Mestrado em Engenharia Informática
Dissertação/Estágio
Relatório Final

Upgrade do OneCare para Arquitetura Microserviços

Mário Alexandre Arzileiro Pereira
maarz@student.dei.uc.pt

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

Mestrado em Engenharia Informática
Dissertação/Estágio
Relatório Final

Upgrade do OneCare para Arquitetura Microserviços

Mário Alexandre Arzileiro Pereira
maarz@student.dei.uc.pt

Orientadores:
Prof. Dr. Bruno Cabral
Eng. António Damasceno

Juri:
Prof.ª Dr.ª Bernarde Ribeiro
Prof. Dr. Carlos Fonseca

Data: 1 de julho de 2016

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

Agradecimentos

Ao Professor Dr. Bruno Cabral pela orientação, motivação, disponibilidade e acompanhamento ao longo desta reta final.

Ao Engenheiro António Damasceno pelo companheirismo, pela amizade, pela confiança que deposita em mim, pelo entusiasmo e por me dar esta oportunidade de conciliar o meu percurso enquanto nadador com o meu percurso enquanto estudante.

Aos meus pais e à minha irmã, porque os últimos tempos não têm sido fáceis, mas que me têm dado mais um exemplo de persistência e de luta por uma vida melhor, deixando sempre que o amor entre nós prevaleça. OBRIGADO!

Á Patrícia, pela paciência, dedicação e empenho que desde sempre me habituou, que me eleva a vontade de continuar a partilhar as experiências da vida, com a certeza de que o amor é a chave para o sucesso. Contigo, tenho a certeza que terei sempre por perto uma mão que me indicará o caminho nos momentos difíceis, e partilhará comigo os sucessos nos momentos de felicidade, sem nunca pedires nada em troca.

A toda a minha família que, mesmo sem se aperceberem, foram dando o seu contributo. Com um carinho muito especial, aos meus avós, que por esta altura seriam, de certo, as pessoas mais orgulhosas de mim.

E àquela que, não o sendo, tem sido a minha família diária, ao Carlos e à Lídia por me apoiarem e me receberem junto deles todos os dias, como um filho, e estarem sempre nas minhas vitórias e derrotas. À Cristina, ao Vitor e a toda a família da Patrícia, também.

Aos meus colegas de faculdade, em especial ao João Sousa por ter feito quase todo o percurso universitário comigo, por ter aturado os meus horários reduzidos para os trabalhos, por não perder a paciência com as minhas ausências constantes e, principalmente, por ser um amigo que levo comigo.

Aos meus colegas da Intellicare e da VPS, em especial à Marisa e à Marta, pela companhia diária, pelas partilhas e por fazerem de mim um profissional melhor.

Ao Miguel, ao Potter, ao João Moreira, ao André Vaz e a toda a equipa do CNAC, companheiros de cloro, de dedicação, de frustrações, mas principalmente de vitórias.

Resumo

A rápida evolução que o mundo vive hoje em dia, acentua a necessidade de adaptar e atualizar os sistemas de informação em termos tecnológicos e de resposta às novas necessidades dos utilizadores e dos clientes. Nas empresas que fornecem serviços de aquisição e controlo de dados de múltiplos dispositivos, essa mudança é mais acentuada pela quantidade de dispositivos inteligentes que são colocados no mercado.

A Intellicare desenvolveu a linha de produtos OneCare, que necessita de ser atualizada com frequência. No entanto, a arquitetura da solução atual não permite proceder a mudanças com a facilidade e rapidez desejada. Utilizando o processo técnico *Architecture Centric Design Method* foram identificadas necessidades da empresa, em termos de desafios tecnológicos e de negócio, e criada uma nova arquitetura de sistema que permite a adição rápida de novos componentes, reduzir os tempos de testes e colocar novas funcionalidades à disposição dos clientes.

Foi produzida uma arquitetura que cumpre os requisitos da Intellicare na rápida progressão e crescimento contínuo, mantendo os padrões do controlo de qualidade. A arquitetura foi validada através da implementação de serviços que, em conjunto, formam um novo produto da empresa. Espera-se que a linha de produtos seja progressivamente migrada para a nova arquitetura.

Palavras-Chave

"Arquitetura de Software", "Dados especialmente sensíveis", "Engenharia de Software", "Gestão de Projeto", "Internet das Coisas", "Micro serviços"

Abstract

The rapid changes that the world is experiencing today, stresses the need to adapt and update the information systems in terms of technology and responding to the needs of users and customers. In companies that provide services for the acquisition and control of data from multiple devices, this change is more pronounced by the amount of smart devices that are placed on the market.

Intellicare product line, OneCare, need to be updated frequently. However, with the current architecture it not possible to not make changes with the desired ease and rapidity. Using the technical process Architecture Centric Design Method, the business needs were identified, in terms of technological and business challenges, and a new system architecture was created to enable the rapid addition of new components, reducing the time spent on testing and making new features available for customers.

An architecture that meets the Intellicare requirements in rapid progression and continuous growth, maintaining the standards of quality control was produced. The architecture has been validated through the implementation of services that together form a new company product. It is expected that the product line gradually be migrated to the new architecture.

Keywords

"Internet of Things", "Microservices", "Project Management", "Software Architecture", "Software Engineer", "Special sensitive data"

Índice

Lista de Figuras

Lista de Tabelas

Glossário

Tabela 1 – Glossário e Acrónimos do relatório

Acrónimo	Descrição
MP	Mário Pereira
AD	António Damasceno
BC	Bruno Cabral
AAL	*Ambient and Assisted Living*
VPS	*Virtual Power Solutions*
EAR	*Enterprise Application Repository*
MD	*Medical Device(s)* – Dispositivos médicos
IoT	Internet das Coisas – original "*Internet of Things"*
FI	*Future Internet* ou Internet do Futuro
CNPD	Comissão Nacional de Proteção de Dados
ISO	Organização Internacional para a Normalização – original *"International Organization for Standardization"*
ASR	Requisito Arquitetonicamente Significante – original: *Architecturally Significant Requirement*
CRUD	*Create, Read, Update, Delete*
ER	Entidade e Relação
CMMI	*Capability Maturity Model - Integration*
TE	*Threads* de execução
OE	Objetivo especifico
PE	Prática especifica
N/A	Não aplicável
WBS	Estrutura Analítica do Projeto – original *"Work Breakdown Schedule"*
CDI	*Contexts and Dependency Injection*
WF	*WildFly*

Capítulo 1.
Introdução

O presente documento tem como objetivo descrever trabalho desenvolvido pelo aluno Mário Pereira durante o seu estágio na empresa Intellicare.

A Intellicare tem no mercado a linha de produtos OneCare, que têm como principal objetivo a aquisição de dados e monitorização de pessoas idosas com sintomatologia de demência. Constituída pelos produtos OneCare *Safe*, que monitoriza a localização de pessoas através de um equipamento móvel dispondo de um botão de alerta e de chamadas com centrais telefónicas especializadas nos cuidados de saúde; OneCare *Sensing*, que recolhe os dados de diversos sensores biométricos e permite a consulta desses dados por parte dos cuidadores e familiares; e o OneCare AAL (*Ambient Assisted Living*), que recolhe informação de sensores de ambiente na casa dos utilizadores e que permite a execução de ações à distância por parte dos utilizadores.

Não sendo produtos maduros, frequentemente aparecem novas necessidades de clientes às quais é obrigatório dar resposta. Para aproveitar janelas de mercado é necessário desenvolver novas funcionalidades, produtos ou serviços. Contudo, o crescimento da solução, a falta de recursos e a dimensão dos testes realizados, faz com que o desenvolvimento abrande para garantir a qualidade da solução.

Alguns exemplos de necessidades pelo qual a Intellicare já vivenciou são: novos sensores de atividade para o OneCare Safe, caso da inclusão dos relógios da *Vivago*, serviço de monitorização de atividade e localização de curta distância; sistemas externos que é necessário integrar ou para os quais é preciso enviar dados, caso do portal da saúde; e importar clientes e utilizadores de outras plataformas existentes, caso do projeto *giraffplus*. Quando estas funcionalidades foram implementadas com os recursos fornecidos pela linha de produtos, e como era de esperar, foram reutilizados os planos de teste da solução, no entanto, a Intellicare deparou-se com o seguinte problema.

Como para todos os casos anteriores, o *deploy* é monolítico, isto é, é instalado um pacote de software no servidor. É construído um pacote EAR que é introduzido numa pasta especifica do servidor e os seus componentes são inicializados através de mecanismos próprios do sistema e é necessário garantir que todas as funcionalidades instaladas estão corretas. No caso de serem feitas atualizações, o facto de reutilizar funcionalidades em cada versão, é necessário rever e acrescentar a bateria de testes de todo o produto para garantir a qualidade. A revisão foca os aspetos que foram objeto de intervenção e são incluídos novos cenários de utilização. Neste momento os testes ao produto, com um recurso a tempo inteiro, necessitam de 40 horas de esforço. Tendo em conta que: há funcionalidades dependentes de deteção de eventos ao longo do dia; o facto de existem tarefas que, independentemente do número de pessoas alocadas, o tempo de espera para que os eventos ocorram é constante, a utilização de duas ou mais pessoas na execução de testes pode reduzir o tempo total de execução de cinco para três dias mantendo o esforço de 40 horas.

Quando a empresa tem necessidade de incluir uma funcionalidade de dimensão reduzida, que não tenha dependência das já existentes ou com dependências limitadas, o esforço de execução de testes é superior ao esperado tendo em conta o esforço de desenvolvimento, que tipicamente deveria ser 30% do total do projeto. Em alguns casos o tempo de desenvolvimento não excedeu 20 horas mantendo-se os testes nas 40 horas.

Para resolver o problema identificado é proposta uma atualização da arquitetura existente. Atualmente a linha de produtos OneCare segue um padrão baseado em eventos (*event-driven architecture*) que em muitos casos, os eventos são criados e executados por agentes independentes, mas que mesmo assim a arquitetura não suporta a versatilidade exigida. O objetivo da atualização da arquitetura do OneCare é alterar a estrutura das aplicações de forma a que seja mais fácil e rápida a atualização, desenvolvimento e testes de novos serviços.

Uma particularidade O OneCare trata dados classificados como dados especialmente sensíveis pela Legislação de Proteção de Dados Pessoais, enquadrando-se no âmbito dos softwares *Medical Devices* (MD). As preocupações relativas à proteção de dados são transversais a toda a solução, já as dos MD colocam exigências de certificação dos componentes que transformam os dados relativos à saúde.

1.1. Enquadramento

O trabalho desenvolvido durante o estágio é realizado na empresa Intellicare, conta com a orientação do Eng. António Damasceno, *Acting*-CEO da Intellicare, e do Doutor Bruno Cabral, professor na Universidade de Coimbra. Teve inicio a 14 de setembro de 2015 e terminou a 15 de julho de 2016.

A Intellicare, sediada no IPN – Instituto Pedro Nunes, edifício D é uma empresa que se dedica à monitorização da saúde e bem-estar da população, à criação de soluções para *ambiente assisted living (AAL)* e auxilio à localização de pessoas e bens. Os principais alvos de mercado são farmácias, serviços domiciliários, residências assistidas, lares, centros de dia, hospitais, centros de saúde e unidades de serviço de família.

O capital da Intellicare é integralmente detido pela Virtual Power Solutions (VPS), empresa dedicada à gestão e monitorização de energia em ambientes domésticos e industriais, ou pelos acionistas da mesma. A Intellicare está incubada na VPS o que significa que, toda a estrutura não produtiva é partilhada por ambas as empresas havendo ainda partilha de alguns recursos técnicos. Para efeitos do estágio é significativo o processo de Qualidade que é partilhado entre as empresas.

Com cerca de 20 anos de experiência no desenvolvimento de soluções inteligentes de monitorização e controlo à distância, a empresa pretende contribuir para o envelhecimento ativo da população, mantendo o seu foco na prevenção. Com sede em Portugal, a Intellicare pretende que o OneCare, chegue a diferentes pontos do globo, contando já com mais 60 mil equipamentos instalados por todo o mundo.

A linha de produtos OneCare é composta por:

- *Safe*, uma solução para acompanhamento de idosos, de pessoas com demência ou com necessidades especiais, em ambiente domiciliário. O utente utiliza um dispositivo que lhe permite acionar um alerta para o prestador de cuidados e/ou familiar, bem como efetuar chamadas e/ou sms. O equipamento permite ainda ao familiar/cuidador saber a localização do utente, sempre que for necessário;
- *Sensing*, um sistema de monitorização remota de parâmetros de bio marcadores (tensão arterial, frequência cardíaca, ECG, glicemia, peso, oximetria, temperatura corporal, entre outros) em ambiente domiciliário. O utente pode ser monitorizado à distância pelo seu médico ou enfermeiro, de forma rigorosa, confortável e económica;
- *Vivago* residencial ou institucional, um sistema de monitorização e acompanhamento de idosos que permite analisar a sua atividade diária, com informação em tempo real, disponível 24/7. Sempre que ocorra um desvio ao padrão normal de comportamento

do utente, será lançado um alerta automático (e.g. em situações de queda ou de inconsciência dos utentes), de forma personalizada, mesmo quando as circunstâncias impedem a utilização do botão de alarme (o que sucede em 80% das situações de emergência com idosos). É um sistema orientado para idosos que vivem sós, ou passam muito tempo sozinhos, trazendo tranquilidade aos seus familiares e contribuindo para o aumento do seu bem-estar. Em contexto institucional várias estações têm capacidade para receber dados de múltiplos utentes portadores do relógio.

1.2. Motivação

A linha de produtos OneCare, apesar de ser composta por várias aplicações, é em grande parte implementada sobre uma arquitetura monolítica, em que os componentes estão, na sua maioria, muito ligados entre si, e onde a verificação da solução é realizada por inteiro. Facto que torna difícil conciliar o desenvolvimento rápido de novos serviços secundários ou adjacentes à solução, com a garantia de um forte controlo de qualidade, colidindo, por vezes, com o tempo para chegar ao mercado. Esta situação piora com o crescimento contínuo do produto. E, apesar de algumas atualizações ao mesmo envolverem apenas alguns componentes, como é normal e esperado, devido à arquitetura, é sempre necessário garantir a qualidade da solução por inteiro executando baterias de testes na sua totalidade. Sendo que no caso da linha de produtos OneCare, a bateria completa de testes pode demorar até cinco dias a ser executada, há a noção clara que a implementação de algumas funcionalidades, como não teve impacto na maioria dos componentes, estes continuam válidos e verificados não sendo necessário a sua verificação novamente.

Dado que alguns dos novos componentes são apenas drivers para novos sensores, novos protocolos de interoperabilidade ou atualizações de protocolos de terceiros, apenas seria necessário provar que os componentes novos ou os que sofrem alterações, e os componentes que com eles interagem, se comportam como esperado, não havendo necessidade de realizar testes totais ao produto. Espera-se que a implementação de uma arquitetura alternativa, permitirá que o tempo de desenvolvimento e testes de componentes possam ser reduzidos acelerando a entrada no mercado de novas funcionalidades.

1.3. Objetivos

O objetivo do estágio é estudar a viabilidade da alteração da arquitetura existente procurando encontrar soluções que resolvam o problema identificado, sendo necessário validar a hipótese de micro serviços. Para cumprir o objetivo torna-se necessário realizar as seguintes atividades: (1) planeamento e monitorização do projeto; (2) produzir um estado de arte (EA) relevante para o trabalho; (3) identificação dos requisitos, atributos de qualidade e restrições técnicas e de negócio da nova arquitetura; (4) a elaboração da documentação da arquitetura criada; (5) a validação da adequação da mesma, (6) o desenvolvimento de um protótipo funcional de serviços OneCare na nova arquitetura.

1.4. Contribuições

A contribuições finais resultantes do estágio são a elaboração de uma documentação de arquitetura que resolva o problema identificado anteriormente e a implementação de um serviço segundo a nova especificação de arquitetura que vá de encontra, aos seus drivers arquiteturais e cumpra com os requisitos do cliente.

1.5. Estrutura do Relatório

O Capítulo 1 pretende descrever o problema, o enquadramento do estágio e apresentar a motivação que leva à sua realização assim como os resultados que se esperam.

No Capítulo 2 o objetivo é identificar e descrever a estrutura das ferramentas de AAL disponíveis, focando na estrutura de plataformas IoT (*Internet of Things*). Atualmente, estes são os pilares no desenvolvimento de soluções para a aquisição, monitorização e processamento de dados de sensores e dispositivos com o foco na qualidade de vida da população idosa.

