Mestrado em Engenharia Informática
Estágio
Relatório Final

Desenvolvimento de um Sistema de Autorizações Médicas com Aplicação Móvel

Marco Pereira
macsp@student.dei.uc.pt

Orientador do DEI-FCTUC:
Professor Doutor Joel Arrais

Orientador da MedicineOne:
Engenheiro Fernando Tinoco

Coimbra, 1 de Julho de 2016

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

Mestrado em Engenharia Informática
Estágio
Relatório Final

Desenvolvimento de um Sistema de Autorizações Médicas com Aplicação Móvel

Marco Pereira
macsp@student.dei.uc.pt

Júris:

Professor Doutor António Dourado

Professor Alcides Fonseca

Coimbra, 1 de Julho de 2016

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

Resumo

O processo de alterações terapêuticas requer que caso o médico não se encontre em regime de presença permanente, se desloque à unidade de saúde para formalizar a alteração. Este procedimento baseia-se num meio de comunicação informal entre enfermeiro e médico, sem nenhum sistema que permita a monitorização da informação. Não obstante, origina custos acrescidos à unidade de saúde, além do tempo necessário para que a alteração terapêutica possa efetivamente ser aplicada.

O presente relatório documenta uma solução tecnológica, com recurso a *smartphones*, que permite uma resolução rápida do processo, contribuindo para a minimização de problemas de responsabilização.

Este estágio direciona-se para o desenvolvimento da solução denominada *Autheras*, composta por uma aplicação cliente para iPhone, assim como de um sistema central que permita que os médicos, ao trabalharem em diferentes unidades de saúde, e com diferentes sistemas *Electronic Health Record*, possam receber no seu telefone móvel todos os pedidos de autorização que lhes são endereçados.

No desenvolvimento do sistema foi dada importância para que o mesmo possa estar preparado para que, no futuro, possibilite oferecer suporte às plataformas Android e Windows Phone.

Palavras chave: *Autorização, Alteração Terapêutica, iOS, Mobile, MedicineOne, Mensagens Clínicas, Profissional de Saúde, Serviços*

Agradecimentos

À equipa da MedicineOne pela simpatia e profissionalismo com que me trataram durante o período deste estágio. Não posso deixar de destacar um especial agradecimento ao João Miguel pela criação deste projeto e pela confiança que depositou em mim para o realizar. Presto o melhor reconhecimento ao Eng.º Fernando Tinoco pela orientação, amizade, confiança, sinceridade e disponibilidade que sempre demonstrou. Agradeço, igualmente, ao Bruno Coelho, Pedro Faustino, Lara Coelho, Nádia Costa, Marco Tinoco e Nuno Pereira pelo apoio e transmissão de conhecimento.

Ao Professor Doutor Joel Arrais quero agradecer o rigor da sua orientação e o valor inestimável dos conselhos na elaboração e revisão deste relatório.

Aos colegas e igualmente estagiários Álvaro Mateus, Celso Mendes e Jaime Correia pela boa disposição e amizade que certamente sairá reforçada para a vida.

À Anabela, minha namorada, pela confiança, carinho, compreensão, paciência para a minha indisponibilidade e por estar sempre presente nos bons e maus momentos deste percurso. À sua família pelo apoio.

Por fim, os mais importante agradecimentos são endereçados às pessoas que desde sempre acreditaram, incentivaram e proporcionaram a oportunidade de apostar na minha formação académica: à minha Mãe e à minha Avó Zulmira. Obrigado. É a vós que dedico o meu trabalho.

Conteúdo

Lista de Acrónimos

ANSI *American National Standards Institute*
API *Application Programming Interface*
APNS *Apple Push Notification Service*
ASC *Acredited Standards Comittee*
ASTM *American Society for Testing and Materials*
BD Base de Dados
CDA *Clinical Document Architecture*
CNPD Comissão Nacional de Proteção de Dados
CRUD *Create, Read, Update e Delete*
DEI Departamento de Engenharia Informática
DICOM *Digital Imaging and Communications in Medicine*
EHR *Electronic Health Record*
FCTUC Faculdade de Ciências e Tecnologia da Universidade de Coimbra
GCM *Google Cloud Messaging*
FHIR *Fast Healthcare Interoperability Resources*
HL7 *Health Level Seven*
HTTP *Hyper Text Transfer Protocol*
HTTPS *Hyper Text Transfer Protocol Secure*
IDE *Integrated Development Environment*
IIS *Internet Information Services*
IP *Internet Protocol*
ISO *International Standards Organization*
JSON *JavaScript Object Notation*
MDF *Message Development Framwework*
MVC *Model-View-Controller*
NCPDP *National Council for Prescription Drug Programs*
NEMA *National Electrical Manufacturers Association*
ORM *Object-Relacional Mapping*
OSI *Open Systems Interconnection*
PIN *Personal Identification Number*

REST *Representational State Transfer*
RIM *Reference Information Model*
SDK *Software Development Kit*
SHA *Secure Hash Algorithm*
SNS Serviço Nacional de Saúde
SOAP *Simple Object Access Protocol*
SQL *Structured Query Language*
SSL *Secure Sockets Layer*
TCP *Transmission Control Protocol*
UI *User Interface*
UML *Universal Modeling Language*
URL *Uniform Resource Locator*
VCS Vocera Collaboration Suite
WCF *Windows Communication Foundation*
WNS *Windows Push Notification Services*
XML *eXtensible Markup Language*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este relatório visa documentar o trabalho realizado na unidade curricular de Dissertação/Estágio, do Mestrado em Engenharia Informática do Departamento de Engenharia Informática (DEI) da Faculdade de Ciências e Tecnologia da Universidade de Coimbra[1] (FCTUC) no ano letivo 2015/2016.

O Estágio decorreu na empresa MedicineOne[2] sob a supervisão do Engenheiro Fernando Tinoco, como orientador da empresa, e do Professor Doutor Joel Arrais, como orientador do DEI.

1.1 A empresa

A MedicineOne é uma empresa tecnológica, sediada em Coimbra, no Instituto Pedro Nunes, dedicada ao desenvolvimento de *software* para o mercado da saúde. Desenvolve soluções para gestão de unidades de saúde públicas e privadas, gerindo de forma integrada a área clínica, administrativa, logística, de gestão, financeira e estatística. A empresa pretende ser uma referência na produção de *software* para a área da saúde, tendo produtos representados em quatro continentes.

Soluções MedicineOne

O produto de bandeira e que dá nome à empresa é o MedicineOne, atualmente na 8ª versão. Este *software*, destinado para profissionais de saúde, foi desenhado para lidar com a complexidade e eficiência no atendimento ao paciente, oferecendo flexibilidade, confiabilidade e rapidez que permitem agilizar todos os processos dentro de qualquer organização de saúde. A empresa também oferece uma solução *mobile* para os dispositivos Apple (iPhone e iPad), mais simplificada e adaptada à prática individual: My MedicineOne 8. Continuamente a pensar no paciente, em março de 2016 foi lançada a aplicação móvel eu+, que, entre outras funcionalidades inovadoras, permitirá ter acesso ao processo clínico individual.

[1] `http://www.uc.pt/fctuc/dei`
[2] `http://www.medicineone.net`

1.2 Enquadramento

O Enquadramento tem o objetivo de expor as **motivações** inerentes a este projeto ao identificar o Contexto do problema e como o Estágio poderá ser solução para o mesmo.

1.2.1 Contexto

No sistema atual, sempre que existe a necessidade de proceder a uma alteração terapêutica a um paciente internado, e o médico responsável pelo internamento não se encontrar na unidade de saúde, existe uma série de procedimentos que exigem recursos extraordinários para formalizar e documentar os pedidos de alteração e evitar possíveis problemas de responsabilização.

A título de exemplo, para racionalizar os custos das unidades de saúde, os médicos alocados para o período noturno são dispensados da obrigatoriedade de presença permanente. Contudo, o médico não está isento da responsabilidade de voltar à unidade de saúde sempre que um paciente, do qual seja responsável, mostrar necessidade de um novo cuidado ou prescrição. Ainda que um enfermeiro possa acelerar o diagnóstico e informar telefonicamente o médico, este ou tem que efetivar a deslocação para validar o processo, ou delibera pela transferência urgente para outra unidade de saúde.

Este problema, identificado por clientes da MedicineOne, tem associado custos e tempo em deslocações que podem comprometer a saúde do paciente.

1.2.2 Estágio

A solução proposta pressupõe a implementação de um sistema de autorizações para que os pedidos emitidos por *Electronic Health Record*s (EHRs) possam ser recebidos e respondidos pelos médicos autorizadores nos seus *smartphones*. O sistema deve ser capaz de responder a pedidos de autorização de vários EHRs que futuramente estejam interessados nesta nova solução, não sendo, por isso, exclusivos do MedicineOne. O procedimento de envio do pedido e receção da autorização pode ser feito em segundos e garantir que todo o processo fique documentado sem comprometer o tempo e aumento de custos.

Neste sentido, o Estágio tem como finalidade criar uma **prova de conceito** de um sistema que permita agilizar a formalização do processo de autorizações terapêuticas. Há, ainda, a salientar que o pedido de autorizações deve ser assegurado pelo *software* MedicineOne 8, pelo que fica fora do âmbito do Estágio.

1.3 Objetivos

O presente projeto tem como principal objetivo o planeamento e desenvolvimento do *Autheras*, uma solução capaz de dar resposta à necessidade apresentada na secção 1.2.1, considerando o estudo e implementação de duas componentes:

- **Sistema agregador:** agnóstico, capaz de receber pedidos de autorização de EHRs em várias unidades de saúde (mensagens) e preparado para comunicar com iOS, Android e Windows Phone. Este sistema é especialmente importante para garantir que os profissionais recebam pedidos das várias unidades de saúde onde trabalham, centrados numa única aplicação móvel;
- **Aplicação móvel:** para a plataforma iOS, que receba os pedidos de autorização oriundos de diversas unidades de saúde, distribuídos eficazmente pelo sistema agregador, permitindo o acompanhamento do seu estado por parte dos requerentes, e a sua consulta e resposta por parte dos autorizadores.

De forma a acrescentar valor ao Estágio e ao produto a desenvolver, foi sugerido:

- Desenvolver a arquitetura e interface da aplicação móvel;
- Especificar a informação a conter nas mensagens para o tipo de dados a transportar entre as unidades de saúde e o sistema agregador;
- Garantir a segurança de todo o sistema através do uso de canais de comunicação seguros e encriptação de mensagens;
- Garantir a autenticação das mensagens transacionadas;
- Tirar partido das potencialidades das versões mais recentes do iPhone, através do uso do sensor biométrico *TouchID*, desenvolvido pela Apple Inc..

1.4 Estrutura do documento

O **primeiro capítulo** pretende contextualizar o leitor sobre o problema, os objetivos e o enquadramento do estágio. O **capítulo dois** apresenta o estudo feito no âmbito do problema, recolhendo informação sobre as abordagens existentes e tecnologias a considerar para o desenvolvimento da solução. No **capítulo três** é dada a conhecer a metodologia, o planeamento e os processos de engenharia utilizados.

O **capítulo quatro** é dedicado à análise de requisitos funcionais e não funcionais, onde é feito o seu levantamento, são identificados os atores do sistema, terminando com a definição de *user stories* e restrições associadas. No **capítulo cinco** é explicada a arquitetura do sistema, detalhando as componentes que o constituem.

O **sexto capítulo** aborda os aspetos de desenvolvimento, enquanto que o **sétimo capítulo** apresenta os testes realizados na aplicação, levando à sua validação. Por último, o **capítulo oito** contém as considerações finais sobre o trabalho desenvolvido, as dificuldades e desafios vividos e o ponto de vista sobre o trabalho futuro.

Capítulo 2

Estado da Arte

O capítulo do Estado da Arte expõe a análise às abordagens e soluções existentes que direta ou indiretamente se podem enquadrar no âmbito das autorizações médicas. São, também, dados a conhecer os standards de mensagens em contexto hospitalar e o que atualmente está disponível ao nível da autenticação do utilizador a partir do *smartphone*.

2.1 Abordagens

O médico é o profissional que, a nível legal, tem poder para prescrever cuidados clínicos e terapêutica farmacológica[1]. Em cenários idênticos aos já descritos na secção que aborda o Contexto do problema, especificamente nos casos onde o médico responsável pelos pacientes internados não se encontra em regime de presença física e, como tal, ausente do local de trabalho, o quotidiano tem demonstrado que nem sempre se justifica a deslocação à unidade de saúde para validar um procedimento perfeitamente assegurável por outros profissionais, como os enfermeiros. Não obstante, é necessário passar pelo processo de autorização para formalizar um determinado procedimento clínico.

A necessidade de autorização médica é um procedimento obrigatório no exercício de cuidados de saúde, tanto para apuramento de responsabilidades, como por motivos de viabilidade financeira. O papel do médico nem sempre é o de autorizador, pois em alguns casos, pode ser considerado como requerente em relação a sistemas de saúde fortemente dependentes das deliberações dos seguros de saúde[2].

A maioria dos processos atuais para a resolução desta imprescindibilidade, **não tem uma abordagem tecnológica**, tendendo a ser moroso e a criar hábitos que não se baseiam no código deontológico [1].

[1] Portaria nº 137-A/2012, de 11 de maio, Artigo 5º

[2] *Prior authorization process*:`https://en.wikipedia.org/wiki/Prior_authorization`

2.1.1 Abordagem não tecnológica

Nesta subsecção, são dadas a conhecer algumas das abordagens utilizadas que podem ser consideradas neste contexto.

Prescrição de terapêutica exclusivamente por via telefónica

A via telefónica é a forma de comunicação de eleição com o médico que está obrigado a permanecer contactável. Permite de forma rápida e pessoal transmitir as razões que carecem da avaliação e, consequentemente, de autorização ou mesmo observação médica. Não é, contudo, um meio eficaz nem válido para prescrição terapêutica, dado que há conhecimento de casos onde o responsável fica apenas pela prescrição por via telefónica, colocando o ónus das responsabilidade do lado do enfermeiro, situação que já foi denunciada pela Ordem dos Enfermeiros [2]. Ainda que a relação de confiança entre o médico e o enfermeiro ou médico assistente o permita, é um meio propício para o erro terapêutico [3] pela possível falha de interpretação na prescrição, dispensa ou administração.

A solução que se propõe vem dar resposta a esta fragilidade, uma vez que permite dar um passo para salvaguardar o não-repúdio do ato médico em situações que o permitam, assim como, a minimização de erros de comunicação entre os vários atores.

Delegação de deveres

A delegação de atos médicos a assistentes ou a pessoas que não o sejam pode ser apropriada em algumas circunstâncias, sempre que seja garantida a boa assistência ao paciente e uso eficiente dos recursos de saúde [4]. Dependendo do país e/ou estado, a jurisdição do processo de delegação pode ser mais ou menos formalizada. No entanto, para o caso em estudo, esta situação pode dar azo a situações de uso abusivo [5] como deixar prescrições em branco previamente assinadas, a delegar, na sua ausência, serviços que estão para além da formação, treino e experiência do profissional de saúde delegado. Assim, torna-se claro que o cuidado do paciente pode ficar seriamente comprometido quando a responsabilidade não é corretamente delegada.

São, ainda, usados pelos profissionais de saúde, formulários específicos para solicitar autorizações de mudança de procedimentos, não necessariamente urgentes, como é exemplo o "*Apêndice 2 do Laudo para solicitação de autorização de procedimento*", usado no Brasil [6].

2.1.2 Abordagem tecnológica

***Electronic Health Record*s (EHR)**

Os sistemas integrados de informação hospitalar visam dar resposta à necessidade crítica de gerir e integrar informação clínica, operacional e administrativa, inerentes às organizações de saúde de todas as dimensões. A cumplicidade entre a necessidade de obter resultados com o bom cuidado ao paciente nunca esteve tão alta, pelo que as soluções de *software* de EHR acrescentam uma real mais valia à eficiência hospitalar.

Estes produtos e serviços são, na sua generalidade, comerciais e complexos, não sendo de fácil acesso para estudos comparativos das suas *features*. Da análise da informação sobre as soluções mais conhecidas [7] (claramente mais direcionada para o *marketing*), não se pode inferir com objetividade a forma como tratam, em particular, as autorizações médicas dentro dos seus sistemas. Contudo, não foi encontrado, até à data, nenhuma oferta com a finalidade de tratar autorizações em contexto hospitalar, independente do sistema EHR e que permita centralizar num único produto várias unidades de saúde.

Poder-se-ão considerar os *providers* de sistemas EHR tanto como "parceiros", visto que as autorizações partem diretamente dos EHRs; como possíveis concorrentes, na medida em que podem vir a oferecer um produto idêntico para tratar as clínicas e hospitais onde já estão implementados. Nesta perspetiva, destacam-se por exemplo, a nível internacional, Cerner [3], Allscripts[4], athenahealth[5] e, a nível nacional, Glintt[6] e Alert[7] (ainda que, neste caso, em contexto de urgência).

Aplicações móveis

A maioria das aplicações móveis ligadas a sistemas de saúde são orientadas a soluções de suporte à decisão clínica, aumento da eficiência operacional, melhoramento do fluxo de trabalho e comunicação segura entre profissionais (médicos, enfermeiros, etc.) [8]. Apesar de não se direcionarem para a validação das autorizações médicas, é interessante verificar a forma como estas aplicações abordam a segurança e a comunicação. Neste sentido, são dadas a conhecer, sem grande enfoque, algumas das funcionalidades das aplicações analisadas, presentes na Tabela 2.1, nomeadamente o **TigerText**[8], **qliqCONNECT**[9] e **docBookMD**[10], por fazerem referência ao tipo de encriptação usada e poderem vir a ser possíveis competidores do produto, na medida em que a solução proposta poderá complementar os serviços que já oferecem. No âmbito do sector da saúde, estas ferramentas também acrescentam valor ao partilharem dados e permitirem integração com EHRs, possibilitando:

- anexar informação ao registo do paciente (por exemplo, imagens, documentação de conversas);
- criar alertas e notificações para os médicos, a partir dos resultados de análises, de acordo com o grau de importância.

Uma ferramenta que se destaca neste estudo é o Vocera Collaboration Suite, na medida em que o fluxo da informação se assemelha ao pretendido.

[3] `http://www.cerner.com/`
[4] `http://www.allscripts.com/`
[5] `http://www.athenahealth.com/`
[6] `sites.glintt.com/glinttconsulting/globalcare/`
[7] `http://www.alert-online.com/`
[8] `http://www.tigertext.com`
[9] `https://qliqsoft.com/solutions`
[10] `https://www.docbookmd.com`

Vocera

A Vocera Communications, Inc.[11], fundada em 2000, é uma empresa líder na criação de soluções com o objetivo de agilizar a comunicação pessoal em tempo real e melhorar o fluxo de trabalho em várias áreas, de onde se destaca a saúde. Nesse sentido, oferece uma vasta gama de serviços e *devices* que permitem a troca de informação através de voz, texto, alarmes e notificações dentro do seu sistema implementado em hospital ou clínica. Uma solução que se destaca é o **Vocera Collaboration Suite**[9, 10] ao permitir:

- **comunicação agnóstica entre dispositivos:** enviar mensagens seguras entre *desktop* e *smartphone*, sem necessidade de saber qual o tipo de dispositivo que o destinatário está a usar;
- **mensagens seguras:** encriptadas, em trânsito, com SSL e com AES no restante fluxo da informação, dentro das normas em vigor[12];
- **conectar instantaneamente:** com o *staff* necessário, dentro ou fora das unidades de saúde, com notificações imediatas e com mensagens instantâneas bidirecionais;
- **entregar:** conteúdo como documentos, *spreadsheets* e imagens para garantir que toda a informação crítica é transmitida.

O **Vocera Collaboration Suite** é compatível com iOS[13] (7.1+ a partir do iPhone 4S e da 2ª geração do iPad), ilustrado na Figura 2.1[14], e Android (4.0+), precisando, também, de ter ligação ao sistema Vocera (Vocera 4.4.2 e Secure Messaging Server 4.9.2)[10]. As aplicações móveis desta empresa têm acesso gratuito mas requerem registo prévio no sistema Vocera que está implementado na unidade de saúde, cujo preço varia do tamanho e dimensão de todo o sistema a implementar. A título de exemplo, recomenda-se a análise de [11].

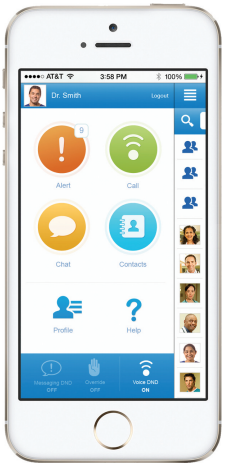


Figura 2.1: VCS para iPhone

A aplicação Vocera Collaboration Suite também está disponível para o Apple Watch[15], permitindo ver notificações de chamadas, mensagens e alertas críticos.

A Figura 2.2 ilustra um diagrama de atividade simplificado de forma a dar a conhecer o fluxo de informação demonstrado nos vídeos promocionais do Vocera Collaboration Suite[16].

[11] http://www.vocera.com/
[12] https://www.hipaa.com/
[13] https://itunes.apple.com/us/app/vocera-collaboration-suite/id806221179?mt=8
[14] fonte: http://i.imgur.com/6CyRwTM.jpg
[15] http://www.vocera.com/watch
[16] https://www.vocera.com/product/vocera-collaboration-suite

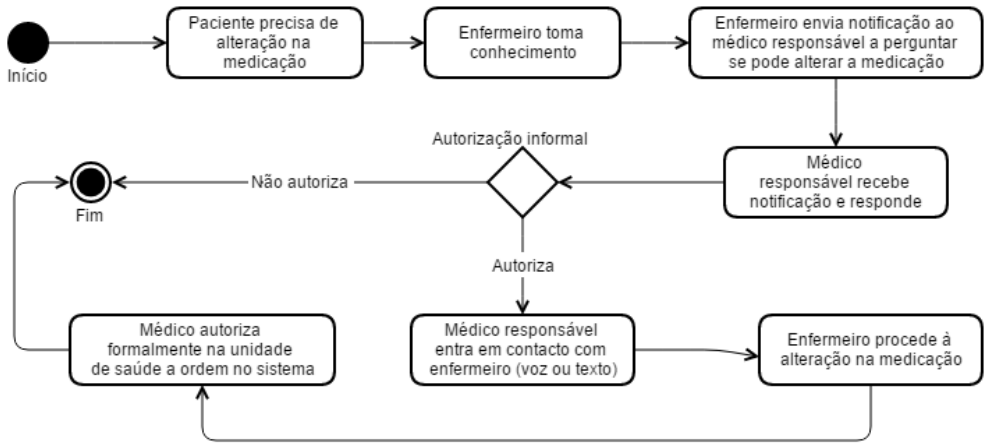


Figura 2.2: Diagrama de atividade simplificado do uso do VCS, com base nos vídeos promocionais da Vocera

Solução	Encriptação	Funcionalidades
TigerText	AES 256 bits	- opção de auto-destruição das mensagens; - definição de tempo de vida das mensagens; - integração HL7.
qliqCONNECT	Chave pública e privada (2048 bits)	- *log* de atividade; - mensagens são diretamente encaminhadas do remetente para o destinatário; - mensagens encriptadas e desencriptadas unicamente nos *devices* de envio e receção; - notificações *custom* de acordo com a sua prioridade; - confirmação de envio; - permite, opcionalmente, arquivar mensagens com o qliqStor; - integração HL7.
docbookMD	AES 256 bits	- desabilitar o *device* remotamente; - lembrete de notificação não vista para e-mail e/ou sms; - integração HL7.

Tabela 2.1: Soluções proprietárias de comunicação segura

2.2 Componentes da aplicação

Nesta subsecção, é apresentado o estudo relacionado com componentes da solução a desenvolver, nomeadamente:

- Mensagens;
- Autenticação do utilizador;
- iOS;
- *Push Notifications*.

2.2.1 Mensagens na área da saúde

Na área da saúde, existem vários standards relacionados com aspetos específicos dos EHRs que são, geralmente, criados por comités técnicos de organizações do sector, como vendedores, reguladores e consultores. A credibilidade das organizações e suas convenções é tanto maior quanto a capacidade de adoção no mercado e a filiação de importantes *players* do sector [12]. A Tabela A.1, presente no Apêndice A.1, lista os standards mais conhecidos para o formato e transação de mensagens, especificando a sua finalidade, onde são usados e quem os desenvolve/suporta [12, 13, 14].

Podendo ser, futuramente, um dos formatos utilizados nesta solução, já na fase de produto, este ponto aborda o Estado da Arte do padrão HL7, visto ter uma maior aceitação de mercado para a troca de mensagens relativas a informação clínica e é, em particular, usado na MedicineOne.

HL7 [12, 15, 16]

O padrão HL7 é suportado pela organização internacional *Health Level Seven*[17] (fundada em 1987), criado com o intuito de fornecer standards para a troca, gestão e integração de informação clínica e administrativa entre os diversos serviços de saúde. O *nível sete* que faz parte do nome da organização e do padrão refere-se ao nível de aplicação (nível mais alto) do modelo ISO-OSI[18]. Atualmente, a *Health Level Seven* dá suporte aos dois principais standards - HL7 versão 2.x, HL7 versão 3 - e FHIR.

- **HL7 versão 2.x (HL7v2.x)** [14, 16, 17, 18]
  Esta versão é a responsável pela aceitação global do HL7, sendo a v2.0 lançada em 1989 e a mais recente - v2.7 - disponibilizada em 2011. As implementações **HL7v2.x** encontram-se em mais de 35 países e conta com a utilização em cerca de 95% das organizações de saúde dos Estados Unidos [18]. A versão exata de HL7v2 que uma dada aplicação use não é problemática, na medida em que as versões são retro-compatíveis.

  O seu sucesso associa-se à sua principal vantagem e, simultaneamente, maior desvantagem − natureza *ad hoc* − que permite adaptar as mensagens às necessidades/propósitos da informação a transmitir mas que, por outro lado, não o

[17] `http://www.hl7.org/`
[18] `https://pt.wikipedia.org/wiki/Modelo_OSI`

favorece na necessidade de escalar para ambientes de maiores dimensões.

As interfaces são desenhadas para serem 80% definidas de acordo com as especificações HL7 e 20% *customizadas* pela implementação local. Este fator faz com que uma mensagem com a mesma função (por exemplo, admissão de um paciente), possa ser feita de várias formas, com vários campos de preenchimento opcionais, não sendo, assim, um ponto facilitador da integração. Torna-se, então, um desafio conseguir implementar uma estrutura standard de mensagem que contenha a informação estritamente necessária.

Encontra-se apresentado no Apêndice A.2 um exemplo da estrutura das mensagens. De seguida, são descritas na Tabela 2.2 as vantagens e desvantagens do HL7v2.x.

Vantagens	**Desvantagens**
- Compatibilidade entre as várias versões 2.x	- Demasiados campos e segmentos opcionais
- Curva de aprendizagem	- Ambiguidade
- Grau de adoção no mercado	- Não é um verdadeiro standard
- Funcional	- Difícil de escalar
- A grande maioria dos fabricantes de equipamentos clínicos fornece interfaces HL7v2.x para os seus equipamentos	- Falta de uma ontologia formal para unificar os conceitos das mensagens com tecnologias que foram surgindo (HL7v3, Web, XML)

Tabela 2.2: Vantagens e desvantagens do HL7v2.x

- **HL7 versão 3 (HL7v3)** [14, 16, 17, 19, 20]
  A terceira versão do padrão foi lançada em 2005, após ter começado a ser desenvolvida por volta de 1996. O HL7v3 é radicalmente distinto da versão anterior, tendo como principais objetivos: 1 − oferecer metodologias formais para modelar os dados e as mensagens; 2 − acrescentar o rigor e a precisão ao standard que a versão 2.x descura com a sua flexibilidade.

  A nova versão mostrou ser também uma alteração no paradigma ao usar princípios do UML e ao ser baseado num modelo orientado a objetos com processos bem definidos - **RIM** (***Reference Information Model***). Assim, em resumo, este modelo consiste numa categoria de três classes principais (*Act*, *Role*, *Entity*) ligadas por três classes de associação (*ActRelationship*, *Participation* e *RoleLink*).

  Ao nível das **mensagens**, a metodologia de desenvolvimento adotada é a ***Message Development Framwework*** que tem por base o modelo de informação do RIM, seguindo todo o processo standard de desenvolvimento de engenharia de *software* (consultar Apêndice A.3). De forma a especificar a estrutura e semântica de um documento clínico, mais orientadas ao conceito de EHR, existe

o ***Clinical Document Architecture***, implementado em XML (podendo incluir texto, imagens, som e outro conteúdo multimédia) e derivado do conteúdo partilhado pelo RIM. As principais vantagens e desvantagens do HL7v3 são apresentadas na Tabela 2.3.

Vantagens	Desvantagens
- Mais próximo do conceito de standard	- Não é compatível com HL7v2.x
- Rigor na definição	- Adoção leva muito tempo
- Melhor para o longo prazo	- Elevado custo de implementação
- Menos opções (menos ambiguidade)	- Elevada curva de aprendizagem
	- As aplicações têm que suportar duas versões nos próximos anos
	- Muito complexo
	- Quota de mercado muito baixa

Tabela 2.3: Vantagens e desvantagens do HL7v3 (Fonte: [16])

Durante o estudo do standard, foram identificadas e testadas algumas das ferramentas como RMIM Designer[19] para o Microsoft Visio e a *framework* Everest[20] para C# que, independentemente de simplificarem o tratamento de mensagens HL7v3, não invalida que se tenha um sólido conhecimento do RIM.

- **FHIR** [21] [21, 22, 23, 24]
  A título informativo, apresenta-se o *Fast Healthcare Interoperability Resources*, uma *framework* criada pela *Health Level Seven*, em 2011, que tenta combinar as melhores funcionalidades das versões 2.x, 3 e CDA, usando os mais recentes padrões da *web* [22]. Surge pela complexidade inerente ao HL7v3 e consequente lenta adoção e dúvida do seu sucesso. Também chamado como HL7v4, o FHIR é um *work in progress* considerado como *Draft Standard* [22, 23].

  O FHIR é baseado em recursos (*resources*), identificados com URLs e com representação em JSON ou XML. Para a sua manipulação, existe uma REST API que permite aplicar operações CRUD.

A Tabela 2.4, baseada em *D. Bender e K. Sartipi* [21], resume alguns dos *trade-offs* a ter em conta entre as versões apresentadas. Pese embora o Estado da Arte aponte para o HL7v3, ainda não existe um completo consenso sobre a mudança de paradigma a que obriga, em contexto hospitalar, como provam a lenta aceitação no mercado e as várias críticas à complexidade associada [21, 25].

[19] `http://gforge.hl7.org/gf/project/visio-rmim-desi/`
[20] `http://te.marc-hi.ca/`
[21] `https://www.hl7.org/fhir/`

Propriedade	HL7v2.x	HL7v3	FHIR
- Paradigma arquitetural	- Mensagens, campos e registos	- Orientado a mensagens	- RESTful
- Ontologia semântica	- Não	- Sim	- Sim?
- Curva de aprendizagem	- Semanas	- Meses	- Semanas
- Tamanho da especificação	- Centenas de páginas	- Milhares de páginas	- Centenas de páginas
- Exemplos de implementação	- Sim	- Mínimo	- Sim
Suporte da indústria e comunidade	- Forte	- Fraco	- Recente
- Grau de adoção	- Muito elevado	- Muito baixo	- Recente
- Onde é mais adoptado	- Estados Unidos; - Mundo em geral	- Turquia, Alemanha, Holanda;	- Recente

Tabela 2.4: Comparação entre standards HL7

2.2.2 Autenticação do Utilizador

Um dos pontos mais importantes em projetos no domínio da saúde, em geral, e no contexto *mobile*, em particular, é a segurança da informação. Direcionando a atenção para o lado do utilizador, considera-se pertinente e necessário conhecer quais os métodos e tecnologias existentes de forma a minimizar os problemas de autenticação.

Passcodes [26, 27]

A validação de palavras-passe ocorre quando o utilizador responde a um pedido de prova de identidade por parte do dispositivo [26], **baseando-se na informação que o utilizador sabe** previamente. No caso das *passcodes*, onde se incluem os PINs e as *passwords* alfanuméricas ou gráficas, tendem a ser negligenciados, como provam os estudos citados por *Pocovnicu* [27], onde mais de 50% dos utilizadores não utilizam PINs nos seus telemóveis, quer por falta de confiança quer pela inconveniência do processo. A Tabela 2.5 identifica algumas das propriedades deste processo que, apesar de ser dos mais vulneráveis, ainda deve ser usado nos dias de hoje e, se possível, em combinação com outras técnicas.

