	NSString	Nome do utente
observações	NSString	Observações associadas à consulta
pk	NSString	Identificador único da consulta
pk_estado	NSNumber	Identificador único do estado de progresso do utente, associado à consulta, na unidade hospitalar
pk_motivo_bloqueio	NSString	Identificador único do motivo de bloqueio de um período de tempo
pk_utente	NSNumber	Identificador único do utente
tipo_marcacao	NSNumber	Identifica o tipo de marcação (Consulta ou bloqueio)
agenda	Agenda	Relação com a tabela Agenda

Tabela 18 - Descrição da tabela Agenda_Consulta

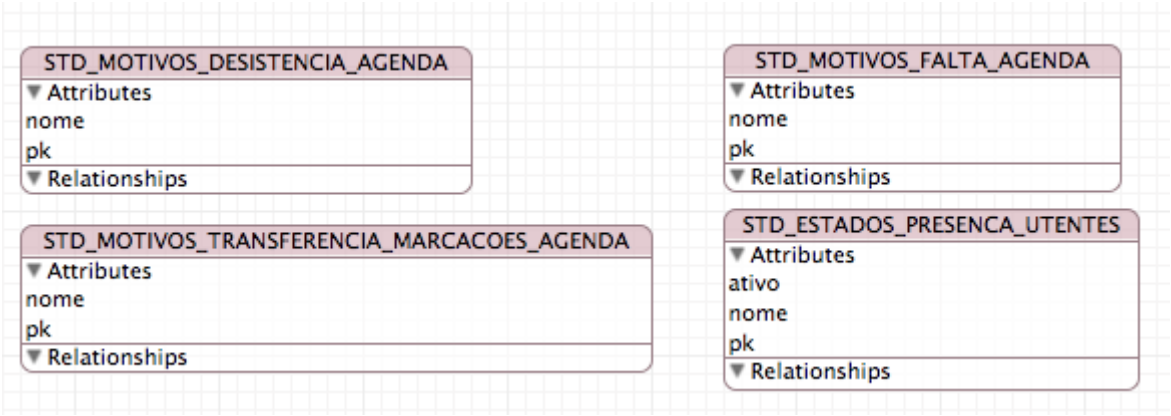

Figura 31 - Tabelas locais

STD_MOTIVOS_DESISTENCIA_AGENDA		
Atributos	**Tipo**	**Descrição**
nome	NSString	Nome do motivo da desistência de uma consulta
pk	NSNumber	Identificador único do motivo de desistência

Tabela 19 - Descrição da tabela STD_MOTIVOS_DESISTENCIA_AGENDA

STD_MOTIVOS_TRANSFERENCIA_MARCACOES_AGENDA		
Atributos	**Tipo**	**Descrição**
nome	NSString	Nome do motivo da transferência de uma consulta
pk	NSNumber	Identificador único do motivo de transferência

Tabela 20 - Descrição da tabela STD_MOTIVOS_TRANSFERENCIA_MARCACOES_AGENDA

STD_MOTIVOS_FALTA_AGENDA		
Atributos	**Tipo**	**Descrição**
nome	NSString	Nome do motivo da falta a uma consulta
pk	NSNumber	Identificador único do motivo de falta

Tabela 21 - Descrição da tabela STD_MOTIVOS_FALTA_AGENDA

STD_ESTADOS_PRESENCA_UTENTES		
Atributos	**Tipo**	**Descrição**
ativo	NSNumber	Indica se o estado deve ser apresentado
nome	NSString	Nome do estado de progresso do utente na unidade hospitalar
pk	NSNumber	Identificador único do estado

Tabela 22 - Descrição da tabela STD_ESTADOS_PRESENCA_UTENTES

Anexo D – Documento de testes

(Em formato digital)

Anexo E – Resultado final do módulo de agendamento

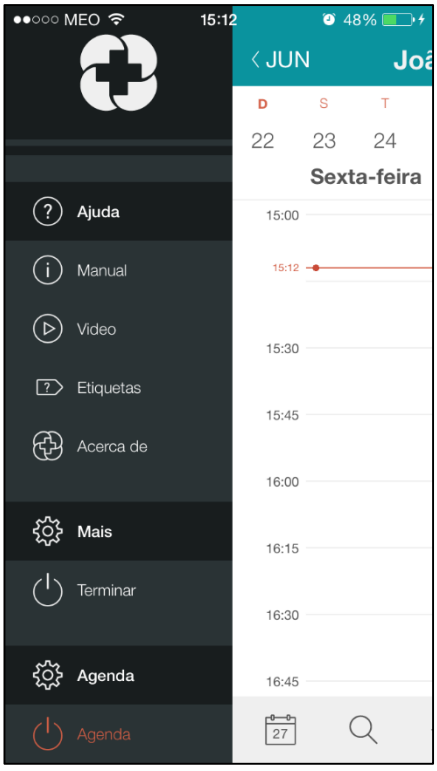

Figura 32 - Integração do módulo de agendamento com a aplicação "MedicineOne"