O primeiro objetivo do estágio é documentado no Capítulo 3 onde são identificados e priorizados os *drivers* arquiteturais, conjunto composto pelos requisitos do sistema, atributos de qualidade e restrições técnicas e de negócio mais importantes da arquitetura alternativa.

O segundo e principal objetivo do estágio é documentado no Capítulo 4 onde são analisadas as arquiteturas derivadas do OneCare, entenda-se como derivadas o padrão da arquitetura atual, a arquitetura dita alvo e uma derivação próxima desta ultima. É apresentada a proposta de arquitetura elaborada para a resolução do problema e é feita uma avaliação se esta é suficiente e se cumpre com os drivers arquiteturais exigidos.

Com o objetivo de validar a arquitetura produzida, o Capítulo 5 documenta o processo realizado. É utilizado um projeto real de um novo produto/serviço da Intellicare para justificar e validar a estrutura elaborada. Como indicado, o novo serviço é tratado como um projeto onde são identificados os seus requisitos, é desenhada a arquitetura com base na especificação da arquitetura produzida no capítulo 4 e documentada a implementação do serviço. É depois validada a estrutura da arquitetura com base nessa implementação.

No Capítulo 6 é dado a conhecer o planeamento das tarefas realizadas e a metodologia dos procedimentos aplicados durante o estágio, com o foco nos processos de gestão de projeto e controlo de qualidade.

Por fim, no Capítulo 7 são apresentadas as conclusões dos resultados obtidos com a mudança de paradigma no desenvolvimento de produtos e serviços através de um novo mecanismo de expansão de uma solução através da decomposição de funcionalidades em serviços independentes. É ainda referido o que se espera realizar como trabalho futuro e é feita uma reflexão profissional e pessoal do trabalho realizado.

Capítulo 2.
Estado da Arte

Um dos entregáveis iniciais de um projeto de *software* é a Arquitetura de Software, está presente ao longo de todo o processo de criação/manutenção de software, contém a estrutura da solução que resolve o problema tendo em conta os drivers arquiteturais. As decisões arquiteturais de estrutura e tecnologia são determinantes na definição de como o sistema irá ser estruturado de forma a satisfazer os requisitos, atributos de qualidade e restrições identificadas no documento de requisitos.

Os drivers arquiteturais definem a qualidade do sistema e têm influencia direta na disposição dos componentes pelas exigências de performance, segurança, etc. Pela necessidade sistemática deste tipo de características em soluções de software, é normal que a decisão arquitetural de uma solução surja de um tipo de arquitetura padrão, bem conhecida, testada e validada, que garanta à partida a confirmação dos drivers arquiteturais.

As soluções do OneCare têm uma grande componente de recolha de dados de diversos dispositivos e monitorização do ambiente no contexto da assistência a idosos, inserindo-se na abordagem do AAL. Na avaliação do estado da arte não vai ser abordada a componente funcional do AAL, mas sim a estrutura não funcional ou arquitetural. A base estrutural das aplicações AAL é o paradigma *Internet of Things* (*IoT*) que suporta a conexão de múltiplos dispositivos utilizando múltiplas plataformas realizando a transmissão e análise de dados *on-demand* (Gubbi et al. 2013)

Com base no paradigma *IoT* existe uma grande oportunidade para extrair conhecimento através dos diferentes dispositivos transversais e multidisciplinares que estão ligados à internet. Devido à quantidade enorme de informação a ser gerada e partilhada surge a necessidade de aumentar o foco das arquiteturas em termos de escalabilidade, integração e segurança. (Oracle, 2014)

O objetivo deste estado de arte é identificar e descrever a componente estrutural de plataformas AAL de referência, e procurar perceber como é que a arquitetura *IoT* se comporta em termos de modificabilidade, testabilidade, escalabilidade e interoperabilidade para essas plataformas, para conseguir extrair táticas que resolvam o problema identificado no OneCare.

2.1. Oracle IoT

O OneCare é implementado em Java, pelo que a arquitetura *IoT* proposta pela Oracle para Java é uma referência importante. A Oracle faz também parte dos grupos de trabalho de standarização pelo que o modelo apresentado na Figura 1, esteja muito próximo do que pode ser o standard de uma arquitetura IoT.

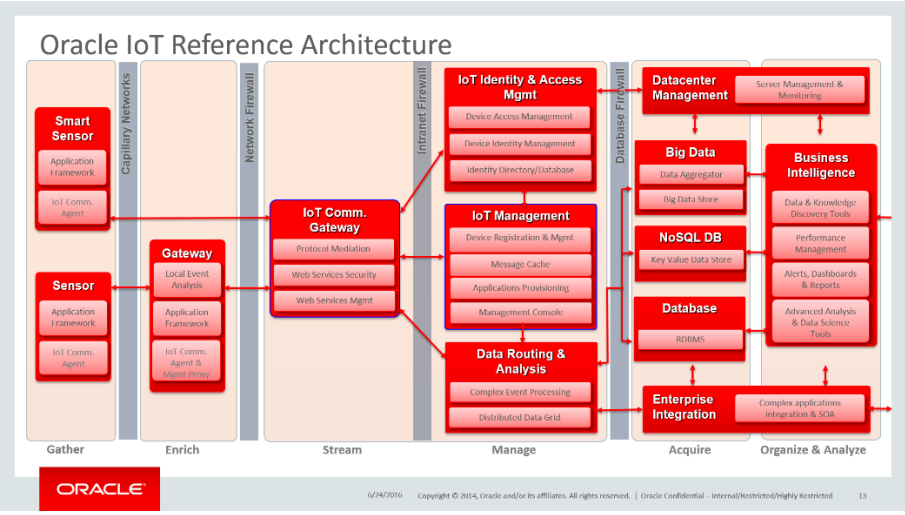


Figura 1 – Arquitetura de referência de IoT [1]

A arquitetura presente na Figura 1 apresenta seis camadas de informação (recolha, enriquecimento, comunicação, gestão, aquisição e análise) separadas por mecanismos de controlo. Um método de comunicação de curta distância de rede capilar, como exemplo, a comunicação Bluetooth e *zigbee*, e três camadas de segurança com *firewalls,* uma que controla o trafego da internet, uma *firewall* interna de intranet e a *firewall* de servidores ou aplicações. Pela complexidade de camadas e interação entre elas a arquitetura privilegia a segurança, o ligeiro acoplamento dos módulos garante não só escalabilidade, mas ao mesmo tempo a sua interoperabilidade.

As soluções IoT devem fornecer acesso controlado e autenticação da informação enquanto mantem a privacidade dos utilizadores (tecnológicos ou humanos). Os sistemas também devem ser resilientes a ataques providenciando graus elevados de fiabilidade. Por existir um modelo de negócio que necessita de ser alimentado com recursos que auxiliem a tomada de decisões, em sistemas IoT a forma como a inteligência gerada através da análise dos dados dos sensores, tem de ser orquestrada e gerida entre os sistemas envolvidos (Oracle, 2014). Como exemplos temos as Smart Cities, com aplicações de encaminhamento de veículos com o objetivo de controlar o estacionamento ou o trafego rodoviário; Smart Grids, com aplicações de gestão energética para o equilíbrio energético das redes, por exemplo, através da transferência das cargas não essenciais das horas de maior consumo, em que a rede está próxima do limite disponível, para as horas de vazio.

O importante a reter desta analise é a segmentação das diferentes camadas, o desacoplamento entre os componentes, e as diversas *boundaries* que tratam da comunicação e da segurança. O que permitiu decidir que uma arquitetura baseada em serviços ligeiramente desacoplados permitiria implementar este modelo de referência.

2.2. AAL4ALL

O projeto AAL4ALL é um projeto mobilizador com a intenção de inserir os produtos e serviços de ambiente assistido no mercado português. O projeto inclui o desenvolvimento de um ecossistema interoperável de produtos e serviços de assistência a pessoas, suportado por sistemas tecnológicos de informação. O AAL4ALL consiste num sistema que permite a agregação e integração de um leque de diferentes aplicações e serviços com a visão de disponibilizar o conteúdo dos serviços AAL aos utilizadores finais.

[1] Robert Stackowiak, (2014) Architecting for the Internet of Things & Big Data – *slide de apresentação*

O sistema AAL4ALL é composto por uma plataforma central e uma local. A plataforma central agrega, orquestra e processa os serviços AAL, ficando responsável por os manter disponíveis para uso através de *Cloud*. A plataforma local, tal como o nome indica, é responsável pelas mesmas ações da plataforma central, mas desta vez, dos serviços locais.

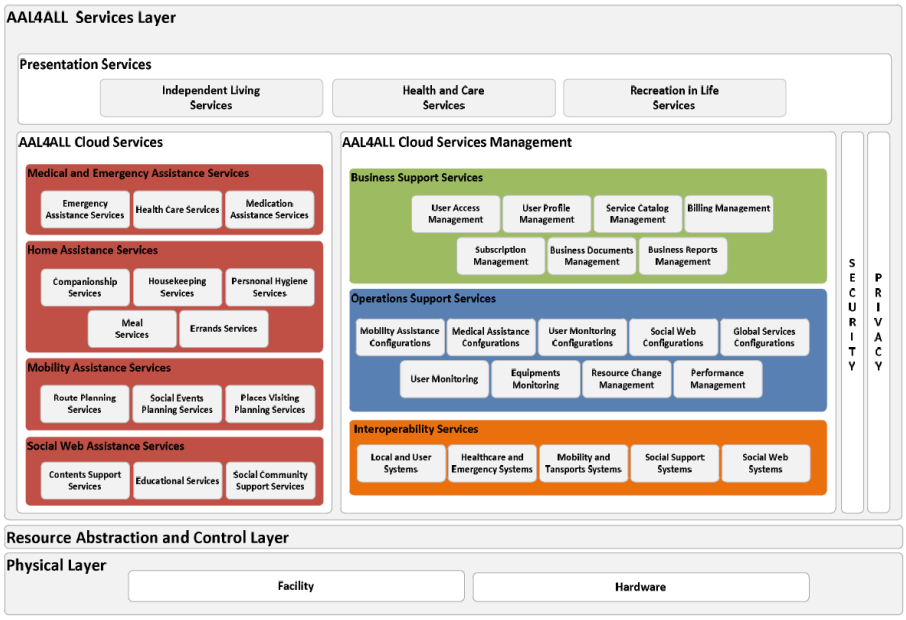


Figura 2 – Arquitetura de referência para o AAL4ALL[2]

A arquitetura presente na Figura 2 é composta por três camadas: serviços, abstração e controlo, e física. A camada de serviços divide-se por: serviços de apresentação, disponibilizando o acesso às funcionalidades organizadas de forma intuitiva aos utilizadores finais; serviços da *Cloud* , bem divididos pelo seu domínio; a gestão dos serviços da *Cloud*, responsável pelos serviços de gestão das aplicações que os programadores desenvolvem e disponibilizam na plataforma; e a segurança e privacidade, transversal a todas as camadas (apesar de não ser explicito na Figura 2) onde estão endereçados todos os requisitos de segurança, autenticação, confidencialidade, integridade e monitorização, e ainda, assegurar a privacidade dos dados dos sensores e dos utilizadores (Pereira et al., 2014).

A Intellicare fez parte do consorcio AAL4ALL, neste momento, a integração efetuada está muito longe de ser um produto que possa ser usado comercialmente.

Não é evidente nesta arquitetura a forma como são cumpridos os requisitos em termos da Legislação da Proteção de Dados pelo que é difícil justificar perante o regulador como os mesmos são cumpridos. Na solução a implementar os mecanismos de segurança deverão ser evidentes facilmente demonstráveis e verificáveis.

2.3. FIWARE

O FIWARE (*Future Internet Software*) é uma comunidade e tem como missão: "construir um ecossistema aberto à volta de plataformas standard públicas, livres de custos e com software modular, que facilitem o desenvolvimento de novas aplicações inteligentes em múltiplos setores". A plataforma disponibiliza um conjunto de *API's* que facilitam o desenvolvimento de aplicações de recolha de informação e auxílio na toma de decisões em múltiplos setores de negócio. As especificações das *API's* são *open-source* e livres de obrigações, permitindo que se

[2] Fonte: (Pereira et al., 2014)

criem novos produtos mais rapidamente e entrem no mercado com um custo de produção inferior. (Ramparany, 2014)

A importância do FIWARE nos sistemas de *AAL* é a possibilidade de fazer uso da informação *open data* disponibilizada pela plataforma, por exemplo, informação contexto proveniente de sensores ambientais. A utilização de informações sobre a temperatura, condições climatéricas ou eventos socias, pode ajudar a prever as ações dos utilizadores finais, a incentivar à atividade tais como promover a saída de casa, ou em contraste, se a previsão é de chuva, o aconselhamento pode ser tirar a roupa do estendal.

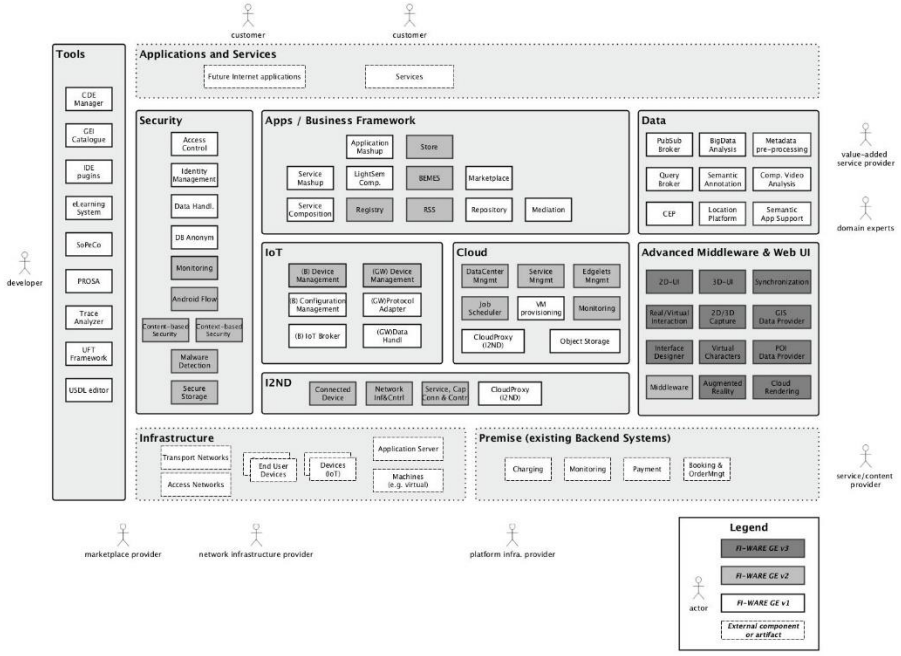


Figura 3 – Arquitetura de referência do FIWARE[3]

Na Figura 3 é ilustrada a arquitetura de referência para o FIWARE que dispõe de oito camadas para a sua estrutura e três de cariz externo. Esta arquitetura privilegia o uso de *Generic Enablers* (*GE's*) que permitem a integração entre as aplicações e serviços existentes na plataforma onde as principais camadas são: Ferramentas que pretendem responder ao desafio de oferecer um conjunto de multiplataformas de desenvolvimento de aplicações FIWARE onde é dado o suporte total ao desenvolvedor desde o desenvolvimento à gestão em produção; Dados: camada responsável por fornecer a recolha e o processamento de quantidades elevadas de dados, dispondo de vários *GE's* de *Big Data Analysis,* Processamento de eventos complexos, entre outros; e Segurança: a internet do futuro (FI) vai ser sempre exposta a ameaças, criar uma estrutura segura e confiável mas mantendo a sua funcionalidade, usabilidade, performance com custo reduzido é um grande desafio. A ambição da segurança do FIWARE é demonstrar que a visão da "Segurança por *design*" é real através do que já se faz e do rápido desenvolvimento nesse sentido (Ferreira, 2015).

Existem ainda desafios por resolver, é impossível criar modelos de negocio em cima de FIWARE uma vez que por imperativo legal é necessário estabelecer contratos entre a entidade responsável pelo processamento de dados e os fornecedores da tecnologia que garanta o cumprimento da legislação da proteção de dados (3.4.3- Legislação de Proteção de Dados) não havendo neste momento *datacenters* que forneçam serviços FIWARE de forma comercial. A separação clara dos *GE's* é interessante como modelo para a definição dos serviços no OneCare.