Tokens [26, 28, 29]

Este tipo de autenticação **baseia-se em informação que o utilizador tem** e é usado como um segundo fator independente de autenticação. Um *token* de segurança funciona como uma *password* que expira após um curto período de tempo e pode vir como:

- *hardware*, em pequenos *devices* que exibem a *password*, não sendo uma solução prática no contexto dos *smart devices*. *Rao et al.* [29] propõe a reutilização dos dispositivos existentes, como o próprio telemóvel e o cartão SIM (*Subscriber*

Identification Module).

- *software*, como programas que geram *one-time passwords* o mais aleatórias possível através de algoritmos desenvolvidos para o efeito[22].

Modo de captura	- Teclado (físico ou emulado) - *Touch-Screen*
Ameaças	- *Keyloggers*
Vantagens	- Pode ser alterado
Desvantagens	- As palavras-passe mais fáceis de adivinhar ainda são as mais usadas; - Pode ser partilhado, roubado;

Tabela 2.5: Propriedades das *passcodes*

Biometria [26, 27, 30]

Por último, a biometria diferencia-se das técnicas anteriores porque identifica o utilizador **com base naquilo que ele é**, tendo como maior vantagem o facto de não poder ser esquecido, partilhado ou roubado facilmente [27]. Segundo *A. K. Jain et al.* [30], existe um conjunto de características a ter em conta aquando da escolha do sistema biométrico:

- **Universalidade:** cada pessoa que está a usar o sistema biométrico deve possuir a característica biométrica;
- **Unicidade:** quão bem é que a característica biométrica separa um indivíduo de outro;
- **Permanência:** quão bem é que um traço biométrico resiste ao envelhecimento;
- **Recolha:** facilidade de aquisição da característica biométrica, sem causar inconvenientes para o utilizador;
- **Desempenho:** precisão, velocidade e robustez da tecnologia utilizada;
- **Aceitabilidade:** grau de aprovação da tecnologia biométrica por parte dos utilizadores;
- **Evasão:** facilidade de utilização de uma imitação da característica biométrica.

Dentro da biometria, distinguem-se a base comportamental (implícita) e a base fisiológica (explicita). No caso da primeira, como por exemplo o registo dos padrões de marcha ou de utilização do *smartphone*, apesar de ter o objetivo de ser menos intrusiva, ainda apresenta muitas limitações, como a necessidade de uma fase de treino (obter dados e treinar as redes neuronais) e o consumo de bateria com processos em *background* [26]. No caso da segunda, destaca-se os reconhecimentos de **voz**, **facial** e **impressão digital** por não precisarem, geralmente, de nenhum *hardware* adicional àquele que os *smartphones* atuais já possuem. As Tabelas 2.6 e 2.7 dão

	Univers.	Unic.	Perm.	Rec.	Desemp.	Aceit.	Evasão
Voz	+−	−	−	+−	−	+	+
Impressão digital	+−	+	+	+−	+	+−	+
Facial	+	−	+−	+	−	+	−

Tabela 2.6: Propriedades dos sistemas biométricos (+ alto; +− médio; − baixo)

	Ameaças de Falsificação	Fatores que podem diminuir a eficácia
Voz	- Gravação da voz do utilizador legítimo	- Doença, mudança natural da voz, barulho de fundo
Impressão digital	- Cópia da impressão digital com plasticina ou fita-cola	- Lesões nos dedos podem alterar o padrão
Facial	- Fotografia do utilizador legítimo	- Problemas de iluminação, maquilhagem, etc.

Tabela 2.7: Problemas associados aos sistemas biométricos

a conhecer uma análise comparativa destes três métodos com base, respetivamente, nas propriedades de *A. K. Jain et al.* e nos problemas que têm associados.

A título de curiosidade, é dada a conhecer a **Autenticação Neuronal**, suportado pelo o estudo [31] de *Blair C. Armstrong et al.*, da Universidade de Binghamton. Este meio de autenticação baseia-se no mapeamento dos padrões da atividade neuronal, únicos de pessoa para pessoa, em resposta a um dado estimulo externo. Como estimulo, foi colocada à disposição de cada candidato uma palavra durante um determinado período de tempo, sendo registada a resposta que o cérebro produzia, usando tal reação como *password* de autenticação. Foram alcançados resultados com uma taxa de sucesso de 97%, no âmbito do artigo estudado.

2.2.3 iOS

O iOS é o sistema operativo desenvolvido pela Apple para execução exclusiva nos seus dispositivos móveis: iPhone, iPad, iPod Touch e, mais recentemente, Apple Watch. Apresentado em 2007, tem as versões principais atualizadas anualmente, indo, atualmente, na versão 9 - **iOS 9** - com cerca de **73,5% de adoção** nos iPhones[23].

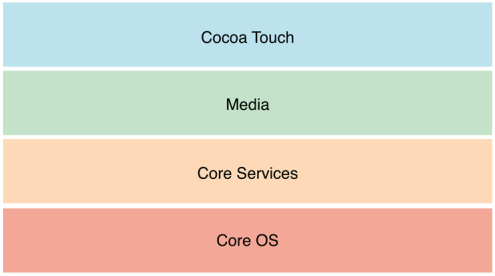


Figura 2.3: Camadas iOS - fonte:[32]

[22] *Time-Based One-Time Password Algorithm*: `https://tools.ietf.org/html/rfc6238`
[23] `https://david-smith.org/iosversionstats/`

A **arquitetura** do iOS é constituída por **quatro camadas** [32] (Figura 2.3), cada uma com o seu conjunto de *frameworks*, onde as que ocupam o nível superior são as mais abstratas e as de nível inferior são mais próximas do *hardware*. A camada **Core OS** situa-se no nível mais baixo, usada pelas superiores através das suas *frameworks*, necessitando de utilização direta quando há estrita necessidade de lidar com segurança ou comunicação com acessórios de hardware. A camada **Core Services** engloba os serviços fundamentais às aplicações, como por exemplo a *CFNetwork Framework*, que é um conjunto de interfaces que utiliza a abstração orientada a objetos para comunicar com protocolos de rede. A camada **Media** contém as tecnologias gráficas, áudio e vídeo, utilizadas para implementar experiências multimédia nas aplicações. Por fim, a camada de mais alto nível, **Cocoa Touch**, que encapsula funcionalidades como multitarefa, reconhecimento gestual (touch), serviços de notificação Apple *push* e interface gráfica (*UIKit*, que contém as classes necessárias para a interface com o utilizador).

Linguagens de programação [33, 34, 35]

Neste contexto, a primeira linguagem a surgir foi o **Objective-C**, uma linguagem orientada a objetos, baseada em C e criada no início da década de 80 pela StepStone Corporation. A linguagem foi licenciada e popularizada pela NeXT, empresa entretanto adquirida pela Apple na década de 90, chegando, assim, ao sistema OS X (baseado no NEXSTEP) e mais tarde ao iOS. O Objective-C não tem uma sintaxe amistosa e ainda precisa de um *mindset* mais próximo do C para ser melhor explorado (por exemplo, ter uma boa noção sobre ponteiros e alocação de memória).

Em 2014, a Apple anuncia o lançamento de uma nova linguagem para o desenvolvimento de aplicações para os sistemas OS X e iOS - **Swift**[24]. Apresenta uma sintaxe mais simples e intuitiva que a do Objective-C, com o objetivo de oferecer desempenho que permita ser um substituto das linguagens baseadas em C (C, C++, Objective-C) e de tornar a escrita e manutenção de código mais fáceis para os *developers*. Já em 2015, foi anunciado **Swift 2.0**, com melhorias na performance, alterações na sintaxe com a criação de novas *keywords* para tornar a linguagem mais natural e expressiva, e novos mecanismos de controlo de erros. Foi tornado *open source* em dezembro do mesmo ano. Com isto, não significa que o Objective-C seja desconsiderado, pois terá ainda um grande peso em contexto de produção, quer para manter os milhões de aplicações desenvolvidas, quer para as migrar para Swift. A Tabela 2.8 pretende dar a conhecer alguns pontos de comparação a ter em conta na escolha da linguagem.

[24] `https://swift.org`

Objective-C	Swift
Mais estável, com cerca de 3 décadas de mercado	É apontado como o futuro para o desenvolvimento de aplicações OS X e iOS
Maioria das aplicações e *frameworks* construídas em Objective-C	Mais adequado para projetos novos e com pouca ou nenhuma necessidade de migração
Sintaxe pouco amigável e *mindset* mais próximo do C	Mais fácil de aprender e sintaxe mais amigável
Esforço e custo elevados ao migrar grandes projectos para Swift	Boa interoperabilidade com Objective-C
Em contexto de produção, será sempre vantajoso ter experiência com a linguagem	Em contexto de formação, a aposta em Swift no presente permite adquirir a experiência necessária que será requisitada a médio e longo prazo.

Tabela 2.8: Objective-C *vs* Swift

IDE

O Xcode é o *Integrated Development Environment* da Apple para desenvolver *software* para OS X e iOS. A primeira *release* data de 2003, estando atualmente (dezembro de 2015) na versão estável 7.2. Trata-se de uma ferramenta bastante madura e completa, que além de oferecer um editor de código e ferramentas de *debug*, disponibiliza um conjunto de funcionalidades que auxiliam na produtividade[25], como:

- **iOS Simulator** que permite simular a interface dos dispositivos móveis da Apple na máquina de desenvolvimento, num ambiente próximo do real;
- **Quick Help**[26] que oferece uma ajuda rápida ao permitir acesso direto à documentação de referência enquanto se programa sem levar a que o utilizador tenha de sair do IDE para procurar ajuda;
- **Interface Builder**[27] que permite construir e testar a interface com o utilizador sem código. Isto é possível porque é utilizada a *pattern* ***Model-View-Controller*** que separa a camada de interfaces da camada de lógica (implementações). Também se destacam as *storyboards* que permitem construir as várias vistas da aplicação e obter uma visão completa do seu fluxo.

[25] `https://developer.apple.com/xcode/features/`

[26] `https://developer.apple.com/library/ios/recipes/xcode_help-general/Chapters/AboutQuickHelp.html`

[27] `https://developer.apple.com/xcode/interface-builder/`

Xamarin[28]

Uma alternativa para o desenvolvimento seria o Xamarin que é uma *framework* de desenvolvimento que permite criar aplicações *cross-platform* para iOS, Android e Windows Phone, utilizando C# de forma nativa, com as **vantagens** de conseguir cerca de 75% [36] de reaproveitação de código entre plataformas e um desenvolvimento mais rápido das aplicações. Em relação ao compilador, dependendo da plataforma, produz uma aplicação nativa (caso do iOS), ou uma aplicatição .NET integrada e em tempo de execução (caso do Android) [37].

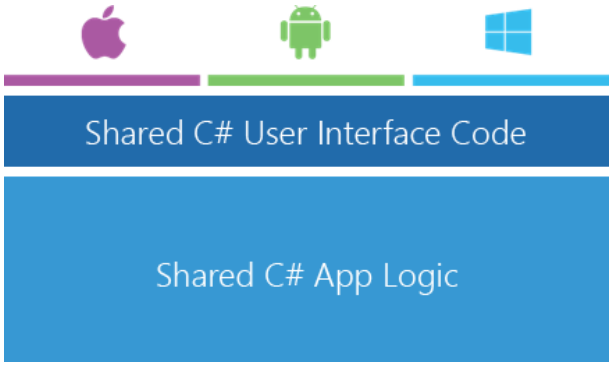


Figura 2.4: Camadas de reutilização de código e especificidade de cada plataforma, usando Xamarin Forms

O Xamarin tem uma boa velocidade de atualização em relação aos novos lançamentos das plataformas nativas e também tem a vantagem de permitir que os recursos de cada SDK sejam acedidos utilizando a sintaxe do C#, atuando como um "espelho", quer do iOS (Xamarin.iOS), quer do Android (Xamarin.Android) [37, 38]. O desenvolvimento pode ser feito com o *Xamarin Studio* ou com o *Visual Studio*.

Na primeira metade de 2014, foi lançado o **Xamarin Forms**[29] (Figura 2.4[30]), uma biblioteca que define uma abstração da *User Interface*, permitindo que o código partilhado entre as diferentes plataformas seja ainda maior, não esquecendo que poderá haver sempre aspetos específicos da plataforma.

A 24 de fevereiro de 2016, a Microsoft anunciou que foi assinado um acordo definitivo para adquirir a empresa Xamarin, fundada em maio de 2011. Esta aquisição permitiu minimizar aquela que será, provavelmente, a maior **desvantagem** - o preço[31]- ao tornar o Xamarin SDK *open-source*. O preço existente até ao início de 2016 foi, inclusive, um dos fatores que não tornou viável o estudo desta plataforma como possível ferramenta de utilização no âmbito do projeto, apesar de dar resposta favorável ao facto de a componente de servidor também ser desenvolvida em C# e facilitar o desenvolvimento multi-plataforma, uma vez que existe o objetivo de, futuramente, colocar a aplicação nos três sistemas móveis.

[28] https://www.xamarin.com/
[29] https://www.xamarin.com/forms
[30] https://blog.xamarin.com/meet-xamarin-forms-3-native-uis-1-shared-code-base/
[31] https://web.archive.org/web/20150905162259/https://store.xamarin.com/

2.2.4 *Push Notifications*

As *push notifications* permitem enviar, diretamente e com base na Internet, informação a um ou vários utilizadores, onde cada transação é iniciada pelo *publisher*/servidor central, contrastando com a tecnologia *pull*[32], em que a solicitação para a transmissão de informação é iniciada pelo *receiver*/cliente. Apresenta **vantagens** quer para quem envia, ao permitir enviar informação útil assim que esteja disponível, sem aguardar por uma requisição; quer para quem recebe, pois não necessita de ter a aplicação ativa ou em *background* para receber informação sobre uma dada aplicação no seu dispositivo. Aquando da instalação das aplicações, os clientes devem subscrever a um canal de informação fornecido pelo servidor, de modo a que, sempre que exista nova informação disponível no canal, esta tenha autorização para ser enviada enviada diretamente para o dispositivo do cliente [39, 40].

Contudo, a informação nunca é enviada diretamente de um *application server* para um dispositivo. Ao invés, a solução passa por a enviar um pedido HTTP/2 para os serviços de *push notification* de cada plataforma móvel (no caso da Apple, o APNS) que trata de reenviar para os dispositivos visados, onde o sistema operativo tratará de controlar a entrega na aplicação. como esquematizado na Figura 2.5 [39].

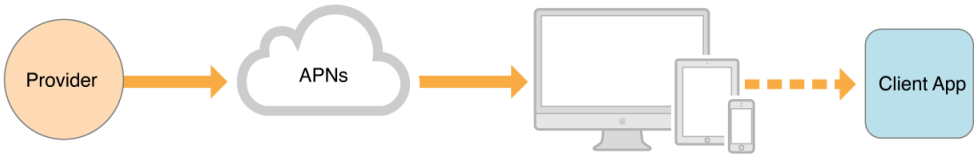


Figura 2.5: Trajeto de uma *Push Notification* - fonte: [41]

Para que uma aplicação esteja habilitada a receber este tipo de notificações, é necessário, aquando do desenvolvimento:

1. registar uma *App Id* que a identifique unicamente;
2. criar um certificado SSL para que o *application server* possa entrar em contacto com o APNS a fim de enviar notificações para a aplicação.

Do lado do dispositivo, para que a aplicação possa receber notificações, é necessário obter o *device token* que contem informação que permite o APNS localizar o dispositivo onde a aplicação está instalada. A geração e partilha do *token* estão esquematizados na Figura 2.6: quando a aplicação aceita a receção de notificações para uma dada aplicação, é feita uma conexão entre o dispositivo e o APNS (1), sendo devolvido o *device token* gerado (2), e posteriormente disponibilizado à aplicação (3) que deve tratar de o enviar para o *application server* (4). Como este *token* pode sofrer alterações (p. ex. desativar notificações, apagar a aplicação, restaurar as definições do dispositivo), deve ser adotada uma estratégia que permita minimizar os casos em que um dado dispositivo possa ser considerado como vários diferentes [41, 42, 43]. A ilustração das conexões entre *provider*−APNS e dispositivo−APNS podem ser consultadas do Apêndice A.4 [41].

[32] `https://en.wikipedia.org/wiki/Pull_technology`

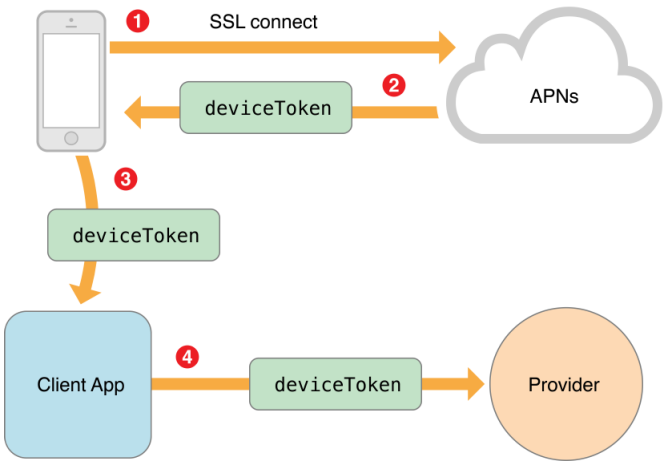


Figura 2.6: Partilha de um *device token* - fonte: [41]

De forma a agilizar a comunicação entre o *application server* e os serviços de notificações já existentes (p. ex. APNS, GCM, WNS), procedeu-se ao estudo de soluções disponíveis no mercado (Apêndice A.5) sob a forma de serviço que, contudo, não deu mostras de ser o mais adequado para a MedicineOne, pois obrigaria a: 1º− estar pendente de *third parties* (e possíveis variações de preços e políticas); 2º− migrar projetos. Não obstante, durante a pesquisa, teve-se conhecimento de que um dos principais *players* deste mercado, Parse[33], iria encerrar até 2017 [44], realçando a opinião anterior.

A solução mais viável passa por utilizar uma biblioteca *server-side*, como acontece atualmente com a biblioteca PushSharp[34], desenvolvida em C#, sendo também a escolha para o presente projeto por também já se encontrar a funcionar com as restantes aplicações da MedicineOne.

[33] `http://parse.com/`
[34] `https://github.com/Redth/PushSharp`

2.3 Análise crítica do Estado da Arte

A aposta num produto independente de EHR, capaz de gerir autorizações médicas de forma a que um profissional de saúde receba, num único *end-point*, todos as requisições que lhe forem feitas vem preencher um setor que ainda não foi explorado.

Pela análise dos *trade-offs* apresentados sobre as versões do formato HL7, pode concluir-se que, no caso de utilizar mensagens em contexto hospitalar, a versão 2.x ainda deve ser considerada compatível com os propósitos desta ideia, na medida em que ainda é o standard mais utilizado. Por não ser prioritário, esta prova de conceito não implementa nenhum dos formatos apresentados em contexto hospitalar, devendo, no entanto, estar preparado para dar suporte a várias versões e standards nos desenvolvimentos futuros.

Em relação à autenticação do utilizador, não se poderá colocar de lado o uso de *passcodes*, pois apesar das fragilidades mencionadas, continua a oferecer alguma segurança, independentemente do dispositivo, sobretudo quando não há possibilidade de acrescentar um segundo fator de autenticação. A utilização de biometria para incrementar a segurança já é uma realidade em algumas das aplicações mencionadas na subsecção anterior, nomeadamente através do reconhecimento da impressão digital. A Apple já permite trabalhar nesse sentido, ao oferecer suporte para trabalhar com impressões digitais no iPhone, a partir da versão 5S [45], com a TouchID API [46].

No que refere à linguagem de programação da aplicação iOS, uma vez que a Apple disponibiliza atualmente duas alternativas, deverá ter-se em conta que apesar de o Objective-C estar no mercado aproximadamente há 3 décadas, ser a base da maioria das soluções já existentes e, como tal, mais maduro e com maior suporte, considera-se que um projeto que começa do zero (sem exigir esforço de migração de projetos anteriores), oferece o cenário ideal para adotar o Swift, visto como o futuro do desenvolvimento para iOS.

Capítulo 3

Metodologia e Planeamento

Neste capítulo, são dados a conhecer a metodologia, processos de engenharia de *software* e o planeamento para os dois semestres.

3.1 Metodologia de desenvolvimento

Foi utilizada a metodologia ágil e incremental, à imagem do que é maioritariamente utilizado na MedicineOne. Esta opção justifica-se pelo desenvolvimento do trabalho com base em iterações, com incremento de novas funcionalidades, em projetos com alterações frequentes. Não é, contudo, seguida uma "definição canónica" de uma metodologia ágil, como por exemplo a *framework* Scrum, uma vez que o desenvolvimento será feito unicamente pelo estagiário. O planeamento é baseado em iterações com períodos de duração entre 2 a 3 semanas. Cada iteração contém um conjunto de atividades que devem ser executadas, com base nos requisitos levantados, de forma a acrescentar valor ao projeto.

Com o objetivo de gerir o trabalho desenvolvido, são realizadas reuniões no final de cada iteração e breves *meetings* diários com o orientador na empresa, para dar a conhecer o que foi desenvolvido no dia anterior e o que está previsto para o dia corrente, identificando impedimentos/dificuldades e priorizar o trabalho. A empresa utiliza, também, a aplicação OnTime[1] para a gestão de *features*, *incidents*, *defects* e de tarefas a realizar. Por fim, para obter um controlo mais visual em relação ao fluxo de trabalho, isto é, olhando para as tarefas como um *work in process* com várias fases, utiliza-se o quadro de *kanban* como é exemplo a ilustração da Figura 3.1.

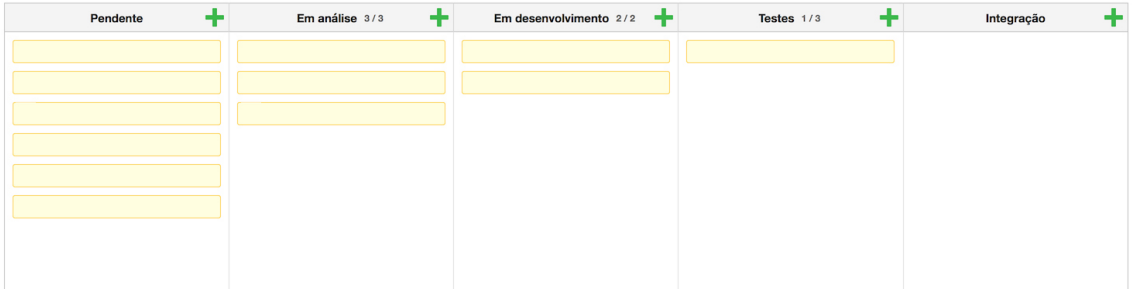


Figura 3.1: Exemplo da aplicação do quadro de *Kanban*

[1] `http://ontimesuite.com/`

O quadro de *kanban* tem colunas que representam o estado (fase) que cada item de trabalho (representado pelos retângulos amarelos) vai experimentando. De forma a evitar sobrecargas, é associado um limite de itens a cada fase.

3.2 Processos de engenharia de *software*

Com base na cadeira de Gestão de Projetos [47], no decorrer do projeto, foram utilizados alguns processos de engenharia de *software* com o objetivo de melhorar a gestão.

3.2.1 Reuniões

Um dos processos de monitorização do projeto foi o conjunto de reuniões com o cliente e orientadores de estágio. O **cliente** deste projeto é o fundador da MedicineOne e criador da ideia, João Miguel, tendo sido agendadas reuniões semanais com o objetivo de acompanhar o desenvolvimento e controlar possíveis alterações aos requisitos.

Ao longo do estágio ocorreram breves reuniões, sempre que possível, com o **orientador na empresa**, para tirar dúvidas e discutir o que foi feito nos dias anteriores. Por fim, com o objetivo de acompanhar o progresso do estágio na vertente curricular, foram agendadas reuniões mensais com o **orientador do DEI**.

3.2.2 Estimativas

A técnica escolhida para estimar o tempo necessário para desenvolver as diferentes tarefas associadas ao projeto foi a *Three-Point Estimation*[2][47] representada pela fórmula:

$$Expected = (1 \times Best + 4 \times Likely + 1 \times Worst)/6$$

3.2.3 Controlo de versões de *software*

Durante o desenvolvimento, foram utilizados o sistema de gestão de código *Team Foundation Server* (TFS), para o servidor, e o Git para a aplicação do cliente móvel.

3.2.4 Análise de riscos

O risco consiste na possibilidade de sofrer uma perda[3], como tal, a sua gestão é um dos processos de engenharia de *software* mais importantes, na medida em que visa minimizar as áreas onde não há controlo. A avaliação dos riscos [47, 48] deve abordar a consequência, a probabilidade de ocorrência (Tabela 3.1), o impacto (Tabela 3.2), a janela de tempo e o plano de mitigação associado.
São, também, definidos termos que compõem a janela de tempo que permite perceber quando é que um dado risco poderá ocorrer:

- **Curto-Prazo:** ocorre desde o início do projeto até às primeiras semanas de desenvolvimento;

[2] `https://www.projectsmart.co.uk/estimating-project-costs.php`
[3] `http://beta.merriam-webster.com/dictionary/risk`

- **Médio-Prazo:** ocorre na fase de desenvolvimento das componentes do projeto;
- **Longo-Prazo:** ocorre na fase final do projeto, ou posteriormente.

Probabilidade	<25%	25% - 50%	51% - 75%	>75%
Descrição	Baixa	Média	Alta	Muito alta

Tabela 3.1: Definição das probabilidades de ocorrência

Impacto	**Descrição**
Baixo	Sucesso do projeto não comprometido, estando dentro do previsto
Médio	Sucesso do projeto não comprometido, mas necessita de pequenos ajustes, com esforço dentro do previsto
Alto	Sucesso do projeto comprometido se não forem feitos ajustes, dependente de esforço adicional
Crítico	Sucesso do projeto comprometido

Tabela 3.2: Definição dos níveis de impacto

Riscos

Neste ponto são dados a conhecer os riscos identificados no âmbito do projeto, considerando a sua importância com base nos níveis descritos.

Elevada curva de aprendizagem

Risco: 01	**Impacto:** Crítico	**Probabilidade:** Muito alta	**Tempo:** Curto/Médio
Facto: Inexperiência no uso das tecnologias associadas ao projeto.			
Consequência: Atraso nas tarefas e possível incumprimento dos requisitos.			
Plano de Mitigação: Reajustamento do tempo necessário para familiarização com as tecnologias. Suporte dos colegas de trabalho mais experientes.			

Tabela 3.3: Risco − Elevada curva de aprendizagem

Disponibilidade da equipa de desenvolvimento

Risco: 02	**Impacto:** Crítico	**Probabilidade:** Alta	**Tempo:** Longo-Prazo
Facto: Falta de disponibilidade da equipa de desenvolvimento para o desenvolvimento da *feature* do pedido de autorizações no MedicineOne 8.			
Consequência: Atrasos no desenvolvimento e nos testes ao sistema.			
Plano de Mitigação: Criação de plataforma que simule um cliente EHR			

Tabela 3.4: Risco − Disponibilidade da equipa de desenvolvimento

Alterações aos requisitos

Risco: 03	**Impacto:** Médio	**Probabilidade:** Baixa	**Tempo:** Curto-Prazo
Facto: Considerando a metodologia de desenvolvimento ágil adotada, poderão existir alterações aos requisitos no decorrer do projeto.			
Consequência: Atrasos no desenvolvimento.			
Plano de Mitigação: Reuniões com o cliente para dar a conhecer o trabalho desenvolvido e discutir possíveis alterações a fazer.			

Tabela 3.5: Risco − Alterações aos requisitos

Riscos verificados

No decurso do primeiro semestre, existiram riscos que se verificaram − #**01** e #**03**. Em relação ao primeiro, fruto da falta de experiência e conhecimento na maioria das tecnologias envolvidas, nomeadamente HL7, foi investido algum do tempo inicial no seu estudo que acabou por não se mostrar suficientemente proveitoso para aplicação no projeto e num primeiro protótipo. Isto não só influenciou a calendarização como interferiu na própria conceção inicial da arquitetura. No segundo caso, já no final do semestre, os requisitos funcionais foram revistos de forma a dar toda a prioridade à gestão de pedidos, deixando para segundo plano a gestão de entidades e, no caso dos requisitos não funcionais, foi esclarecida, já perto do fim do semestre, a restrição do sistema agregador não poder guardar os dados clínicos dos pedidos de autorização.

Durante o segundo semestre, verificou-se a ocorrência dos riscos #**01** e #**02**. O primeiro foi causado pelo atraso na implementação de algumas funcionalidades do sistema, contribuindo para que aspetos calendarizados para períodos posteriores ficassem com uma abordagem mais frágil. Por sua vez, o segundo, mais previsível, levou a que o cliente EHR fosse simulado por uma ferramenta do *Visual Studio*, cujo objetivo é de testar serviços WCF.

3.3 Planeamento

O planeamento enquadra-se em dois semestres, onde o primeiro está direcionado para uma fase de pesquisa e o segundo para a fase de desenvolvimento propriamente dito. O planeamento foi materializado sob a forma do Diagrama de *Gantt* para os dois semestres.

3.3.1 Primeiro semestre

O primeiro semestre, destinado essencialmente ao planeamento e investigação, foi iniciado pelo estudo do contexto do problema, após a sua apresentação.

	Nome	Início	Fim
1	**Primeiro Semestre**	**08/10/2015**	**01/02/2016**
2	Estudo do Contexto do problema	08/10/2015	12/10/2015
3	Estado da Arte	13/10/2015	20/11/2015
4	**Especificação de Requisitos**	**19/10/2015**	**25/11/2015**
5	Levantamento de requisitos funcionais	19/10/2015	21/10/2015
6	Especificação e validação de requisitos	19/11/2015	25/11/2015
7	Estudo e desenvolvimento da arquitetura	25/11/2015	04/12/2015
8	Estudo de tecnologias	25/11/2015	08/12/2015
9	Inicio do desenvolvimento do protótipo	09/12/2015	13/01/2016
10	Análise de riscos	04/01/2016	08/01/2016
11	**Avaliação Intermédia**	**09/11/2015**	**01/02/2016**
12	Escrita do relatório	09/11/2015	21/01/2016
13	Entrega do relatório	22/01/2016	22/01/2016
14	Defesa intermédia	01/02/2016	01/02/2016

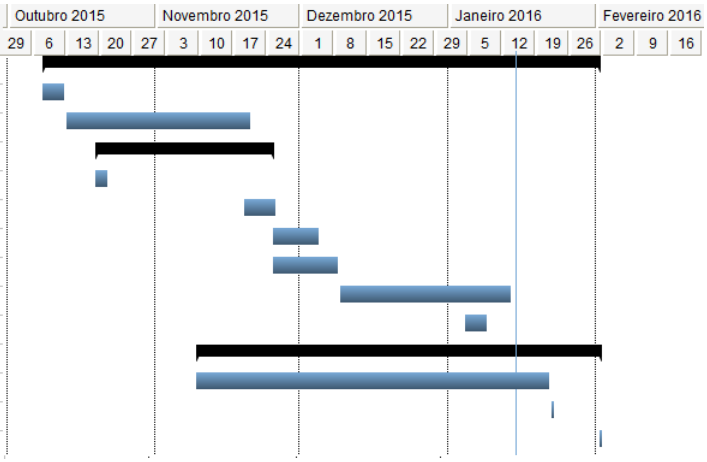
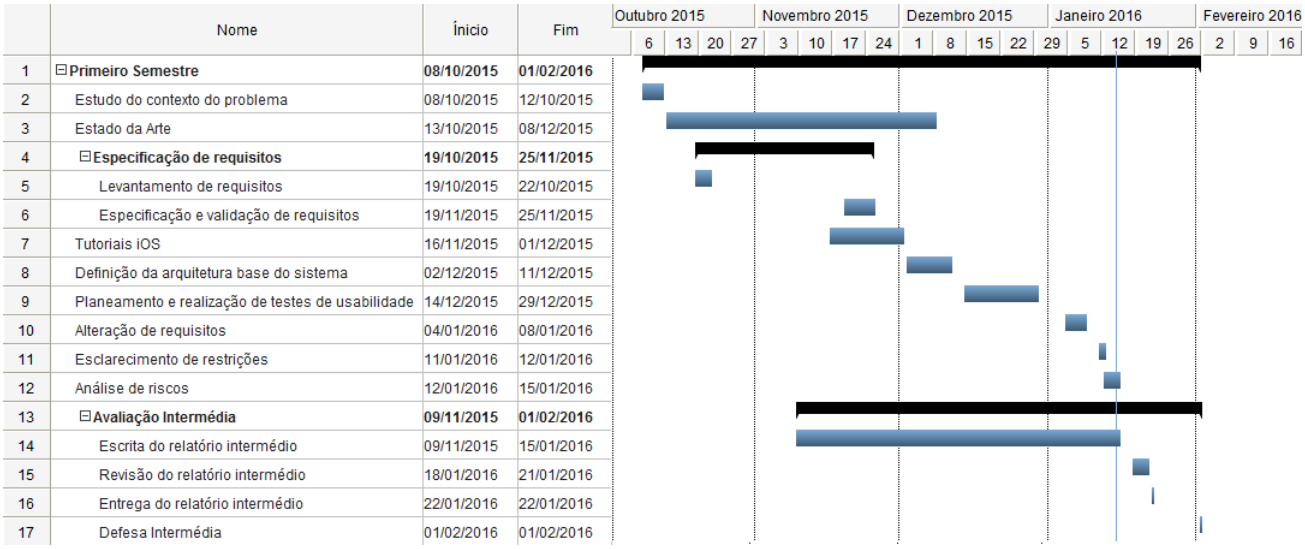


Figura 3.2: Planeamento para o primeiro semestre

A execução das tarefas do semestre registou alguma derrapagem (Figura 3.3) em relação ao plano previsto inicialmente, considerada otimista, na medida em que a complexidade tinha sido subestimada. No caso do Estado da Arte, não contemplou todos os fatores considerados importantes para o sistema e ocupou mais tempo do que o previsto em alguns pormenores. A ocorrência do **Risco#01**, referido em 3.2.4, também contribuiu para que o início e desenvolvimento de um protótipo fosse reajustado para conter o estudo de alguns tutoriais, documentação (nomeadamente iOS e WCF) e do levantamento da necessidade de proceder a testes de usabilidade. Já no mês de janeiro, também por fruto da metodologia utilizada, houve uma alteração aos requisitos (**Risco#04**) de forma a dar prioridade à gestão de pedidos e abstraindo do âmbito do estágio a gestão de entidades (unidades de saúde e profissionais de saúde) e a relação entre os clientes EHR e o sistema agregador. O esclarecimento necessário de que parte da informação não podia ficar guardada no sistema também obrigou a repensar no fluxo que a informação iria tomar. A análise de riscos associados identificou os que efetivamente ocorreram e como isso acabaria por influenciar o planeamento do semestre seguinte.



