Mestrado em Engenharia Informática
Estágio
Relatório Final

Sensor Care - Sistema de apoio à prestação de cuidados remotos de saúde

Nuno Alexandre Almeida de Amicis Rebelo
nunoar@student.dei.uc.pt

Orientadores:
Professor Doutor Carlos Fonseca
Engenheiro Alcides Marques

31 de Agosto de 2012

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

Agradecimentos

Gostaria de agradecer a todos os intervenientes na minha formação pessoal e académica, particularmente ao Professor Doutor Carlos Fonseca e ao Engenheiro Alcides Marques pelo acompanhamento essencial dado ao longo do desenvolvimento deste projecto; ao Professor Doutor Carlos Bento e ao Professor Doutor Paulo Rupino pelos conselhos pertinentes e críticas construtivas; ao Engenheiro Gouveia Leal pelos conselhos inspiradores e incentivos, bem como ao Mestre João Quintas pelo aconselhamento e acompanhamento próximos; ao Miguel Machado pelo contributo ímpar e disponibilidade que sempre demonstrou, a toda a minha equipa, David Cardoso, David Francisco, Diana Guardado, Diogo Laginha, Gonçalo Ferrão, Pedro Catré e Ricardo Vitorino, e à minha família e amigos de sempre, que estão ao meu lado nos bons e maus momentos.

Resumo

O portal We.Can, inserido na iniciativa TICE.Healthy, é um projecto mobilizador a nível nacional que foi pensado com o objectivo de criar um ecossistema de aplicações e serviços, onde pacientes, família e profissionais de saúde possam cooperar das necessidades do dia-a-dia.

O presente estágio foi elaborado para concretizar a visão do que poderá ser uma aplicação do portal: um sistema de monitorização e apoio à prestação de cuidados remotos de saúde, validando a plataforma que o suporta. Assim, foi criada um produto piloto que contempla uma aplicação móvel que agrega, disponibiliza e permite visualizar sinais vitais e uma aplicação web que os disponibiliza em tempo real.

Este relatório tem como objectivo dar a conhecer o trabalho realizado no ano lectivo de 2011/2012 pelo estagiário, para a disciplina de "Dissertação/Estágio", do Mestrado em Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Palavras chave: Saúde, cuidados remotos, tempo real, monitorização, aplicação de *e-health*.

Conteúdo

API	*Application Programming Interface*
Ajax	*Asynchronous JavaScript and XML*
DCP	*Device Control Portocols*
ECG	Electrocardiograma
FCTUC	Faculdade de Ciências e Tecnologia da Universidade de Coimbra
HTTP	*Hypertext Transfer Protocol*
IPN	Instituto Pedro Nunes
MEI	Mestrado de Engenharia Informática
PPS	Produto, Processo ou Serviço
SI	*Socket Intance*
TA	*Target Adapter*
TCP	*Transmission Control Protocol*
TDM	Target Discovery Module
UC	Universidade de Coimbra
UCH	*Universal Control Hub*
UI	*User Interface*
UIPM	*User Interface Protocol Modules*
UIS	*User Interface Socket*
UPnP	*Universal Plug and Play*
URC	*Universal Remote Console*
XML	*Extensible Markup Language*
XMPP	*Extensible Messaging and Presence Protocol*

Tabela 1: **Acrónimos**

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

O presente relatório visa apresentar o trabalho realizado durante o ano lectivo 2011/2012, no âmbito do projecto "TICE.HEALTHY – We.Can", para a disciplina "Estágio/Dissertação" do Mestrado em Engenharia Informática (MEI) da Faculdade de Ciências e Tecnologia da Universidade de Coimbra (FCTUC).

1.1 Enquadramento

O "TICE.Healthy – Sistemas de Saúde e Qualidade de Vida" tem como missão potenciar a presença das empresas e organizações portuguesas, e em particular as englobadas no projecto, nos mercados globais das áreas estratégicas em que está envolvido. Assim, o projecto pretende desenvolver, integrar e testar abordagens tecnológicas inovadoras, que sirvam de base a novos produtos e serviços para os mercados associados ao vector "Saúde e Qualidade de Vida".

O subprojecto em que o estágio se insere, identificado como "Produto, Processo ou Serviço (PPS) 1", é liderado pelo Instituto Pedro Nunes (IPN) e pretende catalisar as empresas do consórcio para a criação de serviços na área da saúde, utilizando como suporte a infra-estrutura Internet, convergindo assim para a disponibilização deste tipo de soluções no mercado.

O PPS 1 está a desenvolver um portal que disponibilizará, de forma centralizada, serviços e produtos orientados à prestação de cuidados de saúde a pessoas que procuram por cuidados informais e por serviços que têm por objectivo promover uma melhor qualidade de vida. Este portal - que sustentará a distribuição, gestão e disponibilização de aplicações - e a plataforma que o sustenta, são designados por We.Can.

O sub-projecto Sensor Care consiste numa primeira aplicação para o portal e contempla a agregação de dados de sensores vitais através de um dis-

positivo *Android*, disponibilizando-os em tempo real. Tendo em conta que a plataforma que sustentará o portal é composta por diversos módulos, o Sensor Care será um protótipo de utilização dos mesmos, possibilitando validar a plataforma e os seus componentes, bem como demonstrar o seu potencial funcional para os actuais e futuros parceiros. Para além disso, será ainda primeira fonte de dados a alimentar o We.Can.

1.2 Motivação

Numa altura em que os gastos com a saúde aumentam mais depressa do que o crescimento económico [1], a temática da saúde revela-se uma das principais prioridades europeias, sendo uma das seis áreas de mercado que a Comissão Europeia considera prioritárias [2].

O envelhecimento da população, que tipicamente se traduz num aumento da necessidade de cuidados de saúde, aliado aos desafios que um mercado cada vez mais competitivo traz, leva a que novas soluções surjam e que este mercado esteja em constante crescimento.

Estima-se que haja cerca de 5000 pequenas e médias empresas na União Europeia a operar no subsector de *e-Health*. A importância do sector da saúde revela-se, também, por empregar 10% da população activa na União Europeia, correspondendo a 9% do Produto Interno Bruto. Calcula-se, ainda, que em 2020 os gastos em saúde atinjam 16% do Produto Interno Bruto dos países pertencentes à Organização para a Cooperação e Desenvolvimento Económico, vulgo OCDE [2].

Estando inserido num sector tão importante, é motivador para o estagiário poder participar num processo de construção de soluções que podem fazer diferença para a sociedade. O projecto proporciona ainda ao estagiário a possibilidade de trabalhar num contexto real e inserido numa equipa ampla e multidisciplinar. Por outro lado, tanto as tecnologias como a metodologia de trabalho utilizadas são uma peça importante para a formação de um Engenheiro Informático.

1.3 Objectivos

Foi idealizado um primeiro cenário de utilização da plataforma We.Can para exemplificar o processo de levantamento de requisitos e especificação de funcionalidades. Este cenário é orientado ao desenvolvimento de projectos piloto que venham a ser construídos sobre a plataforma, bem como para servir de validação da mesma e primeiro exemplo integrador. O presente estágio visa

implementar o dito cenário.

Apresentam-se, de seguida, os principais objectivos do estágio:

- Criar uma aplicação móvel de agregação de sinais vitais;
- Desenvolver uma aplicação web de disponibilização de dados em tempo real;
- Elaborar uma arquitectura orientada à integração com os módulos da plataforma já existentes.

Sobre o projecto, podem-se ainda destacar os seguintes aspectos diferenciadores:

- É um sistema validador da plataforma;
- A aplicação móvel será a primeira a interagir com *hardware* da PLUX [3], um dos parceiros do projecto;
- Desenho do sistema tendo em conta o ecossistema onde se encontra (portal We.Can);
- Extensibilidade relativamente a outros serviços de aquisição de dados, tendo em contra outros cenários de saúde.

Concluíndo, o cenário desenvolvido deverá permitir demonstrar e validar o comportamento da plataforma We.Can ao nível do fluxo de dados e respectivas interfaces de comunicação. O seu propósito é tornar mais fácil o desenvolvimento e a integração de novos e inovadores serviços, orientados para a prestação de cuidados de saúde e comunicação.

Capítulo 2

Definição de requisitos

A definição de requisitos é uma etapa fundamental em projectos de engenharia de *software*. É importante que se faça uma análise atenta e uma documentação cuidada dos mesmos, para que o desenho e construção do sistema sejam validados e estejam alinhados com a visão das partes interessadas. Neste capítulo será descrito o cenário de implementação e será feita a análise e especificação de requisitos. A sua validação foi feita com a gestão do projecto.

Associado a este capítulo estão os anexos "Complemente de requisitos", que contém uma lista com a descrição completa dos requisitos funcionais e "Story tests" com a especificação das respectivas condições de aceitação. Os requisitos rasurados foram descartados sendo as razões para tal, também apresentadas no supracitado anexo.

2.1 Descrição do Cenário

O presente cenário de utilização, relativo à PPS1 do TICE.Healthy, permite demonstrar e validar o comportamento da plataforma We.Can ao nível do fluxo de dados e respectivas interfaces de comunicação.

Este cenário contempla a monitorização de sinais vitais de um utilizador, obtidos pela plataforma através de sensores disponilizados pela PLUX, nomeadamente, um sensor de temperatura periférica, um tríodo de electrocardiograma (ECG) e ainda um acelerómetro de três eixos, que permitem obter as medidas presentes na tabela 2.1:

A monitorização deve feita através de uma aplicação móvel para *tablet* Android e os dados enviados para o módulo PHR, denominado Generic Entity, sendo posteriormente diponibilizados por este módulo para serem visualizados em tempo real numa aplicação web disponível no portal We.Can,

Sensor	**Medidas(Unidades)**
Termómetro	Temperatura(ºC)
Tríodo de ECG	ECG (mV), Ritmo Cardíaco (batimentos por minuto)
Acelerómetro	Dispêndio energético (MET[4] e Counts[5])

Tabela 2.1: **Medidas disponibilizadas pelos sensores PLUX**

podendo ainda ser configurados alertas sobre eles.

Devido à inexistência de um módulo de permissões, foi decidido implementar um protótipo de gestão de permissões.

2.2 Requisitos funcionais

A especificação de funcionalidades foi feita com recurso a *user stories*[6], que serão usadas como forma de obter uma descrição sumária das funcionalidades pretendidas, focando-se no que é essencial. Estas devem ser escritas do ponto de vista de quem usa a aplicação e com o registo linguístico que este usaria.

Uma característica fundamental das *user stories* é o facto de responderem a três questões importantes na definição de funcionalidades:

- O que se pretende?
- Quem o pretende?
- Porquê?

A inclusão das respostas - ainda que superficiais - a estas questões, é essencial para a construção de um plano de desenvolvimento sustentado, que permita não só caracterizar aquilo em que se deve tornar o sistema, como também manter um registo actualizado das necessidades que motivaram a implementação de cada funcionalidade.

Para as *user stories* que contemplam interacção directa com o utilizador, pode ser utilizado o formato:

Enquanto ...<agente do sistema>
Quero ... <acção>
De forma a ...<proposta de valor>

Os requisitos de natureza mais arquitectural, que não estejam relacionados com o utilizador final, poderão ser descritos de forma livre, desde que deinifios de forma simples e concisa. Por exemplo:

Conversão dos ficheiros recebidos para XML

Para que todos os dados sejam guardados e representados pelo *standard*.

Após a redacção e caracterização das funcionalidades, importa contemplar as várias situações passíveis de acontecer no âmbito das mesmas - subprocessos, fluxos alternativos, erros, entre outros - e que deverão ser previstas ao nível da implementação. Estes cenários de operacionalização deverão ser descritos na forma de *story tests*, com recurso ao modelo *GIVEN-WHEN-THEN* (Dado que - Quando - Então). Numa vertente mais técnica, este passo permite a sua rápida e fácil integração com o código fonte em formato de testes de aceitação.

A concretização de funcionalidades em *story tests* permite ainda criar uma base sólida para a definição de requisitos de interface.

A prioritização das *user stories* será feita usando o modelo de MoSCoW[7], popular por usar palavras com significado para o efeito. A escala é apresentada de seguida, com a respectiva descrição do que esta traduz, em português:

- M - *MUST have this* - Representa um requisito necessário para que o projecto seja considerado um sucesso;

- S - *SHOULD have this if at all possible* - Traduz-se num item de prioridade elevada e que se possível deve ser incluído no produto;

- C - *COULD have this if it does not affect anything else* - Caracteriza um requisito que seria desejável ter, mas que não é fundamental e deve ser incluído apenas se o tempo o permitir;

- W - *WON'T have this time but would like in the future* - Item que as partes interessadas acordaram que não será presentemente implementado, mas que deverá fazer parte de uma versão futura.