[3] Fonte: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_Architecture_R3

2.4. OneCare

Atualmente, a solução OneCare, apresentada na Figura 4, está implementada numa arquitetura monolítica e parte da comunicação entre componentes é baseada em eventos. Um evento é qualquer ação que ocorra num determinado ambiente, normalmente, os eventos apresentam-se como forma de mensagens que identificam um problema, um limite alcançado, um desvio, etc. Numa arquitetura baseada em eventos, uma mensagem é produzida por um agente (produtor) e espalhada por todas as partes interessadas (subscritores). Uma ação pode despertar um ou mais serviços, aplicações ou um qualquer processo de negócio, que interpretam a mensagem e opcionalmente executam ações sobre ela. Pelo padrão, a arquitetura é independente e altamente distribuída (Michelson, 2006). No OneCare o uso desta capacidade de fluxo de eventos é usado para distribuir o processamento de um evento pelos diferentes componentes (core, registo, gestão de dados, etc) de forma assíncrona.

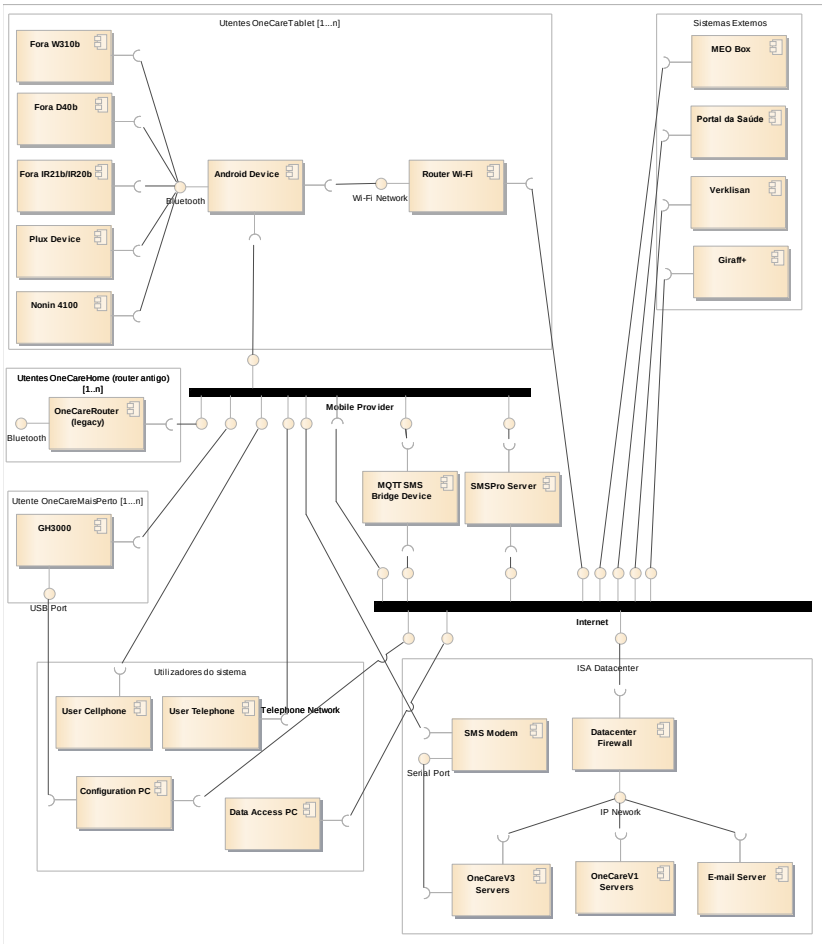


Figura 4 – Diagrama de alocação da arquitetura da linha de produtos do OneCare[4]

Com a análise a esta arquitetura fiquei a conhecer o estado atual do produto em termos estruturais, e perceber melhor a origem do problema a resolver no estágio.

2.5. Análise Comparativa

Foi feita uma análise comparativa às plataformas presente na Tabela 2. É baseada em atributos de qualidade mais significativos para o problema, nomeadamente a modificabilidade e a testa-

[4] Fonte: ISA Intellicare, Especificação da Arquitetura OneCare V3.

bilidade, são ainda incluídos atributos que, apesar de não explícitos no problema, são preocupações habituais de um sistema de software de natureza *IoT* e *AAL*, a escalabilidade, disponibilidade e segurança. A classificação é feita por Baixo (↓), Médio (→) e Alto (↑), consoante o grau de satisfação que a plataforma apresenta em relação às necessidades da Intellicare face ao atributo de qualidade. Sendo que, Baixo – significa que a plataforma não satisfaz as necessidades da empresa, Médio – satisfaz, mas não apresenta valor acrescido para a empresa, Alto – satisfaz, e apresenta vantagens para a resolução do problema. Os seguintes descritores identificam o que a Intellicare pretende para cada atributo de qualidade e o seu critério de avaliação.

Modificabilidade

- Baixa - a plataforma que não permitir a separação dos diferentes componentes ativos do sistema;
- Média - quando a plataforma oferecer a possibilidade do desacoplamento dos componentes por métodos aplicados;
- Alta - que junta à avaliação "média" a independência da linguagem de implementação.

Testabilidade

- Baixa - a plataformas que com a atualização de um componente necessitem de ser testadas na sua totalidade;
- Média - é possível testar serviços em separado, mas as dependências estão documentadas apenas em texto;
- Alta - dependências entre serviços estão formalmente identificadas no sistema sendo possível testar apenas os serviços alvo e os de que eles dependem.

Escalabilidade

- Baixa - apenas permite escalabilidade vertical;
- Média - não permite multiplicar os componentes ativos sem multiplicar a solução por inteiro;
- Alta - permite alocar componentes individuais a diferentes nós do cluster.

Disponibilidade

- Baixa - não dispões de mecanismos de *high availability*;
- Média - tem mecanismos de *high availability* mas está dependente de um único ponto de entrada;
- Alta - tem mecanismos de *high availability* com múltiplos pontos de entrada.

Segurança

- Baixa - sem proteção de canal e de dados no servidor;
- Média - canal encriptado dados não encriptados;
- Alta - canal e dados encriptados.

Tabela 2 – Comparação entre plataformas

	Oracle *IoT*	**AAL4ALL**	**FIWARE**	**OneCare**
Modificabilidade	↑	↑	↑	↓
Testabilidade	→	↓	→	↓

Escalabilidade	↑	→	↑	→
Disponibilidade	↑	↓	↑	→
Segurança	↑	↑	→	↑

A classificação das diferentes plataformas foi feita através da análise das diferentes arquiteturas de referência e pelos documentos de suporte às mesmas.

Podemos extrair desta tabela que nem tudo no OneCare está abaixo do pretendido. A Segurança oferecida pela arquitetura atual do OneCare é elevada devido à sua estrutura monolítica, expondo a aplicação a um certo número de aplicações apenas necessárias ao contexto interno da aplicação. Fato este que justifica a fraca manutenção e escalabilidade do sistema.

A manutenção de um sistema de *IoT* deve ser abordada do ponto de vista do FIWARE, onde são definidas e programadas estruturas (*GE*) e disponibilizadas como dados e processamento com licenças *open-source.*

A Oracle pode ter a solução no que toca à segurança na comunicação de dados de dispositivos pela internet, fortalecendo os limites da transmissão com *firewalls* e mecanismos de segurança adequados.

Modificabilidade AAL4ALL Apesar de estarem desacopladas há uma forte dependência na comunicação.

Capítulo 3.
Requisitos e Atributos de Qualidade

Com a análise de plataformas cujo foco é a avaliação de alternativas e soluções ao problema do OneCare, segue-se a primeira etapa no desenvolvimento de software que é compreender o problema e redigir os documentos de requisitos. Nesse documento, deve estar presente o problema que se pretende ver resolvido e o que seria uma solução satisfatória. Ou seja, pretende-se que no fim do documento seja compreendido os requisitos funcionais os atributos de qualidade e as restrições que em conjunto formam os drivers arquiteturais.

O método que será aplicado ao longo do estágio é o *Architecture Centric Design Model* (ACDM), com algumas derivações. Este modelo determina que nas suas 1ª e 2ª fases se realize o processo de elicitação de requisitos e atributos de qualidade. O objetivo do presente capítulo é descrever o referido processo e os resultados obtidos através do mesmo. A primeira tarefa realizada foi a identificação dos *stakeholders,* demonstrada na Tabela 3.

Tabela 3 – Identificação de stakeholders

Nome	Papel	Posição	Interesse	Poder
Mário Pereira	Gestor de Projeto/Developer	Estagiário	Alto	Alto
António Damasceno	Product Owner/Orientador	CEO, Acting	Alto	Alto
Marta Pinto	Domain Expert	Desenvolvimento	Medio	Baixo
Pedro Marques	Domain Expert	Desenvolvimento	Alto	Baixo

Esta identificação foi realizada em colaboração com o orientador do estágio e validada pelo gestor de projeto que cruzou a atividade e envolvimento de cada *stakeholder* na arquitetura atual. Em relação ao projeto, e consequentemente à nova arquitetura, são identificados: o papel, a posição na empresa, o interesse e o poder sobre a nova arquitetura. Os dois elementos ainda não referidos no presente relatório são colaboradores que desenvolvem software e estão dependentes da arquitetura da linha *OneCare*. Marta Pinto (MPt) é responsável pelo desenvolvimento da aplicação móvel do *OneCare*. Pedro Marques (PM) faz parte da equipa de desenvolvimento da atual arquitetura e da solução web. Estes elementos são importantes para o projeto na medida em que conhecem bem a solução no que toca às suas limitações e necessidades.

No seguimento da identificação de *stakeholders*, foram preparados e realizados dois *workshops* de elicitação de requisitos, como é prescrito na fase 1 do *ACDM*. Na primeira reunião esteve presente apenas o MP e o AD, com o objetivo de apresentar a arquitetura atual e as necessidades de uma nova arquitetura. Foram apresentados os problemas com mais impacto na manutenção e testabilidade da solução e discutidos os valores, em horas, no tempo de execução de algumas tarefas. Desse *workshop* resultou um esboço de requisitos funcionais e atributos de qualidade. O segundo *workshop,* já contou com a presença de todos os *stakeholders*, permitiu identificar e priorizar os requisitos funcionais do sistema e, ainda, reavaliar os atributos de qualidade.

Segundo o processo de *ACDM* os requisitos funcionais são expressos em casos de uso, os atributos de qualidade em seis partes, e restrições negócio e técnicas em texto corrido. Como

resultado das referidas reuniões e cruzamento dos dados, tendo em conta os objetivos e âmbito do presente estágio, foram estabelecidos os drivers arquiteturais (conjunto de requisitos, atributos de qualidade e restrições) resumidos nas subsecções seguintes e evidenciado pelo anexo externo "Requisitos MSA".

3.1. Comportamentos

Antes de apresentar os requisitos do sistema, é necessário entender um conceito de que tudo o que é experienciado num sistema é identificado por comportamentos. Na Figura 5, é possível observar um esquema relativo a um conjunto de comportamentos a que uma solução ou sistema estão sujeitos.

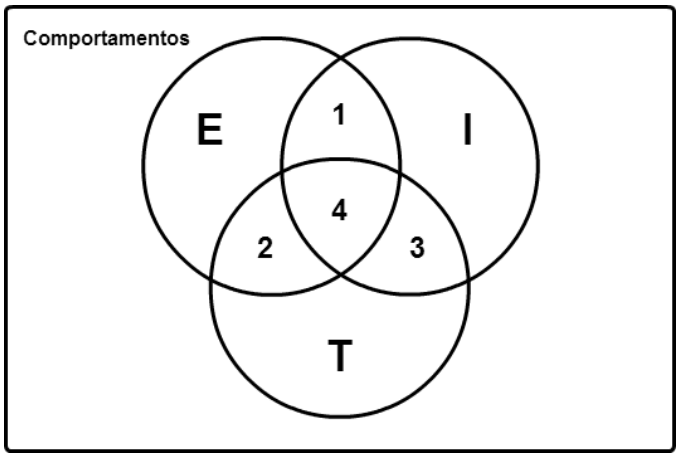


Figura 5 – Diagrama de Venn relativo a comportamentos[5]

Considere um universo de comportamentos como o da Figura 5. Dado uma especificação, um programa, e um conjunto de testes, identifica-se um conjunto de comportamentos especificados (E), um conjunto de comportamentos implementados (I) e um conjunto de comportamentos testados (T). Os comportamentos podem ser: especificados, comportamentos que se sabe que vão ocorrer e que se traduzem em drivers arquiteturais; implementados, comportamentos que por via da estruturação técnica se introduzem neste âmbito; e testados, comportamentos que por meio de testes são incluídos neste conjunto.

Na altura do desenvolvimento são identificados e implementados comportamentos que não foram especificados, ou que, como no exemplo anterior, foram considerados triviais ou generalistas. Para os comportamentos testados a dimensão do conjunto depende do tempo investido nos testes, nas ferramentas e na exigência que a equipa coloca na verificação da solução em termos de ambientes válidos, limites de valores, objetos inválidos, etc. Existem vários cenários em que certos testes corresponderem a comportamentos inesperados. Isto pode dever-se ao facto de a especificação ser insuficiente, ou o *tester* quer garantir que determinados comportamentos não acontecem (região T). À medida que o produto cresce, são identificados comportamentos não especificados que por meio de erros, implementações consideradas triviais ou derivados, se conseguem observar. (Jorgensen, 2008)

Assim, o objetivo é aumentar a interceção dos três tipos de comportamentos ao atualizar os requisitos das versões anteriores da plataforma, para documentar, quer os comportamentos esperados, quer os que, entretanto, foram observados.

Tendo sido identificado que a origem desta limitação, que provem do tipo de arquitetura em vigor, justifica-se a abordagem do desenvolvimento da solução com o foco para uma nova

[5] Fonte: (Jorgensen, 2008)

arquitetura iniciando o processo especificando os comportamentos genéricos que não foram especificados nas versões anteriores do produto ou da linha de produtos.

Dando um exemplo real, numa arquitetura monolítica a cada deploy da aplicação, é disponibilizado/gerado uma interface para o utilizador. Numa arquitetura de micro serviços a cada deploy de um serviço, este é registado, e a interface da solução necessita de ser atualizada. Com isto, na arquitetura atual da plataforma OneCare, a ação de *deploy* da solução no servidor não fazia parte dos comportamentos especificados e, de acordo com o problema do presente relatório, há a necessidade de especificar estes comportamentos pois deixa de ser uma ação trivial.

Os requisitos que foram identificados como novas necessidades para a solução são, na sua base, casos genéricos que podem não se aplicar só ao sistema analisado, OneCare, mas a qualquer plataforma ou aplicação. Estes requisitos pela sua abstração ou generalismos, não foram especificados nas versões anteriores do produto, tendo sido considerados garantidos pela plataforma escolhida para a implementação. Com a necessidade de incluir novas funcionalidades, e de realizar as operações triviais de uma forma diferente do considerado habitual, estes requisitos são agora considerados e avaliados com uma importância elevada.

3.2. Requisitos Funcionais

No que diz respeito a requisitos é possível observarmos a Figura 6, que representa o diagrama de casos de uso da nova arquitetura e que resultou dos workshops referidos anteriormente. Este diagrama está presente no documento de requisitos externo "Requisitos MSA". Tem como objetivo identificar os atores que interagem com o sistema e a forma como é feita essa interação, esclarecendo o propósito ou o papel do ator no sistema ou solução. É usada na elaboração do diagrama a convecção UML 2.0 (Rumbaugh, Jacobson & Booch, 2004).

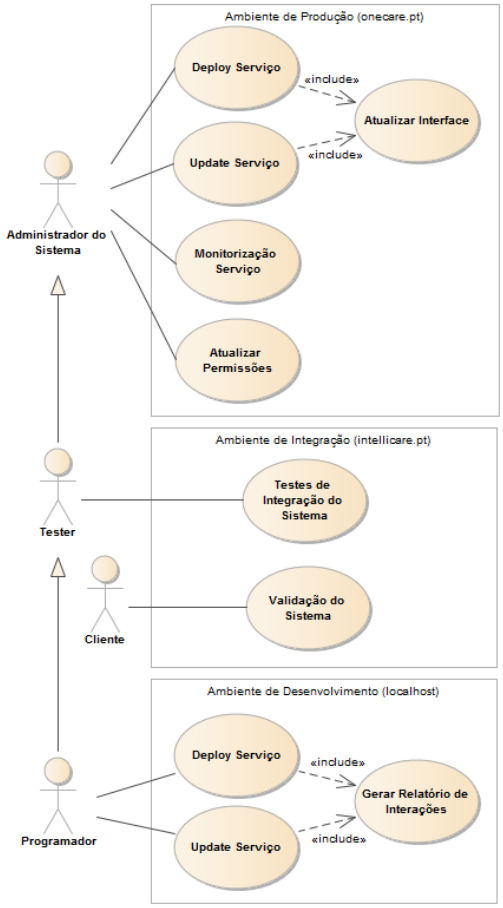


Figura 6 – Diagrama de casos de uso

A subsecções seguintes apresentam o trabalho realizado, na elicitação dos requisitos do sistema, pelo estagiário, onde procura apresentar os requisitos que a Intellicare tem para a solução que resolva o problema identificado no inicio do presente relatório.