	Nome	Início	Fim
1	**Primeiro Semestre**	**08/10/2015**	**01/02/2016**
2	Estudo do contexto do problema	08/10/2015	12/10/2015
3	Estado da Arte	13/10/2015	08/12/2015
4	**Especificação de requisitos**	**19/10/2015**	**25/11/2015**
5	Levantamento de requisitos	19/10/2015	22/10/2015
6	Especificação e validação de requisitos	19/11/2015	25/11/2015
7	Tutoriais iOS	16/11/2015	01/12/2015
8	Definição da arquitetura base do sistema	02/12/2015	11/12/2015
9	Planeamento e realização de testes de usabilidade	14/12/2015	29/12/2015
10	Alteração de requisitos	04/01/2016	08/01/2016
11	Esclarecimento de restrições	11/01/2016	12/01/2016
12	Análise de riscos	12/01/2016	15/01/2016
13	**Avaliação Intermédia**	**09/11/2015**	**01/02/2016**
14	Escrita do relatório intermédio	09/11/2015	15/01/2016
15	Revisão do relatório intermédio	18/01/2016	21/01/2016
16	Entrega do relatório intermédio	22/01/2016	22/01/2016
17	Defesa Intermédia	01/02/2016	01/02/2016

Figura 3.3: Verificação dos desvios ao plano inicial

3.3.2 Segundo semestre

A utilização de *push notifications* assume maior importância no fim do primeiro semestre, pelo que, apesar de já existir uma tecnologia usada na MedicineOne – *PushSharp*[4] – e se ter procurado testar o serviço *Azure Push Notification*[5], optou-se por enriquecer o tema com a inclusão de um breve estudo sobre outras alternativas no início do segundo semestre.

O início do desenvolvimento da aplicação móvel, serviria para estudar e colocar em funcionamento algumas bibliotecas base que já existem nas aplicações móveis da MedicineOne ao nível de, por exemplo, base de dados e conexão à rede.

No que toca às funcionalidades da aplicação, a primeira fase de desenvolvimento corresponderia às tarefas associadas às *user stories* com grau de importância *Must-Have*, enquanto que a segunda fase se destina às tarefas das *user stories Nice-to-Have*.

A implementação de alguns mecanismos de segurança, terá por base o estudo e respetivas decisões, previstos para o início do semestre. Seguindo-se posteriormente o planeamento e execução de testes funcionais sobre a aplicação, de forma a verificar se o seu funcionamento geral vai de encontro com os requisitos especificados. No final seria explorado o requisito *Wishful*, ainda que num contexto mais teórico, pois ainda não existe nenhum dispositivo disponível.

No entanto, existiu uma alteração em relação ao plano inicial (Figura 3.4), verificando-se, também, a ocorrência de uma derrapagem na fase de desenvolvimento. Na prática, na vez de começar a implementar as funcionalidades mais importantes nas duas componentes do *Autheras* –sistema agregador e aplicação móvel– o desenvolvimento foi feito de forma continuada em cada uma delas, existindo uma fase posterior destinada a integração das duas. A falta de experiência com as tecnologias utilizadas, identificada no **Risco#01**, contribuiu para que o término fosse a 12 de maio de 2016.

[4]https://github.com/Redth/PushSharp
[5]https://azure.microsoft.com/en-us/services/notification-hubs/

Por conseguinte, com o tempo disponível, o reforço e implementação de mecanismos de segurança não ficou tão completo quanto o desejado e o planeamento da fase de testes não perspetivou a existência de testes automatizados. Também se concluiu que o requisito *Whishful* deixaria de poder vir a estar no âmbito do projeto, dado a baixa relevância no mercado atual.

	Nome	Início	Fim
1	**Segundo Semestre**	**01/02/2016**	**01/07/2016**
2	Estudo de Serviços de PushNotifications	01/02/2016	02/02/2016
3	Estudo de Segurança	08/02/2016	15/02/2016
4	Modelo de dados	16/02/2016	17/02/2016
5	Inicio de desenvolvimento do sistema agreagador	18/02/2016	23/02/2016
6	Inicio de desenvolvimento da aplicação móvel	24/02/2016	29/02/2016
7	Primeira fase de desenvolvimento	01/03/2016	01/04/2016
8	Segunda fase de desenvolvimento	04/04/2016	26/04/2016
9	Implementação de mecanismos de segurança	27/04/2016	17/05/2016
10	Testes funcionais à aplicação	18/05/2016	02/06/2016
11	Estudo para a realização do requisito GEN_US010 (Wishful)	03/06/2016	17/06/2016
12	Escrita de relatório final	03/06/2016	01/07/2016

Figura 3.4: Planeamento para o segundo semestre

Capítulo 4

Análise de Requisitos

Esta secção é dedicada à formalização da especificação dos requisitos do *software* a desenvolver no âmbito do estágio. São descritos os atores do sistema, os requisitos funcionais e não funcionais do produto, assim como as *user stories* criadas e as restrições a que está sujeito. Maior detalhe sobre os requisitos funcionais do sistema, com base em *user stories*, poderá ser consultado no Apêndice C.1.

A priorização dos requisitos foi dividida em três categorias, de acordo com o grau de importância identificado em conjunto com a *MedicineOne*:

- ***Must-Have*:** o requisito deve fazer parte das funcionalidades implementadas, sendo prioritária a sua execução de forma a ir de encontro com os objetivos de negócio;
- ***Nice-to-Have*:** o requisito pode fazer parte das funcionalidades implementadas, sendo a sua implementação opcional, não comprometendo o sucesso do projeto;
- ***Wishful*:** o requisito foi identificado mas é de carácter opcional. A sua implementação não é prevista até ao fim do projeto.

4.1 Atores do sistema

Esta secção identifica os atores que interagem diretamente com a aplicação, descrevendo o seu papel e função.

Ator	Papel	Função
Médico	Profissional de saúde que utiliza o *smart device* (*smartphone* ou *wearable*) como autorizador	Recebe, visualiza e responde às autorizações
Enfermeiro	Profissional de saúde que utiliza o *smart device* (*smartphone* ou *wearable*) como requerente. Remetente dos pedidos de autorização, a partir das unidades de saúde	Cria e envia requisições de autorização através de EHRs (meio de envio), recebendo a resposta final. Acompanha o estado das autorizações enviadas

Tabela 4.1: Atores do sistema

4.2 *User stories*

A partir da documentação existente e das reuniões internas realizadas, foi possível iniciar o levantamento das funcionalidades que o sistema deveria ter. A formalização da especificação dos requisitos funcionais começou com a utilização de *user stories*, que descrevem uma interação do utilizador com o sistema, focando no valor que daí pode resultar [49]. As *user stories* são geralmente usadas em metodologias ágeis de desenvolvimento de *software*, de forma a fornecer um alicerce para a definição de requisitos com base numa estrutura que visa dar resposta, respetivamente, a **quem**, **como** e **porquê** [50] através:

- **Enquanto** <*utilizador*>
- **Quero** <*ação*>
- **A fim de** <*objetivo*>

As *user stories* têm a vantagem de serem relativamente rápidas de construir e acessíveis, quer ao programador, quer ao cliente. Neste sentido, os artefactos construídos estão presentes no Apêndice C.1.

4.3 Prototipagem

A prototipagem de média fidelidade da aplicação móvel foi concebida e disponibilizada pela *designer* da MedicineOne, com base na experiência das aplicações já desenvolvidas pela empresa e nas funcionalidades idealizadas pelo cliente do projeto, João Miguel. O estagiário organizou o fluxo dos ambientes de navegação com o *software* Flinto[1], idealizando as tarefas e métricas de usabilidade a serem feitas. Estes *mockups* servem como base para a construção da aplicação móvel, podendo sofrer alterações no decorrer do projeto, com vista ao melhoramento do produto final.

[1] `https://www.flinto.com`

4.4 Requisitos funcionais

Os requisitos funcionais representam as principais funcionalidades do sistema e como ele se deve comportar e reagir aos estímulos a que está sujeito durante a sua execução. Com base nos atores identificados e nas *user stories* criadas, foram especificados os requisitos funcionais do sistema. De forma a perceber melhor o contexto, cada requisito tem associado no seu ID uma referência:

- **GEN_RF:** relacionado com o médico e com o enfermeiro;
- **MED_RF:** relacionado com o médico;
- **ENF_RF:** relacionado com o enfermeiro.

As referências estão orientadas à perspetiva da aplicação móvel, ou seja, à funcionalidade. Nesta secção, na Tabela 4.2, são dados a conhecer os requisitos, acompanhados da sua identificação e prioridade.

ID	Nome	Prioridade
GEN_US001	Inscrição de um novo utilizador	*Nice-to-Have*
GEN_US002	Autenticação de um utilizador	*Must-Have*
GEN_US003	Permitir a recuperação da *password*	*Nice-to-Have*
GEN_US004	Aceder à aplicação através de uma notificação, destacando o pedido	*Must-Have*
GEN_US005	Listar autorizações pendentes	*Must-Have*
GEN_US006	Exibir todos os campos dos pedidos de autorização	*Must-Have*
GEN_US007	Consultar o histórico de pedidos	*Nice-to-Have*
GEN_US008	Filtrar o histórico de pedidos	*Nice-to-Have*
GEN_US009	Menu Principal	*Must-Have*
GEN_US010	Receber notificações no *smart watch*	*Wishful*
MED_US001	Autorizar um pedido	*Must-Have*
MED_US002	Rejeitar um pedido, justificando-o	*Must-Have*
ENF_US001	Enviar *reminders* relativos aos pedidos sem resposta	*Nice-to-Have*
ENF_US002	Receber confirmação de leitura do pedido de autorização	*Must-Have*
ENF_US003	Receber resposta ao pedido de autorização	*Must-Have*

Tabela 4.2: Requisitos funcionais da aplicação

4.5 Requisitos não funcionais

Os requisitos não funcionais, também denominados como atributos de qualidade, definem-se como características que o sistema deve ter para além das funcionalidades, definindo capacidades e condições que o sistema deverá cumprir.

Usabilidade

Descrição: A navegação deve apresentar uma interface profissional, fácil de aprender, seguindo a integridade do design das aplicações MedicineOne.

Categoria: Aplicação móvel

Resposta: Criação de *mockups*, testar a usabilidade através da prototipagem antes de iniciar o desenvolvimento. Ver Secção 7.3.

Grau de importância: *Must-Have*

Compatibilidade

Descrição: Esta versão da aplicação deve ser compatível com o sistema operativo iOS, na versão mínima 8.0 e, consequentemente, com os modelos de iPhone que o correm.

Categoria: Aplicação Móvel

Resposta: Devem ser efetuados testes de compatibilidade à aplicação nos modelos 4S, 5S, 6 ou 6S, 6 Plus ou 6S Plus do iPhone a correr, pelo menos, iOS 8.0.

Grau de importância: *Must-Have*

Segurança

Descrição: As questões de segurança são importantes para esta aplicação. Como tal, o sistema deve vedar acessos não autorizados aos serviços, permitindo entrada apenas a utilizadores credenciados

Categoria: Aplicação móvel, Sistema Agregador

Resposta: Informação em trânsito com utilização de HTTPS em conjunto com *Basic Authentication*; Encriptação da informação. Autenticação do utilizador no *smartphone* deve fazer uso de tecnologia biométrica, sempre que possível ou acompanhado de uma *password* escolhida pelo utilizador.

Grau de importância: *Must-Have*

Interoperabilidade

Descrição: O módulo do sistema agregador deve estar preparado para comunicar com diferentes EHRs (apesar de, para esta prova de conceito, o objetivo seja a integração com a solução MedicineOne), presentes em várias unidades de saúde e com os sistemas operativos móveis atualmente no mercado (iOS, Android e Windows Phone).

Categoria: Sistema Agregador

Resposta: Utilização de SOAP na comunicação com os EHRs e de REST com os clientes *mobile*.

Grau de importância: *Must-Have*

Escalabilidade

Descrição: O sistema deve ser capaz de aumentar o desempenho, quando sujeito a um aumento da carga de utilização, sem exigir mudanças ao nível arquitetural, nem comprometer o serviço devido ao tempo necessário para redimensionar os recursos.

Categoria: Sistema Agregador

Disponibilidade

Descrição: O sistema deve ter alta disponibilidade. A definição dos níveis mínimos do *Service Level Agreement*, esperados pelo cliente, envolverá a definição de métricas com base em indicadores como o *Mean Time Between Failures*[2].

Categoria: Sistema Agregador

Estes dois últimos requisitos não funcionais foram identificados como fulcrais na conceção de um produto final. Podem ser garantidos ao nível do *deployment*, graças a tecnologias da *stack* da *Microsoft*, no entanto, como este estágio consiste na realização de uma **prova de conceito**, para validar a ideia deste projeto, estes dois requisitos não funcionais não são prioritários para serem garantidos.

4.6 Restrições

Ao nível das restrições, foram agrupadas de acordo com o seu tipo: **de negócio**, **técnicas** e **legais**.

4.6.1 Restrições de negócio

- **Prazo de entrega:** a primeira versão do protótipo funcional deve estar disponível antes da data de defesa final do Estágio;
- **Mercado-alvo da aplicação:** é direcionado para profissionais de saúde, nomeadamente médicos e enfermeiros;
- **Relutância do cliente EHR para alterações no sistema:** a comunicação entre EHR e sistema agregador acontece somente neste sentido (o EHR envia as mensagens e pede as respetivas respostas);

[2]Tempo médio entre falhas

4.6.2 Restrições técnicas

- Os **serviços** implementados do lado do sistema agregador devem ser **desenvolvidos** com recurso à *framework* .NET, uma vez que a MedicineOne já é parceira da Microsoft, utilizando as suas ferramentas;
- A **aplicação** para iPhone deve suportar iOS 8 e iOS 9;
- **Base de dados do servidor:** Microsoft SQL Server;
- **Expansibilidade:** O sistema será implementado de forma a dar resposta ao caso de uso das necessidades das alterações terapêuticas, contudo, deve ser pensado para suportar qualquer tipo de atividade existente no software EHR que no futuro venha a ser assinalada como requerendo este tipo de autorização.

4.6.3 Restrições legais

O sistema a desenvolver deve estar em conformidade com a legislação nacional aplicável em vigor, regulada pela Comissão Nacional de Proteção de Dados. A Lei nº. 12/2005 de 26 de Janeiro, para a informação genética pessoal e informação de saúde, identifica os dados a tratar neste Estágio. A empresa decidiu que informações relativas ao conteúdo do pedido e à identificação do paciente não podem ser persistidas no servidor.

Capítulo 5

Desenho da solução

A proposta que compõe a solução para o problema descrito inclui a arquitetura e as tecnologias do sistema, assim como o estudo sobre usabilidade da aplicação móvel. Neste âmbito, realça-se que o *Autheras* é fruto de uma ideia apresentada com o objetivo de obter uma prova de conceito criada de raiz no âmbito deste estágio.

5.1 Pressupostos

Antes de dar a conhecer as componentes de desenho da solução, existem alguns pressupostos a ter em conta. A **gestão de entidades**, ou seja, o registo de novos profissionais e unidades de saúde, assim como sua associação, é considerada como menos relevante para esta prova de conceito. A **comunicação com os clientes EHR**, no âmbito do estágio, partirá do princípio que apenas são trocadas mensagens com uma plataforma que simulará o seu funcionamento, recebendo pedidos e disponibilizando respostas.

5.2 Arquitetura

Nesta secção é dada a conhecer a arquitetura do sistema, abordando elementos importantes para a implementação da solução ao nível do agregador central e da aplicação que funcionará como cliente móvel. A arquitetura é ilustrada com recurso a diagramas de diferentes vistas [51].

5.2.1 Vista *bird's eye*

A Figura 5.1 tem o objetivo de expor, de forma simplificada, a ideia pretendida pelo cliente, ou seja, um **sistema agregador central** (servidor) que interage com dois tipos diferentes de cliente, encaminhando as mensagens que contêm os pedidos provenientes dos vários **clientes EHR** (unidades de saúde) para os respetivos *smartphones* de requerente (enfermeiro) e autorizador (médico), funcionando como **clientes *mobile***. O mesmo sistema encaminha as respostas dos clientes *mobiles* de volta para os clientes EHR. Os dois sistemas, servidor e *mobile*, serão construídos de raiz.
De notar que os pedidos de autorização, assim como os pedidos de respostas, partem do lado dos EHRs, feitos por enfermeiros após primeiro contacto telefónico com o médico responsável pelo paciente em questão.

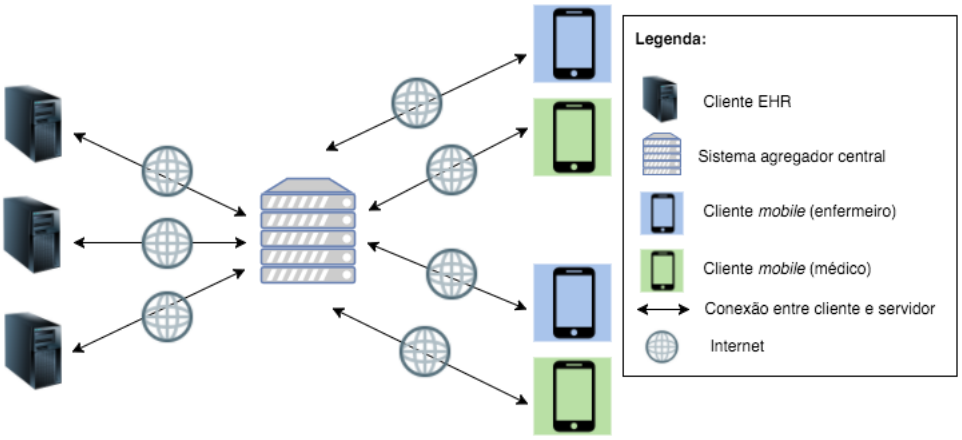


Figura 5.1: Vista *bird's eye*

5.2.2 Vista de componente

A componente delimitada pela fronteira do sistema agregador, Figura 5.2, representa o *back-end*, constituído por três componentes:

- **Servidor de aplicação:** é onde se localizam os *web services* destinados à gestão de utilizadores/profissionais, gestão de unidades de saúde, gestão de pedidos, autorização e autenticação. Os pedidos entre os clientes e o sistema agregador são feitos com recurso a SOAP e REST, respetivamente cliente EHR e cliente *mobile*, por via de HTTPS.

- **Base de dados:** destinada à gestão de entidades e de pedidos de autorização. A comunicação com este módulo e a componente de base de dados é efetuada com recurso a um ORM (*Object-Relacional Mapping*), mapeando, assim, os dados na base de dados.

- **Notificações *push*:** para além do conteúdo da notificação em JSON, contêm referência para o ID da *App* e para o dispositivo do destinatário. Este bloco fornece uma abstração *cross-platform* que permite que uma única API comunique com qualquer serviço de notificação. No caso da presente aplicação, para iOS, o APNS (*Apple Push Notification Service*). Este serviço é utilizado para enviar os pedidos de autorização aos clientes *mobile*, intervenientes num determinado pedido AS.

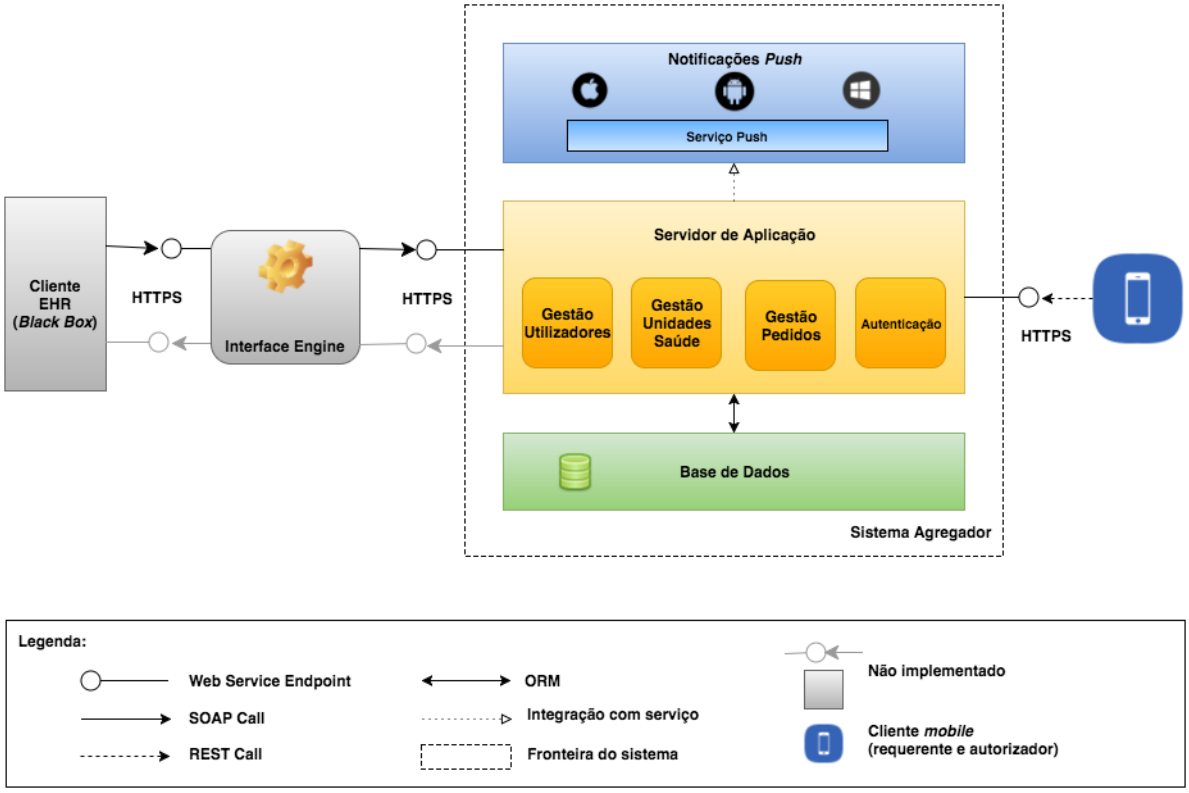


Figura 5.2: Vista Componente

A sugestão de utilização de um *interface engine*[52] dá resposta à necessidade deste sistema poder vir a suportar vários formatos, sobretudo em contexto hospitalar (subsecção 2.2.1). Na prática, este *software* (p. ex., Mirth[1], Iguana[2], Rhapsody[3]) multiplataforma permite o envio bidirecional de mensagens entre sistemas e aplicações com o objetivo de garantir uma maior interoperabilidade. Porém, esta abordagem não foi seguida para a implementação, devido a restrições que são justificadas no último parágrafo desta subsecção.

O funcionamento baseia-se na utilização de conectores de entrada (*income listeners*), como TCP/IP ou *web services*, e de saída (*outcome listeners*), como servidores de aplicação, onde, pelo meio aplica filtros, transformações e encaminhamento nas mensagens, com base em regras definidas (Figura 5.3).

Idealmente, e não existindo a terceira restrição de negócio apresentada na subsecção 4.6.1, o sistema agregador deveria estar preparado para iniciar comunicação com os serviços expostos pelos diferentes clientes EHR, evitando, assim, de os implementar do lado do próprio sistema. Não obstante, a não existência desta mesma restrição também permitiria retirar a importância das *push notifications*, na medida em que a comunicação seria iniciada do lado do cliente móvel, a fim de obter o conteúdo que lhe é destinado. Contudo, como já foi referido, neste projeto os serviços estão implementados no lado do sistema agregador e a comunicação parte sempre do lado do cliente EHR, não existindo a implementação do *interface engine*.

[1] `https://www.mirth.com/`
[2] `http://www.interfaceware.com/iguana.html`
[3] `https://orionhealth.com/us/products/rhapsody/`

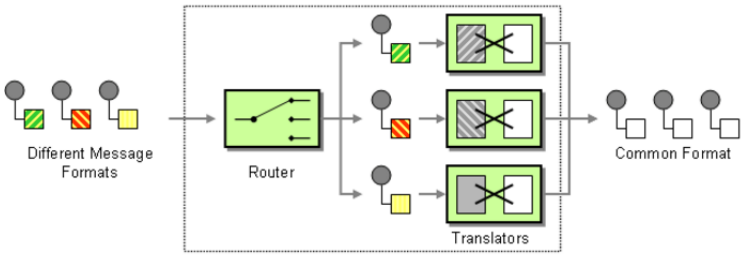


Figura 5.3: *Interface engine* - imagem adaptada [53]

5.2.3 Vista de camada (componente servidor)

Fazendo um *zoom-in* à componente do Servidor de Aplicação (Figura 5.4), pode decompor-se os serviços em 3 camadas de abstração.

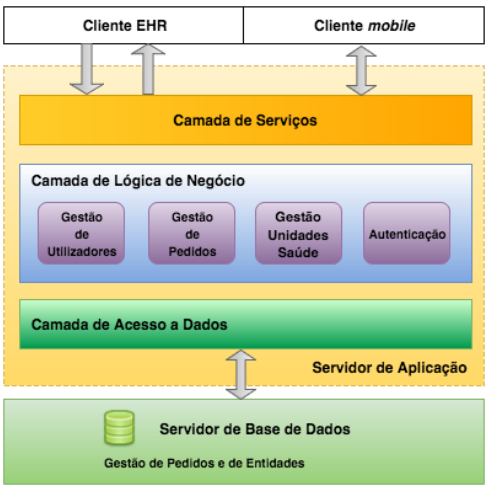


Figura 5.4: Vista de Camada (componente servidor)

A **primeira camada**, com os serviços disponíveis para o exterior, inclui os *service contracts* (dados que entram e saem do serviço) e *operation contracts* (métodos disponíveis) que definem as interfaces que permitem a comunicação com os diferentes *endpoints*.

A **segunda camada**, da lógica de negócio, aplica a lógica às operações dos serviços, analisa pré-condições para cada operação, realiza as atividades de negócio e devolve os resultados para a camada de interface do serviço. Os três módulos representados nesta camada e com nomes auto-explicativos, permite separar responsabilidades, implementando modularidade.

A **terceira** e última **camada**, de acesso a dados, trata das tarefas necessárias para conectar a base de dados subjacente, retirando essa preocupação às camadas superiores [54]. Esta camada utiliza o *Repository Pattern* (Fowler, 2002) [55] para isolar a lógica de acesso a dados de qualquer lógica da aplicação [56].

O *design pattern* em questão, Figura 5.5, é uma abstração que permite testar a lógica de acesso à base de dados em separado, promove a manutenção e confiabilidade de código, permite abstrair a utilização de ORMs e de tipos de bases de dados (podendo ser trocados) e permite desenvolver dados num formato agnóstico. O que na prática

está implementado neste projeto é a criação de vários repositórios com ações específicas de acesso a dados para cada regra de negócio e a criação de um repositório genérico com ações comuns a cada um dos supra-mencionados.

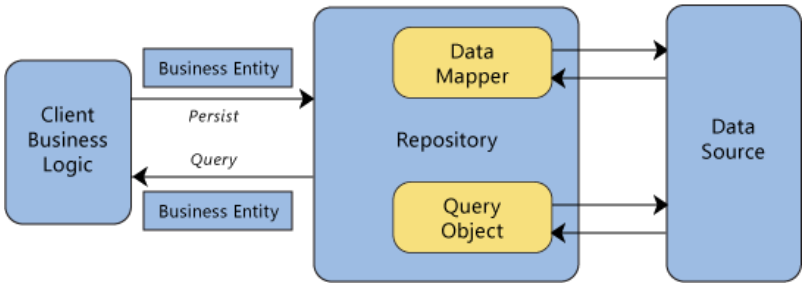


Figura 5.5: *Repository pattern* - fonte: [56]

5.2.4 Vista de camada (cliente *mobile*)

Na Figura 5.6, estão representados os principais módulos da aplicação, separados por camadas. A camada ***View***, responsável pela apresentação e interação dos dados com o utilizador, comunica com o módulo equivalente na camada adjacente (***Controller***), que atua como um intermediário entre uma ou mais ***views*** e um ou mais objetos da camada de dados (***Model***), sendo responsáveis por notificar as vistas sobre alterações nos modelos e vice-versa. A camada de dados, ***Model***, é responsável pelo encapsulamento dos dados e define a lógica que os manipula e os processa. Na MedicineOne, nesta camada, é utilizado um módulo adicional (*Management*), responsável pela obtenção dos dados e/ou configurações (independentemente da fonte, como bases de dados locais ou provenientes de serviços externos − por exemplo, da MedicineOne, permitindo saber a que URL ligar para as chamadas aos serviços), ajudando a retirar muito do "peso"que está geralmente associado à camada *Controller* na *pattern Model-View-Controller* [57].

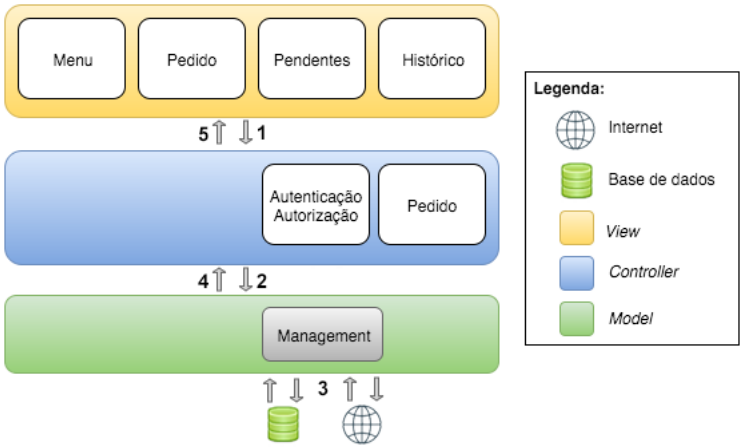


Figura 5.6: Vista de Camada (cliente *mobile*)

No módulo **Menu** existe a integração do menu lateral, responsável pelo acesso às diferentes funcionalidades. O módulo **Pedido**, composto, na camada de *View*, pelo "Pedido", "Pendentes" e "Histórico" trata da gestão do fluxo de pedidos, listando os pendentes e os já processados. O módulo de **Autenticação e Autorização**, diz

respeito à autenticação do utilizador no sistema e à autenticação e autorização para responder ao pedido de autorização que lhe está associado.

Resumidamente, a informação segue o seguinte fluxo: **1** − existe uma ação do utilizador; **2** − a ação detona uma alteração/atualização na camada de dados; **3** − são feitas as operações necessárias para implementar as modificações nos dados; **4** − o *Controller* é notificado da alteração feita; **5** − a camada *View* atualiza a informação que é disponibilizada como resposta à ação do utilizador.