Requisitos sobre a aplicação móvel

User story	Resumo	Prioridade
MOBILE-1	Autenticação/Autorização na plataforma	MUST
MOBILE-2	Listar utilizadores	MUST
MOBILE-3	Escolher utilizador	MUST
MOBILE-4	Mudar de utilizador	MUST
MOBILE-5	Seleccionar sensores	MUST
MOBILE-6	~~Configurar agregador~~	MUST
MOBILE-7	Iniciar aquisição	MUST
MOBILE-8	Terminar aquisição	MUST
MOBILE-9	Aceder ao portal	MUST
MOBILE-10	Sair da aplicação	SHOULD

Requisitos sobre a aplicação web

User story	Resumo	Prioridade
WEB-1	Listar utilizadores e instituições	MUST
WEB-2	Dados em tempo real	MUST
WEB-3	Configuração de alertas	SHOULD
WEB-4	Disparo de alertas	SHOULD
WEB-5	~~Histórico de dados~~	MUST
WEB-6	FAQ	COULD
WEB-7	Adicionar instituição	COULD
WEB-8	Remover instituição	COULD
WEB-9	Adicionar permissões a utilizador	COULD
WEB-10	Adicionar profissional à instituição	COULD
WEB-11	Adicionar paciente à instituição	COULD
WEB-12	Remover profissional da instituição	COULD
WEB-13	Remover paciente da instituição	COULD
WEB-14	Alterar permissões de utilizador	COULD
WEB-15	Alterar permissões de instituição	COULD

2.3 Requisitos de segurança

Requisitos sobre a aplicação web

ID	Resumo	Prioridade
SEG-W-1	Autenticação no portal	MUST
SEG-W-2	~~Ligações seguras~~	MUST
SEG-W-3	Sessões assinadas	MUST
SEG-W-4	Impedir SQL-Injection	COULD

Requisitos sobre a aplicação móvel

ID	Resumo	Prioridade
SEG-M-1	Autenticação no portal	MUST
SEG-M-2	~~Ligações seguras~~	MUST

2.4 Requisitos de privacidade

2.4.1 Aplicação Web

ID	Resumo	Prioridade
PRI-W-1	Autorização de acesso a dados do utilizador	MUST

2.4.2 Aplicação móvel

ID	Resumo	Prioridade
PRI-M-1	Autorização de acesso a dados do utilizador	MUST

2.5 Requisitos de usabilidade

2.5.1 Aplicação Móvel

ID	Resumo	Prioridade
USE-M-1	Lidar com a latência de submissão	SHOULD

2.5.2 Aplicação web

ID	Resumo	Prioridade
USE-W-1	*Layout* fluído e responsivo	SHOULD
USE-W-1	Usar ícones vectoriais para botões	SHOULD
USE-W-1	Submissões por AJAX	SHOULD

2.6 Requisitos de desempenho para a aplicação web

ID	Resumo	Prioridade
PER-W-1	Combinar ficheiros JavaScript	MUST
PER-W-1	Combinar folhas de estilo CSS	MUST
PER-W-1	*Minificar* ficheiros JavaScript	MUST
PER-W-1	*Minificar* folhas de estilo CSS	MUST

2.7 Requisitos tecnológicos

Foi decidido, em conjunto com a gestão do projecto, que a aplicação fosse implementada com vista à utilização em *tablets*, tendo sido escolhida a plataforma Android.

Para usar a aplicação é, assim, necessário um dispositivo Android com a sistema operativo *Honeycomb*[8] 3.2 ou superior e que possua interface Bluetooth e ligação à internet.

2.8 Atributos de qualidade do sistema

Há que definir critérios que se pretendem cumprir relativos a atributos de qualidade relativos a objectivos arquitecturais importantes para o sistema a construir.

2.8.1 Extensibilidade e interoperabilidade

O facto de o projecto desenvolvido durante o estágio apenas integrar com parte do que serão os módulos definitivos da plataforma We.Can traz a necessidade que se criarem camadas de abstracção entre as chamadas às APIs e o tratamento de dados, através da aplicação do "design pattern" Façade.

Deve ser analisado o Dossia, plataforma *open-source*, de modo a criar estruturas que sejam compatíveis com o seu modelo de dados. Isto facilitará a adaptação de aplicações que sigam o exemplo do Sensor Care (como já está acontecer com um dos parceiros do projecto) à interoperabilidade com o Dossia, e permite contribuir para que a especificação do PHR, não a cargo do estagiário, seja mais sólida.

2.8.2 Manutenibilidade

O sistema deve ser implementado prestando especial atenção aos *coding standards* e convenções, a fim de escrever código manutenível. As normas seguidas são as sugeridas para as linguagens Java e JavaScript. Seguindo os *coding standards* e convenções conhecidas, aliados a uma boa documentação, não só se consegue garantir uma melhor manutenção do código produzido, como também melhorar a legibilidade, tornar o *refactoring* mais eficiente, facilitar o *debugging* e melhorar a portabilidade.

Devem também ser seguidas as boas práticas de mensagens de *commit* para o sistema de controlo de versões Git usadas em projectos *open-source*, e aplicada convenção Semantic Versioning[9], para versionamento de código, distinguindo iterações de *release* parciais de modo lógico.

O arquitectura do *back-end* deverá ser estruturado de acordo com a *design pattern* MVC, uma vez que encoraja os developers a dividirem o seu código em blocos auto-contidos e *loosely-coupled*, com objectivos bem definidos – desde gestão de dados, passando por lógica de negócio e código relacionado com a interface com o utilizador.

O sistema deve também ter um mecanismo de *deployment* automatizado.

2.8.3 Permutabilidade e modularidade

Devem ser utilizada uma *template engine* para as vistas da aplicação web, de modo a simplificar a sua reutilização de vistas.

2.8.4 Robustez

A aplicação web deve ser tolerante a falhas na ligação aos componentes com que interage, concretamente no que toca à ligação ao módulo PHR.

A aplicação móvel deve analisar o cumprimento dos seus requisitos tecnológicos.

Capítulo 3

Arquitectura e desenho

Neste capítulo é formalizada a arquitectura do sistema desenvolvido, de acordo com várias perspectivas, bem como é explicitada a lógica, os pressupostos e implicações das decisões que foram tomadas.

3.1 Problema de engenharia

No âmbito deste estágio, colocou-se o desafio de produzir um sistema que:

- agregasse dados vitais provenientes de sensores da PLUX;
- fornecesse dados à plataforma We.Can;
- disponibilizasse dados em tempo real;
- interagisse com diversos módulos descritos na proposta do projecto TICE.Healthy para a arquitectura da plataforma We.Can;
- fosse focada num dos vectores de foco do TICE.Healthy - "Controlo Remoto e Sensores".

Estes pressupostos foram pensados com o fim de demonstrar o potencial da plataforma, para entidades internas e externas ao projecto. Ao mesmo tempo, valida a sua arquitectura, permitindo potenciar o desenvolvimento dos referidos módulos e dando-lhes um uso efectivo, contribuindo para um produto mais robusto.

A figura 3.1 representa esquematicamente a composição da plataforma We.Can nos seus diferentes módulos, que são propostos no projecto. O portal prevê suporte para dois tipos de aplicações web, as *Packaged Apps* e

as *Hosted Apps*. As primeiras consistem em código-fonte HTML, CSS, JavaScript, nos *assets* associados às páginas HTML, o ficheiro de manifesto e respectivos ícones da aplicação, e a sua execução é inteiramente feita no *browser*; as segundas consistem em código-fonte existente num servidor remoto, num ficheiro de manifesto que possui metadados (como o URL para esse servidor, o título da aplicação e ainda informação adicional, como permissões de acesso) e nos ícones da mesma. O proposto é que o Sensor Care consista numa *Hosted App*.

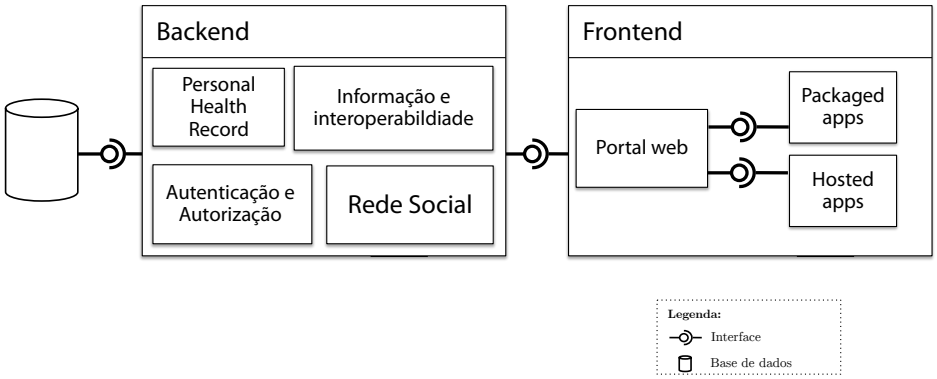

Figura 3.1: **Arquitectura de alto nível do We.Can**

A dimensão do projecto, as interacções complexas que lhe são inerentes, e essencialmente a fase precoce em que este se encontrava aquando do início do estágio, trouxeram bastante incerteza quanto à definição exacta do produto final. Porém, essa incerteza foi trabalhada de modo a se chegar a uma solução sólida e concreta. Numa descrição simplista, o sistema será constituído por uma aplicação Android que adquira sinais de um agregador de sinais PLUX e por uma aplicação web que os permite visualizar e com alertas associados.

Inicialmente estava prevista a integração de uma instância do Universal Control Hub[10], como ponte entre o agregador de sinais PLUX, um dos parceiros do projecto, e a aplicação web. Contudo, dado que o desenvolvimento deste se verificou estar numa fase demasiado embrionária, a ideia foi abandonada, passando, em alternativa, a ser responsabilidade da aplicação móvel desenvolvida fazer essa mesma ponte.

Outro factor que teve de ser contornado foi a inexistência de um módulo de permissões. A solução acordada foi a implementação de um mecanismo básico de permissões que o possa substituir a título demonstrativo. As razões desta tomada de decisão prendem-se com o ter um produto completo e usável, em por outro lado, com o facto de um sistema de permissões integrado num contexto de saúde ser um assunto de complexidade e extensão elevadas, que

não seria concordante com o contexto do estágio. Além do mais, o sistema de permissões é um módulo cuja responsabilidade de implementação não está, à data da escrita do relatório, ainda definida.

Para compreender a evolução final do cenário implementado, o seu impacto ao nível do sistema, e as tomadas de decisão efectuadas é necessário proceder à descrição da arquitectura.

3.2 Descrição da arquitectura

A arquitectura encontra-se descrita em dois níveis de abstracção distintos:

- Nível 0, correspondente a uma visão geral do sistema, sendo apresentadas as suas interacções com componentes externos;
- Nível 1, que reflecte uma visão mais detalhada dos componentes internos do sistema.

3.2.1 Visão de nível 0

Perspectiva estática

A figura 3.2 apresenta uma visão de alto nível estática, com a representação das interacções a entidades externas e bases de dados.

Num ponto de vista macroscópico o sistema comunica com várias entidades externas. Para começar, quando um utilizador acede com o seu *browser* à aplicação, ou quando inicia a aplicação móvel, é necessário estar autenticar-se na plataforma We.Can, bem como dar autorização à aplicação para que esta aceda aos seus dados, interagindo portanto com o servidor OAuth.

Usando a aplicação móvel e uma vez adquiridos os sinais do agregador de sinais PLUX pela aplicação móvel, estes são enviados para o módulo de Personal Health Record (PHR), presentemente denominado Generic Entity, desenvolvido pelo estagiário Pedro Catré, através da sua interface REST, e daí, são dispobilizados através de XMPP, estando a aplicação web à escuta. Caso haja utilizadores com permissões para visualizar esses dados a utilizar a aplicação, estes são redireccionados para o respectivo *browser*.

De referir ainda que, de modo a garantir uma experiência de uso uniforme do portal e das suas aplicações, as páginas geradas pelo aplicação web Sensor Care comunicam com o *iframe* do Portal em que estão inseridas através do método postMessage. Tal é conseguido através da inclusão de um ficheiro Javascript no código da página. Isto permite, por exemplo, que os dois *iframes* se comportem como uma única página e tenham um comportamento

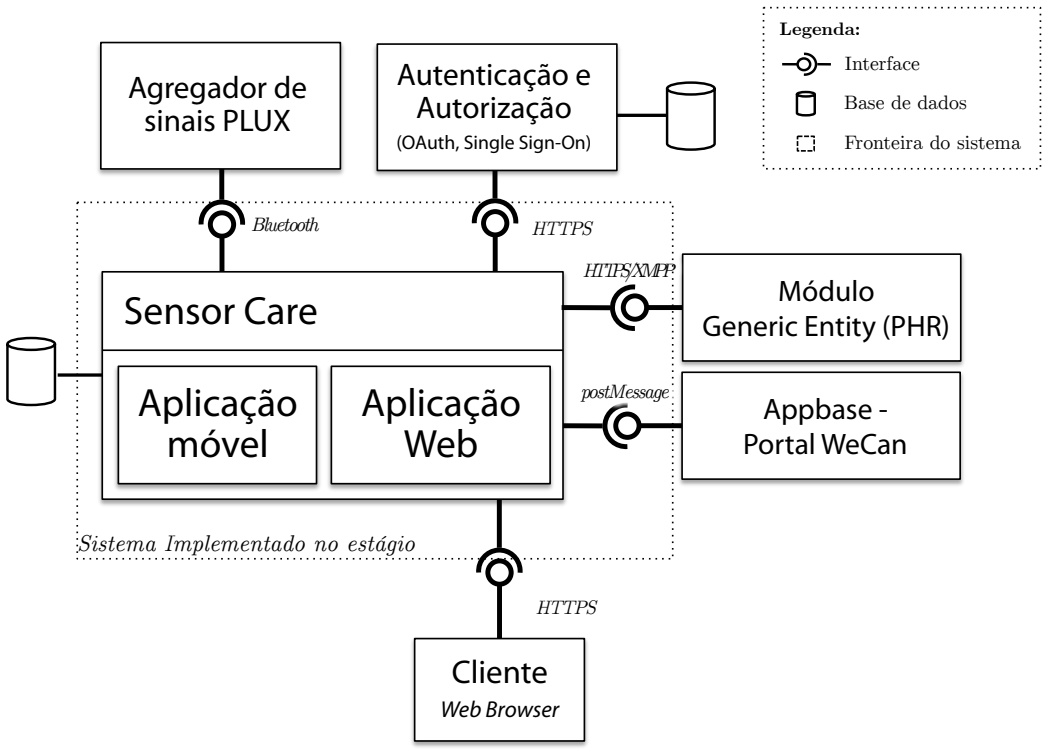

Figura 3.2: **Arquitectura de nível 0 - perspectiva estática**

responsivo, ou seja, que o tamanho dos *iframes* se reajuste à alteração do tamanho da janela do *browser*.

A interacção entre o sistema e uma base de dados está relacionada com o módulo de permissões.

Perspectiva de *deployment*

A figura 3.3 apresenta a perspectiva de *deployment* de nível 0, em que se contextualiza o sistema a nível físico, identificando os agentes externos que com ele comunicam.

A arquitectura We.Can baseia-se no paradigma orientado a serviços, sendo isso notório na figura 3.3, já que os diferentes módulos com que o Sensor Care interage se encontram em servidores distintos. A interacção com os mesmos serve o propósito de validar o seu funcionamento.