3.2.1. Ambientes

Apesar de não ser muito comum, são identificados vários ambientes. Existem certos tipos de requisitos que variam consoante a fase de desenvolvimento que este se encontra e como consequência, o ambiente em que está inserido. A arquitetura tem então de suportar o sistema nos seguintes ambientes apresentados pela Tabela 4:

Tabela 4 – Descrição dos ambientes do sistema

ID – Titulo	Descrição
AM01 – Ambiente de Desenvolvimento	O ambiente de desenvolvimento refere-se ao ambiente ou sistema onde é produzida a solução. Neste caso, é possível haver vários ambientes com o mesmo propósito e características diferentes, visto que, o ambiente é inerente ao computador onde o programador está a desenvolver os novos serviços ou a realizar atualizações aos antigos.
AM02 – Ambiente de Integração	O ambiente de integração refere-se ao ambiente ou sistema por onde a solução passa depois dos seus serviços serem desenvolvidos e testados singularmente no ambiente de desenvolvimento.

	O sistema é alojado no cluster de testes da Intellicare (www.intellicare.pt). Têm acesso a este ambiente clientes específicos com o intuito de testar a integração entre serviços.
AM03 – Ambiente de Produção	O ambiente de produção refere-se ao ambiente ou sistema onde se instala a solução para que esta esteja disponível para toda a internet. A solução já tem de ter passado pelos restantes ambientes, onde esteve sujeito a testes, tendo sido validada e verificada. O sistema pode estar alojado no cluster de produção da Intellicare (www.onecare.pt) ou em clusters de clientes que adquiriram o serviço.

3.2.2. Atores

Os intervenientes na arquitetura denominam-se atores, existem diferentes atores a participar no sistema não só consoante o seu papel como normalmente é apresentado num documento de requisitos, mas também derivado da fase em que o próprio ator se encontra. Os atores estão representados na Tabela 5:

Tabela 5 – Descrição dos atores do sistema

ID – Titulo	Descrição
A01 – Administrador de Sistema	O administrador de sistema é um humano e é responsável pelo sistema e solução no ambiente de produção sendo, por isso, necessário um administrador de sistema por cada cluster em que a solução estiver instalada. Dentro do sistema o ator necessita de poder fazer o *deploy*, *update* e monitorização de serviços, assim como atualizar as permissões dos grupos de utilizadores face aos serviços disponíveis.
A02 – Tester	O *tester* herda do administrador de sistema a interação com o ambiente, mas, neste caso, utiliza o ambiente de integração, sendo assim, responsável pelo mesmo. Além de realizar o *deploy*, *update* e monitorização dos serviços, o ator necessita também de ferramentas de monitorização interna dos serviços para realizar os testes de integração entre serviços.
A03 – Cliente	O cliente é um utilizador, final ou não, que pertence a um grupo de utilizadores do produto. É responsável por usar e testar o produto na perspetiva de utilizador, simulando a sua utilização da forma mais real possível, com a intenção de validar as funcionalidades da solução.
A04 – Programador	O programador herda do tester a interação com o ambiente, mas neste caso utiliza o ambiente de desenvolvimento. Pode haver mais do que um programador, e cada um é responsável pelo seu próprio ambiente. Na herança de necessidades, existe uma pequena diferença no que toca a desenvolver e atualizar um serviço. Neste caso, para qualquer modificação (inserção, modificação ou remoção) de serviços no sistema deverá ser gerado um relatório de interação entre serviços, artefacto a ser utilizado pelo tester em ambiente de integração.

3.2.3. Casos de Uso

O papel dos atores num sistema é fazer o uso do mesmo, na Tabela 6 são apresentados os casos de uso com os requisitos que o sistema necessita que sejam implementados para responder ao problema.

Tabela 6 – Tabela representativa dos casos de uso da nova arquitetural

ID – Titulo	Descrição
UC01 – *Deploy* Serviço	O caso de uso surge da necessidade que o Administrador de sistema tem em adicionar um serviço a um nó do cluster com intuito de testar, validar ou disponibilizar para a internet o serviço. Permite que o administrador de sistema faça o deploy de um serviço, testado e validado, no sistema. O serviço fica disponível no servidor.
UC02 – Update do serviço	Permite que o administrador de sistema faça o *update* de um serviço, no sistema. Este caso de uso surge da necessidade de atualizar um serviço que sofreu alterações quer de correção, quer de inclusão de novas funcionalidades.
UC03 – Monitorização de Serviços	Permite que o administrador de sistema visualize, por meio de gráficos e/ou tabelas, o desempenho e o comportamento dos serviços. O caso de uso surge da necessidade que o administrador de sistema tem em monitorizar o comportamento dos serviços para despistar qualquer mau funcionamento e identificar situações onde existe a necessidade de escalar serviços.
UC04 – Atualizar *Interface*	O caso de uso surge da necessidade que o sistema tem em ser capaz de identificar os serviços disponíveis e disponibilizar os dados e funcionalidades dinamicamente. Permite que a interface seja atualizada consoante os serviços disponíveis na "rede de serviços", ou seja, na solução.
UC05 – Acede às funcionalidades permitidas	O caso de uso surge da necessidade que o Cliente tem em aceder às funcionalidades do portal, aliado à restrição que o Administrador de Sistema aplica aos diversos grupos de utilizadores consoante os planos contratados. Permite que o Cliente possa utilizar os serviços contratados como consultas e edição de elementos. Permite que o Administrador de Sistema restrinja a utilização da plataforma pelos diferentes utilizadores.
UC06 – Testes de integração do sistema	O caso de uso surge da necessidade que o sistema tem em fornecer ferramentas de controlo e monitorização de serviços a um nível de detalhe suficiente para que o A02 – Tester consiga perceber os eventuais erros aquando da realização de testes de integração. É espectável que para realizar esta tarefa seja usado o artefacto, resultante do caso de uso "Gerar Relatório de Integrações", fornecido pelo A04 – Programador. Permite que o Tester tenha um controlo mínimo das variáveis que são transportados entre serviços para a finalidade de testar a interação entre estes.
UC07 – Verificação do sistema	O caso de uso surge da necessidade que a Intellicare tem em verificar o sistema em condições normais de utilização de um cliente

	real. Tem como objetivo verificar a solução em termos de funcionalidades especificas e em geral na interação e integração entre elas. Permite a grupos de utilizadores testarem a aplicação de perspetivas diferentes. O sistema deve providenciar ferramentas de submissão de defeitos, assim como questionários periódicos de usabilidade da solução.
UC08 – Gerar relatório de interações	O caso de uso surge da necessidade de identificar a interação entre os serviços e de otimizar ao máximo os testes de integração de serviços. Tem como objetivo criar um fio condutor de informação entre serviços e ajudar o Tester na execução de testes para validação do sistema. O sistema deve possuir métodos de geração de relatórios aquando a introdução ou modificação de serviços no ambiente de desenvolvimento, nesse relatório devem ser descriminadas as interações entre serviços. Deve surgir a partir daqui um artefacto a ser utilizado pelo tester no ambiente de integração. Espera-se, então, que este relatório seja produzido de duas formas: uma de fácil leitura para um humano, grafos e tabelas, e outra para automatizar o processo dos testes de integração.
UC09 – Atualizar Permissões	O caso de uso surge da necessidade que o Administrador de Sistema tem em disponibilizar certas funcionalidades ou permissões, de acesso a serviços, a grupos de utilizadores específicos. Permite que a interface dos vários utilizadores seja limitada ás suas permissões através da configuração online por parte do Administrador de Sistema.

3.2.4. Legados

Para que seja possível entender facilmente a dimensão da solução do OneCare, na Tabela 7 são apresentados os casos de uso das versões anteriores do OneCare.

Tabela 7 – Tabela representativa dos casos de uso das versões anteriores do OneCare

ID – Titulo
UC01 – Autenticação
UC02 – Configurar equipamento de notificação de localização
UC03 – Configurar equipamento de saúde
UC04 – Configurar limites de alertas de saúde
UC05 – Configurar pessoas monitorizadas
UC06 – Configurar provas de vida
UC07 – Configurar utente
UC08 – Efetuar medição
UC09 – Escalar alertas

UC10 – Inserir alertas de falha de prova de vida
UC11 – Inserir alertas de saúde
UC12 – Obter dados do portal da saúde
UC13 – Processar notificações
UC14 – Publicar dados
UC15 – Receber dados
UC16 – Receber notificações
UC17 – Tratar alertas de dados de saúde
UC18 – Tratar alertas de notificações de localização
UC19 – Verificar medições de saúde
UC20 – Verificar prova de vida
UC21 – Visualizar dados da saúde
UC22 – Obter dados
UC23 – Enviar configuração parcial
UC24 – Configurar botões do equipamento
UC25 – Obter utentes
UC26 – Refrescar lista de pessoas monitorizadas
UC27 – Ativar conta
UC28 – Recuperar acesso
UC29 – Registar medição manual
UC30 – Registar utente
UC31 – Associar equipamento ao utente
UC32 – Configurar credenciais
UC33 – Adquirir Sinais Vitais
UC34 – Receber blocos de medições

3.3. Atributos de qualidade

Após a recolha e especificação dos casos de uso do sistema, é necessário atribuir-lhes parâmetros de qualidade. Os parâmetros de qualidade denominam-se atributos de qualidade e têm por objetivo definir valores ou métricas para a execução de uma determinada funcionalidade. Entenda-se como exemplo, um caso de uso "Um sistema tem de gerar informação sobre os

dados de saúde de um utente" tem o cenário de atributo de qualidade "O tempo que o sistema processa os dados de um utente é sempre inferior a 10 segundos".

Os atributos de qualidade servem para definir a qualidade com que os casos de uso irão ser traduzidos em funcionalidades do sistema. Com estes atributos procura-se: (1) responder aos problemas que originam a necessidade de uma solução; (2) que o sistema esteja de acordo com as restrições e necessidade do negócio; (3) que o sistema cumpra com requisitos legais, como exemplo a proteção de dados pessoais. Normalmente são atribuídas palavras chave a um atributo de qualidade como forma de categorização. Estas categorias indicam, através de uma palavra, o tipo de qualidade que definem. Por exemplo, segurança, performance, usabilidade, disponibilidade, são, entre outros, palavras únicas que definem a que se refere um atributo de qualidade.

Para a definição dos atributos de qualidade da arquitetura, foram escolhidos os casos de uso que mais preocupam os *stackholders*, nomeadamente os casos de uso que vão de encontro ao problema do estágio. Na subsecção seguinte é descrita a qualidade da solução na forma de "atributos de qualidade de seis partes" (Lattanze, 2008).

3.3.1. Descrição Resumida

A descrição de um atributo de qualidade de seis partes é composta por: estimulo, fonte de estimulo, condições do ambiente, elementos arquiteturais, resposta do sistema, e medida de resposta. na Tabela 8 são apresentadas as descrições resumidas dos atributos de qualidade da nova arquitetura resultantes das reuniões realizadas. Os detalhes destes atributos de qualidade em formato de atributos de qualidade de seis partes podem ser consultados no documento "Requisitos MSA" nas páginas 18-19.

Tabela 8 – Tabela descritiva dos atributos de qualidade da nova arquitetura

ID – Titulo	Descrição
AQ01 – Adicionar e testar um serviço ao sistema	O atributo de qualidade define o tempo útil necessário à realização de testes de um componente. Tem como objetivo reduzir o esforço aplicado a testes para verificação do sistema. Esta medida existe para prevenir que com a adição de novos componentes à aplicação, esta não necessite de ser sujeita a testes por inteiro.
AQ02 – Disponibilizar um serviço no sistema	O atributo de qualidade define o tempo útil necessário à disponibilização de uma fonte de dados ou funcionalidade ao cliente. Tem como objetivo reduzir o tempo de atualização do sistema aquando a disponibilização de uma nova fonte de dados ao utilizador. Esta medida existe para garantir que a adição de novos componentes ao sistema é feita através de um sistema de registo e descoberta, onde também a interface é atualizada dinamicamente não sendo necessário efetuar alterações.
AQ03 – Notificação de condições irregulares num serviço	O atributo de qualidade define a o tempo máximo entre um serviço estar sobrecarregado e a receção de aviso por parte do suporte. Tem como objetivo definir um período máximo em que uma máquina onde um serviço está alojado apresenta um valor de processamento ou de memória superior ao estipulado pelo administrador de sistemas. Esta medida existe para prevenir que as máquinas fiquem sobrecarregadas e não consigam processar os pedidos em tempo útil sem que o Suporte saiba do estado das mesmas.

AQ04 – Escalar serviços	O atributo de qualidade define a o tempo máximo que uma instancia de um serviço seja inserido e disponibilizado no sistema. Tem como objetivo estabelecer um tempo limite no que toca a introduzir e disponibilizar uma nova instancia de um serviço já a correr no sistema. Esta medida existe para prevenir que um serviço seja escalado em tempo útil e implica que os dados da nova instância sejam sincronizados antes da sua disponibilização.

3.3.2. Legados

Na Tabela 9 são apresentados os atributos de qualidade das versões anteriores do OneCare onde é possível entender a complexidade da solução. Estes atributos de qualidade legados são importantes para que na nova arquitetura se tenha em conta, quando relevante, as necessidades anteriormente identificadas e especificadas.

Tabela 9 – Identificação dos atributos de qualidade das versões anteriores do OneCare

ID – Titulo
AQ01 – A descoberta da password de um equipamento tem de se lenta e onerosa.
AQ02 – O sistema deve cumprir com a maioria dos dez parâmetros de heurísticas de usabilidade
AQ03 – Tempo para notificar operador em modo operacional normal
AQ04 – *Logout* por *timeout*
AQ05 – Aprendizagem simples da utilização da aplicação
AQ06 – Utilização de sistema de cores para sinalização de alertas
AQ07 – Funcionamento em falha de servidor aplicacional
AQ08 – Indicação do modo de funcionamento
AQ09 – Sempre que há operações deve ser feita sincronização
AQ10 – Aplicação disponível em modo offline
AQ11 – Sincronização obrigatória após alteração de estado para online
AQ12 – Velocidade de processamento de medições
AQ13 – Velocidade de sincronização de dados com o servidor
AQ14 – Funcionamento em falha de servidor
AQ15 – Indicação do modo de funcionamento
AQ16 – Encaminhamento da informação com o mínimo de atraso
AQ17 – Aplicação disponível em modo offline
AQ18 – Encaminhamento dos dados após disponibilização de ligação à Internet

3.4. Restrições

No conjunto dos drivers arquiteturais também se incluem as restrições que podem ser, de negócio, técnicas e, derivadas do negócio, as legais. Qualquer restrição influencia diretamente a arquitetura, sendo por isso um conjunto de decisões que são tomadas previamente, isto é, antes de haver um pensamento inicial, por parte do arquiteto, do desenho da arquitetura. Enquanto que um requisito pode ser flexível e negociado, uma restrição é algo fixo e, em certo modo, imutável.

As restrições técnicas são diretas, e restringem o sistema ao nível da tecnologia utilizada para construir o sistema. Define, muitas vezes, o hardware, software, sistema operativo, componentes legados, com os quais a solução tem de ser pensada. As restrições técnicas podem exercer força na arquitetura dentro do sistema ou podem definir como o sistema interage com os elementos externos. Já as restrições de negócio são indiretas e apesar de não definirem nada em concreto definem como é que os processos e métodos se vão executar. Estas restrições definem normalmente prazos, recursos, métodos, entre outros. (Lattanze, 2008)

3.4.1. Descrição Resumida

O processo de recolha de restrições foi realizado aquando a elicitação de requisitos e atributos de qualidade traduzindo-se nas tabelas seguintes. Na Tabela 10 são apresentadas as restrições de negócio, na Tabela 11 as restrições técnicas e na Tabela 12 as restrições legais do sistema.