5.3 Tecnologias

- **.NET *Framework***
  - **C#:** é a linguagem de programação integrante nesta *framework* para o desenvolvimento do sistema agregador;
  - **WCF:** o desenvolvimento de serviços suporta-se na tecnologia *Windows Communication Foundation*, como parte integrante da *framework* utilizada. Um projeto WCF permite usar SOAP e REST;
  - ***Entity Framework*:** é o ORM presente na *framework*, usado de forma a mapear os dados necessários nas bases de dados.
- **HTTPS:** protocolo de transferência utilizado para comunicação segura que permite a criação de um canal de comunicação encriptado ao usar SSL (*Secure Socket Layer*) como uma sub-camada do protocolo HTTP [58]. No âmbito do projeto, é considerado na comunicação entre o sistema agregador e os dois tipos de cliente.
- **SOAP:** as mensagens baseiam-se em XML, podendo ser transmitidas com recurso a vários protocolos, entre os quais, HTTPS. A opção pelo standard SOAP justifica-se pelo facto de oferecer maior interoperabilidade no lado do cliente EHR, na medida em que este poderá funcionar sobre uma vasta lista de possíveis sistemas de informação desconhecidos ao sistema agregador, provavelmente *legacy*.
- **REST:** utilizado na comunicação cliente *mobile*-sistema agregador, na medida em que é mais leve que o SOAP, correndo sobre HTTPS.
- **JSON:** a comunicação com o cliente *mobile*, considerando também que será utilizado o serviço de notificações *push* para o envio do próprio pedido de autorização para este tipo de cliente, obriga a utilização deste formato. Não obstante, é considerado leve e rápido na execução e transporte dos dados.

Relativamente aos serviços, a comunicação com um cliente ocorre através dos *Endpoints* expostos pelo serviço WCF, constituídos por: - ***Address*** que permite identificar um *Endpoint*, disponibilizando informação do endereço e do protocolo de transporte (HTTP e HTTPS); - ***Contract*** que descreve as funcionalidades de um dado *Endpoint*, onde se realçam os *Services Contracts* (operações disponíveis nos serviços) e os *Data Contracts* (tipos de dados trocados); - ***Binding*** que especifica como comunicar com um *Endpoint*, ao combinar aspetos da comunicação com um serviço, como

protocolos de transporte, modos de codificação da mensagem e aspetos de segurança [59, 60].

5.4 Modelo de dados

A modelação dos dados do lado do servidor (sistema agregador) deve assegurar a gestão de entidades (profissionais e unidades de saúde/EHRs) e de endereçamento dos pedidos. Também existiu a necessidade de criar um modelo de dados ao nível do cliente móvel de forma a poder persistir os pedidos, na integra, provenientes dos diferentes EHRs para os profissionais envolvidos.

5.4.1 Servidor

Como já foi referido anteriormente, o foco do modelo de dados no lado do servidor baseia-se na persistência das entidades envolvidas, assim como o seu envolvimento nos pedidos gerados. A Tabela 5.1 descreve cada uma das entidades do diagrama presente na Figura 5.7.

Tabela	**Descrição**
Professional	Representa os profissionais registados no sistema. Neste caso existe hereditariedade de forma a representar todas as classes profissionais existentes (no caso, médico e enfermeiro)
Specialty	Indica as especialidades que um médico pode ter
Health Organization	Esta tabela representa as unidades de saúde/EHRs de onde cada mensagem é gerada pelos respetivos profissionais
Message	Tabela que indica os intervenientes de cada pedido, assim como o seu estado (urgência, datas de envio e resposta, envio). Como consequência da 3ª restrição de negócio, o conteúdo dos pedidos não é persistido
Country	Representa o país de cada profissional (cliente móvel)
Professional Association	Tabela que representa as ordens profissionais dos profissionais (p. ex: ordem dos médicos, ordem dos enfermeiros, ...)
Device	Indica os dispositivos móveis registados no sistema. Cada dispositivo contém uma identificação única para APNS e só pode estar registado para um e um só profissional
DeviceType	Representa os tipos de dispositivo existentes que suportam o sistema
Notification Attempt	Regista as notificações enviadas para os dispositivos móveis
Device Notification Type	Indica os tipos de notificações a enviar para os dispositivos (mensagem enviada, recebida, lida, pedidos de resposta)

Tabela 5.1: Descrição das entidades do modelo da componente servidor

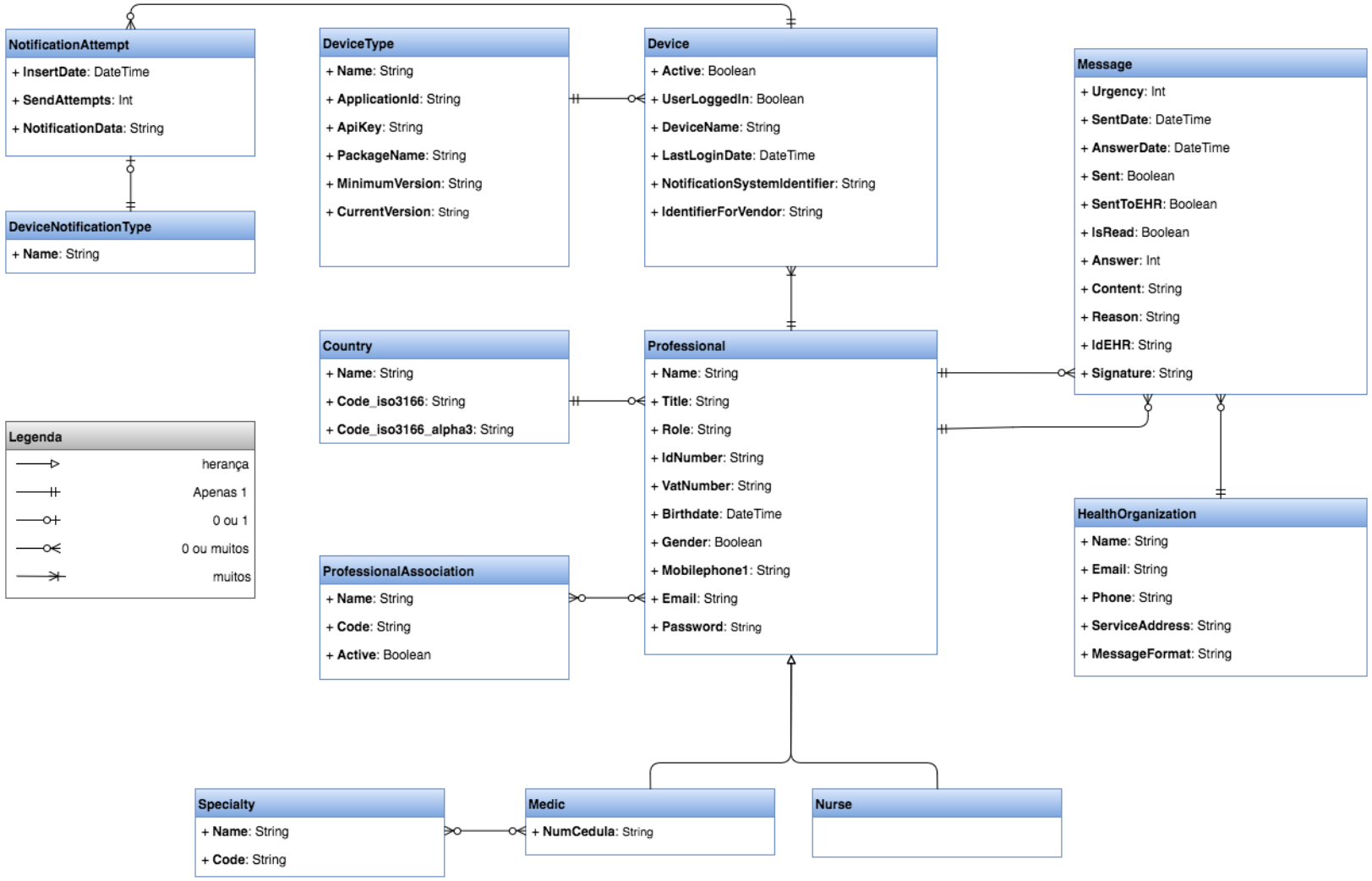


Figura 5.7: Modelo de dados da componente servidor

5.4.2 Aplicação móvel

O modelo de dados do lado do cliente móvel, ilustrado na Figura 5.8, é relativamente mais simples. Existem algumas decisões feitas aquando do planeamento que, já no final, demonstraram necessidade de serem revistas em trabalho futuro. Por exemplo, o *countryCode* e o *gender* deveriam dar lugar a entidades devidamente atualizadas sempre que existissem alterações do lado do servidor. A escolha original justificou-se pelo facto de se ter considerado que estes dois atributos da entidade *User* teriam uma menor probabilidade de serem alterados/atualizados.

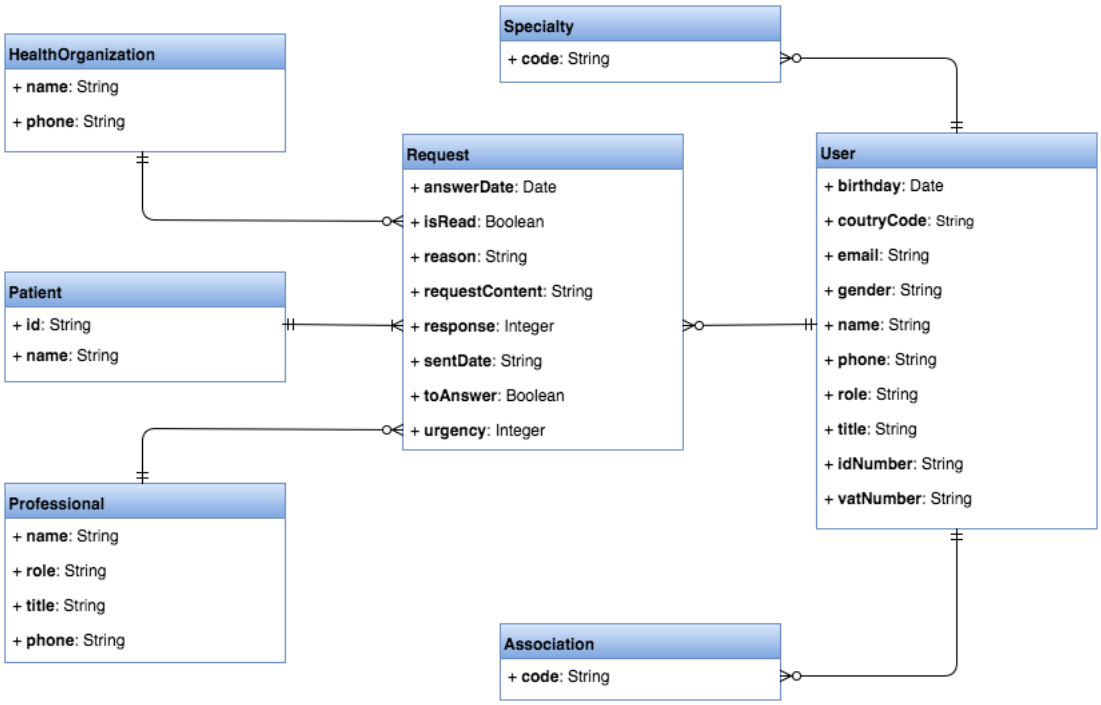


Figura 5.8: Modelo de dados da componente cliente (aplicação móvel)

O conteúdo das tabelas deste modelo de dados é alimentado com base na informação que provem do servidor (sistema agregador), com diferença para a entidade *Patient* que apenas é persistida do lado do cliente móvel (além da parte do cliente EHR, fora do plano deste projeto).

Contrariamente ao modelo do lado do servidor, os pedidos de autorização são associados ao utilizador (autorizador se o atributo *toAnswer* for verdadeiro, ou requerente no caso contrário) do dispositivo e a outro profissional.

5.5 Segurança

Os aspetos de segurança mais importantes no âmbito deste projeto são a autenticação do utilizador, tal como a confidencialidade, integridade e autenticidade das mensagens trocadas entre os serviços e os clientes móveis.

5.5.1 Autenticação na aplicação

O registo de um novo utilizador através da aplicação móvel deve suceder:

- validação do utilizador como profissional através dos seus números de identificação fiscal e civil, em pelo menos uma das unidades de saúde registadas no sistema;
- certificar que o dispositivo móvel escolhido não está associado a mais nenhum profissional registado no sistema.

O mecanismo que limita o acesso aos dados a partir da aplicação móvel consiste num **sistema de *login*** onde cada utilizador deve ter um par de credenciais − **email** e ***password*** (subsecção 2.2.2) − para poder entrar. Caso o dispositivo esteja capacitado com sensor de impressões digitais (tecnologia *TouchId* nos modelos acima do 5S, inclusive, no caso do iPhone da Apple), o utilizador pode realizar a operação de *login* através de **leitura biométrica** (subsecção 2.2.2), apesar de, na prática, comparar a impressão digital fornecida com a que está guardada no dispositivo; se corresponder, envia o par de credenciais supra-referenciado para o servidor.

Considerando que se utiliza o protocolo HTTPS (providenciando confidencialidade e integridade para as mensagens transmitidas [61]) para aceder aos serviços disponibilizados no servidor, o controlo é efetuado com *Basic Authentication*[4]. Apesar deste mecanismo ter algumas desvantagens como as credenciais serem enviadas no cabeçalho do HTTP em cada pedido, codificadas em transito com Base64[5] sem estarem encriptadas, não deixa de ser um standard na Internet e um protocolo relativamente simples que se adapta a esta prova de conceito. Em alternativa, o recomendado seria a utilização de um mecanismo de autenticação baseada em *tokens*, delegando a autenticação para uma entidade certificadora externa (p. ex. *Secure Token Service* [62]) em que tanto cliente como servidor confiam, contribuindo para uma maior escalabilidade e modularidade do sistema, na medida em que há uma clara separação das responsabilidades existentes [63].

5.5.2 Confidencialidade e integridade das mensagens

Para a garantia de confidencialidade e integridade das mensagens trocadas entre as aplicações e serviços da MedicineOne, existe a solução elaborada pelo Eng.º Pedro Faustino, no âmbito da sua Tese de Mestrado na empresa em 2014 [60], não apresentando, contudo, um desenvolvimento maduro nesta prova de conceito. Esta referência oferece segurança − encriptação e autenticação − ao nível da mensagem, *end−to−end*

[4] `https://tools.ietf.org/html/rfc2617`

[5] `https://www.techopedia.com/definition/27209/base64`

[64], e integra a biblioteca *RNCryptor*[6] nos serviços e na aplicação móvel para garantir as duas finalidades supra-ditas. Necessita, contudo, de recorrer aos pontos de extensibilidade do WCF [65] para implementar um *encoder* onde é feita a desencriptação das mensagens antes de serem enviadas para o *encoder* de raiz do WCF que efetua a sua serialização com o formato JSON.

5.5.3 Autenticidade e não repúdio das mensagens

Por outro lado, para garantir uma atuação segura, a solução deve fazer uso da assinatura digital dos conteúdos com certificados digitais garantindo, com valor legal, a sua autenticidade, integridade e não repúdio. Um certificado digital contém informação sobre o período de validade, a entidade para a qual o certificado foi emitido mais a chave pública referente à chave privada que se acredita ser de posse unicamente da entidade especificada no certificado [66, 67].

A assinatura digital, tipicamente tratada como análoga à congénere em papel, está comummente ligada à criptografia de chave pública (criptografia assimétrica), uma das técnicas para gerar documentos digitais com validade legal [68]. Tipicamente, uma assinatura digital resulta:

1º − da aplicação de uma função de *hash* sobre uma mensagem de tamanho variável, gerando um código único e irreversível de tamanho fixo;

2º− da encriptação com a chave privada do emissor, enviado em conjunto com o conteúdo a assinar.

Por existir a geração de um *hash* de tamanho fixo, existe a probabilidade de diferentes mensagens originarem *hashes* iguais (colisões). Contudo, quanto maior for a dificuldade de se criarem colisões, melhor o algoritmo associado (Dang, 2008) [69, 70]. O esquemas de criação e verificação de uma assinatura digital podem ser consultado nos Apêndices D.1 e D.2.

Figura 5.9: Exemplo de leitor de *smart cards* para iPhone e iPad - fonte [71]

Para garantir a integridade do certificado, este deve ser assinado por uma Autoridade Certificadora (um exemplo em Portugal é a Multicert[7]) [72]. Uma das formas de ob-

[6] `https://github.com/RNCryptor/RNCryptor`
[7] `https://www.multicert.com/pt/`

ter um certificado com valor legal é através do Cartão de Cidadão[8] que, do ponto de vista eletrónico, tem um chip de contacto com certificados digitais para autenticação e assinatura eletrónica através de um leitor de *smart card* [73]. Como referido em [74, 75], este mesmo chip permite ainda guardar informação intransmissível do titular, como a sua chave simétrica de autenticação e as chaves privadas dos pares de chaves assimétricas RSA (1024 bits) para autenticação e produção de assinaturas.

A possível utilização de *smart cards* para efeitos de autenticidade nesta solução não é tão linear quanto parece em virtude da discussão sobre o *trade-off* segurança *vs* usabilidade. A necessidade de estar na posse de um leitor de *smart cards* embutido no dispositivo móvel (Figura 5.9) para poder assinar digitalmente cada uma das respostas aos pedidos de autorização, aliado ao elevado preço (entre $90 e $150 para iPhone e iPad, no período de escrita do presente relatório) [71], poderá desmotivar alguns dos possíveis utilizadores finais.

Porém, o recente Despacho[9] de 25 de fevereiro de 2016 que atribui carácter obrigatório à prescrição de receitas sem papel para todas as entidades do Serviço Nacional de Saúde, poderá coadjuvar na aquisição e familiarização dos leitores de *smart cards*, atendendo a que estes serão necessários para a realização de prescrição médica. Por conseguinte, apesar de não se estar perante a solução perfeita, este cenário possibilita minimizar os motivos de inviabilização da usabilidade desta solução aquando da entrada do *Autheras* no mercado.

[8] de acordo com a Lei n.º 7/2007 de 5 de Fevereiro, Artigo 18.º

[9] Despacho N.º 2935-B/2016 - Diário da República N.º 39/2016, 1º Suplemento, Série II de 2016-02-25

Capítulo 6

Desenvolvimento

Esta secção tem o objetivo de dar a conhecer as etapas de desenvolvimento das várias partes que compõem a solução apresentada. Existe uma subdivisão para melhor descrever os serviços desenvolvidos na componente servidor e para os módulos que integram a aplicação móvel.

6.1 Componente servidor

A componente servidor foi desenvolvida em C# com uma solução composta por vários sub-projetos, como ilustram as várias camadas presentes na Figura 6.1. Esta desagregação permite separar responsabilidades, promovendo a manutenção e testabilidade do código.

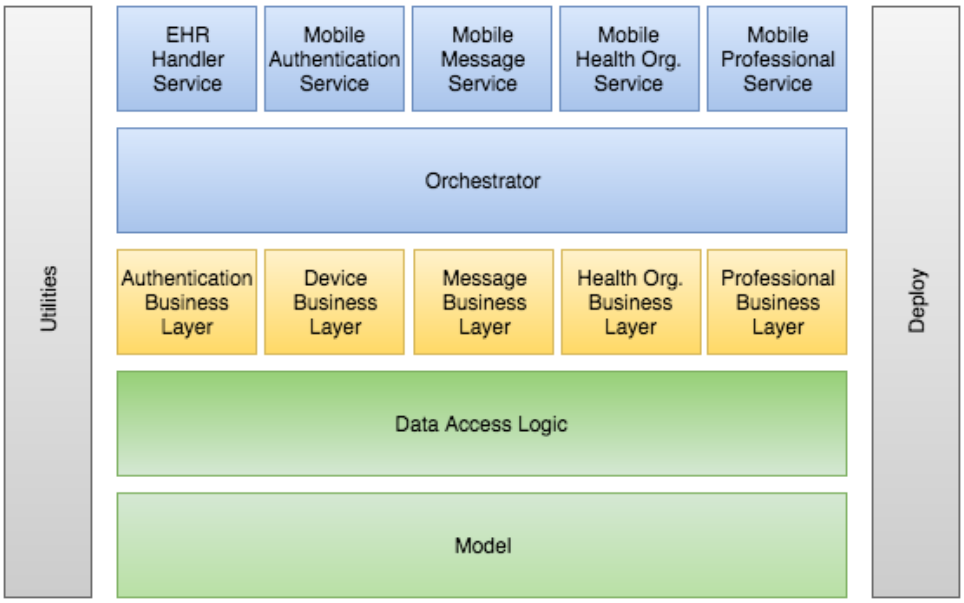


Figura 6.1: Organização da componente servidor

Serviços

Os serviços desenvolvidos visam dar resposta aos processos de negócio estabelecidos para este sistema, ou seja, a gestão de pedidos de autorização, de organizações e de profissionais de saúde (camada superior da Figura 6.1). Os *service contracts*, com respetivos métodos, e os *data contracts* criados estão disponibilizados nos Apêndices

E.1 e E.2. A transmissão de mensagens com a aplicação móvel é feita com o protocolo HTTPS, com recursos representados em JSON, e utiliza o estilo arquitetural REST.

Para a criação de serviços REST com WCF foi necessário recorrer um novo tipo de *binding* (*WebHttpBinding*) para não utilizar, por omissão, envelopes SOAP [76], como é exemplificado no trecho seguinte para o caso da gestão de pedidos de autorização.

```
<service name="MobileMessageHandlerService">
    <endpoint address=""
    contract="IMobileMessageHandlerService"
    binding="webHttpBinding"
    bindingConfiguration="RestBinding"
    behaviorConfiguration="restfulBehavior" />
</service>
```

Os serviços utilizam *Business Data Objects* (Apêndice E.3) no transporte de informação com a camada subjacente: *Orchestrator*. É também neste sub-projeto que está implementado o serviço que permite a *Basic Authentication*, comparando as credenciais recebidas com as que estão na base dados. A configuração no *web.xml* é dada a conhecer no código seguinte:

```
<behavior name="ServiceBehavior">
    <serviceMetadata
    httpGetEnabled="false" httpsGetEnabled="true" />
    <serviceDebug includeExceptionDetailInFaults="false" />
    <serviceAuthorization
    serviceAuthorizationManagerType="ServiceAuthenticator,
AutorizadorLibrary"/>
    </behavior>
</serviceBehaviors>
```

Orchestrator

O sub-projeto esquematizado nesta camada, como o nome indica, trata de orquestrar os mecanismos necessários para o fluxo de cada operação. Na prática, é alimentado pela informação proveniente dos serviços (*Business Data Objects*) e faz uso das camadas subjacentes através do contacto com cada um dos módulos da camada de *Business Logic*. As respostas geradas ou são devolvidas ao serviço que deu início ao fluxo ou são encaminhadas para o serviço de *Push Notifications*, a ser analisado na Subsecção 6.1.1 deste relatório.

A Figura 6.2 ilustra como o *Orchestrator* harmoniza uma das funcionalidades mais importantes: enviar um pedido desde um EHR para a aplicação móvel.
No que respeita a mensagens/pedidos de autorização, além da funcionalidade esquematizada na Figura 6.2, são implementadas outras não menos importantes com fluxo semelhante, como:

- processamento da resposta proveniente da aplicação móvel;
- atualização do estado de leitura;

- processamento do envio de *reminders* de resposta para pedidos pendentes;
- forçar a atualização das respostas dadas em dispositivos que ainda não as tenham.

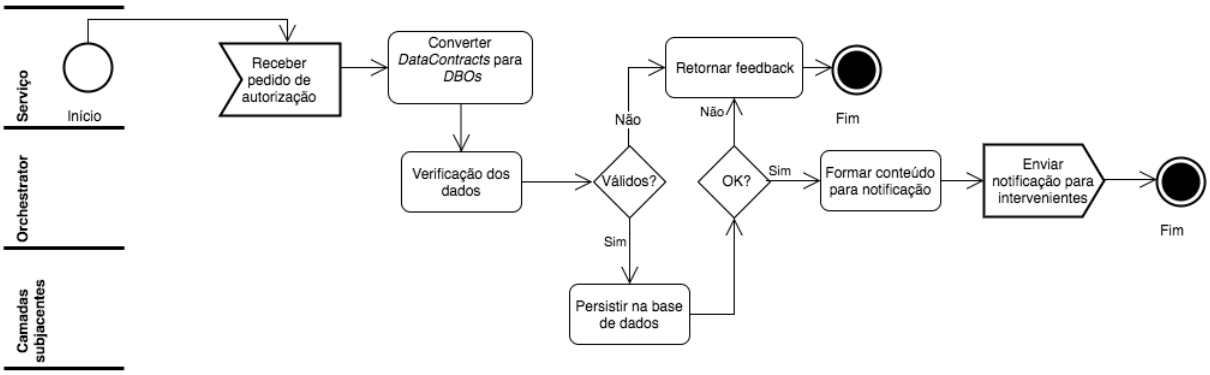


Figura 6.2: Diagrama de atividade do envio de um pedido de autorização

Nos métodos relacionados com **autenticação**, **gestão de profissionais** e de **organizações**, o *Orchestrator* encaminha a informação para a lógica presente nos respetivos módulos da camada seguinte.

Business Layer

Tal como na camada de serviços, existe uma separação de responsabilidades em módulos, de forma a dar resposta aos processos de negócio. O código que compõe os módulos desta camada realiza atividades de negócio e retorna os resultados necessários para a camada sobrejacente. Porém, a maioria dos métodos desenvolvidos não necessita de uma lógica complexa, sendo alimentados pelos resultados das *queries* relacionadas com as camadas de acesso a dados. Só no módulo de autenticação, com os métodos de registo e *login* de um profissional, é que existe a necessidade de mais operações, como gerir os dispositivos móveis associados a cada utilizador após verificar a existência de registos validados.

Data Access Logic e *Model*

A camada de acesso a dados realiza as *queries* necessárias para atualizar a base de dados. Além de um repositório genérico, foi criado um repositório, com respetivas classes e interfaces, para cada entidade (**organização de saúde**, **mensagem**, **notificação**, e **profissional**), disponibilizado no Apêndice E.4, cuja interação é ilustrada na Figura 6.3.
Isto permite que a camada lógica sobrejacente aceda aos dados sem se preocupar como se processa a consulta à base de dados que, neste caso, foi criada a partir das classes C# (*Code-First* do *Entity Framework*).

Figura 6.3: Esquema da comunicação entre camadas com o *Repository Pattern* - fonte [77]

Utilities e *Deploy*

O sub-projeto *Deploy* permite unir todos os outros projetos, disponibilizando para a posterior configuração no IIS. Dentro do sub-projeto *Utilities* encontram-se:

- *resources* para futuro suporte multi-idioma[1];
- definição de enumerações (Tabela 6.1);
- funções para o *hash* das *passwords* dos utilizadores.

Urgency Status		***AnswerStatus***	
Significado	**Valor na BD**	**Significado**	**Valor na BD**
Low	0	*Rejected*	-1
Normal	1	*NoAnswer*	0
Critical	2	*Accepted*	1

Tabela 6.1: Enumerações

Relativamente ao último ponto, a escolha por uma função de *hash*, ao invés de encriptar, justifica-se por ser uma função irreversível, não implicando preocupação com o armazenamento da chave de encriptação para armazenar informação relativa à *password*. Com base em [78], o algoritmo escolhido foi o *BCrypt*[2] (utiliza uma variante do algoritmo de encriptação *Blowfish*[3]), pois é de fácil implementação, permite evitar o armazenamento do *salt*[4] em separado e não foi feito para ser rápido, como, por exemplo, algoritmos mais antigos (p. ex. SHA-1, SHA-2), preocupados em processar muita informação em pouco tempo. Sempre que for feita uma verificação das credenciais de autenticação, é aplicada a função de *hash* à *password* fornecida, comparando o resultado com o persistido na base de dados.

6.1.1 *Push Notifications*

A MedicineOne já implementa *push notifications* através da biblioteca *open-source PushSharp*. De forma a adaptar às necessidades deste projeto, foi necessário, em primeiro lugar, criar uma conta de Apple *Develeoper* e criar certificados .p12 para o

[1] Para o conteúdo *ipsis verbis* das notificações
[2] `http://bcrypt.sourceforge.net/`
[3] `https://www.schneier.com/academic/blowfish/`
[4] `https://pt.wikipedia.org/wiki/Sal_(criptografia)`

APNS [79]. Foi disponibilizado ao estagiário um método específico, invocado no *Orchestrator*, para que a aplicação desenvolvida ficasse habilitada a enviar notificações, sendo necessário passar como parâmetro:

- ***AppId*:** identifica a aplicação no APNS;
- ***deviceToken*:** identifica o dispositivo móvel no APNS;
- ***payload*:** em formato JSON com o conteúdo propriamente dito da notificação para ser processado na aplicação móvel;
- ***stringAlert*:** com o texto, *ipsis verbis*, da mensagem a aparecer no centro de notificações do dispositivo.

```
[
    "aps" : {
        "alert" : "Recebido um novo pedido de autorização",
        "sound" : "default.aiff",
        "content-available": 1
    },
    "data" : {...}
]
```

O trecho de código anterior dá a conhecer o *payload* de uma notificação. As *keys* presentes no dicionário JSON significam:

- **aps:** nome do dicionário que contém as propriedades da notificação;
- **alert:** mensagem de alerta, *ipsis verbis*, que aparece ao utilizador;
- **sound:** som que acompanha a notificação;
- **content-available:** permite trabalhar com notificações silenciosas;
- **data:** conteúdo da notificação a tratar na aplicação móvel, previamente formatado no sub-projeto *Orchestrator*.

Apesar de não ser recomendável confiar nas *push notifications* da Apple para o envio de informação importante[5], considerou-se, ainda assim, uma tecnologia compatível, tendo em conta a terceira restrição de negócio e os pressupostos apresentados na Subsecção 5.1. Recorde-se que é por este mecanismo que a aplicação móvel recebe: **pedidos de autorização**, **respostas**, **lembretes** e **relatórios de leitura e de envio**.

[5]Por não garantir a entrega em alguns cenários

6.1.2 Cliente EHR

Esta prova de conceito ainda não integra com o *software* da MedicineOne. Como tal, o comportamento de um cliente EHR é simulado através da ferramenta **WCF Test Client** que permite consumir serviços WCF. O cenário utilizado para testar o serviço correspondente ao EHR foi sempre dentro do *Visual Studio*.

A Figura 6.4 ilustra o exemplo para o registo de uma nova organização de saúde/EHR. O Apêndice E.5 dá a conhecer o XML correspondente no envelope SOAP e os outros cenários disponíveis.

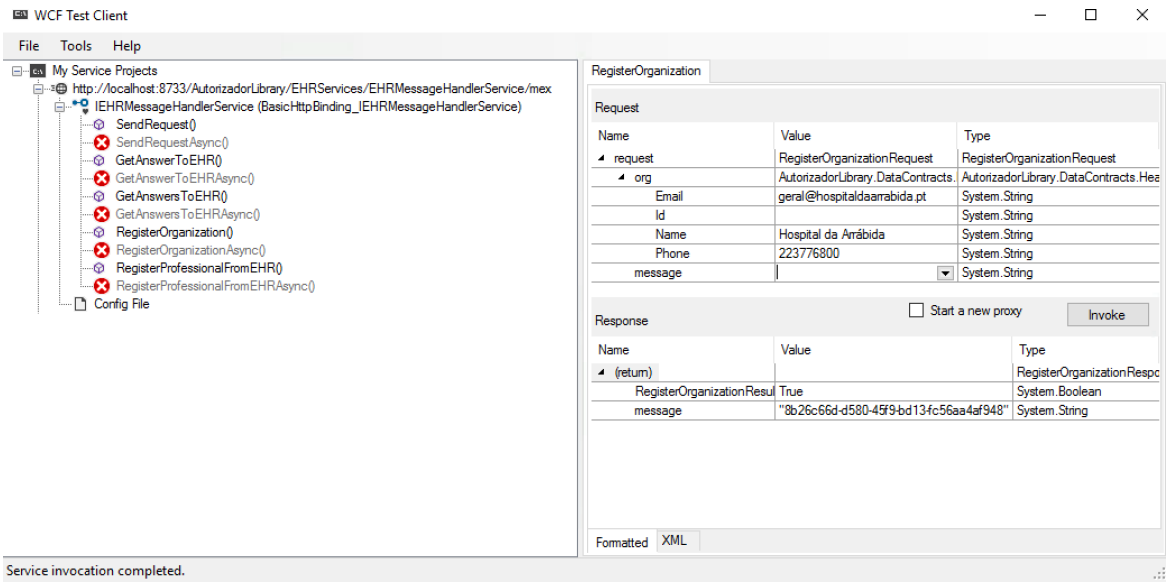


Figura 6.4: Cliente EHR – registo de unidade de saúde

No caso da Figura 6.4 é dado o email, nome e telefone da organização de saúde e é retornado o respetivo *Id* a utilizar nas futuras operações.

6.2 Aplicação móvel

Relativamente à aplicação móvel, desenvolveu-se uma aplicação nativa, onde foram criados os módulos necessários para adicionar as funcionalidades resultantes do levantamento de requisitos funcionais (Subsecção 4.4) após a fase de prototipagem (Apêndice C.2). Neste ponto são dadas a conhecer as bibliotecas utilizadas, a estrutura desenvolvida e as vistas que ilustram o fluxo das funcionalidades.