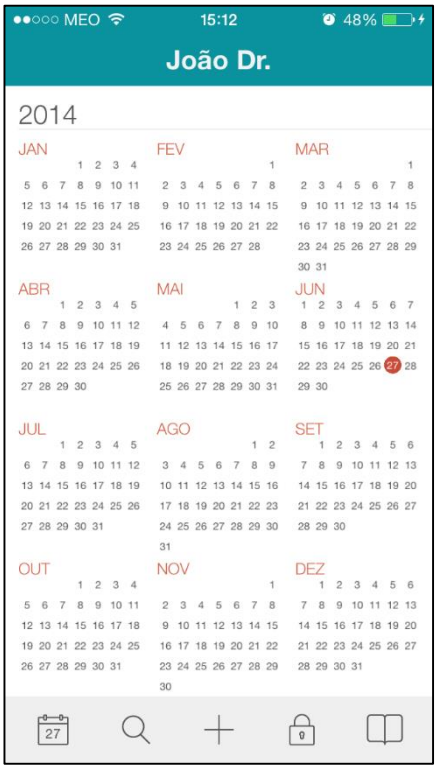

Figura 33 - Funcionalidade de navegação no formato ano

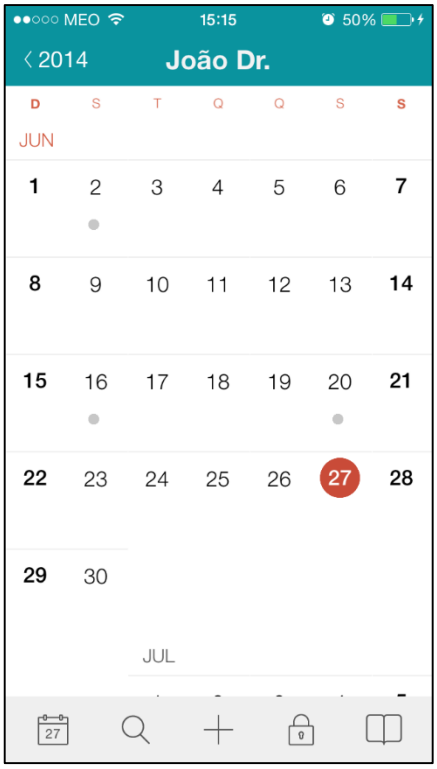

Figura 34 - Funcionalidade de navegação no formato mês

Figura 35 - Funcionalidade de navegação no formato dia

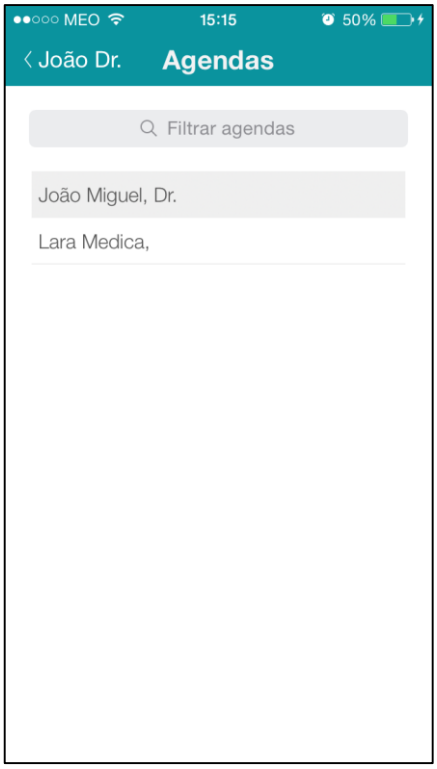

Figura 36 - Funcionalidade de escolha da agenda

Figura 37 - Funcionalidade consultar os detalhes de uma marcação

Figura 38 - Funcionalidade de selecionar o estado referente ao progresso do utente na unidade hospitalar