Tabela 10 – Descrição das restrições de negócio da arquitetura

ID – Titulo	Descrição
RN01 – Data limite de desenvolvimento	O protótipo do novo serviço, com o desenvolvimento baseado na nova arquitetura, tem de estar pronto e em funcionamento até 15 de julho de 2016.
RN02 – Recursos Humanos disponíveis para o projeto	O projeto é realizado com uma pessoa a meio tempo entre 14 de setembro e 7 de fevereiro, e a tempo inteiro de 8 de fevereiro a 15 de julho.
RN03 – Autorização da CNPD para o tratamento de dados pessoais	O sistema tem de ser desenvolvido segundo as especificações e requisitos previsto na Lei 67/98 – Lei da Proteção de Dados Pessoais. Esta restrição existe porque o OneCare trata de dados pessoais sensíveis, os quais o código civil declara como tal.
RN04 – Certificação da solução para usar a Marcação CE	O sistema tem de ser desenvolvido segundo as especificações e requisitos da diretiva 93/42/CEE para que possa ser certificado e comercializado com a Marcação CE para dispositivos médicos.
RN05 – Certificação da qualidade do software para dispositivos médicos	O processo de desenvolvimento do sistema tem de cumprir a norma ISO 13485:2016 para que os serviços possam vir a ser certificados pela ISO (Organização Internacional para a Normalização) com o selo de qualidade correspondente para dispositivos médicos.

Tabela 11 – Descrição das restrições técnicas da arquitetura

ID – Titulo	Descrição
RT01 – Manter a tecnologia atual OneCare	A arquitetura alternativa é considerada um plug-in à solução atual do OneCare.

	O desenvolvimento da nova arquitetura tem de ter em conta que quando esta for aplicada, a linha de produtos OneCare tem de continuar a funcionar.

Tabela 12 – Descrição das restrições legais da arquitetura

ID – Titulo	Descrição
RL01 – Lei 67/98 artigo 15º alínea 1.a (Controlo de entrada nas instalações)	"Impedir o acesso de pessoa não autorizada às instalações utilizadas para o tratamento desses dados."
RL02 – Lei 67/98 artigo 15º alínea 1.b (Controlo dos suportes de dados)	"Impedir que suportes de dados possam ser lidos, copiados, alterados ou retirados por pessoa não autorizada."
RL03 – Lei 67/98 artigo 15º alínea 1.c (Controlo da inserção)	"Impedir a introdução não autorizada, bem como a tomada de conhecimento, a alteração ou a eliminação não autorizadas de dados pessoais inseridos."
RL04 – Lei 67/98 artigo 15º alínea 1.d (Controlo da utilização)	"Impedir que sistemas de tratamento automatizados de dados possam ser utilizados por pessoas não autorizadas através de instalações de transmissão de dados."
RL05 – Lei 67/98 artigo 15º alínea 1.e (Controlo de acesso)	"Garantir que as pessoas autorizadas só possam ter acesso aos dados abrangidos pela autorização."
RL06 – Lei 67/98 artigo 15º alínea 1.f (Controlo da transmissão)	"Garantir a verificação das entidades a quem possam ser transmitidos os dados pessoais através das instalações de transmissão de dados."
RL07 – Lei 67/98 artigo 15º alínea 1.g (Controlo da introdução)	"Garantir que possa verificar-se a posteriori, em prazo adequado à natureza do tratamento, a fixar na regulamentação aplicável a cada sector, quais os dados pessoais introduzidos quando e por quem."
RL08 – Lei 67/98 artigo 15º alínea 1.h (Controlo do transporte)	"Impedir que, na transmissão de dados pessoais, bem como no transporte do seu suporte, os dados possam ser lidos, copiados, alterados ou eliminados de forma não autorizada."
RL09 – Lei 67/98 artigo 15º alínea 2. (Separação de dados sensíveis)	"Os sistemas devem garantir a separação lógica entre os dados referentes à saúde e à vida sexual, incluindo os genéticos, dos restantes dados pessoais."
LR10 – Dispositivo médico	Segundo a norma MEDDEV 2.1/6 o software é considerado como um dispositivo médio.

LR11 – Classificação dispositivo médico	Segundo o Decreto-Lei 145/2009, anexo IX, Grupo III, ponto 5.2.3, o módulo de alarmes gerados através do processamento de dados de saúde recolhidos, é classificado como dispositivo médico de classe IIb sendo, por isso, necessária a sua certificação.

3.4.2. Dispositivos Médicos

As restrições RN04 e RN05 focam na certificação dos dispositivos médicos. Os dispositivos da Intellicare inserem-se no ramo da tele saúde. O conceito consiste na utilização de serviços de saúde usando tecnologias de telecomunicações. No caso do OneCare os dados dos dispositivos são recolhidos através de medidores de tensão arterial, glicémia, entre outros, e consultados por médicos ou enfermeiros.

Se estes mesmos dados forem de alguma forma tratados ou interpretados, o software é considerado um MD (LR10). Por exemplo, o OneCare *Sensing* onde é feita a recolha e consulta dos dados de tensão arterial não é considerado MD. Ao ser introduzido no sistema um algoritmo, ou qualquer outro método de interpretação ou processamento, sobre essa informação de forma a gerar alarmes quando for detetado que os valores estão fora do normal, a solução é considerada um MD.

A utilização de serviços independentes e isolados pode ser solução para garantir que os componentes, ao ser confirmada a sua independência, são certificados de forma singular. Pretende-se com isto certificar apenas alguns componentes e não a solução toda. A vantagem é que pode não ser possível, ou até relevante, certificar a solução toda. Por outro lado, a empresa conseguirá certificar pequenos enxertos do sistema, sob forma de serviços independentes, garantindo a entrada no mercado com produtos com qualidade certificada implicando um custo e esforço de desenvolvimento inferior, comparado com a solução anterior.

3.4.3. Legislação de Proteção de Dados

A Lei 67/98 impõe um conjunto de obrigações a que os sistemas que processam dados pessoais têm de responder. Considerando o facto de se tratar de dados de saúde, para alem das normas gerais, o sistema tem de cumprir os requisitos do artigo 15º, relativo a medidas especiais de proteção.

Os dados tratados pelo OneCare são considerados especialmente sensíveis estando sujeitos aos mecanismos especiais de proteção que com a inclusão nos requisitos das restrições RL01 a RL09, o desenho da arquitetura terá em conta os mecanismos obrigatórios no tratamento dos dados.

A empresa tem experiencia com a CNPD na medida em que já foram pedidos esclarecimentos detalhados sobre a metodologia utilizada para o tratamento de dados pessoais nas suas soluções. O objetivo de incluir e fazer referencia a estas preocupações é em poder responder a qualquer esclarecimento desde o inicio do desenvolvimento da aplicação.

3.4.4. Legados

À semelhança com os requisitos e os atributos de qualidade, na Tabela 13 são apresentadas as restrições de negócio das versões anteriores do OneCare. É importante ter em conta estas restrições para avaliar se devem acompanhar a nova solução ou não.

Tabela 13 – Tabela com as restrições de negócio das versões anteriores do OneCare

ID – Titulo	Descrição
RN01 – *Failover* no acesso ao serviço de receção de dados	A aplicação deverá suportar em regime de *failover* múltiplos serviços de receção de dados e um modo de atualizar a lista de servidores de receção de dados.
RN02 – O acesso ao serviço deverá ser limitado permitindo apenas acessos válidos	O acesso ao serviço deverá ser limitado permitindo apenas acessos realizados com credenciais válidas.
RN03 – Suporte de diversas línguas	As interfaces do utilizador (portal e aplicação Android) têm de suportar PT-PT, PT-BR e EN.
RN04 – Suporte de diversas timezones	O OneCare e o OneCareTablet deverão apresentar os dados na timezone do utilizador.

Na Tabela 14 são apresentadas as restrições técnicas das versões anteriores do OneCare.

Tabela 14 – Tabela com as restrições técnicas das versões anteriores do OneCare

ID – Título	Descrição resumida
RT01 – A aplicação será implementada com base nas tecnologias do OneCareMaisPerto e OneCareTablet	A aplicação será implementada com base nas tecnologias do OneCareMaisPerto e OneCareTablet. A linguagem principal de programação será o Java sendo utilizadas, dependendo do módulo, as seguintes tecnologias: Serviços: · Jboss 6.1 · Seam 2.2 · MySQL 5.5 Aplicação Android: · Android SDK
RT02 – O sistema deve usar equipamento Teltonika GH3000	A solução deve interagir com Teltonika equipamento GH3000, que é um telemóvel adaptado. Documentação pode ser encontrada em: Teltonika - Handheld GPS_GSM Tracker - GH3000.pdf GH3000 Data protocol v1.04.pdf
RT03 – O sistema tem de usar a gateway de SMS da Optimus SMS Pro	A operadora fornece o serviço de gateway de SMS, SMS Pro, sendo necessário interagir com o mesmo através dos webservices SOAP.
RT04 – O sistema deverá armazenar as configurações dos equipamentos em JSON	A estrutura da configuração dos equipamentos depende do tipo de equipamento e o seu armazenamento deverá ser realizado em JSON.

3.5. Priorização

A arquitetura existe para construir um sistema que satisfaça os requisitos, mas para um arquiteto nem todos os requisitos são igualmente importantes. Um ASR (*Architecturally Significant*

Requirement) é um requisito que tem efeito na arquitetura. Neste caso, como os atributos de qualidade já tiveram em conta os requisitos do sistema, os ASR's a ter em conta são os atributos de qualidade, assim como, as restrições.

Como existem ASR's em que o seu cumprimento não depende diretamente da arquitetura ou que não estão fortemente ligados ao negócio. É necessário priorizar os elementos em termos de: (1) importância: se representa um valor acrescido para o negócio ou missão pois, se a arquitetura o vai satisfazer este tem de trazer valor para os stakeholders; e (2) impacto: se entender que tenha profundo impacto na arquitetura em que o cumprimento daquele ASR muito provavelmente irá alterar o desenho da arquitetura se este não for incluído.

A priorização dos ASR's foi realizada durante uma reunião com os stakeholders onde foram apresentados o conjunto de drivers arquiteturais recolhidos no estágio e os legados, procedendo-se depois à sua discussão e finalmente a votação. Para tal, foram dados pontos a cada um dos participantes, no valor correspondente a 2/3 do numero de ASR's, que foram distribuídos de modo priorizar a importância que cada tem para o stakeholder. O processo repetiu-se para a determinação do impacto que o ASR terá na arquitetura.

Com os resultados foram ponderadas as pontuações e definidos três níveis, alto, médio e baixo. O resultado é apresentado na Tabela 15 onde se pode ver, o ID e titulo do ASR, a sua importância para o sistema e o impacto que terá na arquitetura.

Tabela 15 – Avalização dos ASR's

ID	ID – Titulo	Importância	Impacto
ASR01	AQ01 – Adicionar e testar um serviço ao sistema	Alta	Alto
ASR02	AQ02 – Disponibilizar um serviço no sistema	Alta	Alto
ASR03	AQ03 – Notificação de condições irregulares num serviço	Média	Médio
ASR04	AQ04 – Escalar serviços	Média	Alto
ASR05	AQ01(leg.) – A descoberta da password de um equipamento tem de se lenta e onerosa.	Alta	Baixo
ASR06	AQ02(leg.) – O sistema deve cumprir com a maioria dos dez parâmetros de heurísticas de usabilidade	Média	Médio
ASR07	AQ03(leg.) – Tempo para notificar operador em modo operacional normal	Alta	Baixo
ASR08	AQ04(leg.) – *Logout* por *timeout*	Alta	Baixo
ASR09	AQ05(leg.) – Aprendizagem simples da utilização da aplicação	Média	Baixo
ASR10	AQ06(leg.) – Utilização de sistema de cores para sinalização de alertas	Média	Baixo
ASR11	AQ07(leg.) – Funcionamento em falha de servidor aplicacional	Média	Médio

ASR12	AQ08(leg.) – Indicação do modo de funcionamento	Média	Baixo
ASR13	AQ09(leg.) – Sempre que há operações deve ser feita sincronização	Média	Baixo
ASR14	AQ10(leg.) – Aplicação disponível em modo offline	Média	Baixo
ASR15	AQ11(leg.) – Sincronização obrigatória após alteração de estado para online	Média	Baixo
ASR16	AQ12(leg.) – Velocidade de processamento de medições	Média	Baixo
ASR17	AQ13(leg.) – Velocidade de sincronização de dados com o servidor	Média	Média
ASR18	AQ14(leg.) – Funcionamento em falha de servidor	Média	Baixo
ASR19	AQ15(leg.) – Indicação do modo de funcionamento	Baixa	Baixo
ASR20	AQ16(leg.) – Encaminhamento da informação com o mínimo de atraso	Média	Médio
ASR21	AQ17(leg.) – Aplicação disponível em modo offline	Média	Baixo
ASR22	AQ18(leg.) – Encaminhamento dos dados após disponibilização de ligação à Internet	Média	Médio
ASR23	RN01 – Data limite de desenvolvimento	Alta	Médio
ASR24	RN02 – Recursos Humanos disponíveis para o projeto	Alta	Médio
ASR25	RN03 – Autorização da CNPD para o tratamento de dados pessoais	Alta	Alto
ASR26	RN04 – Certificação da solução para usar a Marcação CE	Alta	Alto
ASR27	RN05 – Certificação da qualidade do software para dispositivos médicos	Alta	Alto
ASR28	RT01 – Manter a tecnologia atual OneCare	Alta	Alto
ASR29	RL01 – Lei 67/98 artigo 15º alínea 1.a (Controlo de entrada nas instalações)	Alta	Baixo
ASR30	RL02 – Lei 67/98 artigo 15º alínea 1.b (Controlo dos suportes de dados)	Alta	Baixo

ASR31	RL03 – Lei 67/98 artigo 15º alínea 1.c (Controlo da inserção)	Alta	Alto
ASR32	RL04 – Lei 67/98 artigo 15º alínea 1.d (Controlo da utilização)	Alta	Baixo
ASR33	RL05 – Lei 67/98 artigo 15º alínea 1.e (Controlo de acesso)	Alta	Alto
ASR34	RL06 – Lei 67/98 artigo 15º alínea 1.f (Controlo da transmissão)	Alta	Baixo
ASR35	RL07 – Lei 67/98 artigo 15º alínea 1.g (Controlo da introdução)	Alta	Médio
ASR36	RL08 – Lei 67/98 artigo 15º alínea 1.h (Controlo do transporte)	Alta	Baixo
ASR37	RL09 – Lei 67/98 artigo 15º alínea 3. (Separação de dados sensíveis)	Alta	Alto
ASR38	RL10 – Dispositivo médico	Alta	Alto
ASR39	RL11 – Classificação dispositivo médico	Alta	Alto
ASR40	RN01(leg.) – *Failover* no acesso ao serviço de receção de dados	Média	Médio
ASR41	RN02(leg.) – O acesso ao serviço deverá ser limitado permitindo apenas acessos válidos	Alta	Médio
ASR42	RN03(leg.) – Suporte de diversas línguas	Média	Baixo
ASR43	RN04(leg.) – Suporte de diversas timezones	Média	Baixo
ASR44	RT01(leg.) – A aplicação será implementada com base nas tecnologias do OneCareMaisPerto e OneCareTablet	Baixa	Baixo
ASR45	RT02(leg.) – O sistema deve usar equipamento Teltonika GH3000	Baixa	Baixo
ASR46	RT03(leg.) – O sistema tem de usar a gateway de SMS da Optimus SMS Pro	Média	Médio
ASR47	RT04(leg.) – O sistema deverá armazenar as configurações dos equipamentos em JSON	Baixa	Baixo

3.6. Árvore de Utilidade

O resultado da priorização dos ASR's, é um vasto leque de elementos onde alguns são descartados por falta de recursos ou por não terem a relevância que justifique a sua avaliação. É então necessário definir um critério de limite que leve o arquiteto a restringir o numero de ASR's. Este limite encontra-se classificando o ASR's segundo a sua importância e impacto. O

método utilizado foi atribuir uma pontuação ao ASR foi multiplicar a importância e o impacto associado, segundo a Tabela 16, e com o resultado atribuir a relevância.