6.2.1 Bibliotecas utilizadas

Foram utilizadas algumas bibliotecas com o intuito de agilizar o desenvolvimento de algumas funcionalidades. As preferências apresentadas na Tabela 6.2 tiveram em consideração, sempre que possível, as escolhas já implementadas nas aplicações da MedicineOne.

A gestão das bibliotecas é feita com recurso ao CocoaPods[6], um gestor de dependên-

[6] `https://cocoapods.org/`

cias para projetos Cocoa[7], com mais de dezoito mil bibliotecas. Esta opção acrescenta vantagens como:

- facilidade em encontrar e incluir as bibliotecas no projeto;
- atualização para novas versões disponíveis.

Biblioteca	**Versão**	**Propósito**
SSKeychain	1.3	Um *wrapper* simples para aceder ao sistema de gestão de *passwords* da Apple: *Keychain*
ViewDeck	2.4	Utilizado para o menu principal, que surge na lateral esquerda da aplicação, sempre que solicitado
BRYXBanner	0.5	Permite gerir as notificações com um *dropdown banner* quando a aplicação está em modo ativo
IQKeyboardManagerSwift	4.0	Evita que o teclado deslize para cima de campos editáveis sem necessitar de esforço adicional

Tabela 6.2: Bibliotecas incluídas na aplicação móvel

A adição de novas bibliotecas foi sempre ponderada, pois, apesar de oferecer uma vantagem no uso de algumas APIs, tem inerente uma adição de código externo para o projeto. Assim, a escolha teve em consideração a comunidade de suporte[8] e o valor adicionado:

- **SSKeychain:** simplifica o acesso à *API* do **Keychain** que é bastante antiga e difícil de gerir, aumentando as probabilidades de erro;
- **ViewDeck:** não existe, nativamente, uma gestão de um menu lateral que se torne visível com um deslizar de dedos horizontal;
- **BRYXBanner:** auxilia a visualização das notificações como estas aparecem no sistema nativo;
- **IQKeyboardManagerSwift:** facilita a gestão dos teclados a utilizar nos diferentes campos de texto.

6.2.2 Camada de dados e gestão de *passwords*

Camada de dados

O **modelo de dados da aplicação** foi implementado com a criação das tabelas que podem ser consultadas, com maior detalhe, na Subsecção 5.4.2. Desta forma, o

[7] `https://en.wikipedia.org/wiki/Cocoa_(API)`
[8] Para evitar a possível falta de suporte ao código adicionado

dispositivo móvel tem persistência da informação associada ao utilizador e aos pedidos de autorização, incluindo os restantes atores do sistema.

Todas as tabelas são sincronizadas com as correspondentes no servidor, com exceção para a tabela *Patient* que apenas é persistida localmente na aplicação móvel. Sempre que um utilizador regista um novo dispositivo, apenas é sincronizada a sua informação pessoal, uma vez que com o cenário atual, onde servidor não guarda o conteúdo dos pedidos na integra, não haveria ganho em carregar informação incompleta. O histórico dos pedidos de autorização feitos e/ou respondidos é, portanto, apenas persistido aquando da sua ocorrência nos dispositivos envolvidos. Se por algum motivo um dispositivo não tiver pedidos com as respostas devidamente sincronizadas, pode força-la através do botão assinalado na Figura 6.5, localizado na lateral direita da *navigation bar*.

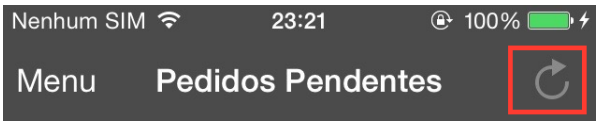


Figura 6.5: Botão para *refresh* de respostas pendentes (lateral direita da *navigation bar*)

Com o objetivo de trabalhar com a *framework* de persistência de dados – **Core Data** – foi utilizado um *bundle* de métodos genéricos (classe *CoreDataMethods*) já existentes nas aplicações da MedicineOne. A classe responsável pelas atualizações das tabelas é a classe *FeedCoreData*, criada pelo estagiário.

Os métodos fornecidos que auxiliam a classe *FeedCoreData* ajudam na adição e obtenção de informação e sua persistência.

Gestão de *passwords*

No que concerne a esta matéria, é utilizado o sistema de gestão de *passwords* da Apple: **Keychain**[9]. Para a implementação foi utilizada a biblioteca *SSKeychain*, de onde se destacam dois métodos :

```
SSKeychain.setPassword(password: ...,
                       forService: ...,
                       account: ...)
```

Para **guardar** uma *password* é necessário defini-la e identificar o serviço (*Bundle Identifier*, no caso: *net.medicineone.AutherasiOS*) e a conta (foi escolhido o *email* do utilizador) a que a *password* está associada para posterior identificação. A função retorna um de dois resultados possíveis:

- **verdadeiro**, se guardou com sucesso;
- **falso**, no caso contrário.

[9] `https://en.wikipedia.org/wiki/Keychain_(software)`

```
SSKeychain.passwordForService(serviceName:..., account:...)
```

Para **carregar** uma *password* é apenas necessário identificar o serviço e conta fornecidos aquando da definição. Em comparação com o método anterior, este método retorna um tipo de resultado diferente, dentro de duas alternativas:

- a ***password*** correspondente se identificou o serviço e conta definidos;
- **nulo**, caso o *Keychain* não tenha nenhuma *password* associada aos parâmetros dados.

Apesar de não se verificar nesta prova de conceito, é imperativo que a informação seja encriptada antes de ser guardada com o Keychain, à semelhança do que já foi descrito na componente servidor.

6.2.3 Tratamento de dados

O **acesso aos serviços** implementados pelo estagiário do lado do servidor é assegurado com a implementação de duas classes:

- *ServiceAccessRequest*: contém métodos para a criação dos URLs[10] e dos respetivos *requests*, utilizando os cabeçalhos do HTTPS para definir o método POST e para incluir a informação a transitar no *body*;
- *ServiceAccess*: reúne métodos para comunicar com os serviços do lado do servidor. É utilizada a classe nativa *NSURLSession* que disponibiliza métodos que permitem fazer *download*, em *background*, dos dados que vêm como resposta aos pedidos efetuados.

Os pedidos feitos a partir da aplicação móvel estão relacionados com: registo, *login*, resposta a um pedido de autorização, envio de lembretes e de confirmações de leitura e resposta e sincronização de entidades (classe *Update*). Neste ponto, fica a nota crítica da necessidade de criar um método mais genérico que possa dar resposta às necessidades de todos os pedidos.

Os **dados** que resultam dos pedidos feitos a partir da aplicação estão no formato *NSData*, sendo serializados para JSON na estrutura de dados idealizada, antes de serem fornecidos para a camada de dados. Estas operações são garantidas com os respetivos métodos da classe *JSONHandler*, também implementada pelo estagiário.

6.2.4 *TouchID*

A fim de utilizar a API do sensor biométrico capaz de ler impressões digitais - *TouchId* da Apple - foi utilizada a *framework* ***Local Authentication***. Esta API está disponível a partir do iOS 8 e tem apenas uma classe nativa com o nome *LAContext*. No âmbito do projeto, este mecanismo é utilizado para a funcionalidade de *login*. É

[10]Baseados no endereço do serviço e no nome do método alvo

possível cancelar a autenticação biométrica e recorrer à manual, para casos de problemas relacionados com o leitor ou apenas por preferência do utilizador.

Em primeiro lugar, verifica-se a disponibilidade do sensor biométrico no dispositivo: se não estiver disponível[11] o utilizador procede para uma autenticação manual com email e *password*; caso contrário, procede-se à leitura da impressão digital.

De seguida, caso a leitura corresponda ao registo que está guardado no dispositivo, é feita a autenticação do utilizador sem a necessidade de inserir as suas credenciais. Todavia, na prática, a aplicação trata de as enviar para o servidor para confrontar com as persistidas na base dados do sistema agregador. No caso da leitura não apresentar correspondência, é feita a avaliação do código de erro, em comparação com os fornecidos pela classe *LAError*. A Figura 6.6 ilustra a modo como o utilizador é convidado a interagir com o sensor biométrico.

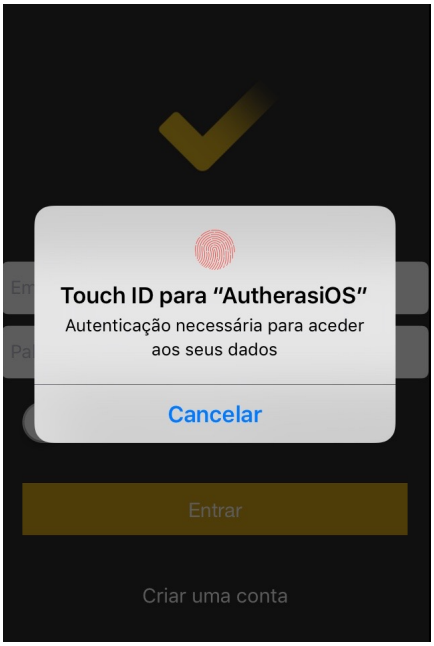


Figura 6.6: Interação com *TouchID*

As versões 4S e 5 do iPhone estão habilitadas a correr esta aplicação mas não suportam o *TouchID*.

6.2.5 Organização do código

Nesta subsecção é resumido o critério em como o código está organizado para dar resposta às funcionalidades pretendidas. Existem diversas diretorias, onde se encontram os ficheiros Swift que compõem as classes desenvolvidas. O Apêndice E.6 dá a conhecer apenas as principais, dada a sua extensão.

Cada uma das funcionalidades relacionadas com **registo**, ***login***, **Menu**, **Definições** e com os **pedidos pendentes** e pertencentes ao **histórico** têm o respetivo código separado de forma a separar responsabilidades, resultando em:

[11] Sem suporte ou inativo

- ***ViewController<NomeFuncionalidade>*:** responsável pelo mapeamento direto com o que ocorre na camada *View*, mais próxima do utilizador;
- ***ViewController<NomeFuncionalidade>Extension*[12]:** permite adicionar funcionalidades a classes criadas (no caso, as *ViewControllers*). Neste projeto, houve um duplo objetivo com as extensões: adicionar a lógica necessária para verificação de dados e servir de porta de entrada para a comunicação com as classes do módulo *Management*; criar suporte para os métodos inerentes à construção de *ViewControllers*;
- ***<NomeFuncionalidade>Management*:** compõe o módulo *Management*, responsável pela comunicação com os serviços e com a camada de dados. Deste modo, esta responsabilidade é retirada dos *ViewControllers*, tornando-os mais leves e fáceis de testar.

De referir que cada ecrã da aplicação é representado com o respetivo *ViewController*, ao qual serve de suporte à estrutura apresentada na enumeração anterior. Todas as classes principais recorrem ao principio de herança com classes nativas. Por exemplo, com exceção do *Login* e dos pedidos em detalhe, todas as funcionalidades têm *ViewControllers* baseados na classe *UITableViewController* e/ou *UITableViewCell* por se apresentarem ao utilizador como tabelas.

6.2.6 Estrutura de navegação

A navegação é feita através dos *ViewControllers*[13] implementados. Com base na Figura 6.7, encontra-se, no topo, a classe *ViewControllerInitial*, herdeira da classe *IIViewDeckController* da biblioteca *ViewDeck*, já adotada pela MedicineOne.

Apesar de haver a possibilidade de gerir três *ViewControllers*[14], apenas se faz uso da lateral esquerda (*ViewControllerMenu*), para o *Menu*[15], e da central, para o conteúdo principal. Neste último caso, a vista central interage com *NavigationControllers*, um tipo de controlador específico para gerir a navegação do conteúdo hierarquicamente. A estrutura de um *NavigationController* é em forma de pilha, adicionando e removendo os *ViewControllers* pelos quais o utilizador navega.

Quando o utilizador acede a qualquer um dos módulos (pedidos pendentes[16], histórico de pedidos e definições), é-lhe apresentado o *ViewController* correspondente, adicionando à pilha os *ViewControllers* seguintes para que seja possível acrescentar animação às transições nas diferentes vistas. Com a Figura 6.7 pretende-se oferecer um ponto de vista simplificado da estrutura de navegação.

[12] `https://www.hackingwithswift.com/read/24/2/creating-a-swift-extension`
[13] Base da estrutura interna da aplicação, gerindo a apresentação do conteúdo
[14] Duas vistas laterais e uma central
[15] Permite listar os módulos pelos quais o utilizador navega
[16] Por omissão, começa com a listagem de pedidos pendentes

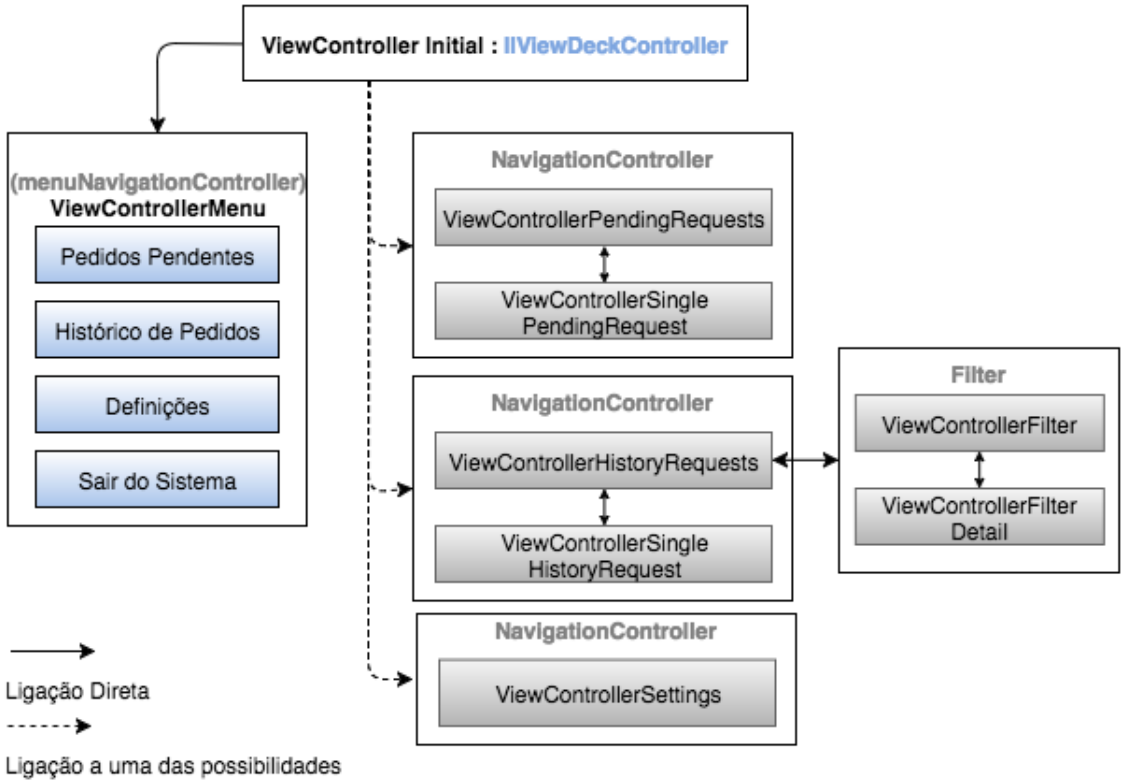


Figura 6.7: Estrutura de navegação

6.2.7 *Push Notifications*

As notificações são uma das componentes mais importantes da aplicação móvel, pois é através deste mecanismo que chega a informação alusiva aos pedidos de autorização. Para o envio de notificações, é necessário um *deviceToken*[17] válido. Por conseguinte, o simulador presente no Xcode não se mostrou compatível, sendo necessário fazer uso de dispositivos reais[18].

Configurações iniciais

Sempre que a aplicação corre pela primeira vez, é necessário autorizar a receção de notificações para o dispositivo, podendo ser gerido através das configurações do sistema. Neste sentido, no que toca ao desenvolvimento da aplicação móvel, equivale ao uso de métodos nativos para requerer autorização ao utilizador, a fim de receber as notificações, e obter o *deviceToken* para a aplicação[19]. A Figura 6.8 ilustra o ecrã resultante.

Uma vez que este processo é de extrema importância[20], o utilizador deve ser alertado para a sensibilidade da operação. **Porém, este facto não é suficientemente óbvio nesta versão, ficando a recomendação da implementação de um alerta de aviso ou, preferencialmente, impedir o *login* se a permissão não tiver sido realizada**.

[17] Identificador do dispositivo no APNS
[18] No desenvolvimento deste projeto, recorreu-se a iPhone e iPad
[19] No caso de já ter sido atribuído, não cria um novo
[20] De outra forma, o utilizador não poderá ter acesso aos pedidos de autorização nesta versão

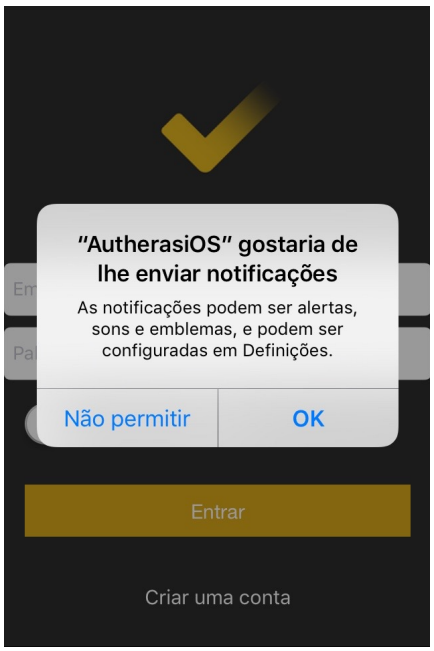


Figura 6.8: Autorização para a receção de notificações

Chegada de uma notificação

As notificações recebidas são, em primeiro, tratadas por métodos nativos, cujo tratamento é diferente, consoante o estado em que se encontra a aplicação (Tabela 6.3 [80]). Posteriormente, é feito um encaminhamento do seu conteúdo para a classe *PushHandler*[21] (excerto de código seguinte), criada para o gerir.

```
PushHandler.Handling(userInfo,
                     window: self.window!,
                     application: application)
```

A informação que é fornecida para a *PushHandler* indica o conteúdo da notificação (*userInfo*), a janela ativa (*window*) e o estado da aplicação (obtido a partir da classe nativa *UIApplication*). A variável *userInfo* é um dicionário JSON, cujo *payload* está definido na Subsecção 6.1.1, com quatro chaves possíveis dentro do parâmetro *data*:

- ***msg*:** para um pedido de autorização;
- ***answer*:** para a resposta a um pedido de autorização;
- ***read*:** recibo de confirmação de leitura de um pedido de autorização;
- ***anserReq*:** lembrete do emissor para que o recetor responda ao pedido.

Após avaliar a chave recebida, é feita a serialização da informação para o tipo de objeto correspondente e persistido na base de dados local. Posteriormente, a classe *NotificationHandler*[21] trata da gestão visual da aplicação, consoante o seu estado (Tabela 6.3). Recorde-se que no estado ativo, é usada a biblioteca *BRYXBanner* para

[21] Criada pelo estagiário

auxiliar a construção de *dropdown banners* com um resumo da notificação recebida.

As Figuras 6.9, 6.10 e 6.11 ilustram as diferentes formas de exibir as notificações. O Apêndice E.7 contem figuras que ilustram todos os tipos de notificações que surgem quando a aplicação está em *foreground*.

Estado do dispositivo	**Estado da aplicação**	**Notificação**
Desligado		Notificações mais recentes ficam em fila e podem ser entregues quando o dispositivo for ligado
Ligado	*Background*	Notificação é exibida ao utilizador, com base nas preferências definidas pelo utilizador nas definições do sistema
Ligado	*Foreground*	Notificação é recebida pela aplicação mas não é exibida automaticamente ao utilizador. É necessário implementar o código que gere o comportamento

Tabela 6.3: Notificações e os diferentes estados da aplicação

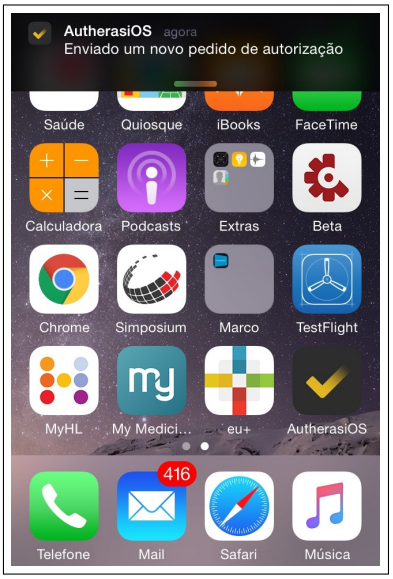


Figura 6.9: Alerta de notificação no *home screen* do iOS

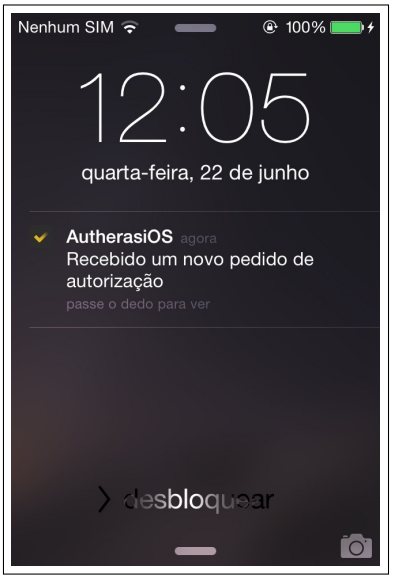


Figura 6.10: Notificação na central de notificações do iOS

Figura 6.11: Notificação exibida com o *Autheras* em *foreground*

6.2.8 Interface

A interface gráfica foi construída com recurso ao *Interface Builder* do Xcode, através de um *storyboard* que esquematiza o fluxo entre ecrãs (controlados por *NavigationControllers* e *ViewControllers*). O maiores desafios encontrados nesta secção

relacionaram-se com o manuseamento do *AutoLayout*[22] e com a preservação dos valores entre *ViewControllers*. Neste caso, a utilização de protocolos mostrou-se vantajosa. Um protocolo especifica um conjunto de comportamentos que uma dada classe deve implementar, sem, contudo, fornecer a implementação desses comportamentos[23]. Um exemplo prático é o caso da escolha de alguns parâmetros de filtragem, onde foi necessário criar um protocolo com um método que permitisse devolver o valor relativo ao parâmetro da i-ésima célula de uma *UITableViewController*, como se pode analisar no seguinte excerto de código:

```
protocol FilterProtocol: class {
    func sendValuesToPreviousVC(row: Int, object: AnyObject)
}
```

Para que uma classe possa implementar os métodos de um protocolo, esta deve fazer uso das propriedades de herança. De seguida, passa-se a apresentar os ecrãs que compõe o fluxo das principais funcionalidades.

Login e Registo

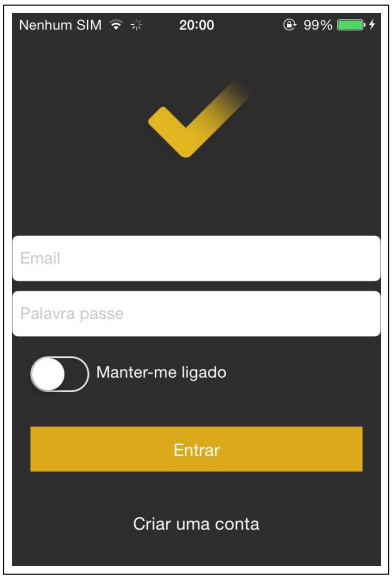


Figura 6.12: Ecrã inicial de *login*

Figura 6.13: Ecrã de validação de dados de *login*

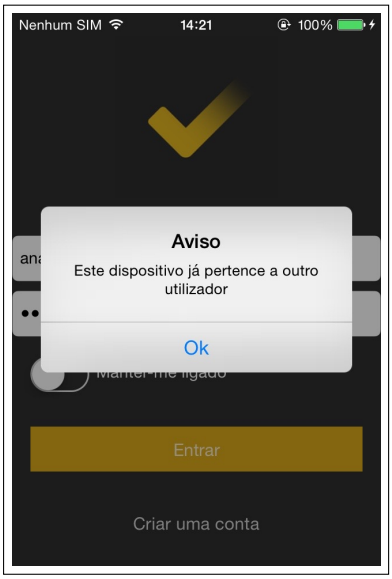


Figura 6.14: Ecrã de registo

O ecrã de *login* permite encaminhar para o ecrã de registo e para o interior do sistema, quando são introduzidas credenciais válidas. Também informa ao utilizador dos possíveis erros, como por exemplo, dados incorretos e dispositivo indisponível. Relativamente ao ecrã de registo, recomenda-se[24] que o país do utilizador seja exibido em separado e em primeiro lugar, de forma a que informação pendente[25] possa ter

[22] `https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/AutolayoutPG/`

[23] Fonte: `http://www.aidanf.net/learn-swift/protocols`

[24] Para a próxima versão

[25] Por exemplo, números civil e fiscal, telefone

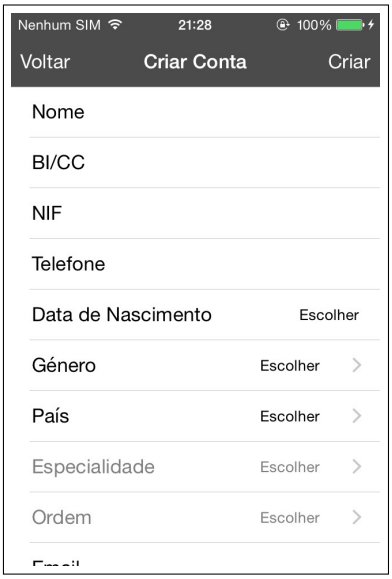


Figura 6.15: Ecrã de registo

Figura 6.16: Ecrã de validação de dados de registo

Figura 6.17: Ecrã de confirmação de registo

máscaras de acordo com os formatos de cada nação. Por outro lado, informação como a especialidade e a ordem profissional devem apenas surgir aos profissionais que efetivamente as possam ter. De notar que a funcionalidade "Manter-me ligado" não se encontra implementada, apesar de surgir no ecrã de *login*.

Menu lateral

Figura 6.18: Menu lateral

Para o caso de existirem novas funcionalidades no sistema *Autheras*, podem ser adicionadas mais opções ao Menu, facilmente e sem mais linhas de código, através da sua referência no ficheiro *property list*[26] (.plist) criado para o efeito.

Pedidos pendentes

A Figura 6.19 ilustra a listagem de pedidos pendentes, com destaque para os emissores (distinguidos com "De") e recetores (distinguidos com "Para"). As diferentes cores no nome dos intervenientes indicam o grau de urgência do pedido (verde: baixo; amarelo: normal; vermelho: crítico). Os pedidos em detalhe[27] (Figuras 6.20 e 6.21)

[26] `https://en.wikipedia.org/wiki/Property_list`
[27] Na parte do superior do pedido é identificado o paciente em causa

Figura 6.19: Ecrã de listagem de pedidos pendentes

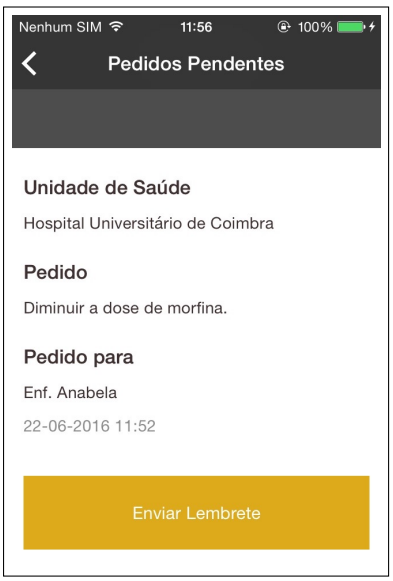


Figura 6.20: Ecrã de um emissor (detalhe)

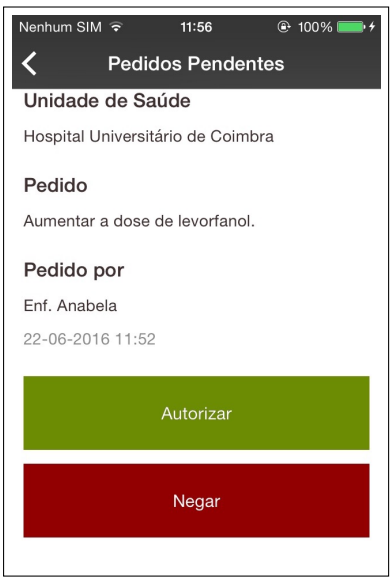


Figura 6.21: Ecrã de um recetor (detalhe)

têm um aspeto diferente para o emissor (disponibiliza o botão para enviar lembretes) e para o recetor (exibe os botões para autorizar ou negar o pedido de autorização).

Histórico de pedidos

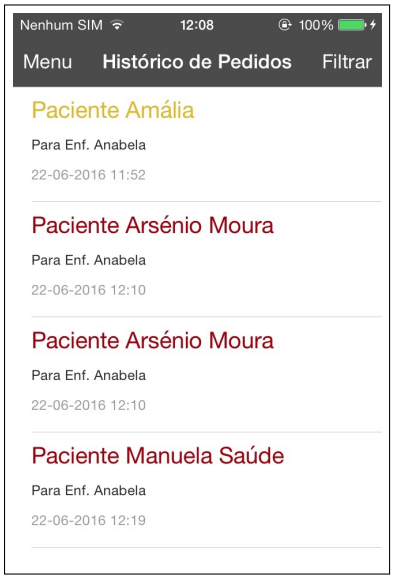


Figura 6.22: Ecrã de listagem de histórico

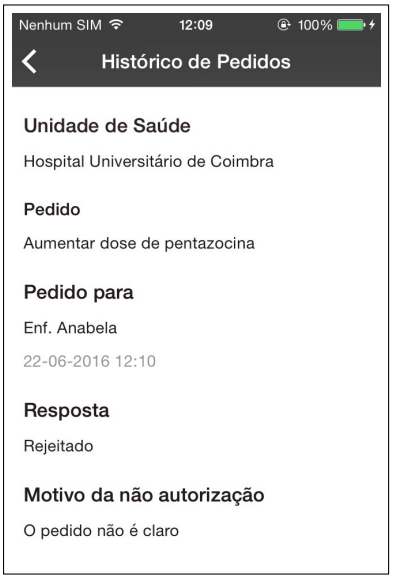


Figura 6.23: Ecrã de um pedido rejeitado

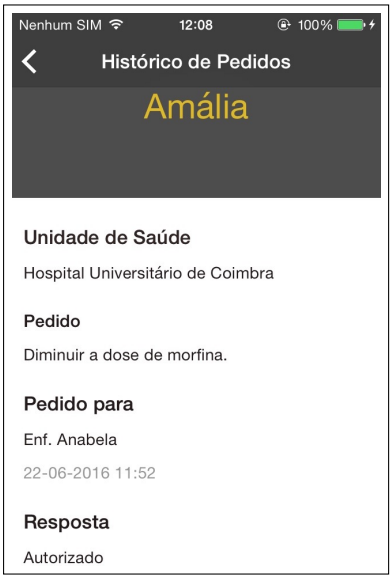


Figura 6.24: Ecrã de um pedido aceite

O ecrã da operação **Filtrar** é ilustrado no Apêndice E.7.

Definições

Nesta versão, as Definições (Apêndice E.7) apenas exibem as informações pessoais do utilizador. Os únicos campos editáveis são os correspondentes à *password*. Neste

caso, apenas seria enviado um pedido para o servidor quando os referidos campos estivessem preenchidos e com valores iguais.

Capítulo 7

Testes

Neste capítulo são dados a conhecer os testes efetuados, assim como alguns resultados obtidos. Mais detalhes acerca desta temática podem ser encontrados nos Apêndices F e G.

Para esta prova de conceito, o maior requisito ao nível de testes visa garantir que o fluxo de cada funcionalidade decorre como o pretendido nos dispositivos compatíveis. Esta secção sairia mais enriquecida com testes automatizados e direcionados a cada uma das componentes[1], contudo, pelo tempo disponível e pela inexperiência do estagiário, não foi possível acrescentar a este projeto.

7.1 Testes de aceitação

Com os testes de aceitação, pretende-se verificar se os requisitos funcionais são cumpridos e se as componentes (servidor e aplicação) que integram cada um deles funcionam como o esperado. A validação dos requisitos funcionais foi feita em duas fases:

- após o desenvolvimento de cada funcionalidade;
- após o desenvolvimento de todo o sistema.