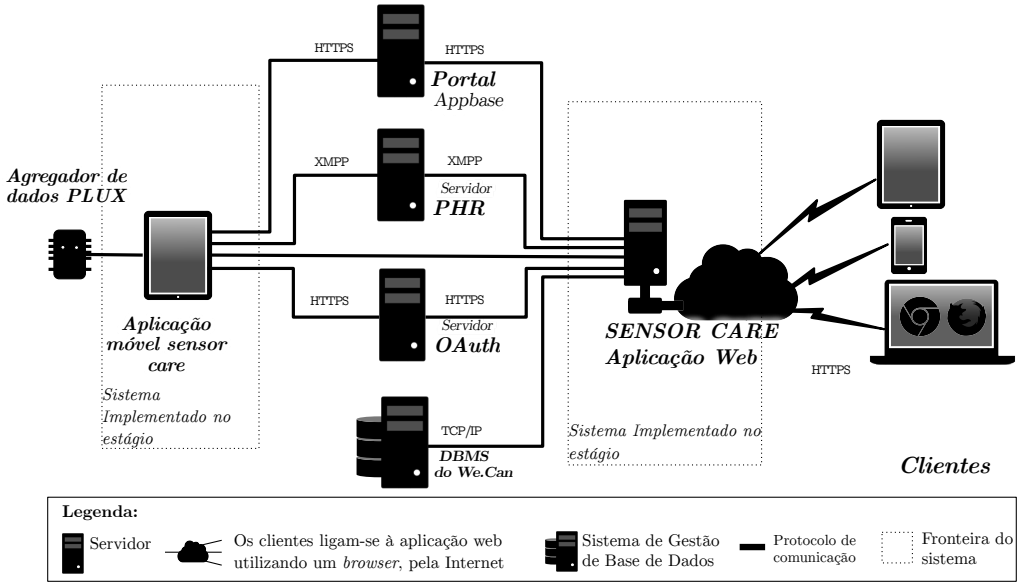

Figura 3.3: **Arquitectura de nível 0 - perspectiva física**

Perspectiva dinâmica

Numa prespectiva dinâmica do sistema, o fluxo de dados mais importante é o explicitado na figura 3.4.

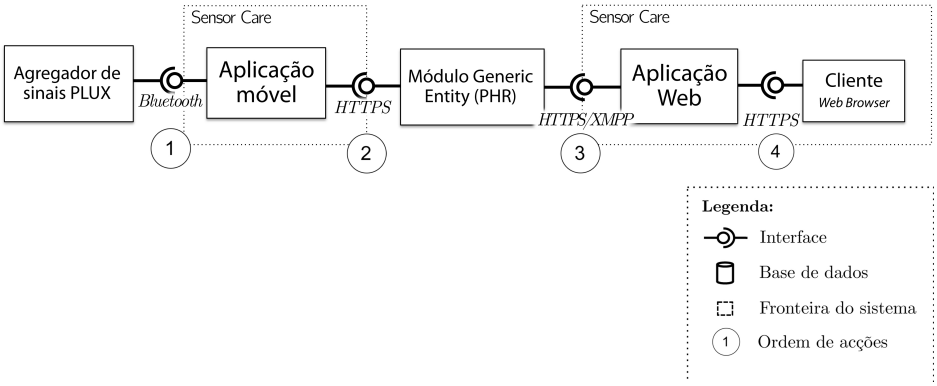

Figura 3.4: **Arquitectura de nível 1 - perspectica dinâmica**

Assim, uma vez adquiridos os sinais do agregador PLUX, pela aplicação Android através de Bluetooth, estes são enviados ao módulo Generic Entity, que por sua vez os disponbiliza à aplicação Web por XMPP, sendo que esta

finalmente os envia para o *browser*.

3.2.2 Visão de nível 1

Nesta secção pretende-se descrever os componentes constituintes do sistema. A visão de nível 1 identifica os seus módulos internos, fluxos e interacções.

Perspectiva estática

Dada a complexidade do sistema, na figura 3.5 apresentam-se as interacções internas entre os vários módulos do Sensor Care com uma maior granularidade. Tal como descrito anteriormente, há duas partes fundamentais do sistema com contextos distintos, a aplicação web e a aplicação móvel. A descrição de cada de uma e dos respectivos módulos é feita de seguida.

Sistema Web - Quando um pedido chega ao *router* da aplicação, este é encaminhado para o módulo da API, ou para a aplicação web, responsável pelo *front-end*. O *Controller*, *View* e *Models* correspondem aos módulos de uma aplicação construída segundo a arquitectura Model-View-Controller (MVC), garantindo a separação das diferentes lógicas aplicacionais, e é responsável por gerar as páginas a enviar para o *browser*. No caso de um utilizador estar na secção de visualização de dados em tempo-real, e caso a aplicação receba dados através do ***Event Handler***, encaminhá-los-á para o *browser* através do ***Event Handler*** respectivo. Quando um utilizador se liga à página do Sensor Care, já autenticado no portal, é-lhe pedido que dê autorização à aplicação para aceder aos seus dados, o que é feito no módulo de *Autenticação/Autorização* que gere a conexão ao servidor de OAuth. O utilizador pode também gerir as suas permissões, tendo o módulo **API de permissões** sido desenvolvido com esse fim.

Aplicação móvel - Quando a aplicação móvel inicia, requer que o utilizador se autentique na plataforma We.Can e autorize o acesso aos seus dados, o que é feito através do módulo de **Autenticação/Autorização**, de seguida é apresentada uma lista de utilizadores sobre os quais este tem permissões para monitorizar dados, podendo escolher o que pretende. Isso sucede através do módulo de permissões, que faz a respectiva *query* ao *back-end* do sistema. Depois, é apresentada a lista de sensores disponíveis e a respectiva configuração. Uma vez iniciada a aquisição de sinais através da ligação Bluetooth com o agregador, a aplicação apresenta os valores recebidos no ecrã, e envia-os ao PHR através de uma chamada REST. No caso dos dados de electrocardiograma, estes são apenas enviados uma vez finda a aquisição, devido ao facto de tanto a PLUX, como médicos contactados, terem descartado a importância de estes serem visualizados em tempo real.

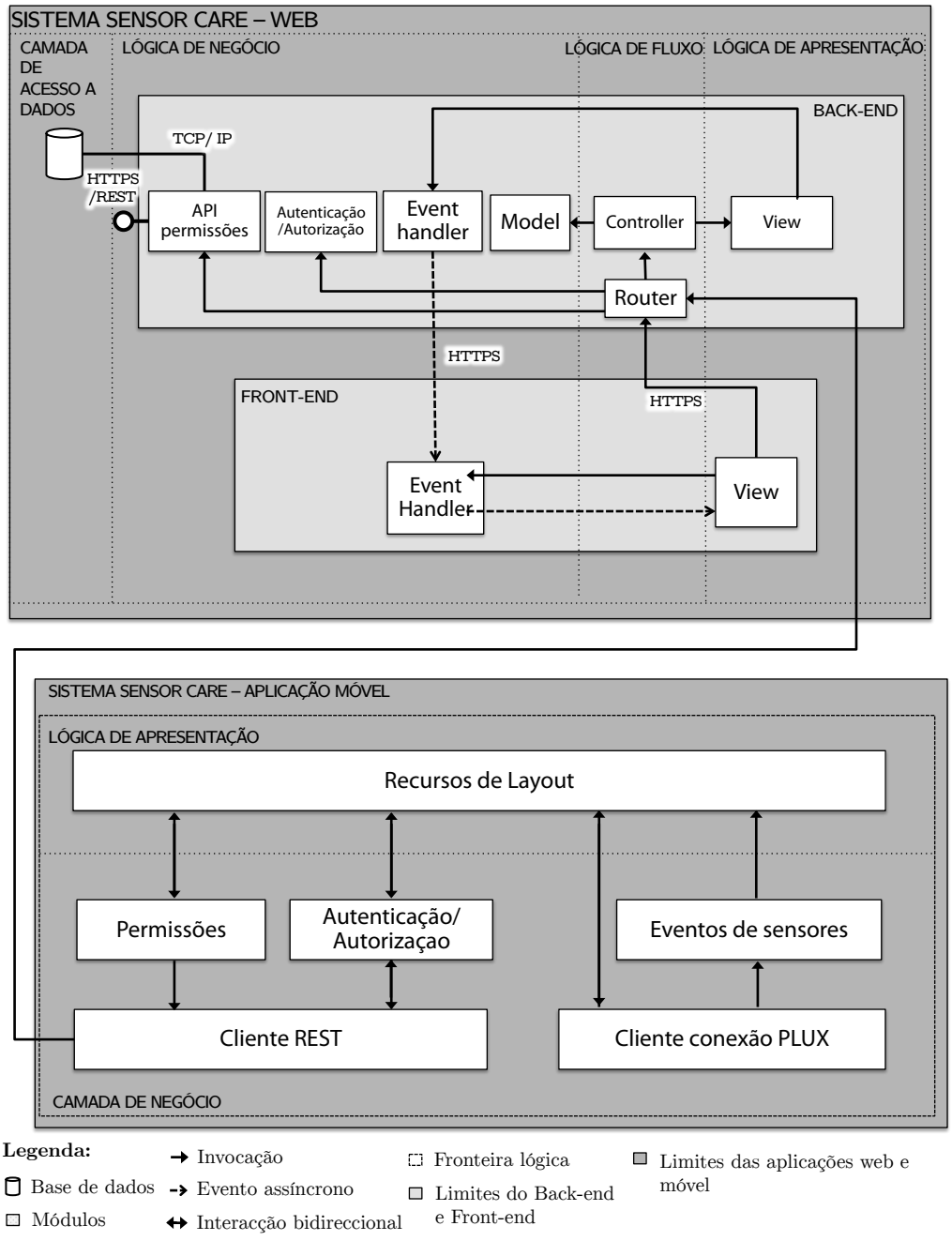

Figura 3.5: **Arquitectura de nível 1 - perspectiva estática**

Perspectiva dinâmica

Há ainda um componente cujo fluxo necessita de uma melhor análise, nomeadamente o fluxo de autorização de acesso a dados, através de OAuth. O protocolo OAuth permite a partilha de recursos privados armazenados num website (neste caso a plataforma We.Can) com outro website (no caso do Sensor Care, para aplicação móvel e web), sem que para isso o utilizador tenha de fornecer credenciais de acesso ao segundo website. No caso do portal, o utilizador, ao autenticar-se perante uma aplicação, está tecnicamente a autorizar a aplicação a ter acesso a dados do seu perfil de utilizador. É de referir porém, que é apresentada ao utilizador uma página em que é explícito que ele está a permitir esse acesso. A melhor forma de perceber o fluxo entre os actores, é observando a figura 3.6.

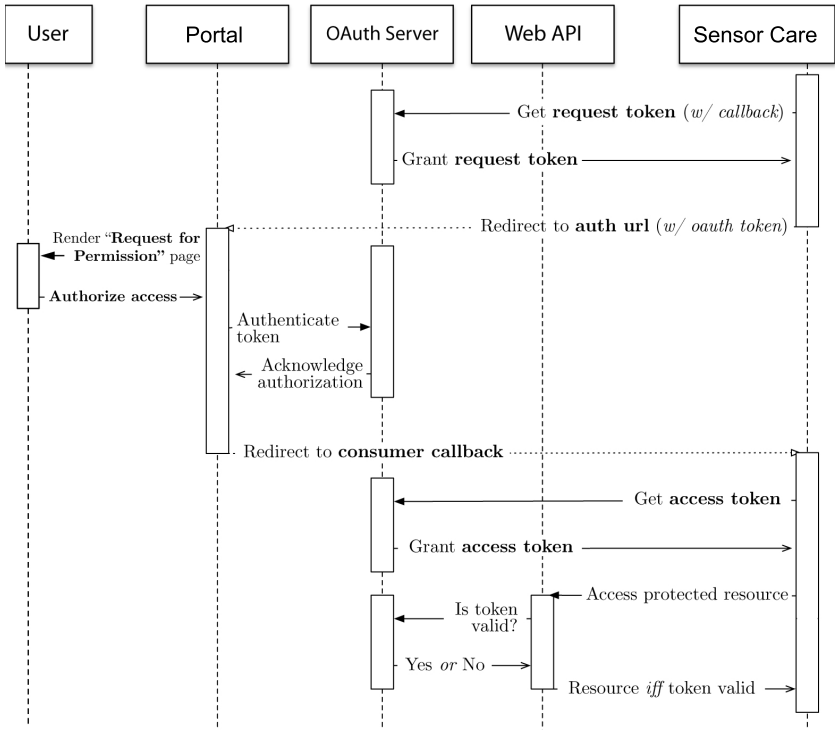

Figura 3.6: **Interacção entre actores no fluxo do OAuth1.0**

No âmbito do estágio, foram implementadas as interacções referentes ao actor Sensor Care.

3.2.3 Modelo de dados

Para os dados dos sensores foi, em primeiro lugar, analisado o modelo de dados da plataforma *open-source* Dossia, relativo a medições[11]. O objectivo foi maximizar a compatibilidade entre os dois modelos de dados de modo a permitir uma fácil integração do Sensor Care no Personal Health Record, e servir de exemplo a aplicações que se desenvolvam para o TICE.Healthy, dotando-as do mesmo potencial de integração. Assim, o modelo escolhido para armazenar os dados no *Personal Health Record*, apresentado na figura 3.7, inclui todos os campos obrigatórios para os dados desta natureza no Dossia.

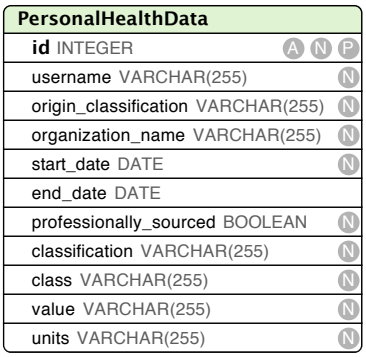

Figura 3.7: **Modelo de dados dos sinais recolhidos**

Dada a importância do modelo, na tabela 3.1 é feita uma breve descrição do significado de cada um dos campos.