Tabela 16 – Atribuição da relevância em função do impacto e importância

Importância ↓	**Pontuação – Relevância**		
Alta – 3	3	6	9
Média – 2	2	4	6
Baixa – 1	1	2	3
Impacto →	Baixo – 1	Médio – 2	Alto – 3

Para a escolha dos ASR's com mais relevância definiu-se que teriam de ter uma pontuação igual ou superior a 6. Assim, é garantido que os ASR's com importância alta e impacto pelo menos médio, e vice-versa, irão fazer parte, e são objetos de avaliação, da arquitetura.

De seguida atribuiu-se uma categoria ao ASR e um tema dentro dessa categoria, e foram ordenados num formato de arvore chamada *Utility Tree* (árvore de utilidade) apresentada na Tabela 17. É a partir desta arvore que os elementos da arquitetura se vão construindo.

Tabela 17 – Utility Tree

	Atributo de Qualidade	**Tema**	**ID (Impa., Impo)**
Utilidade	Manutenção	O tempo de testes realizados na adição de um serviço ao sistema não é superior a 8 horas.	ASR01 (A, A)
		O tempo de disponibilização de um serviço no sistema não é superior a 2 horas.	ASR02 (A, A)
		O sistema legado do OneCare tem de ficar em funcionamento.	ASR28 (A, A)
	Escalabilidade	O tempo de disponibilização de uma nova instância de um serviço no sistema não é superior a 8 horas.	ASR04 (M, A)
	Segurança	O sistema tem de estar em conformidade com a Legislação da Comissão de Proteção de Dados.	ASR25 (A, A)
		O modelo de dados tem de estar em conformidade com a Legislação da Comissão de Proteção de Dados no que toca à separação de dados sensíveis.	ASR37 (A, A)
		O sistema tem de estar em conformidade com a Legislação da Comissão de Proteção de Dados no que toca ao acesso limitado aos dados a pessoas autenticadas.	ASR41 (A, M)

Capítulo 4.
Arquitetura

A arquitetura de um sistema de software é um meio de transição entre os requisitos e a sua implementação. Serve para pensar no esquema da solução antes de investir esforço e partir para a programação do sistema. A implementação exige custos de elaboração e por vezes o esforço nas alterações é elevado.

Existem diferentes níveis de arquitetura dependendo da complexidade do sistema se que pretende desenvolver. Uma prática frequente é nem sequer realizar documentação, ou ficar pelos rabiscos pouco certos que apenas ajudaram no primeiro pensamento, e canalizar esse tempo diretamente para a implementação. Esta abordagem normalmente garante um protótipo funcional mais cedo, mas, dependendo da natureza e complexidade do problema, pode não significar que o produto final siga a mesma tendência.

Quando é tomada a decisão de desenvolver o produto com pouca definição à priori, pode dar-se o caso de que a solução não cumpra com os requisitos pretendidos, não em termos de funcionalidade, mas em termos de atributos de qualidade. A alteração de uma estrutura de código ou dependências técnicas para cumprir os atributos de qualidade e restrições técnicas resulta num investimento adicional ao desenvolvimento da solução. Com a elaboração de uma arquitetura, identificam-se problemas e tomam-se decisões atempadamente reduzindo o risco de haver necessidade de efetuar alterações com impacto elevado no desenvolvimento.

Para elaborar uma arquitetura o método mais acessível é recorrer aos padrões de desenho já utilizados por outras empresas que tenham tido os mesmos problemas e desafios, muitas vezes devidamente documentados e analisados em artigos. Não existe uma solução que respeite todos os requisitos, portanto devem ser articulados diferentes padrões, ponderadas várias soluções reais e balanceado o custo/beneficio das decisões tomadas.

4.1. Análise

Com vista no estudo dos padrões de arquitetura afetos ao problema, a arquitetura monolítica do OneCare, a arquitetura micro serviços alvo, e a mais próxima desta ultima orientada a serviços, devem ser analisadas. É feita então uma análise às referidas arquiteturas no sentido de avaliar as vantagens e desvantagens dos padrões. De acordo com os parâmetros de avaliação definidos em 2.5 a literatura indica que uma arquitetura baseada em Micro serviço seria classificada como alta em todos os parâmetros o que será demonstrado na análise em 0.

4.1.1. Arquitetura Monolítica

A estrutura monolítica caracteriza-se por ser uma arquitetura de suporte a aplicações maioritariamente do lado do servidor. Esta suporta os mais variados tipos de clientes, móveis ou fixos e pode, também, estar exposta a integração de aplicações terceiras (Richardson, 2014). É, ainda assim, um tipo de arquitetura leve. Em alguns casos esta revela-se o melhor padrão de arquitetura para a implementação de uma solução, devido à sua simplicidade. Por exemplo, nos primeiros passos de um projeto baseado numa arquitetura monolítica é mais fácil de realizar o *deploy*. Também quando uma aplicação é pequena a sua utilização facilita na inclusão de

novos colaboradores, permitindo-lhes mais rapidamente perceber o código e também começar a produzir (Namiot & Sneps-Sneppe, 2014).

De um ponto de vista técnico, uma aplicação monolítica é mais simples de produzir, entender e gerir por não ser necessário lidar com métodos remotos e sistemas distribuídos. Em termos de escalabilidade apenas é necessário fazer uma cópia da aplicação e colocá-la noutro sistema (Namiot & Sneps-Sneppe, 2014). É também fácil de ser testada dependendo da dimensão da aplicação e de haver ou não integração com outras aplicações.

As fragilidades da arquitetura aparecem com o decorrer do tempo, devido ao aumento da dimensão da solução. Em ambiente de desenvolvimento, o longo tempo de espera que é necessário para iniciar a aplicação devido à sua dimensão, em termos de espaço que a lógica e gestão de dados ocupam, num projeto é entediante. Resultando em tempo não produtivo e desmotivação por parte dos desenvolvedores. Outro problema está na dificuldade de novos membros da equipa perceberem o extenso código ser elevada e consequentemente, a possibilidade de aumentar o número de colaboradores para lidar com novas funcionalidades torna-se praticamente nula (Namiot & Sneps-Sneppe, 2014).

Em termos de manutenção das aplicações, as novas funcionalidades resultam num alargamento do tamanho da aplicação e, consequentemente, do tempo que leva a fazer o *deploy* da solução. Quando existe a necessidade de corrigir um pequeno bug toda a solução tem de ser testada e exportada novamente, o que resulta num processo bastante dispendioso (Namiot & Sneps-Sneppe, 2014). Quando existe uma nova funcionalidade na solução, muitas das vezes, testar o sistema inteiro é uma perda de tempo, porque se apenas alguns componentes do sistema forem modificados e não houver qualquer interligação com outros, estes não necessitam de ser testados. Claro que, esta situação não se aplica a todos os sistemas porque depende da forma como a atualização é realizada e da criticidade do produto.

A fiabilidade da solução é posta em causa pelo problema da propagação de erros, isto é, se existir um problema de memória na máquina onde o software está instalado toda a solução fica comprometida, uma vez que o sistema é um só, pois o erro acaba por afetar todos os componentes (Richardson, 2014).

Uma solução com uma arquitetura monolítica é fácil de escalar, mas tem normalmente custos acima dos ideais. Se for necessário escalar a aplicação devido a um dos componentes estar sobrecarregado, todos os outros componentes têm de ser duplicados, incluindo componentes não sobrecarregados ou que a utilização não justifica sua replicação. Além disso, se o sistema estiver alojado numa máquina dispendiosa devido às suas características por causa da camada lógica (em termos de processador) e da gestão de dados (em termos de armazenamento), escalar significa a aquisição uma nova máquina igual ou melhor. O desejável seria conseguir escalar os componentes para máquinas específicas para o trabalho que realizam, quer este seja de lógica ou de processamento de dados (Namiot & Sneps-Sneppe, 2014).

Hoje em dia, assistimos ao "lançamento" de novas linguagens ou tecnologias muito frequentemente, sendo que cada uma dessas novidades surge da necessidade de resolver problemas específicos. É bastante provável que as novas tecnologias possam resolver um problema que uma solução enfrentou e que foi resolvido de uma maneira pouco eficaz com os recursos disponíveis na altura (Richardson, 2014). O ideal é refazer o componente, ou parte dele, utilizando a nova tecnologia, mas devido à arquitetura estamos presos à linguagem e às tecnologias que são adotadas no inicio do projeto.

4.1.2. Arquitetura Micro Serviços (AMS)

Não existe, ainda, uma definição consensual, no que toca ao conceito da Arquitetura de Micro Serviços. No entanto, alguns autores têm contribuído para o desenvolvimento do mesmo, como é o caso de Martin Fowler e James Lewis que definem AMS como sendo uma abordagem para o desenvolvimento de uma aplicação singular como um conjunto de pequenos serviços, cada um a executar no seu próprio processo e a comunicar através de mecanismos leves, muitas vezes por API's através de HTTP. Estes serviços são construídos à luz de capacidades de negócio e instalados de forma automática e independente. O controlo centralizado destes serviços é mínimo, e podem ser desenvolvidos em diferentes linguagens e fazer uso de diferentes tecnologias. (Lewis & Fowler, 2014).

Uma arquitetura baseada em micro serviços tem como objetivo reduzir a complexidade nos grandes sistemas, escalando as aplicações pelo eixo-Y do Cubo de Escala (Abbott & Fisher, 2009), o que significa escalar aplicações aplicando uma decomposição funcional. Este particionamento possibilita que diferentes componentes ou módulos sejam tratados independentemente.

Uma vez decompostos, os componentes, agora sistemas mais "pequenos" são chamados de micro serviços, reduzindo assim a sua complexidade. Normalmente esta divisão é feita por grupos de funcionalidades, por exemplo, Utilizadores e Faturas podem ser dois serviços a funcionar independentemente, mas que se usam mutuamente (Richardson, 2014) (Namiot & Sneps-Sneppe, 2014).

Esta característica da AMS traz vantagens sobre a arquitetura monolítica. Uma vez que os serviços são pequenos os novos membros ou equipas não necessitam de perceber o sistema como um todo, podendo restringir-se a cada serviço (Newman, 2015). Já as aplicações acabam por demorar menos tempo a iniciar no ambiente de desenvolvimento. Como resultado obtemos um aumento de produtividade e reduzimos a possível desmotivação inicial de novos elementos.

Uma vez que os serviços são independentes, consegue-se fazer o *deploy* dos serviços separadamente trazendo grandes vantagens no que toca à escalabilidade do sistema. Os novos serviços podem ser postos em ambiente de produção mais frequentemente e ao mesmo ritmo que são desenvolvidos e testados, resultando no crescimento contínuo da solução e tornando-a capaz de responder às necessidades do mercado (Newman, 2015).

A AMS traz, ainda, a possibilidade de aumentar o lucro, baixando os custos. Como os serviços podem ser exportados em máquinas diferentes, encontra-se a possibilidade de preparar ambientes que melhor satisfaçam as necessidades do serviço. Quer isto dizer que, os serviços que estiverem sobrecarregados de pedidos podem ser escalados para máquinas com processamento dedicado (Richardson, 2014). Já os serviços responsáveis por alocar grandes quantidades de informação poderão ser colocados numa *data storage*. Isto não só aumentaria a produtividade dos serviços como reduziria os custos, na medida em que deixa de ser necessário escalar componentes que sejam utilizados uma vez por ano, por exemplo.

Outra vantagem resultante da independência dos serviços é estes poderem ser desenvolvidos em linguagens diferentes, dando liberdade à equipa de escolher qual a linguagem ou tecnologia onde se encaixa melhor, contribuindo para o aumento do seu rendimento em termos de velocidade de produção e, consequentemente, motivação (Lewis & Fowler, 2014). Relativamente ao lançamento de novas linguagens e tecnologias, anteriormente referido, uma vez que os serviços são relativamente pequenos, a inércia à migração de linguagem ou tecnologia é consideravelmente reduzida em relação a uma solução monolítica.

Um outro aspeto que também pode ser escolhido, tendo em conta o objetivo, é o modelo de dados. Se cada serviço necessitar de tipos de bases de dados diferentes, o modelo de dados a usar pode ser específico para satisfazer as necessidades (Newman, 2015). Um serviço não precisa de ficar preso a uma gigantesca base de dados relacional tipicamente vista numa solução monolítica, em vez disso, os dados podem ser armazenados onde se adaptarem melhor, quer seja relacional, não relacional, grafos ou documentos.

Na Engenharia de Software sabe-se que não existem soluções perfeitas e a arquitetura de micro serviços tem, também, as suas contrapartidas. Primeiro que tudo, a tarefa de decomposição de serviços não é fácil, ou seja, identificar as diferentes componentes de um projeto que se iram tornar serviços requer uma análise bastante cuidada na definição dos limites do serviço. O ganho derivado dessa definição de limite apenas vai ser notado com crescimento da solução na altura em que as equipas se separam. É preciso ter em mente que as más decisões vão arrastar-se ao longo do projeto e só numa fase mais avançada é que a independência dos serviços é posta à prova. Estes limites também são definidos em arquiteturas monolíticas, mas nesse caso é muito mais fácil contornar essas barreias, pois, a solução é um grande bloco unido e o perigo consegue ser mais facilmente atenuado (Lewis & Fowler, 2014).

Ainda relativamente à interdependência de sistemas, qualquer que seja a sua dimensão, estes vão comunicar entre si como um sistema distribuído (Namiot & Sneps-Sneppe, 2014). Como, na realidade, isto é apenas uma solução, pode ser complicado coordenar essa interação a um nível quase impercetível. É também difícil lidar com estes sistemas devido à sua barreira linguística. Uma vez que a arquitetura permite que os serviços possam ser desenvolvidos em linguagens diferentes é necessário assegurar que os módulos comunicam entre si numa linguagem comum.

Em relação à persistência ou modelo de dados, como cada serviço tem a sua própria base de dados são garantidas, pelo triângulo CAP (consistência, disponibilidade (*availability*) e particionamento), a disponibilidade e o particionamento (Newman, 2015). E, como não conseguimos encontrar um mecanismo que garanta os três atributos de qualidade, não é garantida uma consistência forte da informação. Ao desenvolver sistemas distribuídos de larga escala obriga a que haja um equilíbrio entre consistência e disponibilidade, este equilíbrio deve ser identificado e garantido pelo programador pois a consistência fraca garante a informação seja replicada ou disponível num espaço de tempo certo. Este período entre a inconsistência e a atualização dos dados, escritos/alterados por um processo, para todos os clientes é chamada janela de inconsistência (Vogels, 2009). Todas as atualizações consideradas essenciais, estão protegidas com *two phase commit*.

No pós-desenvolvimento de uma solução, baseada numa arquitetura micro serviços, uma das dificuldades passa pelos mecanismos de teste numa perspetiva de interação entre serviços (Newman, 2014). Uma vez que não existe controlo explícito sobre o comportamento dos componentes entre si, é necessário investir esforço no desenvolvimento ou na configuração de mecanismos de monitorização, aquando a realização dos testes de integração (Lewis & Fowler, 2014). No entanto, e sendo prioridade testar os componentes individualmente e as interações diretas que o serviço é sujeito, a anulação do *overhead* relativo à realização de baterias de testes completos a uma solução de grandes dimensões compensa o investimento.

Existe a necessidade de criação de um serviço de descoberta que conhece todos os serviços disponíveis e consegue indicar a qualquer um dos outros a localização do serviço desejado. Este serviço é bastante importante e faz parte do padrão da arquitetura micro serviços, como tem conhecimento dos serviços disponíveis consegue balancear a carga e o trafego entre serviços e pode, por isso, ficar responsável por alertar os administradores do sistema quando ocorrerem anomalias com os serviços como, por exemplo, um serviço que fique indisponível

(Richardson, 2014). Como vimos anteriormente, o *deployment* dos serviços pode ser feito em diferentes máquinas e isto significa que o par *host/port* pode variar frequentemente, devido a *Deny of Service*, escalabilidade dinâmica, etc. Escrever no código todas as possibilidades de localizações de serviços não só não é a melhor solução de desenvolvimento como na eventualidade de serem usadas novas infraestruturas esses endereços vão ser todos diferentes. Por exemplo, se for necessário atualizar à mão endereços de 200 serviços vai custar tempo a alterar o código, a testar e a realizar o *deployment* de cada um dos componentes afetados.