Assim, pretende-se confirmar se as funcionalidades existem e obedecem ao comportamento esperado (Tabela 7.1) através de testes não-automatizados. O Apêndice F.1 indica, para cada funcionalidade, a descrição do teste, a *User Story*, o resultado obtido. Os casos em que o comportamento esperado foi considerado "Não aplicável" justifica-se pelo facto de, por não terem sido aplicados no sistema, não se torna possível aferir o seu comportamento

[1] P. ex., testes unitários

Funcionalidade	**Implementado**	**Comportamento esperado**
Inscrição de um novo utilizador	Sim	Sim
Autenticação de um utilizador	Sim	Sim
Permitir a recuperação da *password*	Não	Não aplicável
Aceder à aplicação através de uma notificação, destacando o pedido	Sim	Sim
Listar autorizações pendentes	Sim	Sim
Exibir todos os campos dos pedidos de autorização	Sim	Sim
Consultar o histórico de pedidos	Sim	Sim
Filtrar o histórico de pedidos	Sim	Sim
Menu principal	Sim	Sim
Receber notificações no *smart watch*	Não	Não aplicável
Autorizar um pedido	Sim	Sim
Rejeitar um pedido, justificando-o	Sim	Sim
Enviar *reminders* relativos aos pedidos sem resposta	Sim	Sim
Receber a confirmação de leitura do pedido de autorização	Sim	Sim
Receber resposta ao pedido de autorização	Sim	Sim

Tabela 7.1: Validação dos requisitos funcionais

7.2 Testes de compatibilidade

Considerando que a aplicação móvel *Autheras* deve ser compatível com todas as versões do iPhone − a partir da versão 4S − e com as duas últimas *major versions* do iOS[2] − 8 e 9 − foi necessário testar o seu funcionamento em dispositivos diferentes. Uma vez que as notificações apenas funcionam em dispositivos reais, somente o *layout* foi testado com recurso ao simulador do Xcode. Foram realizados testes de aceitação com os dois dispositivos disponibilizados para teste - iPhone 4S com iOS8 e iPad mini 3[3] com iOS9 - com o objetivo de verificar o correto funcionamento das funcionalidades, cujos resultados são apresentados na Tabela 7.2.

Dispositivo	Sistema Operativo	*TouchId*	Resultado
iPhone 4S	iOS 8.4	Não	Passou
iPad mini 3	iOS 9.2.1	Sim	Passou

Tabela 7.2: Testes de compatibilidade

[2] À data de escrita deste relatório

[3] Único dispositivo disponível com *TouchID*

7.3 Testes de usabilidade

Como já referido no Capítulo 4, foram utilizados protótipos com o objetivo de realizar **testes às funcionalidades e interfaces** idealizadas para a solução da aplicação móvel. Os protótipos utilizados para os testes com utilizadores classificam-se como **funcionais**, de **média fidelidade**, com recurso a um sistema computacional para simular o seu comportamento, **horizontais** (com várias tarefas implementadas) e com **baixa profundidade** (apenas com respostas estáticas às ações dos utilizadores) [81].

A prototipagem nesta fase permite recolher medidas de usabilidade, avaliar se o *feedback* ao utilizador é o indicado e promove a deteção de problemas de usabilidade, minimizando a sua propagação para contexto de produção. Este ponto foi considerado bastante positivo e inovador pela empresa, na medida em que não existia o hábito de proceder a este tipo de testes, ao nível do protótipo em forma de *mockup*.

Na fase de testes, já na parte final do segundo semestre, executou-se uma demonstração de testes com base em heurísticas de Nielsen [82], que não envolveram utilizadores.

Medidas gerais de usabilidade

Segundo a norma ISO para a usabilidade, ISO 9241-11[4], as medidas de usabilidade devem abranger a **eficácia** (qualidade com que o utilizador atinge os objetivos), **eficiência** (recursos gastos para atingir os objetivos) e **satisfação** (reação subjetiva dos utilizadores à utilização do sistema). Como tal, para estudar a **usabilidade global** da aplicação, foram utilizadas as medidas da Tabela presente no Apêndice G.1 para cada uma das seis tarefas realizadas pelos utilizadores em dois testes. Recomenda-se a análise dos Apêndices G.4 e G.5 para uma análise mais detalhada aos resultados obtidos.

7.3.1 Testes com utilizadores

Durante o primeiro semestre, foram realizadas **duas fases de testes com utilizadores**, com observação direta, de forma a medir o desempenho e satisfação ao realizarem as tarefas propostas (Apêndice G.3), assim como identificar possíveis erros de usabilidade. Os utilizadores desta aplicação são profissionais de saúde, essencialmente médicos e enfermeiros. Seria interessante obter uma amostra destes dois conjuntos, preferencialmente com diferentes faixas etárias. Não sendo possível obter tais utilizadores nesta fase, foi feito o convite a um grupo de 5 pessoas (colegas de trabalho) com o objetivo de representarem um grupo heterogéneo de utilizadores − Apêndice G.2.

A primeira fase direcionava-se a encontrar problemas de usabilidade na interface e proceder ao seu melhoramento. Já a segunda fase foi realizado com a finalidade de certificar se os problemas encontrados no primeiro foram corrigidos com sucesso.

[4] `http://www.usabilitynet.org/tools/r_international.htm#9241-11`

Resultados

Ainda que frágeis, os testes de usabilidade decorridos no primeiro semestre foram realizados com recurso a um computador com o *software* Flinto[5] a simular o *layout* do iPhone, permitindo a interação do utilizador através do rato. Apesar destes testes não terem envolvido um grande número de utilizadores, foi possível tirar várias conclusões que levaram a alterações ao protótipo.

Considerando a **primeira fase de testes**, com base nos resultados obtidos, pode concluir-se que a avaliação da usabilidade foi, no geral, bastante positiva, assim como a satisfação global com a aplicação. Os problemas identificados são considerados críticos apenas na Tarefa 3, dado que foi a tarefa com menor eficiência, com pior tempo médio (1º:03s:320ms) e a única onde existiram erros na execução, contribuindo para que apenas 3 dos 5 utilizadores a executassem com sucesso. Os problemas associados identificaram pouca clareza na disposição da informação com parâmetros de pesquisa do histórico de pedidos.

Após a **segunda fase de testes**, os utilizadores mostraram agrado com as alterações feitas na Tarefa 3, o que contribuiu para a melhoria observada nos resultados das medidas de satisfação e de eficácia. Uma vez que se estava perante os mesmos utilizadores, pode também concluir-se que existiu memorização da interface da aplicação, como comprova a melhoria das medidas de eficiência com os registos temporais significativamente menores em praticamente todas as tarefas.

7.3.2 Testes sem utilizadores

A realização de testes, sem envolver utilizadores, decorreu após o desenvolvimento da prova de conceito. Para tal, utilizou-se a avaliação das dez heurísticas de Nielsen, uma técnica de inspeção[6] desenvolvida por Jakob Nielsen, cuja implementação é considerada acessível, rápida e pouco dispendiosa [81, 83]. Segundo Nielsen, este tipo de teste deve ser executado por um conjunto de três a cinco peritos, de forma independente, a fim de poder encontrar cerca de 65% a 75% dos problemas de uma interface [81]. Contudo, dada a natureza do estágio, a avaliação foi feita apenas por uma pessoa, podendo descobrir cerca de 35% [81] do total dos problemas de usabilidade.

Os resultados (Apêndice G.6.4) foram recolhidos na execução das tarefas presentes no Apêndice G.6.1, com base numa amostra de vinte e cinco sub-heurísticas especificadas em [83], enquadradas nas heurísticas de Nielsen [82] (Apêndice G.6.3). De forma a priorizar a resolução dos problemas encontrados, é atribuído um grau de severidade [81, 82] (Apêndice G.6.2) a cada um deles.

Resultados

Foram diagnosticados cinco problemas na execução das oito tarefas, onde os mais críticos tornam prioritária a futura implementação de mais e melhores meios de resposta às ações do utilizador. A maioria incide na primeira heurística de Nielsen –

[5] https://www.flinto.com/

[6] Dada a sua natureza, este método não pode ser executado automaticamente [83]

visibilidade do estado do sistema − ao não providenciar respostas a todas as ações do utilizador.

7.4 Teste de carga à memória da aplicação

Este tipo de teste teve o objetivo de avaliar o desempenho de memória da aplicação, assim como a tentativa de encontrar possíveis fugas (*memory leaks*). Os testes foram efetuados com a ferramenta *Instruments*, disponível no Xcode, e percorreram todos os ecrãs da aplicação de modo a analisar o comportamento de cada um deles.

Em cada ecrã foram realizados vários testes às funcionalidades que lhes são inerentes, de forma a verificar:

1. se existiam padrões na utilização de memória, evitando que esta crescesse indefinidamente sem ser libertada;

2. se existia ocorrência de fugas de memória, isto é, alocação de objetos que já não estivessem referenciados.

De acordo com os resultados disponibilizados no Apêndice F.2, verificou-se a ocorrência de algumas fugas de memória que, pelo tempo disponível, não foi possível de resolver. Atendendo aos padrões existentes, concluiu-se que apesar de existir um aumento do consumo de memória nas mudanças de ecrã, esta é libertada, não correndo o risco de ser terminada pelo sistema por crescer indefinidamente. Idealmente, estes testes deveriam ter sido efetuados após o desenvolvimento de cada funcionalidade, a fim da sua análise poder acrescentar uma real mais valia neste projeto.

Capítulo 8

Considerações finais

Este capítulo apresenta uma abordagem geral sobre o período de estágio na empresa MedicineOne. Assim, será feita uma análise ao trabalho realizado, sem deixar de identificar os principais obstáculos que surgiram e como foram ultrapassados. O capítulo termina com as considerações finais sobre o futuro do sistema *Autheras*.

Trabalho realizado

O presente estágio foi pensando para o planeamento e desenvolvimento do sistema *Autheras* em dois semestres. O primeiro, dedicado ao planeamento, começou com o estudo das abordagens atuais, standards utilizados e das componentes que viriam a integrar o sistema desenvolvido. Incluiu, também, a análise de requisitos, onde foram levantadas e priorizadas as funcionalidades que deviam ser garantidas, assim como os atributos de qualidade e restrições inerentes. Esta fase terminou com o desenho da solução, onde se inclui a arquitetura planeada, e com a realização de testes de usabilidade aos *mockups* com utilizadores, procedimento não existente na empresa até então.

O segundo semestre foi mais orientado para o desenvolvimento das componentes que compõem o sistema *Autheras* − servidor e aplicação móvel. Começou com a consolidação da fase anterior, com a definição dos modelos de dados e posterior implementação das componentes planeadas no semestre anterior. Esta fase englobou, ainda, uma abordagem sobre aspetos de segurança antes da definição e elaboração de testes ao sistema.

Principais obstáculos

No decorrer deste estágio, foram tomadas decisões a fim de oferecer resposta ao cumprimento das funcionalidades e requisitos propostos. O facto de no final do primeiro semestre ter existido uma clarificação de restrições, onde o sistema agregador (servidor) não poderia preservar parte da informação nem estaria habilitado a iniciar contacto com os EHRs, contribuiu para que o sistema de notificações tivesse ganho um papel fundamental/chave no funcionamento do *Autheras*, afetando todo o fluxo da informação. O desenvolvimento acabou por refletir o respeito pelas restrições, contudo, considera-se que estas devem ser revistas com o objetivo de diminuir a necessidade de guardar tanta informação nos dispositivos móveis e permitir uma maior integração com diferentes EHRs.

A falta de experiência contribuiu para a verificação do primeiro risco, ou seja, um atraso no desenvolvimento das funcionalidades. Uma vez que esta prova de conceito tinha como principal objetivo a implementação das componentes do sistema, a fim de verificar o funcionamento do fluxo de informação dos pedidos de autorizações, essa foi a prioridade, contribuindo para que atributos também importantes, que acrescentariam qualidade, como segurança, estivessem menos presentes na implementação. Dada a dimensão do projeto, este atributo, assim como a usabilidade, poderiam originar outros projetos de forma a serem focados de uma forma mais madura.

Reflexão crítica

O principal objetivo deste estágio foi planear e desenvolver o sistema *Autheras*, capaz de dar resposta ao cenário de pedidos de autorização, através de uma prova de conceito constituída pelo sistema agregador e por um protótipo de uma aplicação móvel iOS. Deste modo, considera-se que o objetivo foi atingido, pois o resultado da planificação e desenvolvimento contempla as funcionalidades mais importantes, permitindo que seja continuado o seu estudo e melhoria para uma possível evolução para produto.

A dimensão do projeto podia ter dado azo à especialização de um atributo de qualidade[1], contudo, optou-se por tentar abordar um pouco de todos os aspetos que constituem a solução, sendo desafiante tentar perceber e implementar cada um deles em novas tecnologias para o estagiário. O ponto da dimensão, aliado à inexperiência, contribuiu, então, para que os atributos referidos tenham uma abordagem algo frágil.

Analisando os pontos identificados como mais-valias, considera-se que foram maioritariamente cumpridos, na medida em que a base da arquitetura e da interface da aplicação ficam feitas, foi especificada a informação a ser transacionada e tira-se partido do uso do sensor biométrico do iPhone. Dado que a aplicação móvel é nativa, a escolha do recém Swift não é um problema, porque além de ser considerado o futuro para o desenvolvimento para MacOS e iOS, existe uma boa interoperabilidade com outras linguagens e as *major changes* são bem documentadas pela Apple.

Trabalho futuro

Além da revisão das restrições, já referida neste Capítulo, o sistema *Autheras* deve ter um foco mais profundo em aspetos como segurança e usabilidade[2], apesar deste último ponto já ter relevância no presente relatório. Não obstante, a integração com um sistema EHR deve começar pelo MedicineOne 8, levando a que este tenha a criação da *feature* para tratamento dos seus pedidos de autorização, geridos pelo *Autheras*.

O requisito *Whishful* − GEN_US10 − destinado à receção de notificações no *Apple Watch*, não foi utilizado neste projeto pela sua baixa priorização e pouca relevância no mercado atual[3], não deixando de ser, ainda assim, um bom ponto de interesse para acrescentar funcionalidades e novas formas de interação com o utilizador.

[1] Tornando o seu estudo mais completo

[2] Considerando os seus *trade-offs*

[3] `http://www.theregister.co.uk/2015/07/07/apple_watch_slice/`

Existe, ainda, o objetivo de dotar o sistema de novas funcionalidades, de forma a que este permita reduzir a dependência do utilizador com o sistema EHR. A título de exemplo, a implementação de uma funcionalidade que permita iniciar um pedido de autorização a partir de um dispositivo móvel sem que o utilizador tenha de se dirigir a um computador. Para além desta nova funcionalidade, com o avançar deste projeto para produção, propõe-se que seja oferecido suporte para outros sistemas móveis, em particular para Android pela sua posição no mercado atual.

Bibliografia

[1] "Código deontológico." https://www.ordemdosmedicos.pt/?lop=conteudo&op=9c838d2e45b2ad1094d42f4ef36764f6&id=cc42acc8ce334185e0193753adb6cb77. Última visita: 6-11-2015.

[2] "Prescrição de terapêutica por via telefónica - carta aberta dirigida à ordem dos médicos." http://www.ordemenfermeiros.pt/sites/acores/informacao/Paginas/Prescri%C3%A7%C3%A3oTerap%C3%AAuticaViaTelef%C3%B3nica-CartaAbertadirigida%C3%A0OrdemdosM%C3%A9dicos.aspx. Última visita: 6-11-2015.

[3] "O erro terapêutico: causas e estratégias de prevenção." http://repositorio.hff.min-saude.pt/bitstream/10400.10/1010/1/o%20erro%20terap%C3%AAutico.pdf. Última visita: 5-11-2015.

[4] "Delegation of a medical act." https://www.cpsbc.ca/files/pdf/PSG-Delegation-of-a-Medical-Act.pdf. Última visita: 6-11-2015.

[5] "Delegation of duties by a physician to a non-physician." https://tma.custhelp.com/ci/fattach/get/27841/0/filename/Delegation+of+Duties+OGC-+June+2010.pdf. Última visita: 4-11-2015.

[6] "Manual técnico do sistema de informação hospitalar." http://bvsms.saude.gov.br/bvs/publicacoes/07_0066_M.pdf. Última visita: 5-11-2015.

[7] "Top 20 most popular emr software solutions." http://www.capterra.com/infographics/top-emr-software. Última visita: 17-11-2015.

[8] C. L. Ventola, "Mobile devices and apps for health care professionals: uses and benefits," *Pharmacy and Therapeutics*, vol. 39, no. 5, p. 356, 2014.

[9] "Improving clinician communication and collaboration across the health system (case study)." http://ibm.idcimpshowcase.com/showcase/detail.cfm?id=31. Última visita: 9-11-2015.

[10] "Vocera collaboration suite - for iphone and android devices." http://www.vocera.com/sites/default/files/resources/sb-collaboration-suite-vocera-usa-1.pdf. Última visita: 9-11-2015.

[11] "First amendment to agreement by and between the county of santa clara and vocera communications inc.." http://sccgov.iqm2.com/Citizens/FileOpen.aspx?Type=30&ID=36486. Última visita: 9-11-2015.

[12] M. Katherine Kim, "Clinical data standards in health care: Five case studies," pp. 5–6, 2005.

[13] S. M. Huff, "Clinical data exchange standards and vocabularies for messages," in *Proceedings of the AMIA Symposium*, p. 62, American Medical Informatics Association, 1998.

[14] M. Ferreira and N. Felix, *Interoperabilidade numa perspectiva Hospitalar*. Universidade do Minho, Braga, Portugal, 2009.

[15] P. Schloeffel, T. Beale, G. Hayworth, S. Heard, H. Leslie, *et al.*, "The relationship between cen 13606, hl7, and openehr," 2006.

[16] "Hl7 evolution, (corepoiunt health)." `https://www.corepointhealth.com/sites/default/files/whitepapers/hl7-v2-v3-evolution.pdf`. Última visita: 15-11-2015.

[17] J. Henriques and P. Carvalho, *HL7 version v2.x: slides das aulas de Informática Médica*. Universidade de Coimbra, Coimbra, Portugal, novembro 2013.

[18] "Hl7 messaging standard version 2.7, (health level seven international." `http://www.hl7.org/implement/standards/product_brief.cfm?product_id=146`. Última visita: 16-11-2015.

[19] G. W. Beeler, "Hl7 version 3—an object-oriented methodology for collaborative standards development," *International journal of medical informatics*, vol. 48, no. 1, pp. 151–161, 1998.

[20] K. W. Boone, *The CDA Tm Book*. Springer Science & Business Media, 2011.

[21] D. Bender and K. Sartipi, "Hl7 fhir: An agile and restful approach to healthcare information exchange," in *Computer-Based Medical Systems (CBMS), 2013 IEEE 26th International Symposium on*, pp. 326–331, IEEE, 2013.

[22] "Introducing hl7 fhir." `https://www.hl7.org/fhir/summary.html`. Última visita: 17-11-2015.

[23] G. Grieve, E. Kramer, and L. McKenzie, "Introduction to hl7 fhir." `http://pt.slideshare.net/HINZ/introduction-to-hl7-fhir`. Última visita: 17-11-2015.

[24] E. Muir, "What is 'fhir' and why should you care?." `http://www.interfaceware.com/blog/what-is-fhir-and-why-should-you-care/`. Última visita: 17-11-2015.

[25] A. Hasman *et al.*, "Hl7 rim: an incoherent standard," in *Ubiquity: Technologies for Better Health in Aging Societies, Proceedings of Mie2006*, vol. 124, p. 133, 2006.

[26] T. Stockinger, "Implicit authentication on mobile devices," in *The Media Informatics Advanced Seminar on Ubiquitous Computing*, 2011.

[27] A. Pocovnicu, "Biometric security for cell phones," *Informatica Economica*, vol. 13, no. 1, pp. 57–63, 2009.

[28] F. Aloul, S. Zahidi, and W. El-Hajj, "Two factor authentication using mobile phones," in *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on*, pp. 641–644, IEEE, 2009.

[29] T. V. N. Rao and K. Vedavathi, "Authentication using mobile phone as a security token," *IJCSET*, vol. 1, no. 9, pp. 569–574, 2011.

[30] A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 14, no. 1, pp. 4–20, 2004.

[31] B. C. Armstrong, M. V. Ruiz-Blondet, N. Khalifian, K. J. Kurtz, Z. Jin, and S. Laszlo, "Brainprint: Assessing the uniqueness, collectability, and permanence of a novel method for erp biometrics," *Neurocomputing*, vol. 166, pp. 59–67, 2015.

[32] "About the ios technologies." `https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007898-CH1-SW1`. Última visita: 23-12-2015.

[33] "Swift vs objective-c, 10 reasons the future favors swift." `http://www.infoworld.com/article/2920333/mobile-development/swift-vs-objective-c-10-reasons-the-future-favors-swift.html`. Última visita: 23-12-2015.

[34] "Learning swift vs objective-c." `http://blog.teamtreehouse.com/learning-swift-vs-objective-c`. Última visita: 23-12-2015.

[35] "Getting started with ios: Objective-c vs swift." `http://www.codenewbie.org/blogs/getting-started-with-ios-objective-c-vs-swift`. Última visita: 23-12-2015.

[36] "Xamarin e visual studio." `https://msdn.microsoft.com/pt-br/library/Mt299001.aspx`. Última visita: 28-05-2016.

[37] "Entendendo a xamarin." `https://comocriaraplicativos.com.br/entendendo-a-xamarin/`. Última visita: 28-05-2016.

[38] "Porque escolher o xamarin." `http://blog.oncedev.com/mobile/2014/06/25/porque-escolher-o-xamarin/`. Última visita: 28-05-2016.

[39] "Push technology." `https://en.wikipedia.org/wiki/Push_technology`. Última visita: 06-05-2016.

[40] "Push notifications explained." `https://www.urbanairship.com/push-notifications-explained`. Última visita: 06-05-2016.

[41] "Apple push notification service." `https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html`. Última visita: 06-05-2016.

[42] "Setting up push notifications on ios." http://code.tutsplus.com/tutorials/setting-up-push-notifications-on-ios--cms-21925. Última visita: 06-05-2016.

[43] "How to create apns certificates." http://quickblox.com/developers/How_to_create_APNS_certificates. Última visita: 06-05-2016.

[44] "Facebook shutters its parse developer platform." http://techcrunch.com/2016/01/28/facebook-shutters-its-parse-developer-platform/. Última visita: 28-01-2016.

[45] "About touch id security on iphone and ipad." https://support.apple.com/en-us/HT204587. Última visita: 20-11-2015.

[46] "Working with touch id api in ios 8 sdk." http://www.appcoda.com/touch-id-api-ios8/. Última visita: 20-11-2015.

[47] M. Vieira, *Slides das aulas de Gestão de Projetos*. Universidade de Coimbra, Coimbra, Portugal, novembro 2014.

[48] A. J. Dorofee, J. A. Walker, C. J. Alberts, R. P. Higuera, and R. L. Murphy, "Continuous risk management guidebook.," tech. rep., DTIC Document, 1996.

[49] "New to user stories?." https://www.scrumalliance.org/community/articles/2010/april/new-to-user-stories. Última visita: 30-11-2015.

[50] "Advantages of the as a user i want user story template." http://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template. Última visita: 30-11-2015.

[51] P. Clements, D. Garlan, L. Bass, J. Stafford, R. Nord, J. Ivers, and R. Little, *Documenting software architectures: views and beyond*. Pearson Education, 2002.

[52] "Interface engine." http://wiki.hl7.org/index.php?title=Interface_Engine. Última visita: 01-06-2016.

[53] G. Hohpe and B. Woolf, *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004.

[54] M. Liu, *WCF Multi-layer Services Development with Entity Framework*. Packt Publishing Ltd, 2014.

[55] M. Fowler, *Patterns of enterprise application architecture*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[56] "The repository pattern." https://msdn.microsoft.com/en-us/library/ff649690.aspx. Última visita: 01-06-2016.

[57] "Massive-view-controller." http://appledoc.gentlebytes.com/blog/2015/01/30/massive-view-controller/. Última visita: 06-01-2016.

[58] "Https (http over ssl or http secure) definition." http://searchsoftwarequality.techtarget.com/definition/HTTPS. Última visita: 07-01-2016.

[59] J. Lowy, *Programming WCF Services: Mastering WCF and the Azure AppFabric Service Bus*. "O'Reilly Media, Inc.", 2010.

[60] P. Faustino, "Modulo de agendamento para a aplicacao MedicineOne iPhone," Master's thesis, DEI-FCT, Universidade de Coimbra, Coimbra, Portugal, 2014.

[61] "Ws transport with message credential." https://msdn.microsoft.com/en-us/library/aa354508(v=vs.110).aspx. Última visita: 26-05-2016.

[62] "Brokered authentication: Security token service (sts)." https://msdn.microsoft.com/en-us/library/ff650503.aspx. Última visita: 27-05-2016.

[63] "The ins and outs of token based authentication." https://scotch.io/tutorials/the-ins-and-outs-of-token-based-authentication. Última visita: 27-05-2016.

[64] "Chapter 7: Message and transport security." http://msdn.microsoft.com/en-us/library/ff648863.aspx. Última visita: 25-05-2016.

[65] "Custom message encoder: Custom text encoder." https://msdn.microsoft.com/en-us/library/ms751486(v=vs.110).aspx. Última visita: 09-06-2016.

[66] "Certificado digital." https://pt.wikipedia.org/wiki/Certificado_digital. Última visita: 27-05-2016.

[67] "Msign - assinatura e autenticacao." https://www.multicert.com/pt/produtos/desmaterializacao/msign-assinatura-e-autenticacao/. Última visita: 27-05-2016.

[68] J. Zhou and K.-Y. Lam, "Securing digital signatures for non-repudiation," *Computer Communications*, vol. 22, no. 8, pp. 710–716, 1999.

[69] Q. Dang, *Recommendation for applications using approved hash algorithms*. US Department of Commerce, National Institute of Standards and Technology, 2008.

[70] "Sha-3 standardization." http://csrc.nist.gov/groups/ST/hash/sha-3/sha-3_standardization.html. Última visita: 11-06-2016.

[71] "ipad and iphone pkard reader bundles." http://www.thursby.com/products/pkard-reader. Última visita: 29-05-2016.

[72] "Tipos de autoridades de certificação." https://technet.microsoft.com/pt-pt/library/cc732368.aspx. Última visita: 15-06-2016.

[73] "Cartao de cidadao." https://www.cartaodecidadao.pt/index.php_option=com_content&task=view&id=19&Itemid=29&lang=pt.html. Última visita: 28-05-2016.

[74] J. P. Silveira, "Aplicacoes de Criptografia Baseada em Identidade com Cartoes de Identificacao Eletronica," Master's thesis, Universidade da Beira Interior, Covilha, Portugal, 2013.

[75] "Cartao de cidadao - prova de conceito." https://www.cartaodecidadao.pt/index.php_option=com_content&task=view&id=20&Itemid=31&lang=pt.html. Última visita: 29-05-2016.

[76] "Criando serviços rest com wcf." https://msdn.microsoft.com/pt-br/library/dd941696.aspx. Última visita: 15-06-2016.

[77] "Implement step step generic repository pattern." http://blog.falafel.com/implement-step-step-generic-repository-pattern-c/. Última visita: 02-05-2016.

[78] "How safely store passowrd in 2016." https://paragonie.com/blog/2016/02/how-safely-store-password-in-2016. Última visita: 02-05-2016.

[79] "How to configure and send apple push notifications using pushsharp." https://github.com/Redth/PushSharp/wiki/How-to-Configure-&-Send-Apple-Push-Notifications-using-PushSharp. Última visita: 14-05-2016.

[80] "Push notifications tutorial for ios9." http://www.intertech.com/Blog/push-notifications-tutorial-for-ios-9/. Última visita: 14-05-2016.

[81] M. J. Fonseca, P. CAMPOS, and D. GONÇALVES, *Introdução ao design de interfaces*. Lisboa: FCA-Editora de Informática, 2012.

[82] J. Nielsen, "Enhancing the explanatory power of usability heuristics," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 152–158, ACM, 1994.

[83] R. Yáñez Gómez, D. Cascado Caballero, and J.-L. Sevillano, "Heuristic evaluation on mobile interfaces: A new checklist," *The Scientific World Journal*, vol. 2014, 2014.

[84] M. Contreras, G. Moore, and H. Nguyen, "Vocera communications system hca: Healthcare management information systems," 2013.

[85] R. Soeiro, *HIE A standards-based distributed Healthcare Integration Engine*. Universidade Técnica de Lisboa, Lisboa, Portugal, 2009.

[86] "2012 hl7 interface technology survey results." http://corehealthtechnologies.com/main/wp-content/uploads/2012/10/2012-HL7-Interface-Technology-Survey-Results.pdf. Última visita: 20-11-2015.

[87] V. Bicer, G. B. Laleci, A. Dogac, and Y. Kabak, "Artemis message exchange framework: semantic interoperability of exchanged messages in the healthcare domain," *ACM Sigmod Record*, vol. 34, no. 3, pp. 71–76, 2005.

[88] M. Kaeo, *Designing network security*. Cisco Press, 2003.

Apêndice A

Estado da Arte

A.1 Standards para o formato e transação de mensagens em contexto hospitalar

Standard	HL7
Developer	*Health Level Seven*
Descrição	- utilizado por entidades governamentais, especialistas em interfaces clínicas e ao nível da informática médica; - intercâmbio de dados em todas as áreas da saúde; - possui uma enorme adoção no mercado; - mensagens flexíveis (versão 2.x); - constante evolução.
Standard	DICOM
Developer	*National Electrical Manufacturers Association* (NEMA)
Descrição	- gestão, armazenamento e transmissão de imagens médicas; - desenvolvido para operações em redes TCP/IP; - direcionado a profissionais que trabalham com diagnóstico por imagem.
Standard	NCPDP
Developer	*National Council for Prescription Drug Programs* (NCPDP)
Descrição	- estrutura para transmitir pedidos de prescrições electrónicas; - relativo a informação farmacêutica;
Standard	ASC X12
Developer	*American National Standards Institute* *-Acredited Standards Comittee* (ANSI-ASC)
Descrição	- usado para comunicação entre serviços de saúde com o estado e seguradoras.
Standard	ASTM E1238
Developer	*American Society for Testing and Materials*
Descrição	- primeiro standard para transferir comunicação clínica entre computadores; - usado para informação laboratorial.