No que toca às permissões, recorde-se que este módulo é meramente um protótipo que foi criado com o fim de tornar o estágio um produto usável, e ignora as particularidades dos actores envolvidos no projecto TICE.Healthy e dos requisitos legais que lhe são inerentes. O modelo de dados das permissões desenvolvido pode ser na figura 3.8. Uma descrição sucinta de cada uma das entidades é feita na tabela 3.2.

id	Identificador e chave primária do registo
username	Utilizador a quem pertence o registo
origin_classification	Indica o indivíduo ou sistema de origem do registo
organization_name	Nome da organização fonte dos dados
start_date	Início da medição
end_date	Fim da medição
professionally_sourced	Se é uma fonte profissional certificada ou não
classification	Tipo de dispositivo que dez a medição[12]
class	Tipo de medição
value	Valor da medição
unit	Unidades em que a medição foi feita

Tabela 3.1: **Modelo para dados dos sensores**

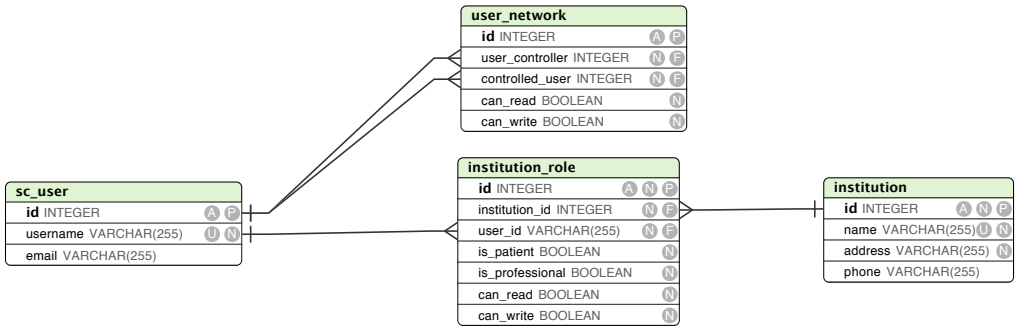

Figura 3.8: **Modelo de dados para permissões**

sc_user	tabela de utilizadores
user_network	tabela que relaciona utilizadores
institutional_role	tabela que relaciona utilizadores com instituições
institution	tabela de instituições

Tabela 3.2: **Entidades do modelo de dados para permissões**

Capítulo 4

Estado da arte

Neste capítulo é apresentado o trabalho de pesquisa e análise em que se baseia a construção do Sensor Care.

De seguida, foca-se um dos pontos nucleares do trabalho desenvolvido, e neste sentido foram avaliadas as várias alternativas para o processo de comunicação em tempo real entre servidor e *browser*.

4.1 Soluções na área da saúde

Os sistemas de saúde estão, cada vez mais, sob pressão para responder aos novos desafios que a tecnologia promove, aliados à maior facilidade de mobilidade e concorrência. Por conseguinte, há uma procura constante da optimização de processos, nomeadamente quanto à redução dos custos dos serviços, quer a nível de tempo, quer a nível financeiro. Assim, é cada vez mais comum encontrar aplicações que procuram, de uma forma ou de outra, aliar as novas tecnologias à prestação de cuidados de saúde.

O estudo apresentado de seguida servirá para compreender o contexto em que se inserem as aplicações ligadas a portais de saúde, bem como em contribuir para a fase de elaboração de requisitos, o que é importante para o Sensor Care em si, mas também para o TICE.Healthy, projecto em que está inserido, permitindo aumentar a base de conhecimento que o sustenta.

Para o efeito, foram seleccionados o Sapo Saúde, por ser uma solução nacional, e os portais internacionais com mais peso: o Microsoft Health Vault e o Dossia. Da análise foi excluído o Google Health, que, embora pertença a uma entidade de peso no mundo *Web*, é um serviço descontinuado.

Sapo Saúde

Direccionado a cidadãos portugueses, o Sapo Saúde é um portal da Portugal Telecom que pretende cativar pessoas com preocupações relacionadas com a saúde e bem estar e contém diversa informação relacionada com saúde, como doenças, nutrição ou estética. A tabela 4.1, resume a análise feita ao portal.

Sapo Saúde	
Região	Portugal
Descrição	Espaço que agrega informação sobre peso e nutrição, estética, saúde, medicina e bem estar. Tem uma área pessoal onde se pode registar informação pessoal relacionada com a saúde.
Positivo	• A área pessoal permite registar informação e consultar o seu histórico; • Contém uma secção de resposta a perguntas frequentes que dada a diversidade de pessoas que pode consultar o *site*, pode ser bastante importante.
Negativo	• Embora a navegação na zona pessoal esteja mais bem construída, as páginas de informação geral têm demasiados elementos gráficos e publicidade, dificultando a navegação; • Poucas funcionalidades e integração com outras entidades; • Há apenas um perfil de utilizador. Todos interagem com o portal como pacientes.
Endereço	`http://saude.sapo.pt/`

Tabela 4.1: **Sapo Saúde - análise**

Microsoft Health Vault

O Microsoft Health Vault permite guardar informação relacionada com saúde e bem estar. Graças à aposta na interoperabilidade viabiliza a agregação de informação de diversos sensores, tipicamente através de uma aplicação própria. Uma descrição mais detalhada pode ser observada na tabela 4.2.

Microsoft Health Vault	
Regiões	Estados Unidos da América e Reino Unido
Descrição	Permite aceder a informação útil, integra com alguns sensores para a aquisição de dados, disponibiliza planos ligados à melhoria de saúde e bem estar. Permite a interligação com instituições e profissionais da área.
Positivo	• A secção pessoal permite monitorização através de sensores e a consultar do seu histórico de dados; • A plataforma interliga com uma vasta variedade de parceiros; • Possui secções explicativas com instruções direccionadas a pessoas com interesses e necessidades distintas.
Negativo	• A monitorização de sinais depende de uma aplicação que é exclusiva para sistemas operativos Windows; • Uso limitado aos Estados Unidos da América e Reino Unido.
Endereço	`http://www.microsoft.com/en-us/healthvault/`

Tabela 4.2: **Microsoft Health Vault - análise**

Dossia

Dossia é uma plataforma desenvolvida por um consórcio de empresas americanas, nomeadamente a AT&T, a Applied Materials, a BP America, a Cardinal Health, a Intel, a Pitney Bowes, a Sanofi-aventis, a Walmart, a Abraxis, a BioScience e a Vanguard Health Systems, com o objectivo de transformar os cuidados de saúde nos Estados Unidos da América, reduzindo o desperdício e desenvolvendo um registo controlado pelos utilizadores[13]. Focado numa primeira fase nos seus próprios funcionários, o objectivo era que estes pudessem tomar decisões mais informadas e fundamentadas para a sua saúde e do seu agregado familiar. O Dossia é baseado na tecnologia *open-source* Indivo[14].

Dossia	
Região	Estados Unidos da América
Descrição	Agrega informação de saúde de diversas fontes, permite ao utilizador decidir quando com quem partilhá-la
Positivo	• Possui uma loja de aplicações direccionadas a cuidados de saúde e ao uso de sensores; • É baseado em código aberto (no inglês, *open source*); • Possui secções explicativas com instruções direccionadas a pessoas com interesses e necessidades distintas; • Refere explicitamente que a informação dos utilizadores apenas é disponibilizada a quem e quando este desejar; • A informação pode ser inserida por médicos; • Tem funcionalidades dirigidas a famílias; • Aposta na demonstração de funcionalidades.
Negativo	• O seu uso é dirigido a funcionários de instituições; • Uso limitado aos Estados Unidos da América.
Endereço	`http://www.dossia.org/`

Tabela 4.3: **Dossia - análise**

Conclusões

Como se pode constatar através da análise à tabela 4.3, há uma clara preocupação tanto com a possibilidade de consultar um histórico de informação, como em ajudar os utilizadores a usar a plataforma, pelo que estes terão de ser aspectos a ter em conta no TICE.Healthy. A motivação para haver secções de ajuda está provavelmente relacionada com o facto de haver intenção de colaboração de familiares no uso dos portais, que pode incluir faixas etárias que tipicamente têm menos familiaridade com tecnologias. Pode-se também concluir que a integração com sensores, presente no Microsoft Health Vault e Dossia, dá ênfase à importância do Sensor Care para a plataforma We.Can.

Por outro lado, a existência de um standard aberto a ser usado em produção, como é o caso do Dossia, pode ser vista como um factor que deve ser

tido em conta na construção do sistema.

4.2 Transmissão de dados em tempo real

A forma como se aborda a questão da comunicação entre servidor e cliente depende do âmbito da solução que se pretende desenvolver. Quando há necessidade de actualizações frequentes, é comum considerar-se um modelo de comunicação inverso ao tradicional, ou seja, em vez de fazer pedidos directamente ao servidor, num intervalo de tempo específico, ser o próprio servidor a enviar os dados ao cliente [15].

Tendo em conta os recentes avanços da comunicação em tempo real para aplicações *web*, há uma série de tecnologias que visam reduzir a latência na transmissão de mensagens entre servidor e cliente. De entre a variada oferta de opções para implementar um cenário deste tipo, foram seleccionadas algumas pela sua popularidade, que serão de seguida analisadas, de modo a criar uma base para escolher a mais adequada ao que se pretende para a implementação do Sensor Care.

4.2.1 Comet

Comet é uma designação que agrega uma série de mecanismos que utilizam ligações HTTP persistentes para promover a comunicação entre servidor e cliente. Todos eles têm como denominador comum o facto de usarem funcionalidades que os *browsers* suportam nativamente. As implementações mais comuns recorrem a *Asynchronous JavaScript and XML* (Ajax)[16], com Long Polling e Streaming.

Sucintamente, Ajax consiste num conjunto de abordagens que permite actualizar uma página Web, ou uma secção da mesma, sem a recarregar [16]. Long polling, uma técnica que procura minimizar a latência, consiste em o servidor não responder de imediato a pedidos do cliente, mas apenas quando há eventos para transmitir. Caso não haja eventos ou dados disponíveis, o cliente espera um determinado tempo até voltar a fazer um pedido ao servidor. No caso do Streaming, o servidor nunca termina nem a ligação com o cliente, nem o pedido em si, actualizando o cliente quando necessário.

Embora tanto uma técnica como outra tenha as suas desvantagens [17], a opção de Streaming revela-se problemática, devido a cenários que envolvem intermediários como *proxies*, *transparent proxies* ou *gateways*, cujo funcionamento não garante que estes enviem os pacotes que lhes cheguem de forma imediata [17]. A solução de Streaming não se pode, então, considerar adequada para um ambiente de produção [18], para casos em que a prioridade

seja a rapidez da entrega de dados.

Na tabela 4.4 podem-se observar as vantagens e desvantagens do Comet.

Positivo	• Funcionamento universal.
Negativo	• Ser uma tecnologia de baixo nível pode ter alto impacto no tempo de desenvolvimento; • *Overhead* - A comunicação gera "HTTP headers" de tamanho elevado, tipicamente maiores que 2kB[19].

Tabela 4.4: **Comet - pontos positivos e negativos**

4.2.2 WebSocket

WebSocket é uma tecnologia que permite comunicação em canais *full-duplex* (bi-direccional, simultaneamente em ambos os sentidos), sobre um único *socket* Transmission Control Protocol (TCP). A sua especificação foi desenvolvida para simplificar a comunicação e gestão de ligações, através de uma abstracção de mais alto nível face à existente com Comet e Ajax. O seu uso é limitado devido ao número de "browsers" que ainda não suporta esta especificação. À data da escrita deste parágrafo apenas o Firefox 7 em diante, Chrome 14 e Internet Explorer 10 suportam esta tecnologia[20][21].

Na tabela 4.5 estão assinalados os seus maiores prós e contras.

Positivo	• Application Programming Interface (API) de alto nível
Negativo	• Limitado a poucos *browsers* em produção

Tabela 4.5: **WebSocket - pontos positivos e negativos**

4.2.3 BOSH e XMPP

BOSH significa Bidirectional-streams Over Synchronous Hypertext Transfer Protocol (HTTP) e emula uma ligação TCP sobre HTTP [17]. Genericamente pode-se dizer que o busílis desta tecnologia está no funcionamento

com base em *long polling* com múltiplos pares de pedido/resposta síncronos. O servidor XMPP adia a resposta de um pedido até que haja dados para enviar. Assim que o cliente recebe dados, envia um pedido ao servidor, de modo a garantir que este esteja novamente disponível a enviar dados.

Estão assinalados na tabela 4.6 os destaques positivos e negativos do uso de BOSH e XMPP.

Positivo	• API de alto nível; • Funcionamento muito abrangente em termos de *browsers*.
Negativo	• Requer o uso de um módulo extra - o servidor de XMPP; • Latência [22].

Tabela 4.6: **BOSH e XMPP - pontos positivos e negativos**

4.2.4 Socket.IO

A tecnologia Socket.IO[15] foi criada com vista a tornar possível a comunicação de aplicações em tempo real em todos os *browsers*, incluindo ambientes móveis.

O seu funcionamento consiste em seleccionar a forma mais eficiente de transporte em *runtime* através da detecção do *browser* cliente e do conhecimento prévio do melhor método para cada *browser*, sem afectar a API de comunicação.

Assente em Node.js[23], de entre as soluções híbridas (ou seja, que combinam várias tecnologias de modo a oferecer um leque de possibilidades de utilização mais amplo) destaca-se, exactamente, pelo número de tecnologias suportadas, nomeadamente WebSocket, Adobe Flash Socket, AJAX long polling, AJAX multipart streaming, Forever Iframe e JSONP Polling[24].

Este perfil multifacetado do Socket.IO permite-lhe, então, promover a comunicação em Internet Explorer 5.5+, Safari 3+, Google Chrome 4+, Firefox 3+, Opera 10.61+, iPhone Safari, iPad Safari, iPod Safari, Android WebKit e WebOs WebKit[24].

Na tabela 4.7 podem-se observar os seus pontos positivos e negativos.

Positivo	• API de alto nível; • Funcionamento muito abrangente em termos de *browsers*, com recurso a *fallbacks*; • Comunicação optimizada para cada caso.
Negativo	• Embora esteja pronto para produção, uma análise aos problemas indicados no GitHub[25], revela que ainda há por onde melhorar

Tabela 4.7: **Socket.IO - pontos positivos e negativos**

4.2.5 Conclusão

Feita a análise das tecnologias envolvidas, e tendo em conta a tabela 4.8, optou-se pela utilização de Socket.IO, por ser a alternativa que conjuga mais vantagens (tecnologia de alto-nível, grande abrangência, e não necessitar de um módulo suplementar exclusivo para a comunicação). A desvantagem assinalada ao Socket.IO não é limitativa, já que, como foi dito, está pronta para produção, sendo inclusivamente utilizada em diversos produtos[26]. O melhor exemplo será a sua utilização pela aplicação Zynga Poker [27][26], que actualmente tem uma média de utilizadores por dia superior a sete milhões [28].