Como a independência de serviços não é trivial, para realizar atualizações quando estas envolvem a comunicação com diferentes serviços é necessário que a transformação seja pensada e executada por fases, nunca desprezando a dependência que existe entre eles (Alagarasan, 2015). Uma solução para melhorar este processo é a realização de alterações, numa perspetiva *bottom-up,* preparando os serviços de *back*-end e atualizando gradualmente os serviços até chegar aos componentes responsáveis pela vista da aplicação. É, contudo, importante que as *API's* antigas sejam apenas retiradas quando o "topo" do sistema já estiver atualizado (Preston-Werner, s.d.). Esta mudança gradual visa garantir que a atualização não provoca *downtime* à aplicação em geral e que este seja mínimo nos diferentes.

4.1.3. AMS vs Arquitetura Orientada a Serviços

Existe um número de profissionais que identificam a AMS sendo apenas uma nova terminologia utilizada para descrever a Arquitetura Orientada a Serviços (AOS). Para aqueles que têm usado a AOS como micro serviços isto acaba por ser verdade. Os Micro serviços são uma parte de AOS, também chamada "AOS usado de forma correta" ou ainda "granularidade fina da AOS". Mas, existem algumas diferenças que a arquitetura orientada a serviços tem como base e que não são usadas pela arquitetura micro serviços (Lewis & Fowler, 2014).

Uma aplicação desenvolvida baseada em AOS é, na verdade, uma grande componente singular com fluxos e processos bem definidos que usa serviços externos para alcançar os objetivos de uma funcionalidade. Isto é, executa um fluxo de eventos alterando o estado do programa como um *pipeline* de processos, passo a passo, até alcançar o objetivo. Com isto a arquitetura foca-se na reutilização de software (Rogers, 2005).

A base de comunicação de uma arquitetura orientada a serviços é o *Enterprise Service Bus* (ESB), responsável pela transação de elementos entre a camada lógica e o modelo de dados. O ESB é uma camada de abstração no topo da implementação de um sistema de mensagens, não sendo focado na lógica de negócio, mas sim na infraestrutura. Este tem como principal objetivo a abstração dos métodos e da informação dos serviços permitindo a comunicação semi-transparente entre eles. A dependência criada por este ESB faz com que seja necessário refazer mais um elemento, neste caso bastante complexo, sempre que é feita uma criação ou atualização num dos serviços ou fluxos. Por outro lado, a arquitetura micro serviços faz uso de mecanismos mais rápidos de comunicação como filas de mensagens, uma vez que estamos a retirar a camada de lógica inerente a um ESB produzindo, assim, um sistema de comunicação mais leve (Namiot & Sneps-Sneppe, 2014).

Em termos da gestão de dados, em AOS é necessário recorrer a serviços próprios de tratamento de dados ou mesmo ao ESB para realizar transações de escrita ou de leitura. Em micro serviços, por outro lado, como cada serviço tem a sua própria base de dados, este tem independência e isolamento suficiente para processar os dados (Newman, 2015).

4.2. Notação

Na secção 4.3 é demonstrada e descrita a proposta de arquitetura para o OneCare. Como os diagramas podem não ser percetíveis a todos os leitores, é necessário apresentar a notação utilizada na especificação da arquitetura.

As vistas seguintes seguem a especificação UML 2. Aqui são descritos os conceitos básicos da notação de modo a que as especificações que se seguem sejam claras. Em todos os documentos de arquitetura, a especificação é constituída por duas partes, o diagrama e a descrição escrita. Estes componentes devem existir sempre em conjunto.

4.2.1. Alocação

A vista de "Alocação", na Figura 7, apresenta a notação utilizada quando se pretende especificar os componentes físicos do sistema e o modo como comunicam entre eles. Os componentes de alocação são elementos que têm presença física, por exemplo, servidores, routers, etc. Os conetores representam os mecanismos de interação entre os componentes físicos.

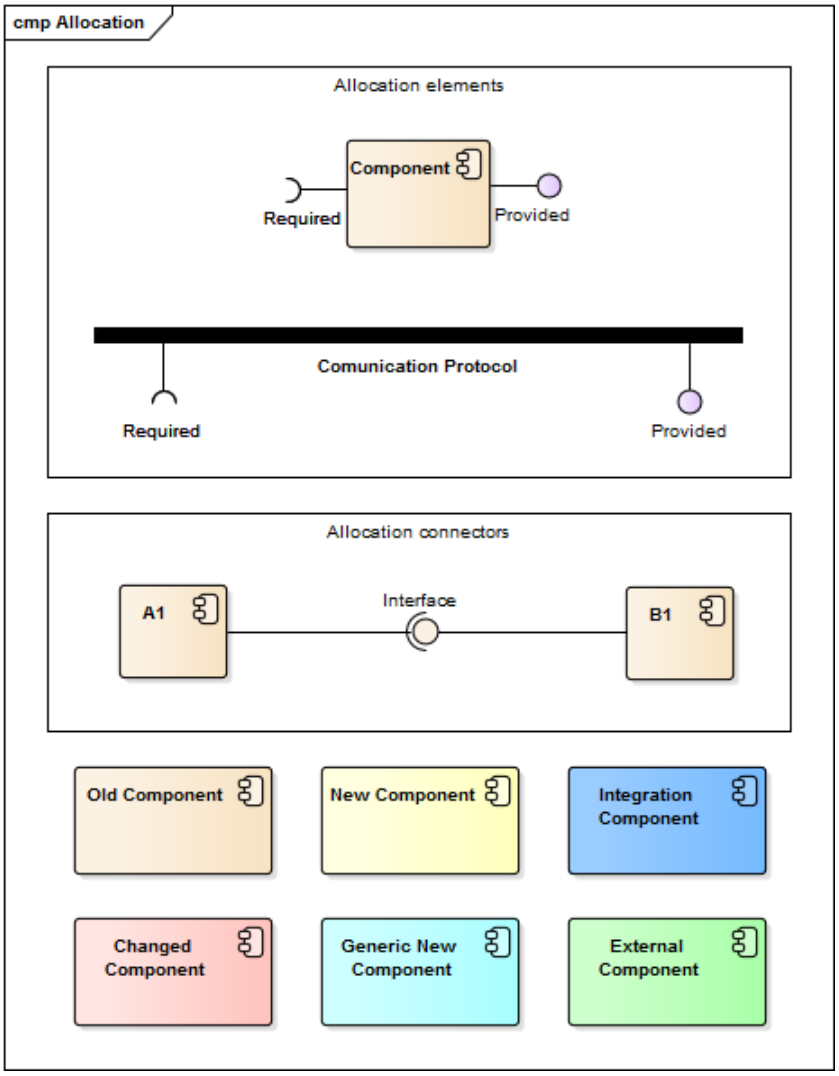


Figura 7 – Notação da vista de alocação

Como em todos os documentos de arquitetura, a especificação é feita em duas partes, o diagrama e a prosa. A Tabela 18 apresenta a descrição dos componentes presentes na Figura 7.

Tabela 18 – Descrição da notação da vista de alocação

Elementos	
Component	Um elemento físico, tratado como um componente.
Comunication Protocol	Um elemento ou conjunto de elementos correspondendo a um bus de mensagens transferidas utilizando protocolos standards.
A1, B1	Instancias de componentes físicos.
Old Component	Componente proveniente da arquitetura anterior e não são realizadas alterações.
New Component	Componente novo da arquitetura. É introduzido um novo componente físico na infraestrutura e/ou rede.
Changed Component	Componente antigo na arquitetura que necessita de alterações com a inclusão de uma nova solução.
External Component	Componente externo à arquitetura que acede ou é acedido pela mesma.
Conetores	
A1 -co- B1	O componente A1 liga-se à interface "*Interface*" exposta pelo componente B1.
Component -o	Interface providenciada e exposta pelo componente *Component*.
Component -c	Requisição de interface pelo componente *Component*.

4.2.2. Componentes e conetores

A vista de "Componentes e Conetores", na Figura 8, apresenta a notação utilizada quando se pretende especificar os componentes ativos do sistema e o modo como comunicam entre eles. Os componentes são elementos que têm presença na execução das funcionalidades do sistema, por exemplo, processos, serviços, aplicações, etc. Os conetores representam os mecanismos de interação entre os componentes.

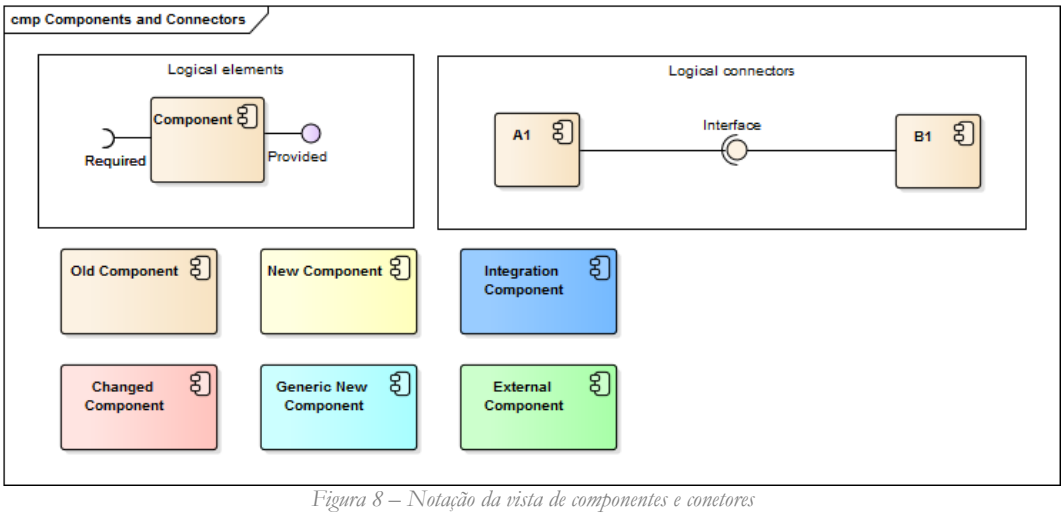


Figura 8 – Notação da vista de componentes e conetores

A Tabela 19 apresenta a descrição dos componentes presentes na Figura 8.

Tabela 19 – Descrição da notação da vista de componentes e conetores

Elementos	
Component	Componente ativo do sistema.
A1, B1	Instancias de componentes ativos.
Old Component	Componente proveniente da arquitetura anterior e não são realizadas alterações.
New Component	Componente novo da arquitetura. São implementadas funcionalidades do novo sistema.
Integration Component	Componente novo na arquitetura. São implementadas funcionalidades que dependem da arquitetura anterior. As funcionalidades ficam a funcionar no antigo e no novo sistema.
Changed Component	Componente antigo na arquitetura que necessita de alterações com a implementação de um novo sistema.
Generic New Component	Componente novo na arquitetura. Existem vários componentes deste género para as diferentes funcionalidades de gestão ou análise de parâmetros vitais ou de outro modelo de dados ou negócio.
External Component	Componente externo à arquitetura que usa ou é usado pela mesma.
Conetores	
A1 – B1	O componente A1 invoca a interface "*Interface*" exposta pelo componente B1.
Component -o	Interface providenciada e exposta pelo componente A1.
Component -c	Requisição de interface pelo componente A1.

4.2.3. Entidade e Relação (ER)

A vista de "ER", na Figura 9, é apresentada a notação utilizada quando se pretende especificar o modelo de dados do sistema e a relação entre eles. Tabelas, pacotes e interfaces representam estruturas de dados relacionais. Os conectores representam o mecanismo de interação relacional entre tabelas.

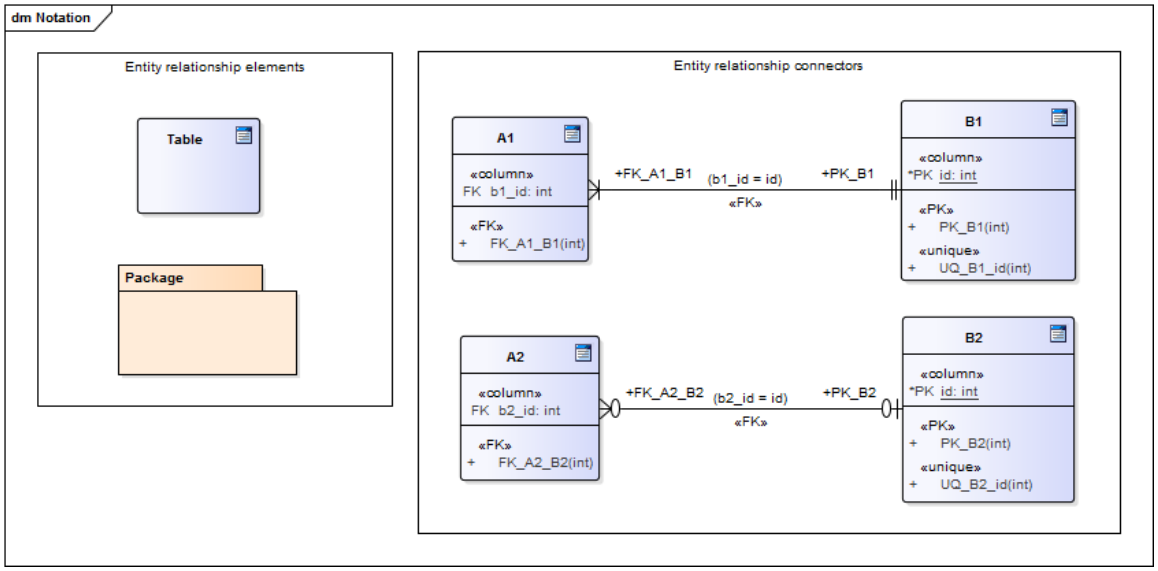


Figura 9 – Notação da vista de ER

A Tabela 20 apresenta a descrição dos componentes presentes na Figura 9.

Tabela 20 – Descrição da notação da vista de ER

Elementos	
Table	Um elemento da tabela ER.
Package	Um pacote de elementos.
A1, A2, B1, B2	Instancias de tabelas ER.
Conetores	
A1 – B1	Uma e uma só instancia de B1 é associada a pelo menos uma instancia de A1. A relação é efetuada pela chave primária de B1 (PK_B1) e a chave forasteira de A1 (FK_A1_B1).
A2 – B2	Havendo instancia de B2 é única e é associada a zero ou mais instancias de A2. A relação é efetuada pela chave primária de B2 (PK_B2) e a chave forasteira de A2 (FK_A2_B2).

4.2.4. Módulos

A vista de "Módulos" na Figura 10, apresenta a notação utilizada quando se pretende especificar os pacotes de dados e interação das classes. Classes, packages e interfaces representam artefactos concretos de programação orientada a objetos. Conetores representam os mecanismos de interação entre classes, interfaces e/ou packages.

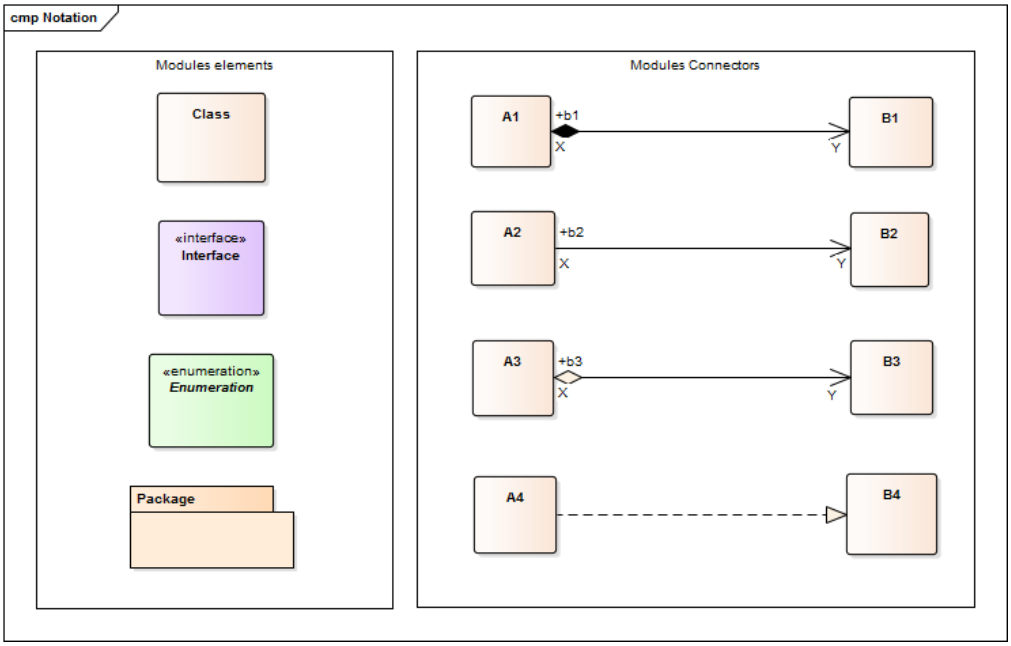


Figura 10 – Notação da vista de módulos

A Tabela 21 apresenta a descrição dos componentes presentes na Figura 10.