Figura 39 - Funcionalidade de editar as observações de uma consulta

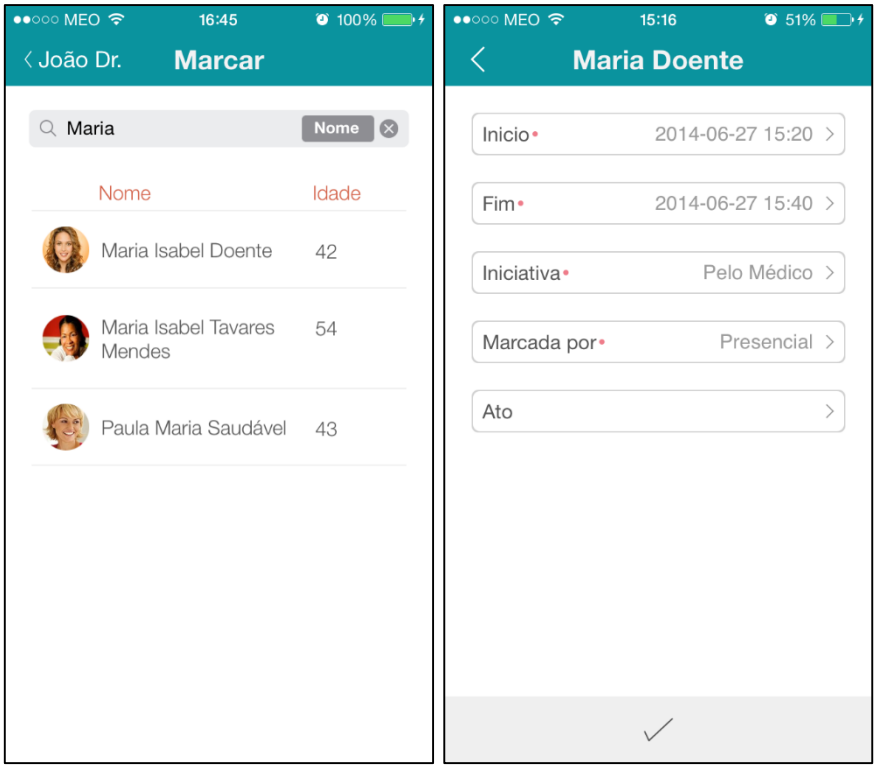

Figura 40 - Funcionalidade de marcar uma consulta

Figura 41 - Funcionalidade de desmarcar uma consulta

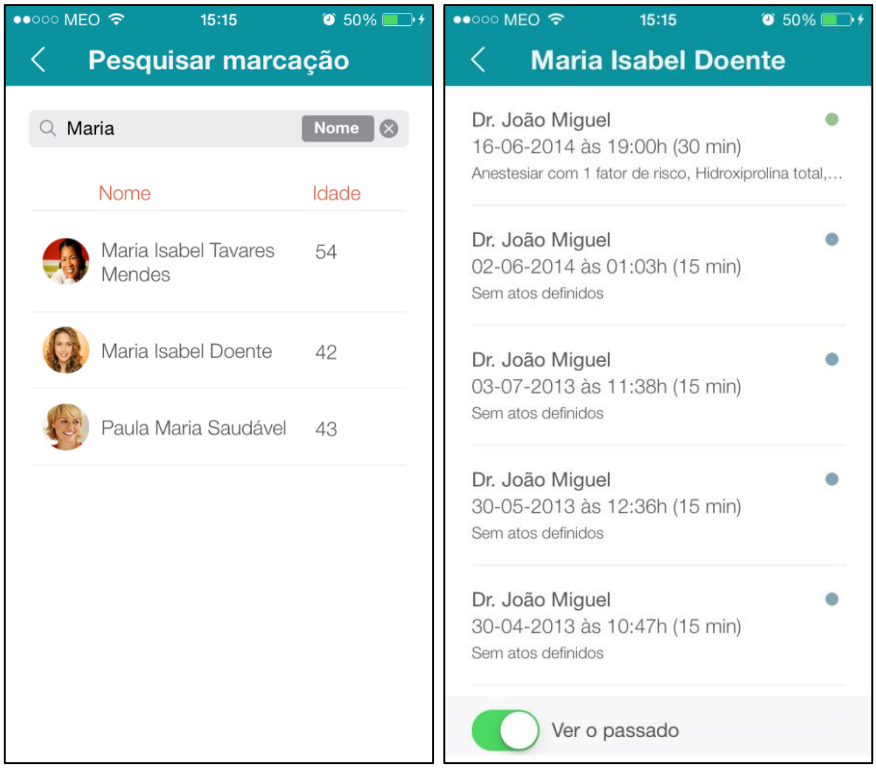

Figura 42 - Funcionalidade de pesquisar marcações por utente