Tecnologia	**Alto-nível/baixo-nível**	**Abrangência**	**Dependências**
Comet	Baixo	Grande	Não
WebSockets	Alto	Pequena	Não
BOSH e XMPP	Alto	Grande	Servidor XMPP
Socket.IO	Alto	Grande	Não

Tabela 4.8: **Comparação tecnologias de comunicação servidor-cliente em tempo real**

A fim de validar a opção, foi adaptado um módulo feito anteriormente com WebSockets para o projecto TICE.Mobilidade - projecto irmão do TICE-Healthy focado na mobilidade urbana, passando a comunicação a ser feita com Socket.IO. Verificou-se que a tecnologia opera de acordo com o esperado e que as funcionalidades descritas na sua documentação se revelam eficazes nos vários *browsers* descritos.

4.3 *Web application frameworks* de Node.JS

Nesta secção são analisadas as várias alternativas para *Web application frameworks* de Node.JS. Foi feito um levantamento inicial das várias soluções de código aberto e com licença permissiva, e que portanto podem ser aplicadas em projectos comerciais. De notar que as razões para a escolha de Node.JS estão explicadas na secção "Desenho da solução".

4.3.1 Análise e comparação

À data da escrita do relatório, o número de *frameworks* para Node.JS, listado no respectivo repositório de módulos[29] chega quase às cinco dezenas. De modo a filtrar os módulos a analisar foram excluídos, os que comunidades menos activas têm. Estando todos os projectos alojados no Github, foi possível usar como métricas de actividade o número de seguidores do projecto, "open issues" e "closed issues", que são vistas como sinónimo de maturidade do projecto e do uso efectivo que têm. Como há diferenças claras na ordem de grandeza destas métricas foram excluídas as opções com menos de mil seguidores, menos de cem "open issues" e mais de cem "forks" e "closed issues". A versão reduzida da comparação pode ser vista na tabela 4.9 enquanto a tabela completa pode ser vista no capítulo "*Web application frameworks* de Node.JS" dos anexos do relatório.

Projecto	**Número de seguidores**	***Open issues***	***Closed issues***	**Número de *forks***
Express[30]	7,309	76	1,223	787
Geddy[31]	1,055	13	161	101
SocketStream[32]	2,090	9	277	128

Tabela 4.9: **Tabela comparativa de *Web application frameworks***

Das três alternativas apresentadas, foi descartado o SocketStream, por ser orientado a aplicações de uma única página [32]. Verificou-se que Geddy e Express são também *frameworks* que recebem grande atenção em comunidades como Stack-Overflow, Quora e Twitter, meios de comunicação típicos da comunidade *open-source*.

Seleccionadas estas duas tecnologias, e após verificar se eram compatíveis com uma configuração *Model-View-Controller* passou-se a uma análise mais aprofundada das mesmas. Para uma melhor leitura, esta foi separada em duas tabelas: 4.10 e 4.11.

Projecto	**Data de criação**	**Contribuidores**	***Versão mais actual***
Express[30]	2009-06-26	71	3.0.0rc3
Geddy[31]	2010-03-14	31	0.4.5

Tabela 4.10: **Tabela comparativa de *web application frameworks 1***

Projecto	***API Reference***	***Commits*/dia**	**Projectos em que é usado**
Express[30]	✓	3.3	Learnboost, Storify, Geekli.st, Klout, Prismatic, Clipboard, Persona
Geddy[31]	✓	1.0	Não há informação oficial

Tabela 4.11: **Tabela comparativa de *web application frameworks 2***

Pode-se verificar que Express tem uma comunidade maior e mais activa a participar na sua evolução e suporte. É, também, um projecto que existe há mais tempo e há exemplos concretos de que está a ser usada em projectos em produção.

4.3.2 Conclusão

Da análise a ambas as soluções, verificou-se que a documentação da *framework* Express é mais evoluída. Na análise aos "open issues" verificou-se que havia problemas com a integração com Socket.IO, tecnologia também analisada e seleccionada neste capítulo; por conseguinte, e tendo em conta também a sua maior maturidade, a tecnologia Express revela-se a escolha indicada.

4.4 *Template engines para Express*

Após escolher a *web application framework* Express foi necessário definir um sistema de *Template Engines*, a fim de definir as vistas de forma modular e garantindo a separação entre as lógicas da aplicação. Esta *framework* suporta dois Template Engines: EJS[33] e Jade[34].

Entre as duas a escolha recaiu sobre EJS, pois, tendo uma sintaxe baseada a HTML, não exigindo aprender uma sintaxe nova, o que é importante quando é possível que haja outras pessoas manter ou evoluir o projecto. Por outro lado, EJS permite criar templates *client-side*, enquanto Jade está limitado ao servidor, o que poderá ser uma vantagem no futuro caso venha a ser

um requisito.

Capítulo 5

Desenho da solução e decisões tencológicas

Tendo em conta a arquitectura idealizada, a investigação feita e os requisitos levantados, desenhou-se uma solução para implementação do sistema, que é apresentada de seguida.

5.0.1 Aplicação móvel

A aplicação móvel, por decisão da gestão do projecto, foi desenvolvida para a plataforma Android e com vista a ser utilizada em *tablets*.

De modo a seguir as mais recentes boas práticas de desenvolvimento de aplicações sugeridas pela Google[35], proporcionando uma melhor experiência de utilização aos utilizadores, e garantindo uma interface concordante com as mais recentes directivas, foi usada a biblioteca ActionBarSherlock[36]. Esta é uma extensão da biblioteca de compatibilidade nativa do Android e facilita a construção de aplicações, integrando componentes apenas desenvolvidos em Software Development Kits mais recentes (Ice Cream Sandwich[37] e Jelly Bean[38]), proporcionando a mesma experiência de utilização.

Para implementação do OAuth foi usada a biblioteca Signpost, que permite criar o fluxo OAuth assinando mensagens HTTP para aceder a conteúdos protegidos com a devida autorização. De referir ainda, a utilização da biblioteca GSON, que permite fazer mapeamento entre objectos Java e JSON.

Para a ligação ao agregador de sinais PLUX e respectiva aquisição de sinais, foi usada uma biblioteca desenvolvida pela PLUX. Esta foi desenvolvida a par do trabalho do estagiário que se manteve em contacto com a empresa, tendo reportado diversas falhas, bem como apresentando sugestões para melhoria da documentação, contribuindo para a evolução da biblioteca que será

usada pela PLUX nos seus produtos. Até à data de escrita do relatório não foi possível integrar uma versão plenamente funcional da biblioteca, sendo que o estagiário teve de contornar os problemas da mesma com aquisição de sinal para alguns sensores em modo cru e tratando-o posteriormente.

Uma vez adquiridos os sinais estes são enviados para o módulo *Generic Entity*, um editor de entidades configurável, que se propõe a implementar o Personal Health Record, que dará suporte à plataforma We.Can. Este disponibiliza-os à aplicação web. Tem sido desenvolvido pelo estagiário Pedro Catré, e o Sensor Care foi o primeiro caso de uso do mesmo, o que contribui para a sua evolução e validação.

5.0.2 Aplicação Web

O motor de base de dados escolhido para utilizar no projecto foi PostgreSQL, que consiste num sistema de gestão de base de bados *open-source* objecto-relacional, *cross-platform* e amplamente suportado pela comunidade. A principal razão para a escolha é a directiva de gestão para uniformizar as tecnologias utilizadas, aliada ao facto de esta não chocar com as necessidades do Sensor Care.

Após sugestão da gestão técnica do projecto, e perante o tipo de aplicação em causa, optou-se por desenvolver a aplicação web em Node.JS. As razões que sustentam essa decisão prendem-se com não só com a uniformização de linguagens, tanto no Sensor Care, havendo componentes de back-end e front-end em JavaScript, como no projecto em si, tendo na equipa sido desenvolvido uma aplicação no projecto irmão TICE.Mobilidade na mesma tecnologia; mas também o facto de ser uma tecnologia *event-based*, com soluções direccionadas ao desenvolvido de aplicações em tempo real.

Como descrito no Estado da Arte, optou-se por usar EJS para os *templates* das vistas; Express, para modelar a arquitectura *Model-View-Controller* que serve de base à aplicação e Socket.IO para gerir os eventos dos sensores em tempo real.

A aquisição de dados do canal XMPP que o módulo *Generic Entity* disponibiliza é feito através da biblioteca simple-xmpp

Capítulo 6

Desenho de interface

Neste capítulo é abordado o processo de construção das interfaces para as aplicações web e móvel. Ambos foram fruto de um processo iterativo. Este método facilita alinhar a visão daquilo que se pretende construir, criando oportunidade de se receber *feedback* construtivo promovendo a redução do tempo de implementação que se torna mais precisa[39]. Podem ser vistos Ecrãs complementares aos apresentados

6.0.3 Desenho do ícone da aplicação

O ícone da aplicação móvel Sensor Care, seguiu as boas práticas indicadas pela Google[40] para ícones para o *launcher* Android.

Assim, foi escolhido um ícone que fizesse lembrar a saúde, através da cruz, que se identificasse com o TICE.Healthy, cujo design tem como azul a cor principal, e que promove-se o Sensor Care, através da presença da letra 'S'. Foram seguidas as directivas de ter um desenho simples, não ter componentes cortados nem demasiado finos, e de usar transparência e um toque apelativo, aplicados através do gradiente na côr, e no efeito tridimensional (condizente com a perspectiva usual), respectivamente. Este pode ser visto na figura 6.1.

Figura 6.1: **Ícone da aplicação móvel**

O ícone foi adoptado para representar a aplicação no portal, de modo a criar uma identidade unificada entre os dois componentes do sistema.

6.1 Aplicação móvel

O processo de desenho de interface da aplicação móvel começou por protótipos de baixa fidelidade, como o que pode ser visto na figura 6.2. Dado que esta só ficou definida já no segundo semestre, daí partiu-se para a implementação, o que se demonstrou essencial, já que permitiu o contacto imediato com a realidade das interfaces de *tablet* Android, que se revelou um pouco diferente do que havia sido desenhado, e que trouxe alguns desafios, explicados de seguida.

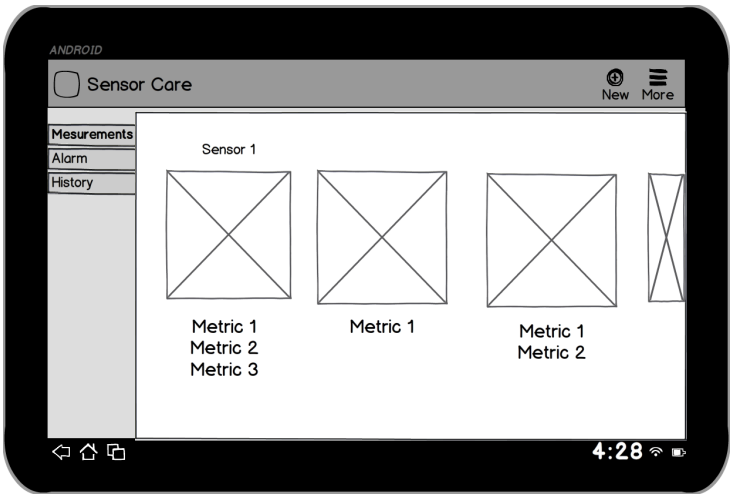

Figura 6.2: **Protótipo de baixa fidelidade**

O desenho da aplicação, em si, procurou também seguir as mais recentes linhas de orientação disponibilizadas[41]. Exemplos disso são a utilização de uma "action bar" e "action items" para a navegação e principais acções dentro da aplicação, presentes na figura 6.4. A grelha de visualização de utilizadores presente na figura 6.3 foi implementada de modo a comportar qualquer número de utilizadores, e personalizada de forma incluir elementos destintos, nomeadamente imagens - os *avatars* dos utilizadores, e texto - o respectivo nome.

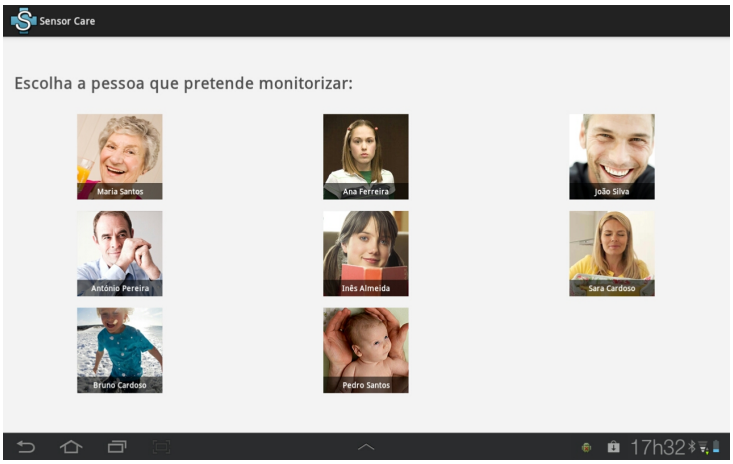

Figura 6.3: **Grelha de utilizadores**

Figura 6.4: **Aquisição de sinais**

Ao longo do *sprint* 6 o *layout* foi modificado, sendo que a principal alteração consiste na pre-definição dos canais de cada sensor, em vez de dar a opção ao utilizador de os configurar. Esta modificação foi feita após identificado um problema de usabilidade e tendo sido discutido com o Mestre João Quintas, que é um elemento da projecto com experiência na área de interacção entre pessoas e *hardware*. O texto foi também tornado mais conciso.