Tabela A.1: Standards para o formato e transação de mensagens em contexto hospitalar

A.2 Estrutura das mensagens HL7v2.x [17]

De forma a disponibilizar um meio para agrupar a informação de forma lógica, as mensagens HL7v2.x são constituídas por **segmentos** (opcionais, representados por '[]'; repetidos, representados por '{}') que, por sua vez, podem conter vários **campos** (separados pelo carácter |) com vários **componentes** (separados pelo carácter '^'). As possibilidades de combinação são muitas, dificultando a integração, mas, ainda assim, deixa as mensagens relativamente fáceis de ler sem necessidade de grande conhecimento técnico. A título de exemplo, na Figura A.1, MSH, EVN, PID e PV1 constituem os segmentos onde: **MSH** (*Message Header*) identifica o tipo de mensagem, o remetente e o destinatário; **EVN** (*Event Type*) identifica o tipo de evento; **PID** (*Patient Identification*) identifica o paciente; **PV1** (*Patient Visit Information*) fornece informação sobre a internação hospitalar do paciente (por exemplo, localização atribuída, médico responsável).

```
MSH|^~\&|ADT1|MCM|LABADT|MCM|198808181126||ADT^A01|
MSG00001|P|2.3<cr>
EVN|A01|198808181123<cr>
PID|||PATID1234^5^M11||JONES^WILLIAM^A^III||19610615|M||C|1200 N ELM
STREET^^GREENSBORO^NC^27401-1020||(919)379-1212|(919)271-3434||S||
PATID12345001^2^M10|123456789|<cr>
PV1|1|I|2000^2012^01||||004777^LEBAUER^SIDNEY^J.|||SUR||||ADM|A0<cr>
```

Figura A.1: Exemplo de uma mensagem HL7v2.x (fonte: [17])

A.3 Diagrama HL7v3-MDF

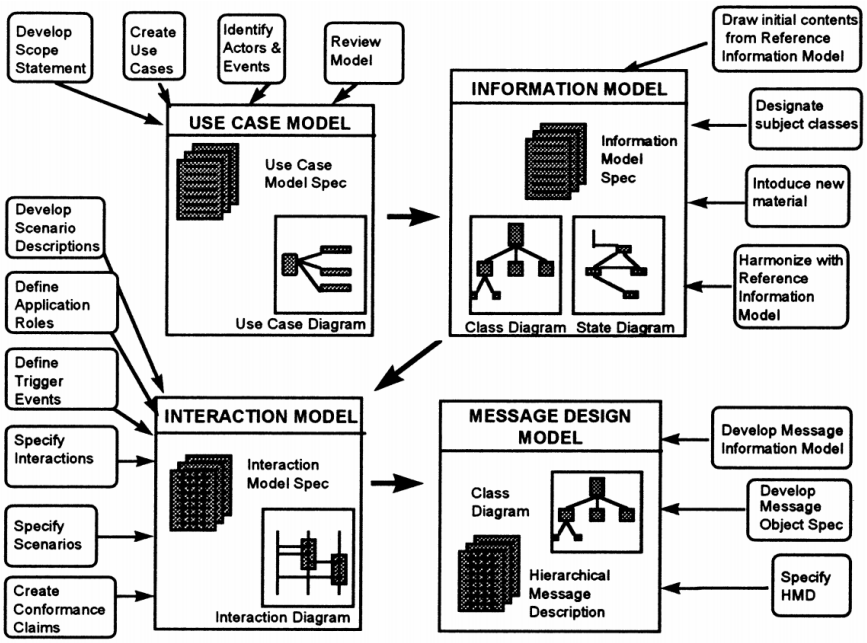


Figura A.2: Diagrama dos modelos primários, etapas de desenvolvimento e resultados documentados especificado pela versão HL7v3-MDF

A.4 *Apple Push Notification Service* [41]

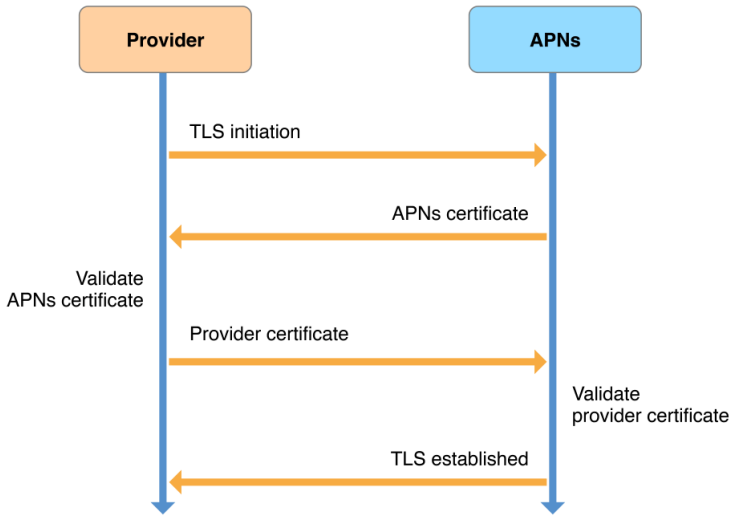


Figura A.3: Conexão de confiança entre *provider* e APNS

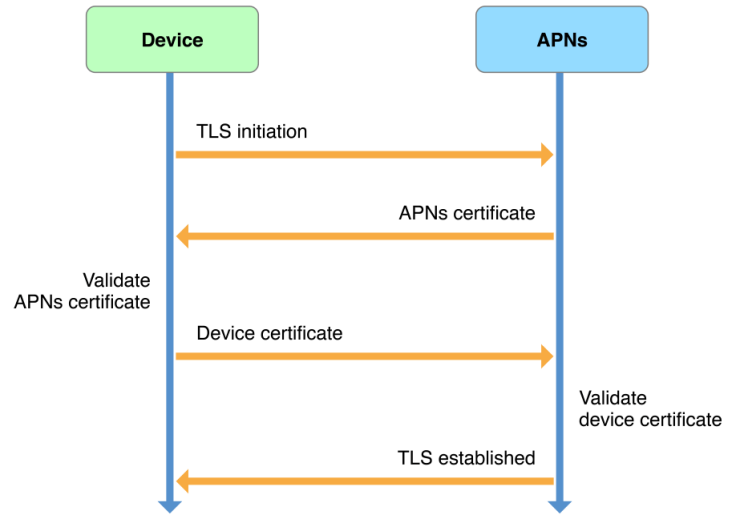


Figura A.4: Conexão de confiança entre dispositivo e APNS

A.5 Comparação de serviços *push notification*

Serviço	OneSignal	PushWoosh	UrbanAirship	Amazon SNS
Principais características	- Boa compatibilidade; - Fácil configuração; - Escalabilidade; - *Real time tracking*	- Escalabilidade; - Compatibilidade; - *Geozones*	- Escalabilidade; - *Real time tracking*; - Não limita o número de *push notifications*; - Facilidade de utilização	- Confiabilidade; - Escalabilidade; - Baixo custo
Preços p/ mês	Gratuito e com soluções (custom) para clientes (enterprise)	- Premium ($49,.95); - Gold ($249.95); - Platinum ($749.95)	- Basic ($99); - Essential ($149); - Comprehensive ($349)	1 milhão de solicitações gratuitas; $0.50 por cada milhão seguinte
Protocolos de transporte	HTTP, HTTPS, XMPP	HTTP, HTTPS, SMTP	HTTP, HTTPS, SMTP	HTTP, HTTPS, SMTP
Formato dos dados	JSON	JSON	JSON	JSON
Plataformas suportadas	APNS, GCM, WNS, ADM	APNS, GCM, WNS, ADM	APNS, GCM, Microsoft, Blackberry	APNS, GCM, WNS, ADM
Clientes	Uber, zynga, 9GAG, MTV, ...	Leica, Ubisoft, WWF, MasterCard, HP, Deloitte, ...	adidas, wall street journal, bankinter, sky, ...	yelp, Wunderlist, tinder, ...

Tabela A.2: Serviços de *push notifications*

Apêndice B

Planeamento

B.1 Estimativas e priorização

ID	Prioridade	*Best Case*	*Most Likely*	*Worst Case*	*Expected*
GEN_US002	*Must*	15	18	22	18,2
GEN_US004	*Must*	15	18	22	18,2
GEN_US005	*Must*	14	17	23	17,5
GEN_US006	*Must*	13	15	18	15,2
GEN_US009	*Must*	8	10	13	10,2
MED_US001	*Must*	20	22	26	22,3
MED_US002	*Must*	18	20	24	20,3
ENF_US002	*Must*	10	13	18	13,3
ENF_US003	*Must*	12	14	19	14,5
GEN_US001	*Nice*	14	16	20	16,3
GEN_US003	*Nice*	12	14	17	14,2
GEN_US007	*Nice*	12	14	17	14,2
GEN_US008	*Nice*	10	11	14	11,3
ENF_US001	*Nice*	13	16	20	16,2
GEN_US010	*Wishful*	45	60	80	60,8
		231	**278**	**353**	**282,7**

Tabela B.1: Estimativas para cada *user story* (valores em horas)

Apêndice C

Análise de Requisitos

C.1 *User Stories*

User Story	**GEN_US001**
Nome	Inscrição de um novo utilizador
Categoria	Aplicação Móvel
Descrição	**Enquanto** profissional de saúde (médico ou enfermeiro), **quero** registar a minha conta **a fim de** usufruir das funcionalidades do sistema
Prioridade	*Nice-to-have*
Dependências	-

User Story	**GEN_US002**
Nome	Autenticação de um utilizador
Categoria	Aplicação Móvel
Descrição	**Enquanto** profissional de saúde (médico ou enfermeiro), **quero** autenticar-me na aplicação **a fim de** usufruir das funcionalidades do sistema
Prioridade	*Must-have*
Dependências	GEN_US002

User Story	**GEN_US003**
Nome	Permitir a recuperação da *password*
Categoria	Aplicação Móvel
Descrição	**Enquanto** profissional de saúde (médico ou enfermeiro), **quero** recuperar a minha *password* **a fim de** voltar a utilizar a minha conta no sistema
Prioridade	*Nice-to-have*
Dependências	GEN_US002

User Story	**GEN_US004**
Nome	Aceder à aplicação através de uma notificação, destacando o pedido
Categoria	Aplicação Móvel
Descrição	**Enquanto** profissional de saúde (médico ou enfermeiro), **quero** aceder à aplicação **a fim de** destacar, diretamente, um pedido de autorização
Prioridade	*Must-have*
Dependências	GEN_US002, GEN_US006

User Story	**GEN_US005**
Nome	Listar autorizações pendentes
Categoria	Aplicação Móvel
Descrição	**Enquanto** profissional de saúde (médico ou enfermeiro), **quero** listar os pedidos de autorização pendentes **a fim de** analisar cada um em detalhe
Prioridade	*Must-have*
Dependências	GEN_US002, GEN_US006

User Story	**GEN_US006**
Nome	Exibir todos os campos dos pedidos de autorização
Categoria	Aplicação Móvel
Descrição	**Enquanto** profissional de saúde (médico ou enfermeiro), **quero** aceder a um pedido de autorização **a fim de** ter acesso a toda a sua informação detalhada
Prioridade	*Must-have*
Dependências	GEN_US002

User Story	**GEN_US007**
Nome	Consultar o histórico de pedidos
Categoria	Aplicação Móvel
Descrição	**Enquanto** profissional de saúde (médico ou enfermeiro), **quero** listar o histórico de autorizações **a fim de** aceder aos pedidos em que estive envolvido
Prioridade	*Nice-to-have*
Dependências	GEN_US002

User Story	**GEN_US008**
Nome	Filtrar o histórico de pedidos
Categoria	Aplicação Móvel
Descrição	**Enquanto** profissional de saúde (médico ou enfermeiro), **quero** filtrar o histórico de autorizações **a fim de** tornar a minha pesquisa mais direcionada
Prioridade	*Nice-to-have*
Dependências	GEN_US002, GEN_US007

User Story	**GEN_US009**
Nome	Menu principal
Categoria	Aplicação Móvel
Descrição	**Enquanto** profissional de saúde (médico ou enfermeiro), **quero** ter acesso a um menu principal na aplicação móvel **a fim de** poder escolher rapidamente qualquer uma das funcionalidades disponíveis
Prioridade	*Must-have*
Dependências	GEN_US002

User Story	**GEN_US010**
Nome	Receber notificações no *Apple Watch*
Categoria	Aplicação Móvel
Descrição	**Enquanto** profissional de saúde, **quero** receber uma notificação de um pedido de autorização no meu *Apple Watch* **a fim de** o sincronizar com o meu *iPhone*
Prioridade	*Wishful*
Dependências	GEN_US002

User Story	**MED_US001**
Nome	Autorizar um pedido
Categoria	Aplicação Móvel
Descrição	**Enquanto** médico, **quero** responder a um pedido de autorização **a fim de** o autorizar
Prioridade	*Must-have*
Dependências	GEN_US002, GEN_US005, GEN_US006

User Story	**MED_US002**
Nome	Rejeitar um pedido, justificando
Categoria	Aplicação Móvel
Descrição	**Enquanto** médico, **quero** responder a um pedido de autorização **a fim de** o rejeitar, justificando-o
Prioridade	*Must-have*
Dependências	GEN_US002, GEN_US005, GEN_US006

User Story	**ENF_US001**
Nome	Enviar *reminders* relativos aos pedidos sem resposta
Categoria	Aplicação Móvel
Descrição	**Enquanto** enfermeiro, **quero** poder enviar alertas para o médico autorizador **a fim de** o poder notificar da urgência do pedido
Prioridade	*Nice-to-have*
Dependências	GEN_US002

User Story	**ENF_US002**
Nome	Receber confirmação de leitura do pedido de autorização
Categoria	Aplicação Móvel
Descrição	**Enquanto** enfermeiro, **quero** poder receber notificações no meu *smart device* **a fim de** saber quando é que o meu pedido de autorização foi visto pelo médico
Prioridade	*Must-have*
Dependências	GEN_US002

User Story	**ENF_US003**
Nome	Receber resposta ao pedido de autorização
Categoria	Aplicação Móvel
Descrição	**Enquanto** enfermeiro, **quero** poder receber notificações no meu *smart device* **a fim de** confirmar a resposta de um pedido
Prioridade	*Must-have*
Dependências	GEN_US002, MED_US001, MED_US002

C.2 Protótipos da aplicação móvel

C.2.1 Ecrãs de *login* e de menu principal

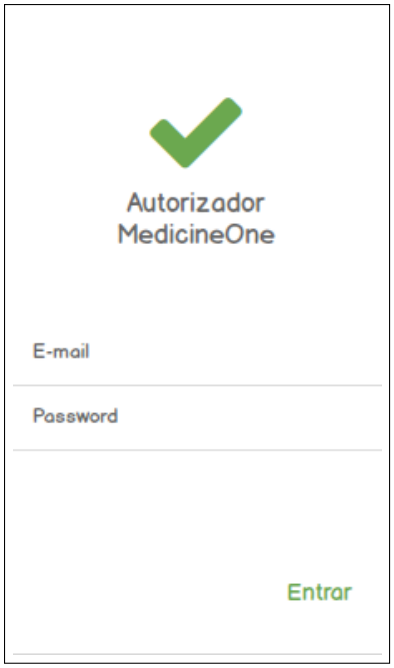


Figura C.1: Ecrã inicial

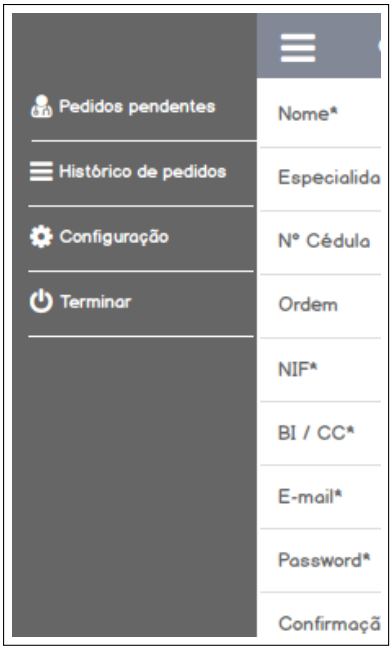


Figura C.2: Menu principal

C.2.2 Ecrãs de pedidos pendentes

Figura C.3: Pedidos pendentes

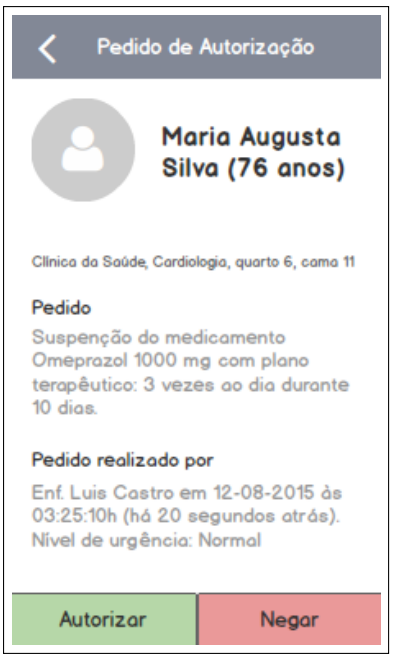


Figura C.4: Pedido na íntegra

Figura C.5: Pedidos pendentes após uma resposta

C.2.3 Ecrãs para autorizar e rejeitar pedidos

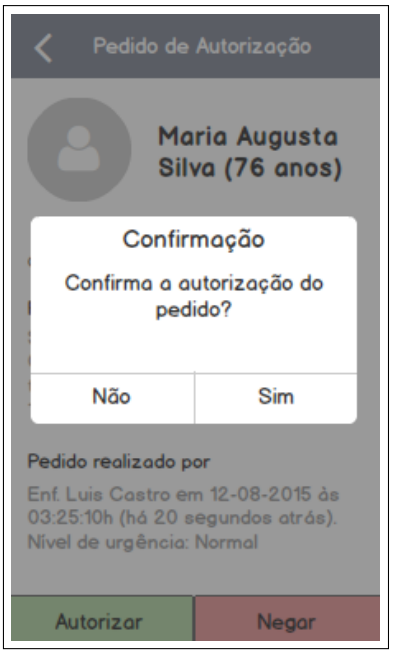


Figura C.6: Confirmação de autorização de pedido

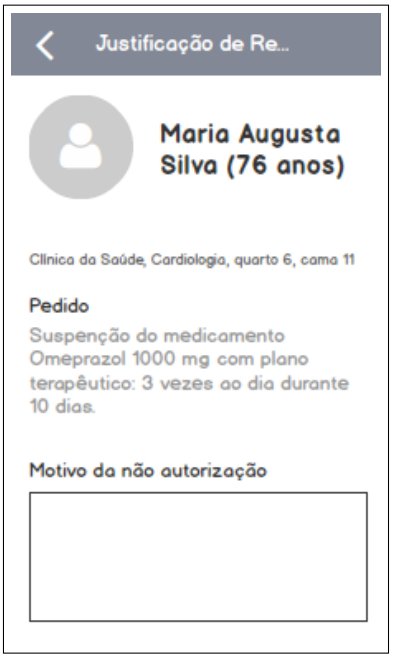


Figura C.7: Ecrã para justificar rejeição de pedido

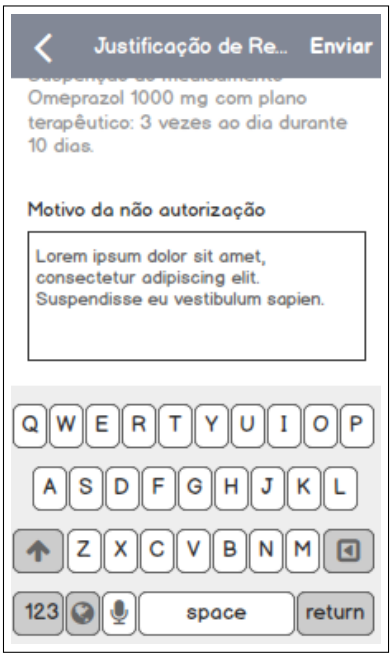


Figura C.8: Ecrã para justificar rejeição de pedido

C.2.4 Ecrãs para histórico de pedidos

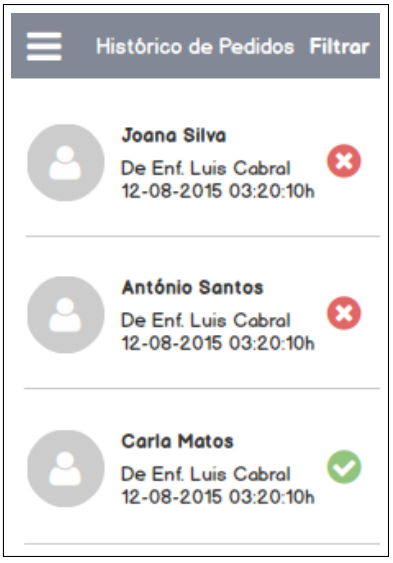


Figura C.9: Histórico de pedidos

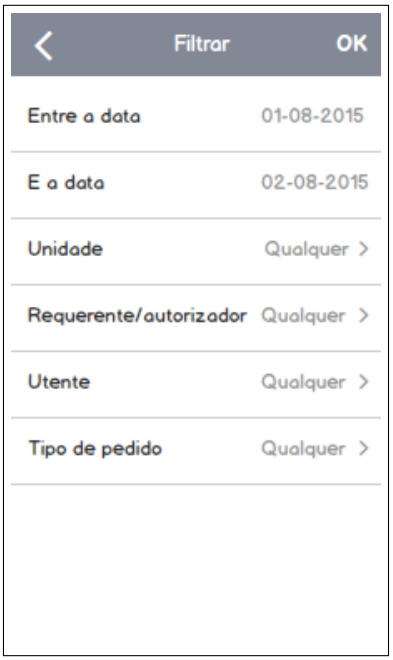


Figura C.10: Parâmetros de filtragem

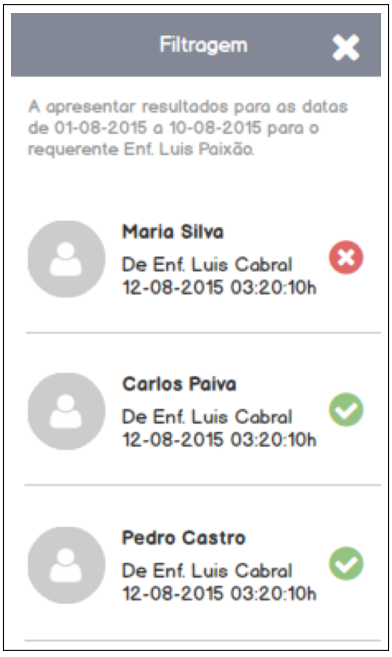


Figura C.11: Histórico de pedidos com filtro aplicado

Apêndice D

Desenho da solução

D.1 Criação da assinatura digital [88]

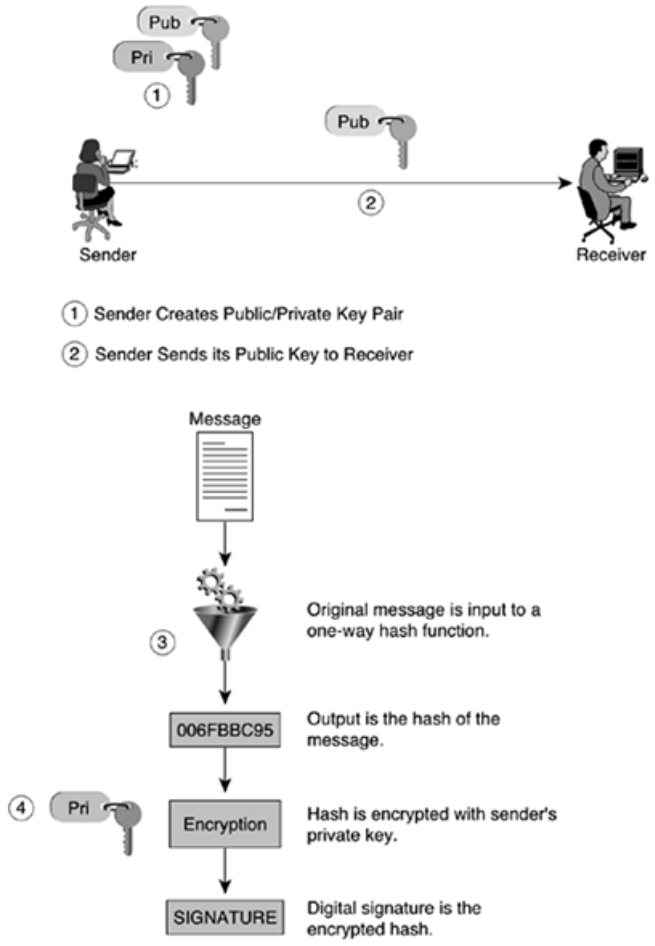


Figura D.1: Criação da assinatura digital - fonte [88]

- **Passo 1:** Bob cria o par de chaves (pública e privada);
- **Passo 2:** Bob fornece a sua chave pública a Alice;
- **Passo 3:** Bob escreve a mensagem para Alice e usa o documento como *input* para a função de *hash*;

- **Passo 4:** Bob encripta o *output* do Passo 3 (*hash*) com a sua chave privada, resultando na assinatura digital.

D.2 Verificação da assinatura digital [88]

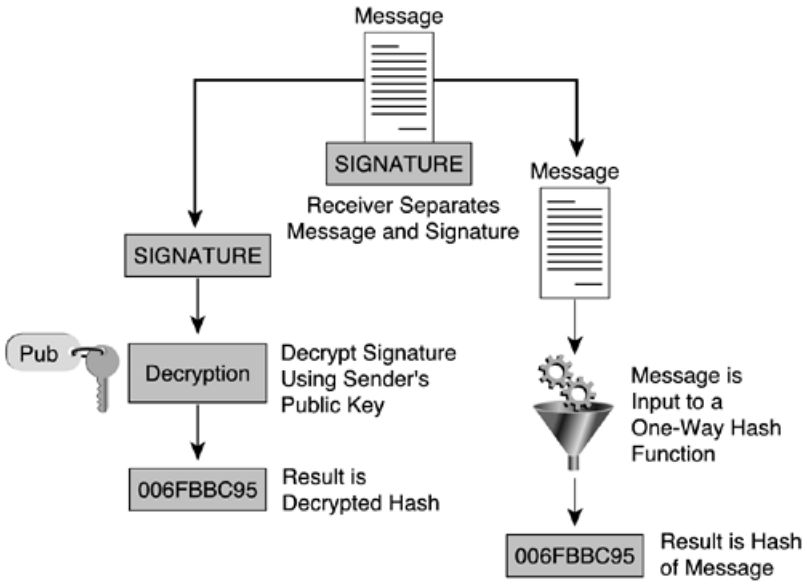


Figura D.2: Verificação da assinatura digital - fonte [88]

Apêndice E

Desenvolvimento

E.1 *ServiceContracts* (componente servidor)

Figura E.1: *Service Contracts* (componente servidor)

E.2 *DataContracts* (componente servidor)

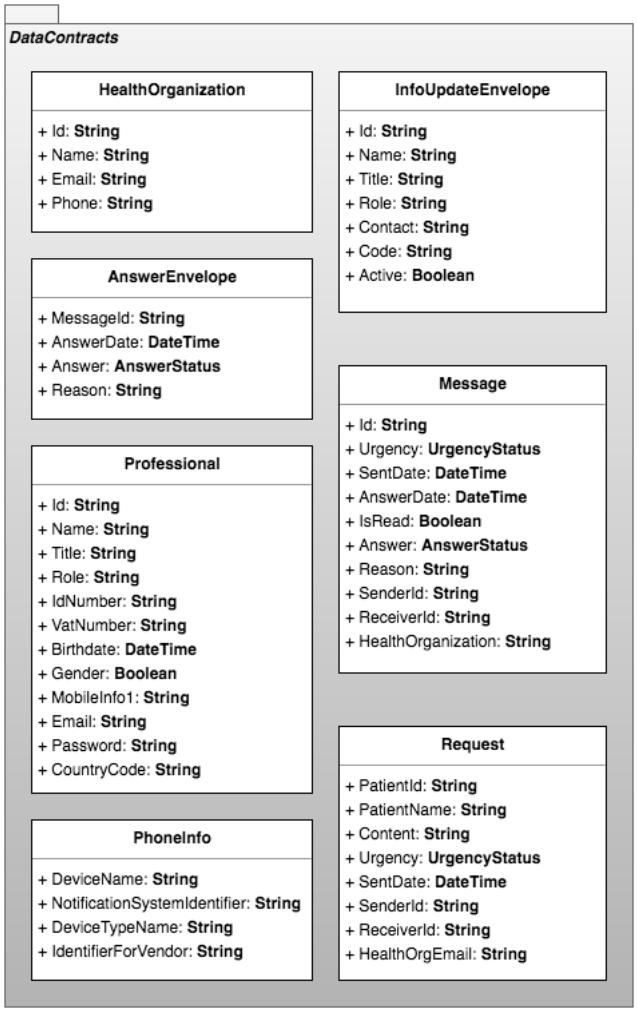


Figura E.2: *Data Contracts* (componente servidor)

E.3 *Business Data Objects*

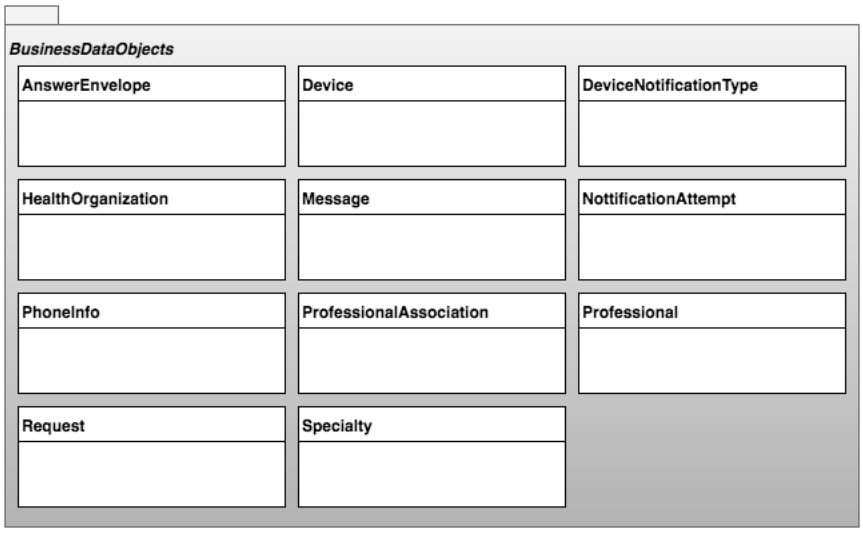


Figura E.3: *Business Data Objects*

E.4 Repositórios

Figura E.4: Repositórios

E.5 Cliente EHR

Listing E.1: XML correspondente ao registo de uma unidade de saúde

```
<s:Envelope
    xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    ...
  </s:Header>
  <s:Body>
    <RegisterOrganization xmlns="http://tempuri.org/">
      <org ...>
        <d4p1:Email>geral@hospitaldaarrabida.pt</d4p1:Email>
        <d4p1:Id />
        <d4p1:Name>Hospital da Arrabida</d4p1:Name>
        <d4p1:Phone>223776800</d4p1:Phone>
      </org>
      <message />
    </RegisterOrganization>
  </s:Body>
</s:Envelope>
```

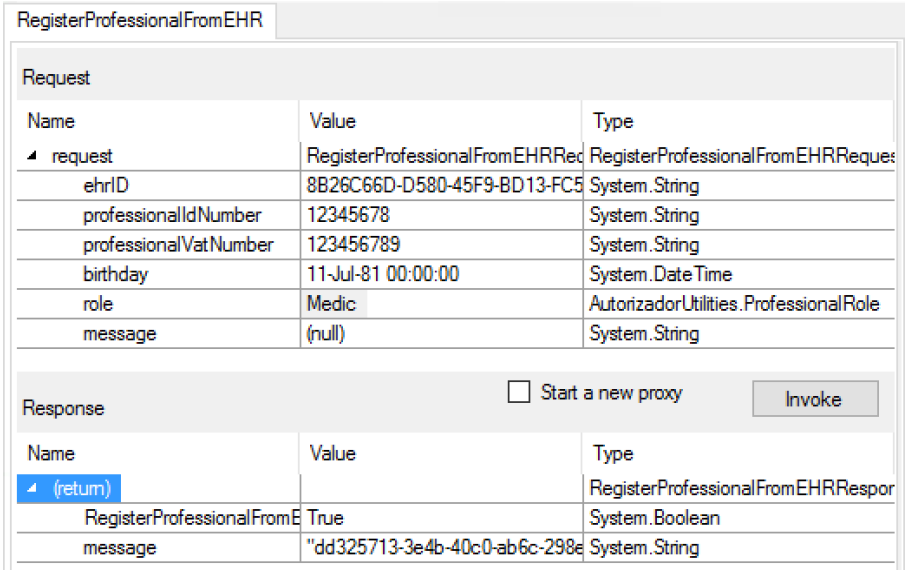


Figura E.5: Cliente EHR – pré-registo/validação de profissional de saúde

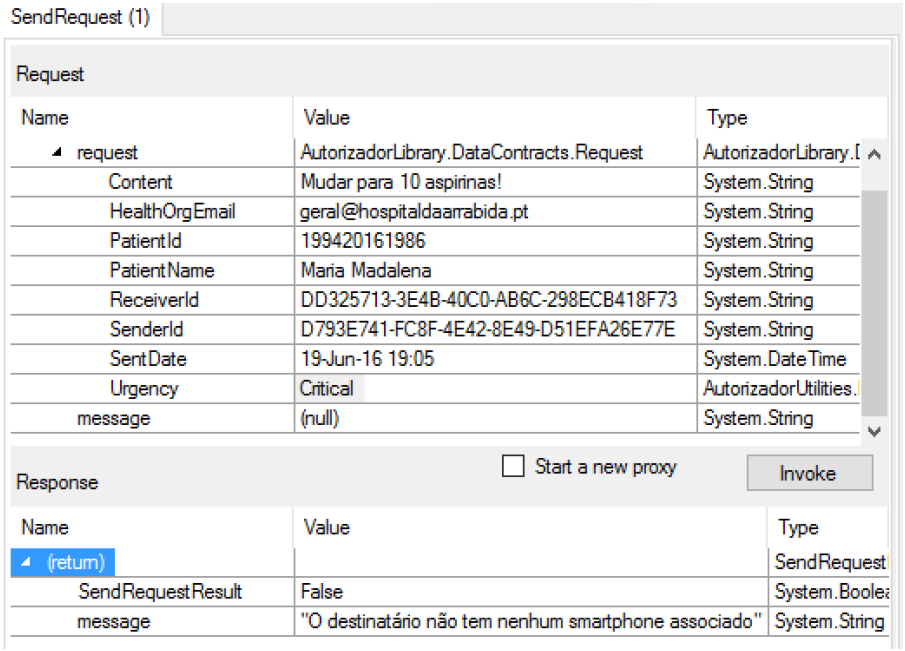


Figura E.6: Cliente EHR – Envio de um pedido de autorização

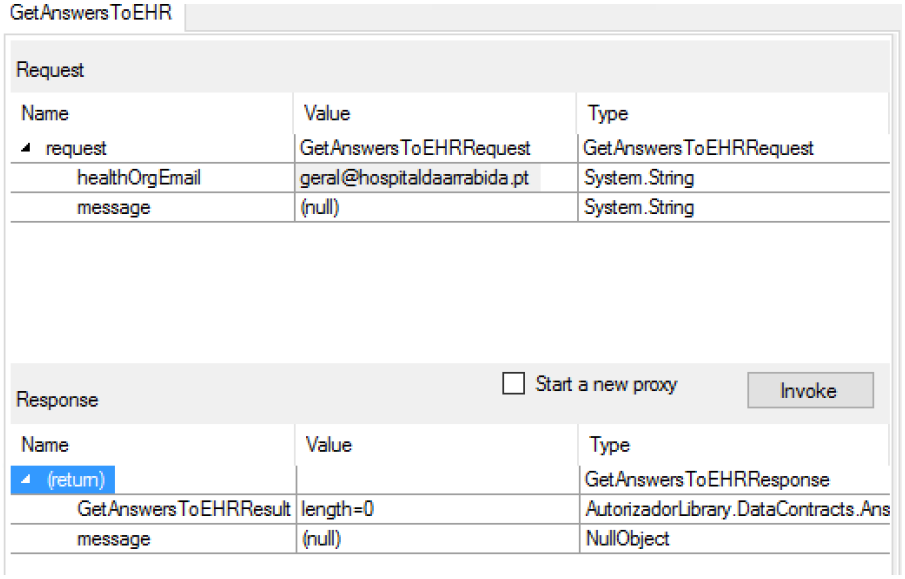


Figura E.7: Cliente EHR – Recolha de respostas de um EHR (0 disponíveis)