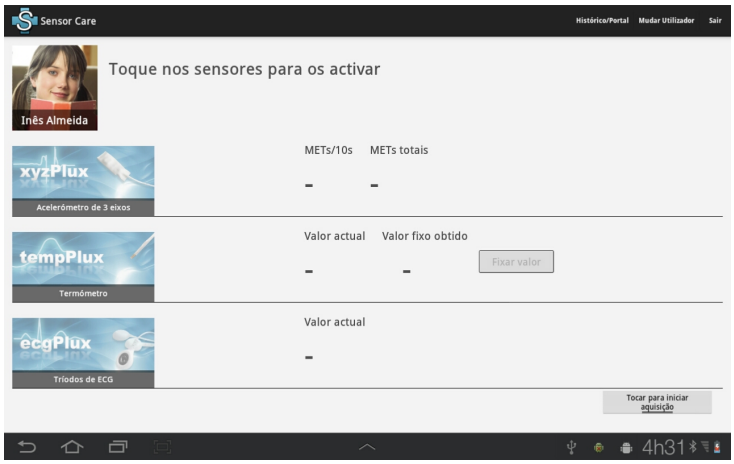

Figura 6.5: **Aquisição de sinais**

6.2 Aplicação web

Para a aplicação web começaram por ser criados protótipos de baixa fidelidade, que consistem em esboços da interface gráfica idealizada.

Seguem-se exemplos desta primeira iteração do processo:

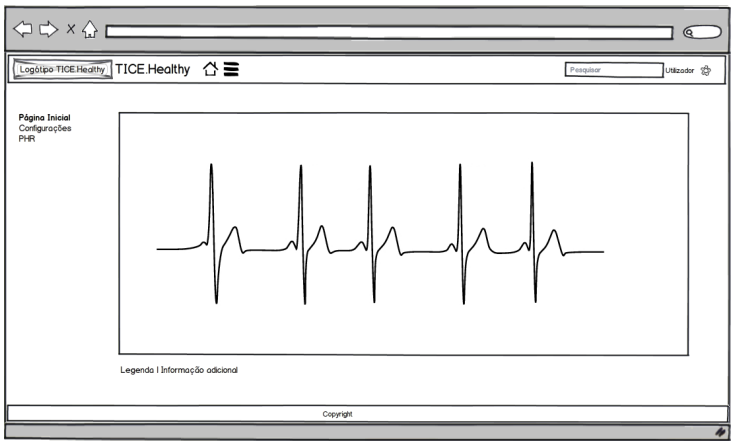

Figura 6.6: **Visualização de sinais vitais**

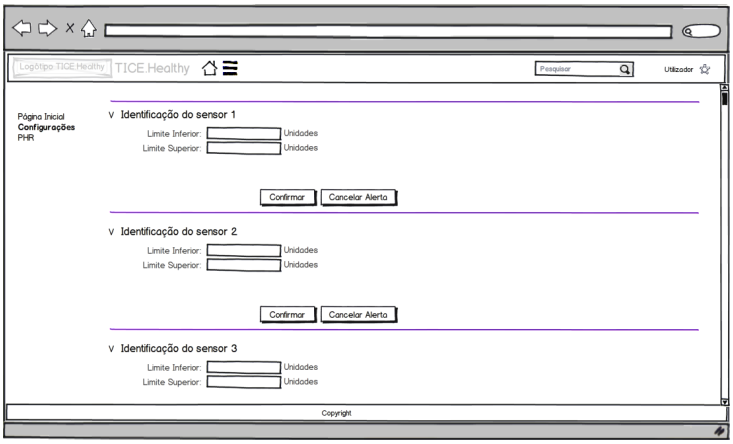

Figura 6.7: **Configurações**

De seguida foi construído um protótipo com maior nível de detalhe, denominado de alta fidelidade, já com tecnologias que a serem empregues. A partir desta iteração passou a usar-se Twitter Bootstrap na construção dos layouts, que oferece nativamente *layout* fluido e responsivo.

Na figura 6.8 pode ser observada uma captura de ecrã referente ao dito protótipo, estando inserido no portal We.Can. A sua integração requereu apenas uma linha de *Javascript*, a fim de que a apresentação visual fosse correcta de modo a proporcionar uma experiência de utilização agradável.

De referir que os dados dos sensores estão a ser simulados e transmitidos para o *browser* com Socket.IO.

Figura 6.8: **Protótipo de alta fidelidade de aquisição de sinais**

Seguiu-se a implementação da aplicação com um *design* que foi drastica-

mente alterado durante o *sprint* 6, conforme pode ser visto nas figuras 6.9 e 6.10. Nesta fase foram também adicionadas ícones vectoriais nos botões e submissões por Ajax às páginas construídas.

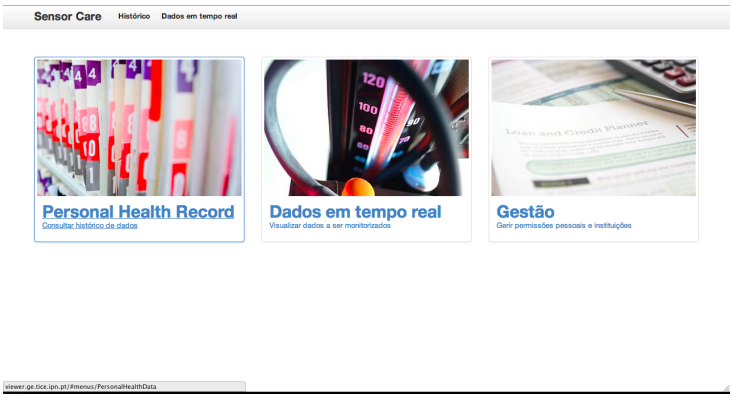

Figura 6.9: **Primeira versão do *design* final**

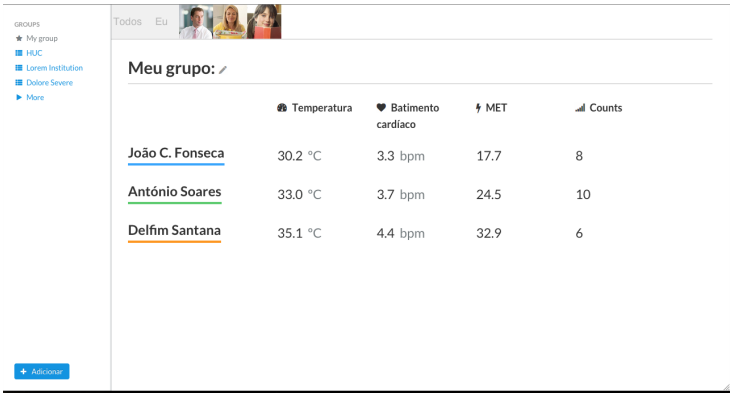

Figura 6.10: **Segunda versão do *design* final**

Capítulo 7

Testes

Neste capítulo é descrito o processo de testes efectuado. De forma complementar podem ser vistos os testes funcionais e de integração referentes às "user stories" definidos no anexo " Complemento de requisitos".

7.1 Testes de aceitação

Foram especificados e elaborados testes não-automatizados de aceitação pelo utilizador (do inglês *User Acceptance Testing*). Este tipo de testes tem como objectivo verificar se as *user stories* implementadas funcionam da forma esperada pelo utilizador final. Definem, desta forma, o valor de negócio que cada *user story* deve entregar.

Desta forma, no final de cada *sprint*, de acordo com as práticas da equipa de trabalho em vigor, foram feitas sessões de demonstração à equipa de gestão na qual se incluem:

- Eng. José Gouveia Leal, Director Executivo dos projectos TICE, pelo IPN;
- Professor Doutro Carlos Bento, Director do LIS;
- Professor Doutro Jorge Dias, Director do Laboratório de Automática e Sistemas;
- Eng. Alcides Marques, Coordenador do projecto TICE.Mobildiade e Director Adjunto do LIS, na qualidade de *product owner*;
- Mestre João Quintas, Coordenador do projecto TICE.Healthy;
- Eng. Miguel Laginha Machado, Coordenador Técnico dos projectos TICE, na qualidade de *Scrum Master*;

- Eng. António Cunha, Director Adjunto do LAS.

Estes elementos pudera, desta forma, criticar trabalho da perspectiva de negócio, dando um contributo positivo para o progresso de estágio e do projecto.

7.2 Testes de usabilidade

Optou-se por fazer testes qualitativos de usabilidade com base nos métodos sugeridos por Jakob Nielsen[42] e Steve Krug[43]. Estes testes não seguem métodos científicos, são antes baseados no que se pode apreender com a simples utilização do sistema por parte dos utilizadores daí a sua designação de *Simple User Tests*. Steve Krug equipara este tipo de testes a fazer *debug* a um *design*.

Simple user testing consiste em observar os utilizadores a utilizar o sistema de forma autónoma com a intenção de o tornar mais fácil de usar, e baseia-se no princípio de que os problemas mais sérios tendem a ser fáceis de encontrar. Este processo deve ser feito com regularidade, com utilizadores diferentes, de forma a validar e/ou melhorar os sistemas.

Foram dadas tarefas específicas a três utilizadores que nunca tinham utilizado o sistema e observado o seu comportamento, com ajuda do seu *feedback*, e registaram-se os seguintes problemas:

- Aplicação móvel - os utilizadores não liam o texto indicativo para a configuração dos sensores e ficaram confusos com o que fazer de seguida. Esta dificuldade foi identificada pela primeira vez numa sessão de demonstração da aplicação, sendo que estes resultados corroboraram a existência do problema.

- Aplicação web - os diferentes níveis de navegação na página não eram de utilização clara, ficando os utilizadores indecisos quanto a como chegar a cada funcionalidade.

Para a aplicação móvel, foi pedida a colaboração do Mestre João Quintas, que com a sua experiência em interacção com *hardware* indicou que a solução passaria por pré-determinar a configuração dos sensores.

No caso da aplicação web, a equipa de trabalho colaborou para a alteração do modo de navegação.

Capítulo 8

Planeamento

Este capítulo apresenta o plano de estágio, a metodologia de desenvolvimento de software aplicada, e ainda a análise de riscos identificados durante o estágio, com a indicação da sua ocorrência ou inocorrência.

8.1 Metodologia de Trabalho

Em Engenharia de Software, definir uma metodologia de trabalho eficaz e eficiente é não só uma base importante como também um contributo fundamental para o sucesso de um projecto. Neste caso, a dimensão global do projecto traz responsabilidades acrescidas quanto à adopção de boas práticas, dado o potencial de interacção com os vários parceiros, presentes ou futuros.

Nesta secção serão descritos os métodos de trabalho adoptados, bem como as boas práticas mais relevantes no decorrer do desenvolvimento do Sensor Care.

8.1.1 Processo de desenvolvimento

O processo de desenvolvimento de software aplicado durante o estágio, tem por base a metodologia ágil de gestão de projecto Scrum. A sua natureza ágil é vantajosa em projectos como o presente, dada a interacção e colaboração com parceiros externos, bem como a existência de requisitos pouco estáveis, sendo expectável a ocorrência de constrangimentos.

O Scrum é composto por três fases: *pre-game*, que corresponde a planeamento e desenho; *game*, correspondente à fase de implementação; e *post-game*, para preparar o produto para o ambiente de produção.

Durante a fase de implementação, esta metodologia terá por base a execução de vários ciclos, denominados Sprints[44]. Estes ciclos são compostos por

desenho, implementação e testes e no seu final é feita uma entrega incremental do produto em questão. Um diagrama demonstrativo do funcionamento desta abordagem pode ser visualizado na figura 8.1. A duração dos Sprints é, em si, meramente exemplificativa, podendo variar consoante o *backlog* ou outros factores da ordem da gestão.

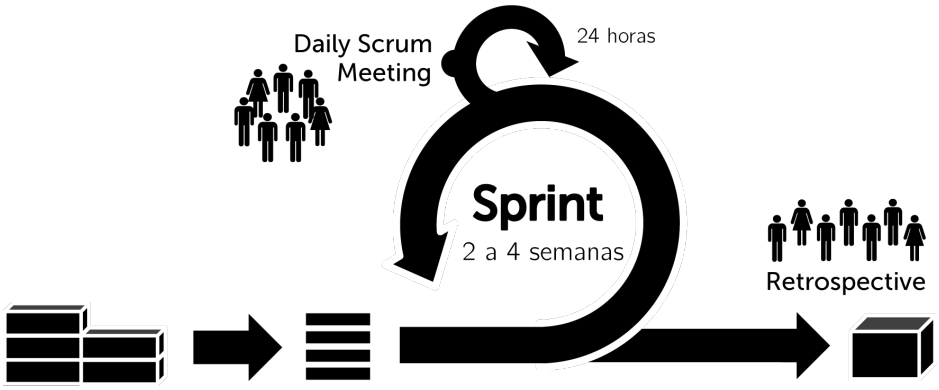

Figura 8.1: **Diagrama representativo do fluxo do Scrum**

Durante os Sprints há reuniões diárias, denominadas Daily Scrum Meetings, de curta duração, em que cada participante fala sobre o progresso conseguido, o trabalho a ser realizado e o que o impede de avançar. No final de cada Sprint é feita uma revisão do trabalho realizado, Sprint Review, e o planeamento do Sprint seguinte, Sprint Planning Meeting. O planeamento é feito com base no Product Backlog, que consiste numa lista de requisitos a cumprir pelo produto, sendo que a cada iteração é definido qual ou quais serão tratados, ficando definido um Sprint Backlog [44]. Estes foram geridos através da ferramenta web Redmine[45]. É de destacar ainda a existência de uma reunião denominada de Retrospectiva do Sprint [44], que consiste numa oportunidade para a equipa Scrum se inspeccionar e criar um plano de melhorias ao seu processo produtivo a ser seguido durante o próximo Sprint.

Relativamente à equipa, é de assinalar a sua constituição e tarefas de maior destaque:

- Product Owner [44] - Engenheiro Alcides Marques - é responsável por maximizar o valor do produto e o trabalho da equipa. É ele que gere o Product Backlog e define as suas prioridade
- Scrum Master [44] - Engenheiro Miguel Machado - que tem responsabilidades para com o Product Owner, relacionadas essencialmente com planeamento e cumprimento do Product Backlog, e para com a Equipa

de Trabalho, que dizem respeito mais a tarefas de orientação da equipa e remoção de obstáculos a mesma.

- Equipa de desenvolvimento [44] - no caso composta por oito elementos, é uma equipas de desenvolvimento multi-disciplinar cujo objectivo é criar os incrementos do produto potencialmente comercializáveis.

O processo termina quando o produto final do projecto em questão cumpre o *backlog* estipulado e, portanto, se encontra em versão estável.

Face à metodologia Scrum, há aspectos que foram alterados para se adaptarem à realidade do projecto, e ao relatório em si, sendo de destacar:

- Os gráficos Burndown apresentados são referentes ao estagiário e não à equipa;
- Para seguir os propósitos do estágio o trabalho de cada estagiário foi planeado antes da fase de Game;
- O Product Owner nem sempre está presente na reunião de Sprint Retrospective.

É política do Laboratório de Informática e Sistemas do IPN haver apresentações temáticas de periodicidade mensal, que visam partilhar conhecimento entre os vários membros do laboratório. Estas abordam a investigação sobre um determinado tema ou sobre o trabalho que está a ser desenvolver.

8.1.2 Sistema de controlo de versões

O desenvolvimento de software passa, muitas vezes, por experiências empíricas: seja para validação, por falta de documentação, ou até por alterações de requisitos. Durante este processo, é comum ter de voltar atrás, por exemplo, para adoptar uma opção previamente abandonada. Se numa linha de execução linear já é normal haver erros, ou não tenha o adágio popular *Errare humanum est* chegado aos nossos dias, neste caso essa situação há que ser ainda mais tida em conta.

A utilização de um sistema de controlo de versões permite voltar atrás facilmente, mas não só. Outras vantagens são a possibilidade de visualizar o histórico de evolução do projecto com diversos tipos de atomicidade, bem como identificar os responsáveis pelas alterações. Com a adopção de boas práticas, o número de vantagens aumenta tornando-se o sistema numa ferramenta fundamental.

No presente caso, optou-se pela utilização do sistema Git[46], infra-estrutura já utilizada no Laboratório de Informática e Sistemas do IPN desde 2007.

8.1.3 Gitflow

Para uma melhor gestão do repositório foi adoptado o modelo de desenvolvimento proposto por Vincent Driessen[47], o Gitflow [48]. Este modelo procura melhorar a organização do repositório orientando-a à produção, não interferindo nas acções habituais de desenvolvimento. Esta ferramente funciona a nível local e adiciona uma camada de abstracção sobre os comandos de Git.

O Gitflow permite criar e fundir *branches* para funcionalidades, lançamento de novas versões, desenvolvimento e correcção de erros, conforme as necessidades reais do desenvolvimento e de forma transparente. Por exemplo, se se pretender corrigir um erro, quando se termina esse ramo, ele vai ser fundido tanto com o ramo *master*, que corresponde à versão de produção, como com o ramo *development*. O uso correcto do Gitflow permite sempre identificar qual a versão estável de desenvolvimento, bem como perceber a evolução do desenvolvimento.

8.1.4 Semântica de versionamento de código

Durante o processo de desenvolvimento é importante que haja uma métrica definida para etiquetar as versões do software desenvolvido. O método mais comum é atribuir-lhes números. Quanto maior for o projecto, mais dependências há, e mais importante se torna torna ter uma semântica de versionamento uniforme e eficaz.

De modo a disciplinar as métricas de atribuição que etiquetam as versões, para que haja incrementos lógicos e com significado real, foi usado o sistema baseado no proposto por Tom Preston-Werner[9].

O sistema consiste na definição de versões segundo o modelo vX.Y.Z, correspondendo X a incrementos unitários que representem alterações que criam incompatibilidades com versões anteriores, como alterações na API pública; Y à adição de novas funcionalidades ou alterações que não produzam incompatibilidades com o previamente existente; e Z à correcção de erros. Esta métrica, tem ainda a vantagem de encaixar facilmente na fluxo de trabalho adoptado pelo Gitflow. De mencionar ainda, que à primeira versão pública é atribuída a versão “v1.0.0”.

8.2 Plano de Estágio

Este capítulo foca-se em questões relacionadas com a gestão do projecto, nomeadamente, organização e planeamento de tarefas.

8.2.1 Primeiro Semestre

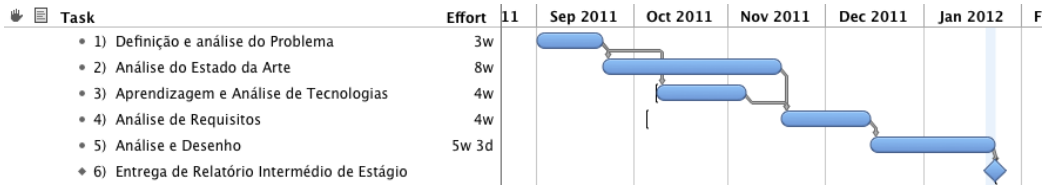

Figura 8.2: **Primeiro Semestre - planeamento**

Na figura 8.2 pode-se observar o diagrama de Gantt relativo ao planeamento para o primeiro semestre, em que o trabalho é divido em seis fases que espelham a modelação de negócio (We.Can e análise do problema, estado da arte e aprendizagem de tecnologias), requisitos (que inclui protótipo de baixa fidelidade) e análise e We.Can (arquitectura e protótipo de alta fidelidade).

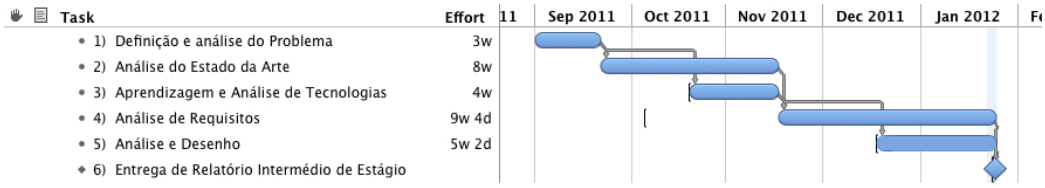

Figura 8.3: **Primeiro Semestre - execução real**

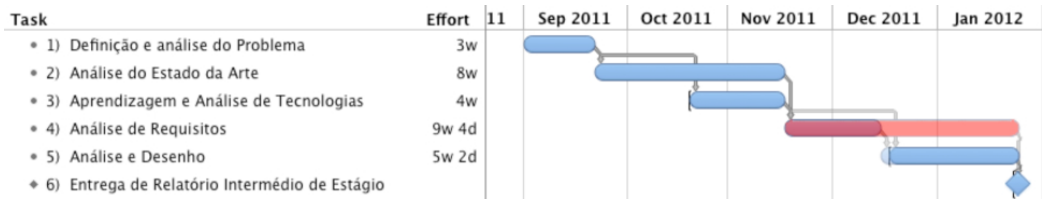

Figura 8.4: **Primeiro Semestre - desvio**

Relativamente ao planeamento inicial, houve um ligeiro atraso. A forma como decorreu pode ser vista na figura 8.3 e o desvio na figura 8.4. Este deveu-se essencialmente à falta de visão do projecto e falta de contacto com

um prestado de cuidados de saúde que levantasse requisitos reais. Este risco foi identificado e a respectiva estratégia de mitigação aplicada, conforme descrito na secção "Análise de Riscos" do presente documento.

8.2.2 Segundo semestre e extensão até Setembro

O trabalho do segundo semestre, a que se refere a figura 8.6, foi orientado em *sprints* com duração de quatro semanas: três de desenho, implementação e testes; e uma de documentação. O esforço de trabalho foi definido da seguinte forma:

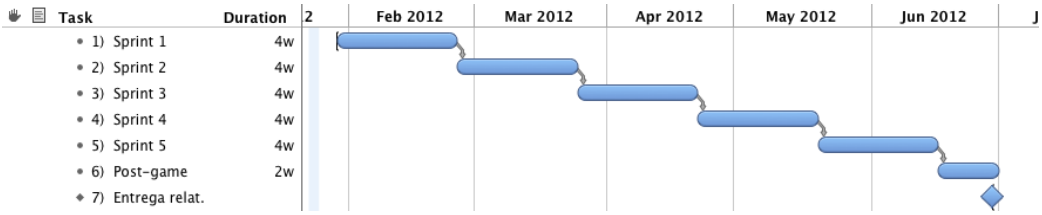

Figura 8.5: **Segundo semestre - planeamento inicial**

No fim do quinto *sprint* houve uma fase reflexão e em conjunto com a gestão do projecto decidiu-se que se poderia acrescentar valor ao produto final procedendo à entrega em época especial.

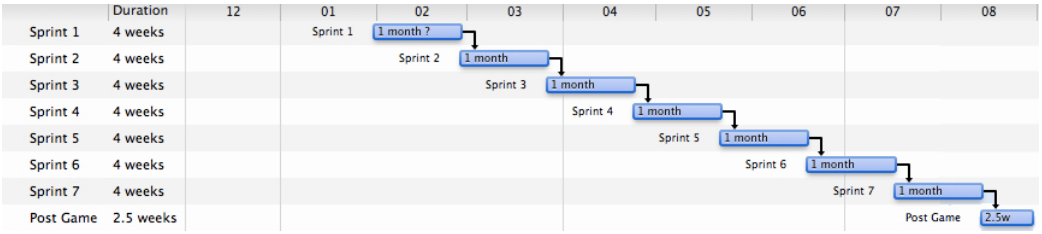

Figura 8.6: **Segundo semestre - execução real**

No início do segundo semestre houve um realinhamento do projecto, devido à exclusão da integração do UCH na arqutectura e redefinição da aplicação móvel a desenvolver.

As tarefas a realizar em cada *sprint* foram definidas nas reuniões de Sprint Planning, que se decorreram no início de cada *sprint*. Na tabela descrição do

que foi feito. Para uma descrição mais detalhada das tarefas realizadas deve ler-se no Anexo "Artefactos da Metodologia de Desenvolvimento Ágil".

Na fase de *Post-game*, procedeu-se à escrita de documentação e preparou-se o produto para estar em produção, procedendo ao seu *deployment*.

8.3 Análise de Riscos

Um risco é um evento cuja ocorrência tem impacto negativo no alcançar de um ou mais objectivos.

Em projectos de software, como na vida, é normal que o fluir dos acontecimentos nem sempre seja o planeado. Os riscos são, por conseguinte, uma constante e devem ser identificados tão cedo quanto possível, para que se possa agir atempadamente caso ocorram de modo a não afectar o planeamento ou a qualidade do produto.

A análise de riscos deve ser feita de modo cíclico. São identificados, a cada momento, os riscos presentes, podendo o impacto, ou probabilidade de ocorrência, ser alterados.

Esta secção tem como objectivo analisar os riscos que podem assolar o projecto desenvolvido e delinear estratégias de mitigação.

Nas tabelas 8.3, 8.3, 8.3, 8.4 e 8.5, são apresentados os principais riscos identificados, bem como a sua descrição, probabilidade de ocorrência, impacto no projecto, plano de contingência e se ocorreu.

Tabela 8.1: **Risco número 1**

Título	O UCH não está pronto para integração
Descrição	O módulo do Universal Control Hub não está pronto para produção sem haver previsão de quando estará
Tipo	Técnico
Probabilidade	Alta
Impacto	Baixo
Plano de contingência	Agregar os sinais no dispositivo Android
Ocorrência	Ocorreu e o plano de contingência foi aplicado.

Tabela 8.2: **Risco número 2**

Título	Não há modulo de permissões
Descrição	O módulo de permissões não está implementado, compromentendo as funcionalidades do sistema
Tipo	Técnico
Probabilidade	Alta
Impacto	Alto
Plano de contingência	Implementar um mecanismo de permissões para efeitos demonstrativos
Ocorrência	Ocorreu e o plano de contingência foi aplicado.

Tabela 8.3: **Risco número 3**

Título	Não há *tablet* Android para desenvolvimento
Descrição	O projecto não dispõe de um dispositivo que suporte os requisitos tecnológicos que a aplicação móvel requer
Tipo	Técnica
Probabilidade	Alta
Impacto	Médio
Plano de contingência	Usar o emulador para simular as funcionalidades que este suportar
Ocorrência	Ocorreu e o plano de contingência foi aplicado.

Tabela 8.4: **Risco número 4**

Título	Módulo Generic Entity não está desenvolvido.
Descrição	O PHR pode não ser desenvolvido a tempo, inviabilizando a sua integração durante o tempo de estágio.
Tipo	Técnico
Probabilidade	Média
Impacto	Médio
Plano de contingência	Implementar um sistema equivalente para efeitos demonstrativos.
Ocorrência	Não ocorreu.

Tabela 8.5: **Risco número 5**

Título	O projecto não tem uma visão definida.
Descrição	O TICE.Healthy encontra-se numa fase de desenvolvimento pouco evoluída e há falta de contacto com um prestador de cuidados de saúde, dificultando o levantamento de requisitos.
Tipo	Negócio
Probabilidade	Alta
Impacto	Alto
Plano de contingência	Basear a construção de requisitos na análise pessoal do problema, tentando perceber as necessidades dos utilizadores, e no estado da arte. Alinhar a visão com a gestão do projecto.
Ocorrência	Ocorreu e o plano de contingência foi aplicado

Capítulo 9

Notas finais

Findo o tempo de estágio, há que fazer uma análise ao trabalho feito e tecer considerações quanto ao passado, presente e futuro do projecto.

9.1 Principais obstáculos

O atraso patente no desenvolvimento do projecto TICE.Healthy foi o principal obstáculo encontrado, por diversos motivos. Em primeiro lugar, trouxe vários focos de incerteza ao longo do tempo, nomeadamente quanto à integração do sistema com outros componentes, sendo exemplos disso a Generic Entity, o UCH, ou o módulo de permissões Por outro lado, levou a um alongar do tempo de desenvolvimento do sistema, sendo o melhor exemplo disso a integração da API para Android de ligação ao agregador de sinais PLUX, que ao longo do tempo, e graças à contribuição do estagiário que identificou os seus problemas, foi tendo um crescendo de correcções, embora sem nunca chegar a apresentar um comportamento totalmente funcional. Por outro lado, acabou por influenciar a indefinição do âmbito do Sensor Care, pois, embora a agregação e disponibilização de dados tenha sido um objectivo fixo, o demais teve de ser ajustado ao longo do tempo com uma base de sustentação difusa.

Durante o desenvolvimento, teve de ser despendido tempo relevante em aprendizagem de tecnologias. Este facto foi, contudo, considerado como um desafio e uma mais valia para a formação do estagiário, que ganhou novas competências especialmente no que toca a tecnologias web emergentes.