E.6 Organização do código da aplicação móvel

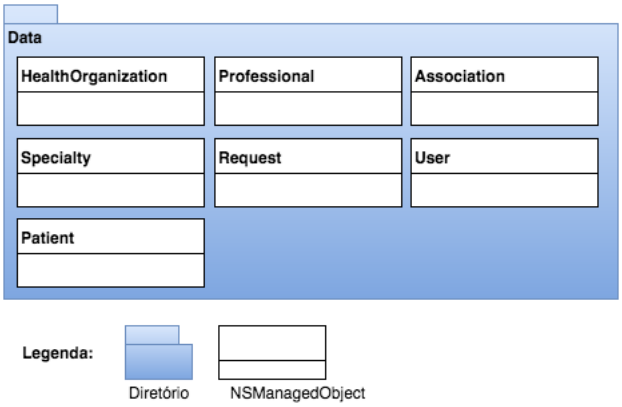


Figura E.8: Organização do diretório com *NSManagedObjects*

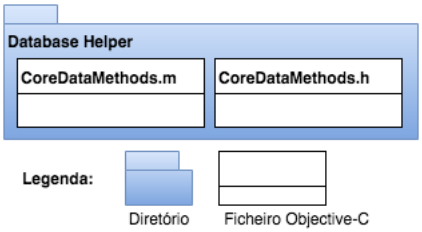


Figura E.9: Organização do diretório com classes de suporte ao Core Data

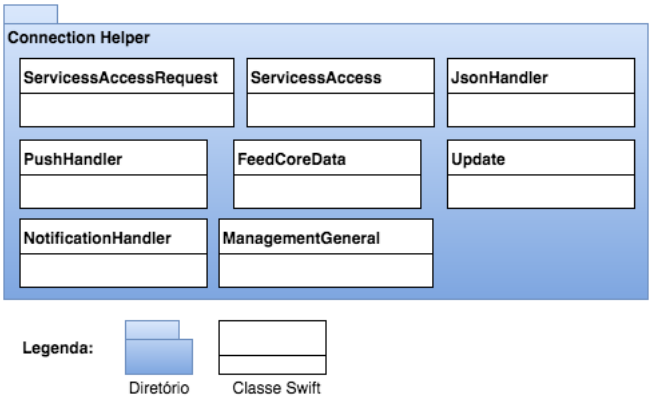


Figura E.10: Organização do diretório com as classes que gerem notificações, serialização e acesso a serviços e camada de dados

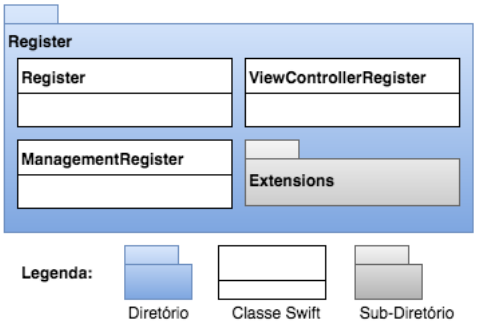


Figura E.11: Organização do diretório com classes relacionadas com o Registo

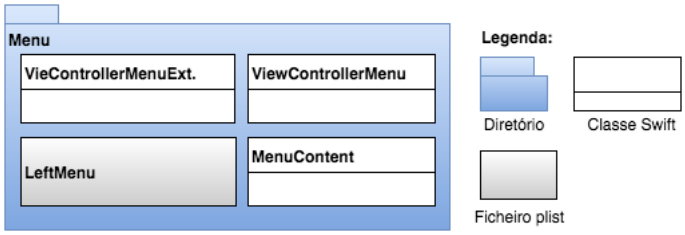


Figura E.12: Organização do diretório com classes relacionadas com o Menu lateral

Figura E.13: Organização do diretório dos pedidos de autorização (pendentes e histórico)

E.7 Ecrãs da aplicação móvel

Figura E.14: Notificações recebidas com a aplicação em *foreground*, destacando um lembrete e um recibo de confirmação de leitura

Figura E.15: Notificações recebidas com a aplicação em *foreground*, resumindo os pedidos enviados, recebidos e as possíveis resposta

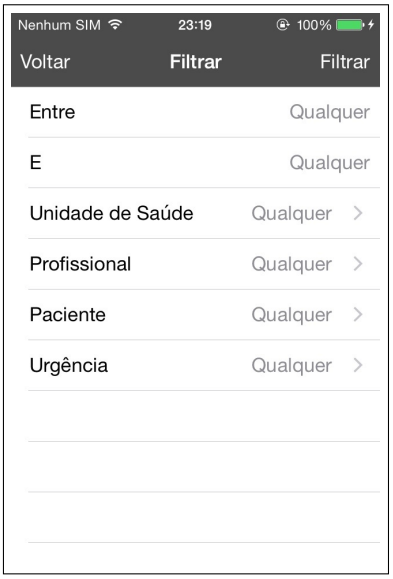


Figura E.16: Ecrã de filtro de pedidos

Figura E.17: Ecrã para confirmar autorização

Figura E.18: Ecrã para justificar rejeição

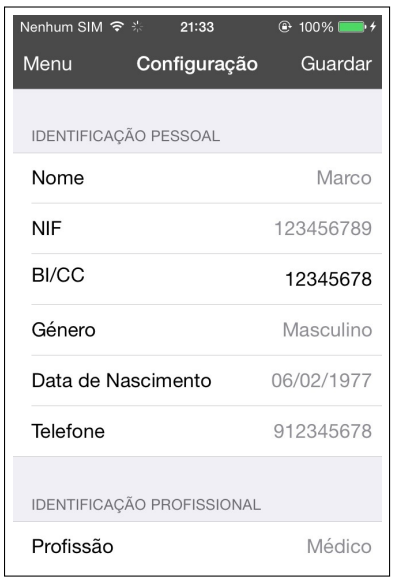


Figura E.19: Ecrã de definições

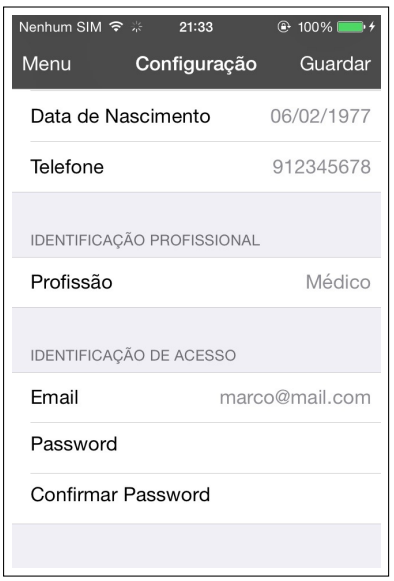


Figura E.20: Ecrã de definições (continuação)

Apêndice F

Testes

F.1 Testes de aceitação

# Teste	Descrição do teste	*User Story*	Peso	Resultado
1	Preencher os campos de registo (nome, números civil e fiscal, data de nascimento, género, país, email, password e confirmação	GEN US001	*Nice*	OK
2	Preencher os campos de *login* com email e password	GEN US002	*Must*	OK
3	Aceder à aplicação através de uma notificação, e verificar se destaca o pedido, visualizando o seu conteúdo	GEN US004	*Must*	OK
4	Aceder ao menu principal e escolher a opção "Pedidos pendentes"	GEN US005	*Must*	OK
5	Aceder ao menu principal, escolher a opção "Pedidos pendentes" e selecionar um pedido em particular e visualizar toda a informação	GEN US006	*Must*	Parcial[1]
6	Aceder ao menu principal e escolher a opção "Histórico de pedidos"	GEN US007	*Must*	OK
7	Aceder ao menu principal, escolher a opção "Histórico de pedidos", selecionar a opção "Filtrar" e escolher parâmetros de pesquisa	GEN US008	*Must*	Parcial[2]
8	Executar o Teste 2, deslocar o dedo na horizontal no sentido da direita para a esquerda	GEN US009	*Must*	OK
9	Como autorizador, aceder ao menu principal, escolher a opção"Pedidos pendentes", selecionar um pedido em particular, aceitar e confirmar	MED US001	*Must*	OK

[1] É exibida a informação pretendida mas o *layout* apresenta incongruências
[2] A filtragem funciona mas não existe verificação do intervalo de datas

10	Como autorizador, aceder ao menu principal, escolher a opção"Pedidos pendentes", selecionar um pedido em particular, rejeitar e justificar	MED US002	*Must*	OK
11	Como requerente, aceder ao menu principal, escolher a opção "Pedidos pendentes", selecionar um pedido em particular e enviar lembrete	ENF US001	*Nice*	OK
12	Como autorizador, aceder ao menu principal, escolher a opção"Pedidos pendentes", selecionar um pedido em particular pela 1ª vez e verificar se o requerente recebe uma notificação no seu dispositivo	ENF US002	*Must*	OK
13	Como autorizador, aceder ao menu principal, escolher a opção"Pedidos pendentes", selecionar um pedido em particular, responder e verificar se o requerente recebe uma notificação no seu dispositivo	ENF US003	*Must*	OK

Tabela F.1: Conjunto de testes de aceitação realizados ao *Autheras*

F.2 Testes de memória da aplicação

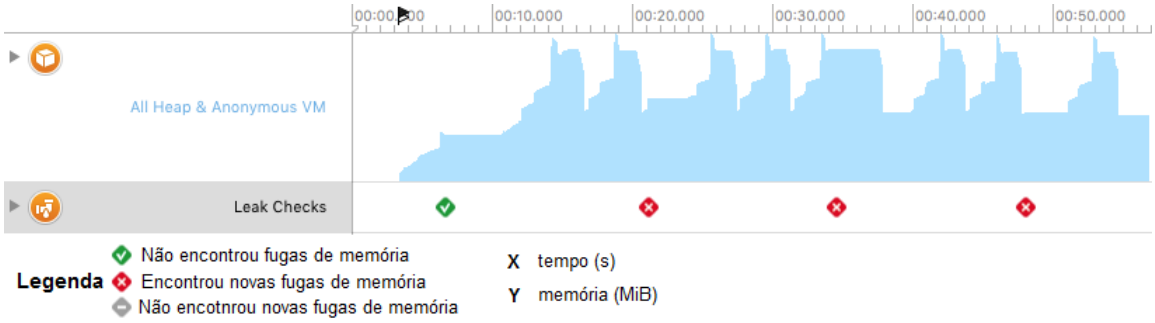


Figura F.1: Teste de memória − ecrãs de *login* e de registo

Pela análise da Figura F.1, conclui-se que após o início sem *memory leaks*, existe uma ocorrência, verificada sempre que se entra no ecrã de registo (no código referente ao *date picker* para a data de nascimento). Por outro lado, apesar de existir um aumento do consumo de memória quando há uma mudança de ecrã, esta é libertada, onde o padrão indica que não há memória alocada indefinidamente.

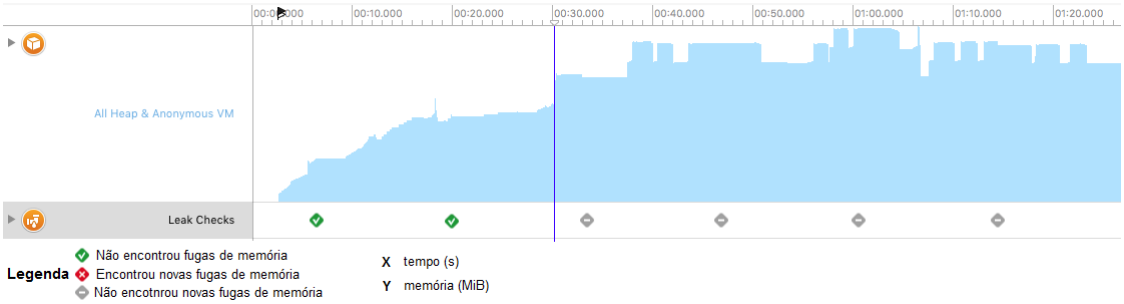


Figura F.2: Teste de memória – ecrã de listagem de pedidos pendentes

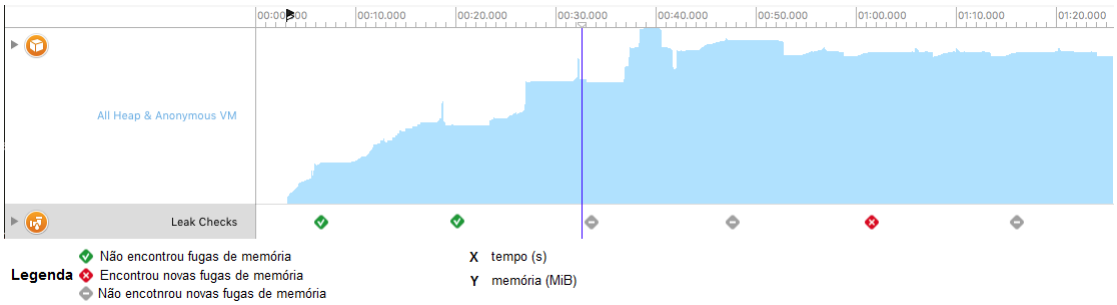


Figura F.3: Teste de memória – ecrã de um pedido detalhado (aceitar pedido)

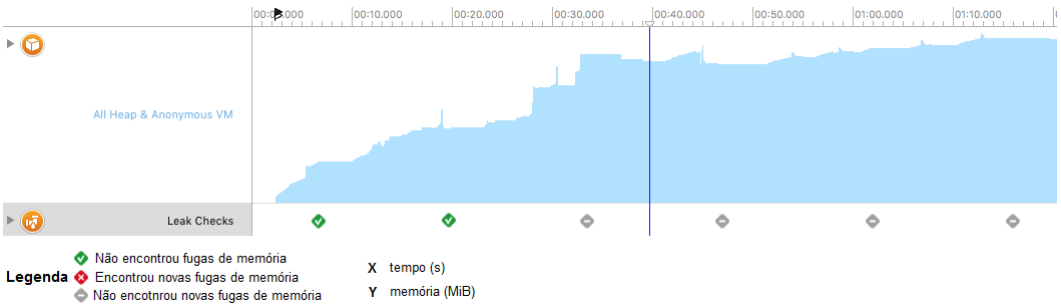


Figura F.4: Teste de memória – ecrã de um pedido detalhado (rejeitar pedido)

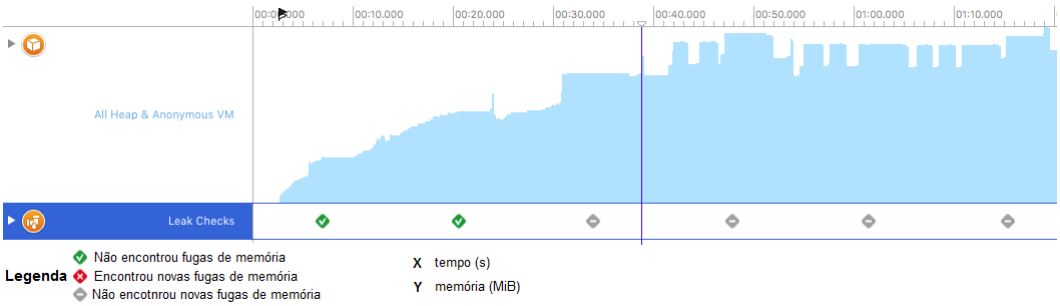


Figura F.5: Teste de memória – ecrã de listagem de histórico de pedidos

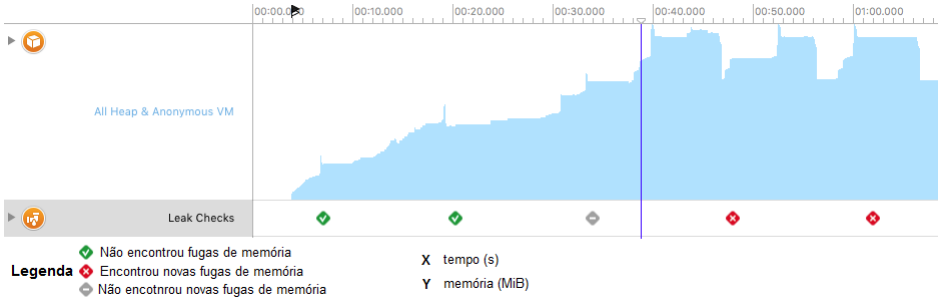


Figura F.6: Teste de memória – ecrã de filtro de pedidos

As Figuras anteriores identificam um aumento do uso de memória em qualquer uma das operações que, contudo, apesar de se manter elevada, não cresce indefinidamente. Realça-se para o facto de que o início do respetivo teste de cada imagem conta a partir da barra azul. Somente o ecrã de filtro de pedidos voltou a encontrar um novo *memory leak*, sendo que o motivo é idêntico ao referido para a Figura F.1.

Apêndice G

Usabilidade

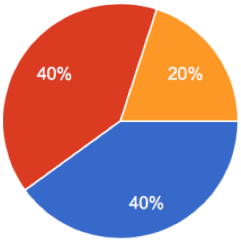
G.1 Medidas de usabilidade

Medidas de eficácia	- nº de chamadas de ajuda; - nº de erros para executar uma tarefa; - nº de utilizadores que concluíram uma tarefa.
Medidas de eficiência	- tempo para completar uma tarefa; - tempo para completar todas as tarefas.
Medidas de satisfação	- facilidade de utilizar a aplicação; - aprendizagem na utilização da aplicação; - organização da interface; - satisfação geral.

Tabela G.1: Medidas de usabilidade

G.2 Descrição de utilizadores

Os utilizadores que estiveram presentes nas duas fases de testes de usabilidade representam diferentes faixas etárias, nomeadamente: jovens adultos (com idade entre 18 e 29 anos), adultos (com idade entre 30 e 49 anos) e sénior (com idade igual ou superior a 50 anos).



Representação de utilizadores por faixa etária			
Faixa etária	Total	Percentagem	Cor
Jovem Adulto	2	40%	
Adulto	2	40%	
Sénior	1	20%	

Figura G.1: Distribuição de utilizadores por faixa etária

G.3 Lista de tarefas para as duas fases de testes com utilizadores

1. Aceder aos pedidos pendentes e autorizar o pedido de autorização menos urgente do "Enf. Luís Cabral";
2. Consultar o histórico de pedidos e visualizar o pedido do paciente "António Santos";
3. Consultar o histórico de pedidos e filtrar com a data definida por omissão:
   (a) Consultar o pedido do paciente "Carlos Paiva";
   (b) Retroceder para o histórico de pedidos (remover filtros).
4. Aceder ao menu de configuração e, de seguida, guardar;
5. Aceder aos pedidos pendentes e negar o pedido de autorização menos urgente do "Enf. Luís Cabral";
6. Consultar o histórico de pedidos e:
   (a) Visualizar o pedido do paciente "Carla Matos";
   (b) Retroceder para o histórico de pedidos (sem filtros filtros).

G.4 Resultados (medidas de eficácia e eficiência)

Cada uma das tarefas tem duas linhas que correspondem, respetivamente, à primeira e segunda fase de testes de usabilidade. É notória a melhoria verificada de uma fase para a outra.

G.5 Resultados de satisfação

Após a realização dos testes de usabilidade, os utilizadores foram convidados a preencher um formulário com o objetivo de medir a satisfação sentida durante a utilização do sistema. A escala siuta-se entre 1 e 5, com:

- **1** − Discordo plenamente;
- **5** − Concordo plenamente.

Os gráficos da **esquerda** correspondem à **primeira fase**, enquanto que os da **direita** correspondem à **segunda fase**, com:

- Eixo do xx: nível de concordância;
- Eixo do yy: número de inquiridos.

Tarefa	Tempo médio (mm:ss:ms)	Nº de reclamações	Nº total de ajudas	Nº total de erros	Utilizadores que terminaram com sucesso
1	00:24,044	1	2	0	5
	00:09,575	0	0	0	5
2	00:17,830	0	0	0	5
	00:12,745	0	0	0	5
3	01:03,320	3	0	3	3
	00:30,770	3	0	2	4
4	00:18,970	0	0	0	5
	00:11,618	1	0	0	5
5	00:26,316	1	0	0	5
	00:20,320	0	0	1	5
6	00:20,460	0	0	0	5
	00:16,623	0	0	0	5

Figura G.2: Resultados de eficácia e eficiência

A aplicação é fácil de usar

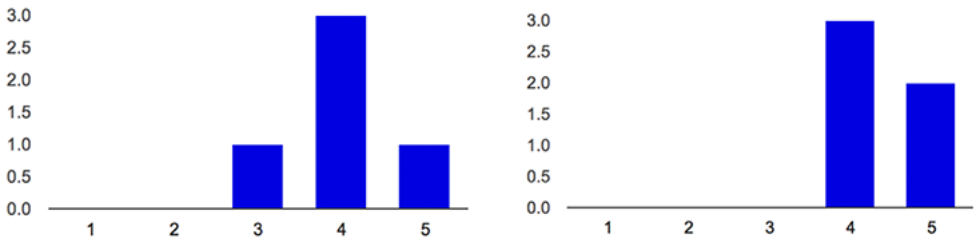


Figura G.3: Distribuição de respostas à Questão 1

Considero que aprendi rapidamente a utilizar a aplicação
A organização da interface é intuitiva

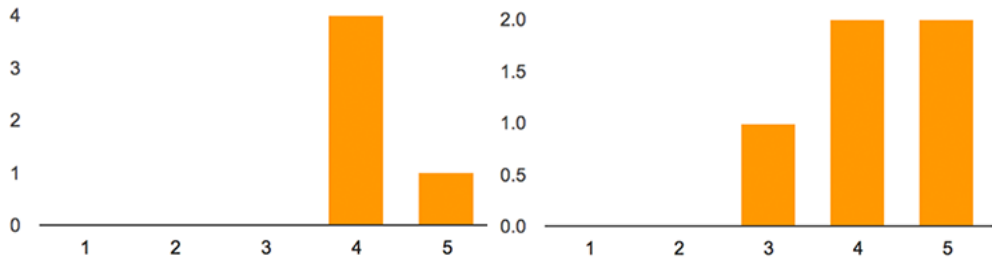


Figura G.4: Distribuição de respostas à Questão 2

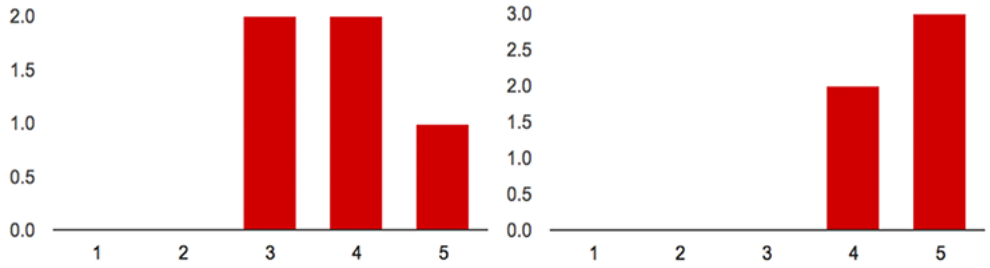


Figura G.5: Distribuição de respostas à Questão 3

Posso afirmar que o meu grau de satisfação geral com esta aplicação é positivo

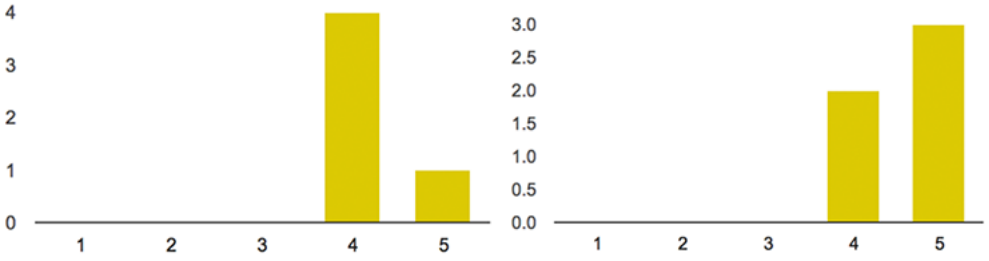


Figura G.6: Distribuição de respostas à Questão 4

G.6 Testes sem utilizadores

G.6.1 Tarefas

Estas tarefas são parcialmente baseadas nas criadas para a fase de testes com utilizadores (Apêndice G.3).

1. Criar um novo registo com o preenchimento de todos os campos pedidos, utilizando o par de números "123" – "321" para, respetivamente, identificação civil e fiscal e o email "test@m1.com";

2. Efetuar *login* no sistema com a conta criada na Tarefa 1;

3. Aceder à listagem de "Pedidos pendentes" a partir do menu lateral e procurar o pedido de autorização da "Enf. Anabela Saúde". Forçar *update* de respostas;

4. Aceder à listagem de "Pedidos pendentes" a partir do menu lateral, procurar e escolher o pedido de autorização da "Enf. Anabela Saúde", ler o seu conteúdo e aceitar;

5. Aceder à listagem de "Pedidos pendentes" a partir do menu lateral, procurar e escolher o pedido de autorização do "Enf. Carlos Mateus", ler o seu conteúdo e rejeitar, justificando;

6. Aceder à listagem de "Histórico de pedidos" a partir do menu lateral e procurar o pedido de autorização do "Enf. Carlos Mateus";

7. Aceder à listagem de "Histórico de pedidos" a partir do menu lateral e filtrar com o intervalo de datas entre "12−05−2015" e "13−05−2015". Regressar para a listagem sem filtros;

8. Aceder às "Configurações" a partir do menu lateral e, em seguida, "Guardar".

G.6.2 Graus de severidade [81, 82]

Cada um dos problemas de usabilidade encontrados é classificado de acordo com uma escala entre 0 e 4, onde:

1. **Problema estético:** a resolver apenas se existir tempo e recursos;

2. **Problema de usabilidade menor:** problema de baixa prioridade;

3. **Problema de usabilidade maior:** importante que seja corrigido, pelo que deve ter uma prioridade elevada;

4. **Problema de usabilidade catastrófico:** correção obrigatória antes de lançar para produto.

Estes pontos têm em conta a frequência com que o problema ocorre, o impacto no desempenho dos utilizadores e a sua persistência.

G.6.3 Avaliação heurística [81, 82, 83]

O presente Apêndice descreve as dez heurísticas de Nielsen, apresentando um teste por cada sub-heurística, baseado em [83].

H1. Visibilidade do estado do sistema

O utilizador deve ser informado sobre o que está a acontecer, através de informação apropriada em resposta à sua ação.

1. ***feedback*:** o sistema oferece resposta a todas as ações do utilizador?

2. **informação e identidade:** existe referência sobre a proteção de dados de carácter pessoal?

3. **tempo de resposta:** existe algum *delay* observável?

4. **seleção de dados:** existe *feedback* visual quando os objetos são selecionados?

H2. Correspondência entre o sistema e a realidade

O sistema deve utilizar palavras, frases e conceitos familiares ao utilizador, seguindo as convenções do mundo real.

1. **metáforas:** os ícones são específicos e familiares?
2. **navegação:** existem muitos níveis de navegação?
3. **menu:** as escolhas efetuadas correspondem aos significados atribuídos?
4. **simplicidade:** o idioma apresentado é o mesmo do utilizador?

H3. Livre arbítrio do utilizador

O utilizador é quem controla o sistema, realizando as tarefas pela ordem que entender, o que pode levar a erros que necessitem de "saídas"dos estados indesejáveis.

1. **interface explorável:** existem saídas claramente identificadas?
2. **processo de confirmação:** o utilizador é informado das ações com consequências mais drásticas?
3. **cancelar:** o utilizador pode cancelar operações que se encontram em progresso?
4. **controlo de menu:** os menus são equilibrados?

H4. Consistência

O utilizador não deve ter dúvidas sobre o significado de palavras ou ações, de forma a minimizar o fator surpresa.

1. **design:** foi evitado o uso excessivo de letras maiúsculas?
2. **convenção de nomes:** existe consistência no nome dos objetos do sistema?
3. **menu:** os títulos correspondem às escolhas do utilizador?

H5. Prevenção de erros

A interface deve evitar a ocorrência de erros.

1. as caixas de texto indicam o número de caracteres disponíveis?
2. as escolhas do menu são distintivas e exclusivas?

H6. Reconhecer em vez de lembrar

O sistema deve minimizar a carga cognitiva, ao utilizar elementos de diálogo que auxilie nas escolhas do utilizador.

1. **minimizar memória utilizada:** os controlos de uma tarefa estão agrupados e refletem o seu significado no desempenho da tarefa?
2. **uso de pistas visuais:** é usada a cor em conjunto com outra pista visual?
3. **menu:** os itens de menu inativos encontram-se omitidos (ou acinzentados)?

H7. Flexibilidade e eficiência

O sistema deve ser acessível para principiantes e, ao mesmo tempo, flexível o suficiente para possibilitar a personalização das ações mais frequentes por utilizadores mais experientes.

1. **pesquisa:** o sistema dispõe de caixas de procura?
2. **navegação:** existem atalhos que permitem navegar mais rapidamente para um dado tópico?

H8. Design estético e minimalista

Os diálogos da interface devem ser adequados e com informação relevante. Menos é mais.

1. **ícones:** foi evitado o detalhe excessivo no detalhe dos ícones?
2. **menus:** os títulos são breves o suficiente para comunicar?

H9. Ajudar o utilizador a reconhecer, diagnosticar e recuperar de erros

As mensagens de erro devem ser claras e simples, a fim de ajudar o utilizador a resolve-los, ao invés de o intimidar.

1. o utilizador é sinalizado do erro que cometeu? É orientado para a resolução?

H10. Ajuda e documentação

A interface deve ser fácil e intuitiva, evitando o recurso a documentação. Caso seja necessária, deve ser facilmente localizada, contextual e concisa. **Não aplicada.**

G.6.4 Resultados

Na Tabela G.2, além da identificação dos testes que apresentaram problemas, é feita uma descrição, quando ocorre, uma possível resolução e a proposta de severidade associada.

Teste: H1.1 **Problema:** Falta resposta para uma dada ação. **Quando:** Tarefa 3 **Descrição:** Como ao carregar no botão para forçar possíveis respostas pendentes, não é dada nenhuma informação, o utilizador poderá ser forçado a executar algo de forma errada. **Correção:** Acrescentar um alerta com um diálogo claro do resultado obtido. **Severidade:** 4
Teste: H1.1 **Problema:** Falta resposta para uma dada ação **Quando:** Tarefa 8 **Descrição:** Ao carregar no botão "Guardar", não é dada nenhuma confirmação ao utilizador, podendo induzi-lo em erro sobre o estado do sistema. **Correção:** Acrescentar um diálogo a confirmar sucesso da operação. **Severidade:** 3
Teste: H1.2 **Problema:** Falta de referência para o tratamento de dados pessoais **Descrição:** O utilizador não é elucidado sobre o facto de a aplicação fazer uso de dados pessoais. **Correção:** Acrescentar um ecrã com uma referência para questões legais **Severidade:** 3
Teste: H5.1 **Problema:** Falta de sinalização do número de caracteres disponíveis. **Quando:** Tarefas 1 e 2 **Descrição:** Os campos de introdução/edição de texto não indicam o número de caracteres disponíveis nem sinalizam o limite. **Correção:** Acrescentar um diálogo a informar o limite de caracteres atingido. **Severidade:** 3
Teste: H7.1 **Problema:** Falta caixa de procura para procurar informação em listas. **Quando:** Tarefas 3 e 6 **Descrição:** Nos ecrãs passíveis de ter muita informação, como não existem caixas de pesquisa, o utilizador levará mais tempo para encontrar o que pretende **Correção:** Adicionar caixas de pesquisa nas listagens de pedidos **Severidade:** 2

Tabela G.2: Problemas identificados com a avaliação heurística