9.2 Trabalho futuro

Os próximos passos passam por evoluir o sistema no sentido de se integrar com mais módulos desenvolvidos, tais como permissões, e mecanismos adicionais de segurança que sejam necessários para a interacção com o Personal Health Record, tal como previsto na evolução do módulo Generic Entity.

Por outro lado, está prevista a integração de mais tipos de sensores com o Sensor Care, tanto de parceiros internos ao TICE.Healthy (outros disponibilizados pela PLUX), como externos, estando idealizada uma colaboração com a Polar[49].

9.3 Contributo

O trabalho desenvolvido representou um contributo importante para o projecto TICE.Healthy, na medida em que o Sensor Care representa uma alavanca para o seu desenvolvimento. Em primeiro lugar, o Sensor Care representa a primeira aplicação que alimenta a plataforma de dados; por outro lado, o modelo de dados adoptado promove uma fácil integração de aplicações desenvolvidas para a plataforma Dossia, o que permite minimizar o esforço de adaptação de aplicações desenvolvidas de uma para a outra plataforma, tendo este modelo sido partilhado e reutilizado por parceiros do projecto. O estagiário compilou as boas práticas para "commits" em sistemas de versionamento de código que foram apresentadas ao laboratório e que as passou a adoptar, e contribui disseminar conhecimento pelos vários parceiros do projecto garantindo a comunicação transparente dos avanços efectuados no projecto e respectiva documentação, sendo um dos maiores contribuidores da "Knowledgebase" construída na aplicação web que suporta o projecto (Redmine), informação essa disponível para todos os parceiros do projecto.

O modo de desenvolvimento da aplicação Android, com a aplicação do "design pattern" Façade, permite que seja disponibilizada uma biblioteca Android para a comunicação com o módulo Generic Entity, para que esteja já enquadrado com o modelo de dados aplicado à realidade dos sensores, funcionando como API Android para a dita comunicação, abstracção sob a chamada REST.

O estagiário contribuiu ainda, de maneira efectiva, para a melhoria dos sistemas com que interagiu, através da descoberta de *bugs* e contribuindo ainda para que o módulo OAuth se tornasse mais concordante com as implementações mais comuns, evoluindo de forma convergente para a feita pelo Google.

Para finalizar, o estagiário reportou ainda um erro do Socket.IO ao seu autor, tendo este sido corrigido, pelo que contribuiu também para a comunidade *Open-Source*.

9.4 Lições aprendidas

O projecto representou um enorme contributo para a formação do estagiário. De destacar a aplicação de conceitos de Engenharia de Software em ambiente real, que permitiu adquirir conhecimentos sobre aspectos fundamentais para o desenvolvimento de software com qualidade, relacionados com levantamento de requisitos, gestão de riscos, e criação de uma arquitectura. Dada a exposição da equipa de trabalho a um projecto com elevado número de interacções foi também possível também perceber as vantagens de uma metodologia de desenvolvimento de software ágil, como o Scrum.

Por fim, foram ganhas novas competências relacionadas com as chamadas "soft-skills", ao fazer parte de uma equipa com um número relevante de elementos, bem como um melhor conhecimento do tecido empresarial, através das vivências do dia-a-dia e do contacto com os diversos parceiros, assim como com os processos organizacionais associados ao trabalho desenvolvido.

Pode-se dizer que a experiência adquirida foi extremamente enriquecedora e que a componente de "Dissertação/Estágio" proporcionada pelo MEI uma peça fundamental na formação de um futuro engenheiro.

Bibliografia

[1] European Commission. ehealth week 2011: Information technology provides a cure for europe's mounting healthcare costs. http://ec.europa.eu/information_society/newsroom/cf/document.cfm?action=display&doc_id=789, 2011.

[2] European Commission. ehealth - industrial innovation - enterprise and industry. http://ec.europa.eu/enterprise/policies/innovation/policy/lead-market-initiative/ehealth/, 2011.

[3] PLUX. ecgplux | plux wireless biosignals | miniaturized emg biofeedback systems. http://www.plux.info/sensors, 2011.

[4] Metabolic equivalent - wikipedia, the free encyclopedia. hhttp://en.wikipedia.org/wiki/Metabolic_equivalent, 2011.

[5] Faq - what are counts? http://support.theactigraph.com/faq/counts, 2011.

[6] Courtney Johnston. User stories: A beginner's guide. http://www.boost.co.nz/blog/agile/user-stories/, 2012.

[7] Coley Consulting. Moscow prioritisation method. http://www.coleyconsulting.co.uk/moscow.htm, 2012.

[8] Google. Honeycomb | android developers. http://developer.android.com/about/versions/android-3.0-highlights.html, 2011.

[9] Tom Preston-Werner. Semantic versioning. http://semver.org, 2011.

[10] Universal control hub reference implementations. http://myurc.org/tools/UCH/, 2011.

[11] Dossia schemas - dossia wiki. http://wiki.dossia.org/index.php/Dossia_Schemas#Measurement, 2011.

[12] Dossia schemas - dossia wiki. http://wiki.dossia.org/index.php/Dossia_Enumerated_Listings#ClassificationEnumType, 2011.

[13] Dossia. Dossia | dossia personal health platform. http://www.dossia.org/, 2012.

[14] The indivo personally controlled health record. http://indivohealth.org/, 2011.

[15] Guillermo Rauch. Websockets everywhere with socket.io. http://howtonode.org/websockets-socketio, 2010.

[16] Ajax introduction. www.w3schools.com/ajax/ajax_intro.asp, 2012.

[17] S. Loreto, P. Saint-Andre, and G. Wilkins. draft-loreto-http-bidirectional-07 - known issues and best practices for the use of long polling and streaming in bidirectional http. http://tools.ietf.org/html/draft-loreto-http-bidirectional-07, 2011.

[18] Kenneth P Kufluk. Building mclaren.com - part 2: Serving telemetry. http://kenneth.kufluk.com/blog/2010/04/building-mclaren-com-part-2-telemetry/, 2010.

[19] Axod. Axod's hack: Websocket - some numbers. http://axod.blogspot.com/2009/12/websocket-some-numbers.html, 2011.

[20] Tavendo. Websockets implementation test report. http://www.tavendo.de/autobahn/testsuite/report/clients/index.html, 2011.

[21] Microsoft. Html5 (windows developer preview). http://msdn.microsoft.com/library/hh673546.aspx, 2011.

[22] Jack Moffitt. Xmpp is better with bosh » metajack. http://metajack.wordpress.com/2008/07/02/xmpp-is-better-with-bosh/, 2011.

[23] Inc Joyent. Node.js evented i/o for v8 javascript. http://nodejs.org/, 2011.

[24] Guillermo Rauch. Socket.io. http://socket.io/, 2011.

[25] Github - social coding. https://github.com/, 2012.

[26] Guillermo Rauch. Learnboost/socket.io - github. https://github.com/LearnBoost/socket.io/issues?page=2&state=open, 2010.

[27] Zynga poker. `www.zynga.com/games/zynga-poker`, 2012.

[28] WebMediaBrands Inc. Texas holdem poker - facebook application metrics from appdata. `http://www.appdata.com/apps/facebook/2389801228-texas-holdem-poker`, 2012.

[29] Inc Joyent. Modules · joyent/node wiki · github. `https://github.com/joyent/node/wiki/Modules#wiki-web-frameworks`, 2012.

[30] Express. `https://github.com/visionmedia/express/`, 2011.

[31] Geddy. `https://github.com/mde/geddy/`, 2011.

[32] Socketstream. `https://github.com/socketstream/socketstream`, 2011.

[33] Ejs- javascript templates. `http://embeddedjs.com/`, 2011.

[34] Jade - template engine. `http://jade-lang.com/`, 2011.

[35] Best practices | android developers. `http://developer.android.com/guide/practices/index.html`, 2011.

[36] Actionbarsherlock - home. `http://actionbarsherlock.com/`, 2011.

[37] Google. Android - introducing ice cream sandwich. `http://www.android.com/about/ice-cream-sandwich/`, 2011.

[38] Google. Android - android 4.1, jelly bean. `http://www.android.com/about/jelly-bean/`, 2011.

[39] Marcus Westlin. Building fast webapps, fast - velocity 2010. `http://slidesha.re/rrTJhL`, 2011.

[40] Google. Launcher icons | android developers. `http://developer.android.com/guide/practices/ui_guidelines/icon_design_launcher.html`, 2011.

[41] Google. User interface | android developers. `http://developer.android.com/guide/topics/ui/index.html`, 2011.

[42] Jakob Nielsen. How to conduct a simple user test with jakob nielsen. `http://www.youtube.com/watch?v=r0A6IW2TFFI`, 2011.

[43] Steve Krug. *Rocket Surgery Made Easy - The Do-It-Yourself Guide to Finding and Fixing Usability Problems.* New Riders Press, 2009.

[44] Scrum.org. *Scrum.org | Training, Assessments, Certifications - Scrum.org*, 2012.

[45] Jean-Philippe Lang. Overview - redmine. http://www.redmine.org/, 2012.

[46] Git - fast version control system. http://git-scm.com/, 2012.

[47] Vincent Driessen. A successful git branching model » nvie.com. http://nvie.com/posts/a-successful-git-branching-model, 2011.

[48] Vincent Driessen. nvie/gitflow - github. https://github.com/nvie/gitflow, 2011.

[49] Polar | listens to your body. http://www.polar.fi/en, 2011.

[50] Kaazing Corporation. Websockets everywhere with socket.io. http://websocket.org/, 2011.

[51] Alex Russell, Greg Wilkins, David Davis, and Mark Nesbitt. The bayeux specification. http://svn.cometd.com/trunk/bayeux/bayeux.html#toc_1, 2011.

[52] Meticube. *UCH Developer's Guide 1.0*, 2011.

[53] Upnp forum. http://www.upnp.org/, 2011.

[54] Universal control hub » pluggable user interfaces. http://myurc.org/publications/2006-Univ-Ctrl-Hub.php, 2011.

[55] PLUX. ecgplux | plux wireless biosignals | miniaturized emg biofeedback systems. http://www.plux.info/ecg, 2011.

[56] PLUX. tempplux | plux wireless biosignals | miniaturized emg biofeedback systems. http://www.plux.info/temp, 2011.

[57] PLUX. xyzplux | plux wireless biosignals | miniaturized emg biofeedback systems. http://www.plux.info/accelerometer, 2011.

[58] Scrum.org. Scrum.org | training, assessments, certifications - scrum.org. http://www.scrum.org/, 2012.

[59] Jack Moffitt. Strophe.js - an xmpp library for javascript. http://strophe.im/strophejs/, 2011.

[60] Alexey Shchepin. ejabberd community site. `http://www.ejabberd.im/`, 2012.

[61] Rabbitmq - messaging that just works. `http://www.rabbitmq.com/`, 2012.

[62] actionhero. `https://github.com/evantahler/actionHero/`, 2011.

[63] archetype. `https://github.com/jefftrudeau/archetype/`, 2011.

[64] aries. `https://github.com/edjafarov/aries`, 2011.

[65] blueprint. `https://github.com/hankejh/blueprint`, 2011.

[66] broke. `https://github.com/brokenseal/broke/`, 2011.

[67] Capsela. `https://github.com/capsela/capsela/`, 2011.

[68] Cargobox. `https://github.com/inruntime/cargobox`, 2011.

[69] chain. `https://github.com/hassox/chain/`, 2011.

[70] Coffeemate. `https://github.com/kadirpekel/coffeemate`, 2011.

[71] Coke. `https://github.com/dreamerslab/coke`, 2011.

[72] Crux. `https://github.com/kbjr/node-crux/`, 2011.

[73] Derby. `https://github.com/codeparty/derby/`, 2011.

[74] djangode. `https://github.com/simonw/djangode`, 2011.

[75] drty. `https://github.com/drtyhbo/drty`, 2011.

[76] Drumkit. `https://github.com/chrisjpowers/drumkit`, 2011.

[77] Ext core for nodejs. `https://github.com/mycoding/Ext-Core-for-NodeJS`, 2011.

[78] Flatiron. `https://github.com/flatiron/flatiron`, 2011.

[79] Genji. `https://github.com/zir/genji`, 2011.

[80] Grasshopper. `https://github.com/tuxychandru/grasshopper/`, 2011.

[81] Jaxserver. `https://github.com/jaxcore/jaxserver`, 2011.

[82] jimi. https://github.com/colingourlay/jimi, 2011.

[83] josi. https://github.com/thatismatt/josi, 2011.

[84] Kassit. https://github.com/marxus85/kassit, 2011.

[85] Katana. https://github.com/Shogun147/Katana, 2011.

[86] Kiss.js. https://github.com/stanislavfeldman/kiss.js, 2011.

[87] Locomotive. https://github.com/jaredhanson/locomotive, 2011.

[88] merlin. https://github.com/brynbellomy/node-merlin, 2011.

[89] Meryl. https://github.com/kadirpekel/meryl, 2011.

[90] Mojito. https://github.com/yahoo/mojito/, 2011.

[91] Monorail.js. https://github.com/runexec/Monorail.js, 2011.

[92] N-ext. https://github.com/xcambar/n-ext, 2011.

[93] node-extjs. https://github.com/egorFiNE/node-extjs, 2011.

[94] nodemachine. https://github.com/tautologistics/nodemachine, 2011.

[95] nodepress. https://github.com/zir/nodepress, 2011.

[96] Piejs. https://github.com/fakewaffle/piejs, 2011.

[97] Protos. https://github.com/derdesign/protos, 2011.

[98] Quickweb. https://github.com/leizongmin/QuickWeb, 2011.

[99] Railwayjs. https://github.com/1602/express-on-railway, 2011.

[100] Seek. https://github.com/fk1blow/Seek/, 2011.

[101] spludo. https://github.com/DracoBlue/spludo, 2011.

[102] Stick. https://github.com/olegp/stick, 2011.

[103] Tower. https://github.com/viatropos/tower, 2011.

[104] oauth-signpost - simple oauth message signing for java - google project hosting. http://code.google.com/p/oauth-signpost/, 2011.

[105] Actionbarsherlock - home. `http://code.google.com/p/google-gson/`, 2011.

[106] arunoda/node-simple-xmpp · github. `https://github.com/arunoda/node-simple-xmpp/`, 2011.