Tabela 21 – Descrição da vista de módulos

Elementos	
Class	Uma classe
Interface	Uma interface
Enumeration	Uma enumeração
Package	Um pacote
A1 – A4, B1 – B4	Classes estáticas
Conetores	
A1 → B1	B1 é um componente de A1, uma variável com o nome 'b1' na classe A1 é usada para a composição e o conetor só é navegável de A1 para B1. O valor de 'X' indica a multiplicidade do lado A1 do conector e o valor de 'Y' indica a multiplicidade do lado B1 do conector.
A2 → B2	B2 é associado com A2, uma variável com o nome 'b2' na classe A2 é usada para a associação e o conetor só é navegável de A2 para B2. O valor de 'X' indica a multiplicidade do lado A2 do conector e o valor de 'Y' indica a multiplicidade do lado B2 do conector.
A3 → B3	A3 agrega elementos de B3, uma variável com o nome 'b3' na classe A3 é usada para a agregação e o conetor só é navegável de A3 para B3. O valor de 'X' indica a multiplicidade do lado A3 do conector e o valor de 'Y' indica a multiplicidade do lado B3 do conector.
A4 → B4	A4 realiza B4. Todos os atributos e operações contidas em B4 estão também incluídas em A4.

4.3. Descrição Geral

O foco do trabalho realizado no segundo semestre foi maioritariamente na elaboração da solução em termos de estrutura para corresponder às necessidades do sistema. Este trabalho de arquitetura foi realizado pelo estagiário com o *mentoring* do AD.

Nas subseções seguintes é apresentada a proposta da nova arquitetura, uma vista de alocação, uma de componentes e conetores e uma vista de módulos que sustenta a transmissão de pacotes de dados do sistema.

4.3.1. Alocação

Na Figura 11 é apresentado o diagrama geral de alocação de componentes físicos da arquitetura proposta.

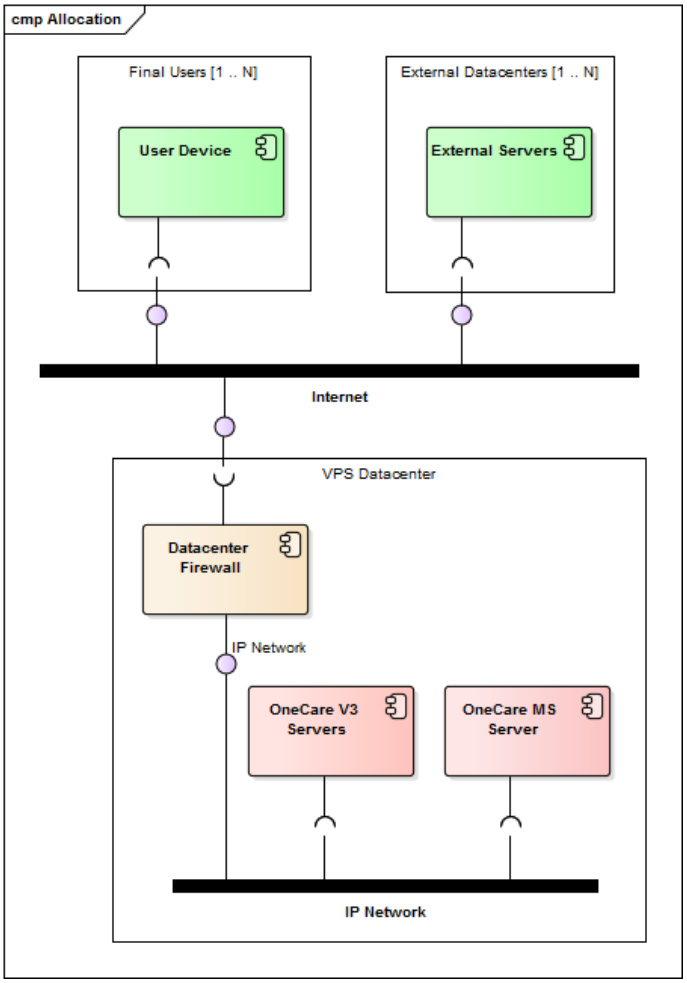


Figura 11 – Diagrama da vista de alocação

Na Tabela 22 é descrito os componentes presentes na Figura 11.

Tabela 22 – Descrição da vista de alocação

Componente	**Descrição**
User Device	Máquina utilizada pelo utilizador final para aceder ao sistema.
External Server	Servidor de um sistema externo.

Datacenter Firewall	Máquina da VPS responsável pela firewall da comunicação da rede.
OneCare V3 Servers	Maquina da Intellicare onde estão a correr o sistema legado.
OneCare MS Servers	Maquina da Intellicare onde vão ser colocados os micro serviços a correr.
Internet	Meio de comunicação do *VPS Datacenter* com o exterior.
IP Network	Meio de comunicação das diversas máquinas e servidores dentro da rede da VPS.

4.3.2. Componentes e Conectores

Na Figura 12 é apresentado o diagrama geral de componentes e conectores da arquitetura proposta.

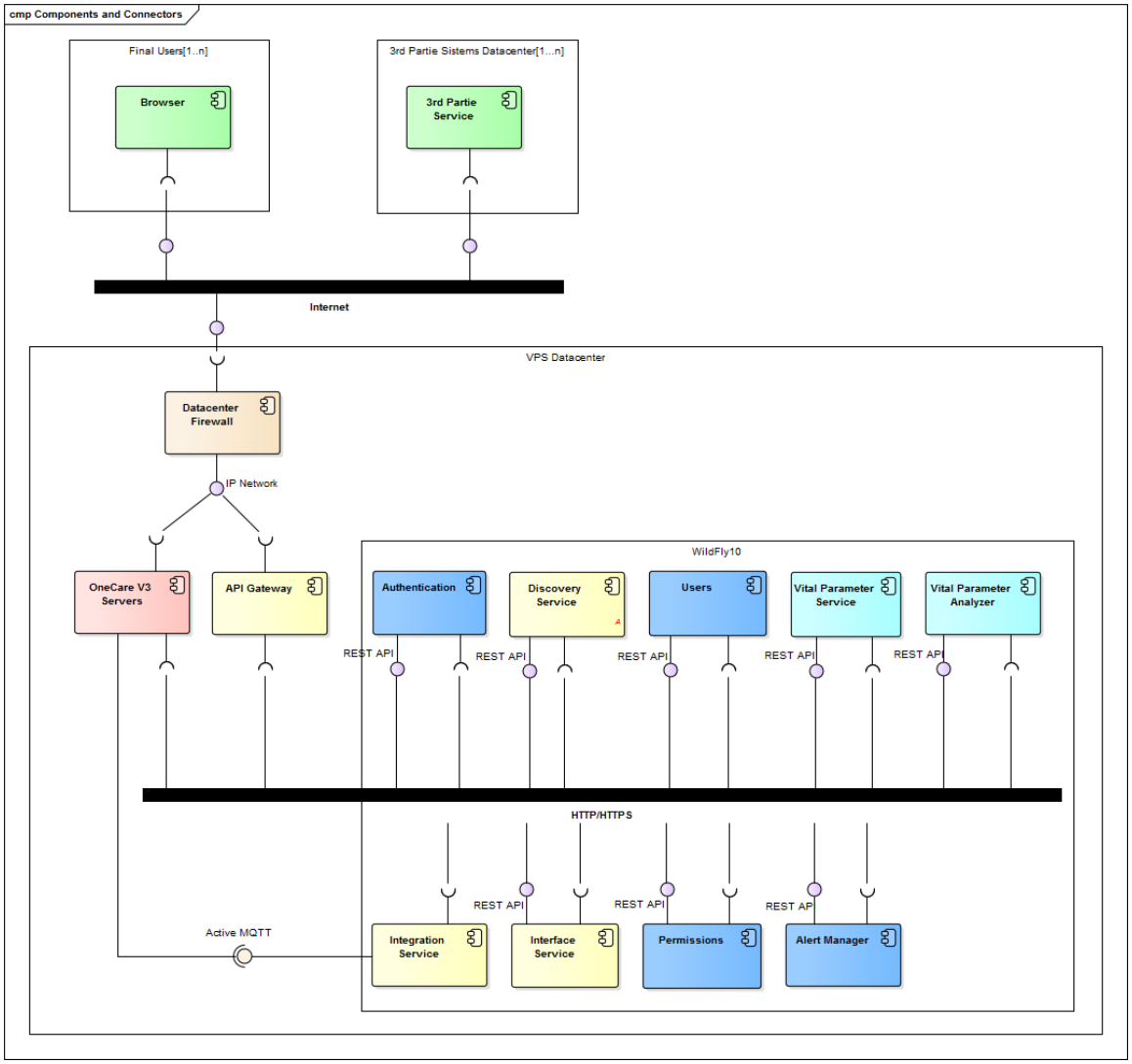


Figura 12 – Vista de componentes e conectores da arquitetura

A Tabela 23 descreve os componentes do diagrama de componentes e conetores.

Tabela 23 – Descrição dos componentes e conectores

Componente	Descrição
Browser	O *Browser* é o software que os utilizadores usam para aceder à página do Portal OneCare.
3rd Partie Service	É qualquer aplicação externa que utiliza os serviços da Intellicare
Datacenter Firewall	Componente que tem por objetivo criar restrições de acesso entre as redes existentes recorrendo a políticas de segurança no conjunto de protocolos TCP/IP.
OneCare V3 Servers	Conjunto de servidores que disponibiliza os serviços de *routing* e armazenamento da informação do OneCare.
API Gateway	Serviço responsável pelo *routing* e transformação dos pacotes externos para os micro serviços do OneCare. A transformação de pacotes é útil para responder de forma diferente às aplicações móveis e de *desktop*, por exemplo, devido às suas capacidades de processamento e de largura de banda. Em concreto no OneCare, e de forma muito simplificada, uma aplicação móvel efetua um pedido para obter as ultimas 10 medidas de tensão arterial recebendo uma resposta com os 10 valores, enquanto uma aplicação web pode receber 10 respostas, cada uma com uma medida diferente. A API Gateway consolida as 10 medições numa resposta, no caso da aplicação móvel, mantendo as respostas que recebeu do serviço de medições.
*Authentication Serv*ice	Serviço responsável pela autenticação dos serviços e dos utilizadores. O serviço guarda a informação dos *tokens* das sessões e serviços ativos. É espectável que o serviço realize as seguintes operações: • Verificar os pares *user/pass*; • Gerir e enviar *tokens* de sessão; • Verificar se os *tokens* que recebe estão válidos ou não.
*Discovery Servic*e	Serviço responsável pelo registo e localização dos serviços. O serviço guarda a informação dos serviços ativos. É espectável que o serviço realize as seguintes operações: • Registo de serviços (Figura 13); • Gerir os serviços através de *heartbeats* mantendo a sua base de dados de serviços ativos atualizada (Figura 14); • Providenciar a localização dos serviços (Figura 13).

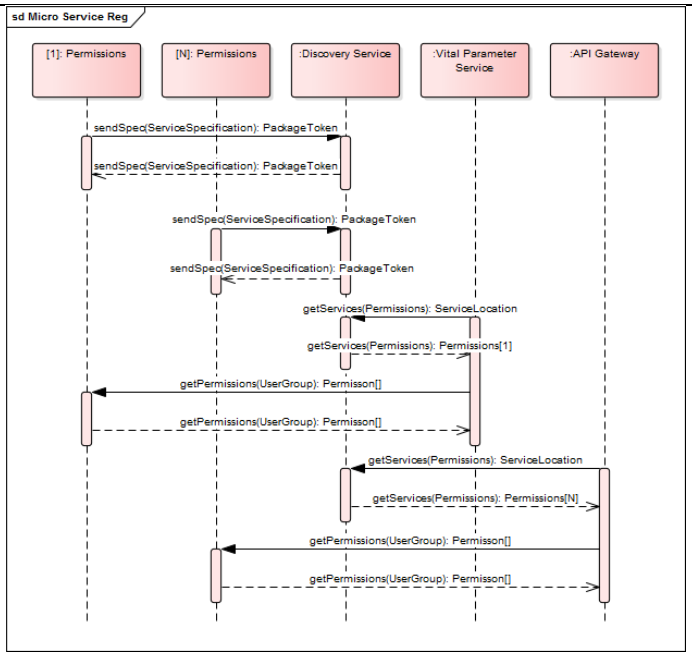


Figura 13 – Diagrama de sequencia registo e chamada a serviços

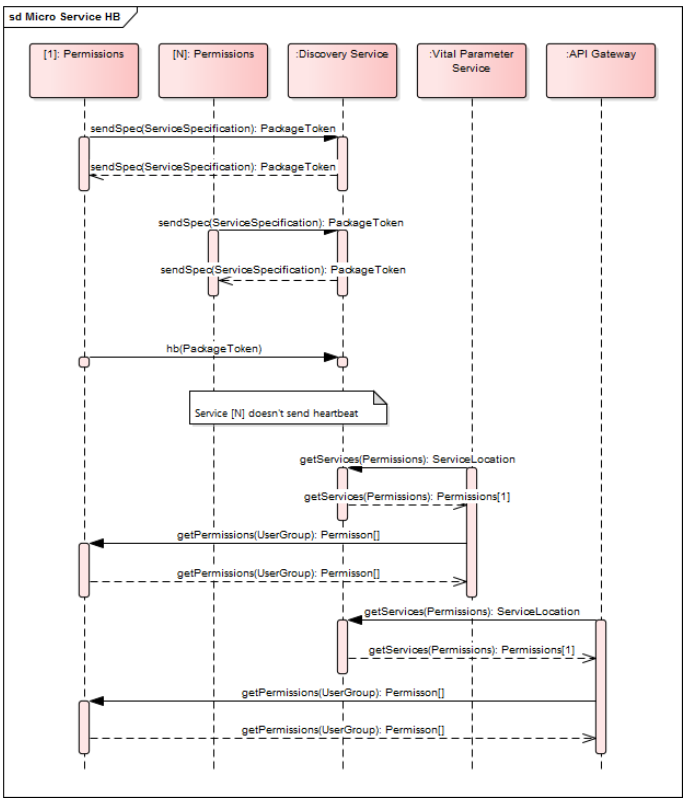


Figura 14 – Diagrama de sequencia heartbeat e chamada a serviços

Users	Serviço responsável pelo registo e gestão dos utilizadores. O serviço guarda a informação dos utilizadores do sistema. É espectável que o serviço realize as seguintes operações: • Registo e gestão de utilizadores; • Fornecer os *UserGroups* do sistema; • Fornecer o *UserGroup* de um utilizador específico.
Integration Service	Serviço responsável pela integração da aplicação legada com os novos micro serviços implementados.

	É espectável que o serviço realize operações de recolha da informação das *message queue's* legadas e transmita as mensagens para os micro serviços, desta forma é possível orquestrar a informação e estabelecer uma ponte entre o sistema antigo e a nova implementação.
Interface Service	Serviço responsável por gerir e transmitir a componente de visualização aos utilizadores. O serviço guarda a informação da interface dos serviços. É espectável que o serviço realize as seguintes operações: • Forneça ao utilizador a página que pertente; • Autenticar o utilizador que utiliza o sistema através do *Authetication Service*; • Verificar se a ação que o utilizador pretende realizar está contemplada nas suas permissões. Este serviço existe porque é uma forma de estabelecer mecanismos de segurança entre o utilizador e a informação. A interface de cada serviço poderia ser fornecida diretamente do mesmo, mas entende-se que dar ao serviço essas funcionalidades, poderá expolo a ataques e comprometer a segurança do serviço.
Permissions	Serviço responsável pela gestão de permissões do sistema. O serviço guarda a informação das permissões dos utilizadores, grupos de utilizadores e serviços. É espectável que o serviço realize as seguintes operações: • Fornecer permissões através de um *UserGroup*; • Fornecer permissões através de um *token* que recebe; • Fornecer ao *Interface Service* as permissões completas; • Permitir que sejam criadas e atualizadas as permissões de serviços e de utilizadores.
Vital Parameter Service	Este serviço é uma generalização de um serviço de gestão de parâmetros vitais. Como exemplo é previsível que existam os serviços dedicados a pressão arterial, peso, glicémia, etc. Serviço responsável pela gestão dos dados dos utilizadores. O serviço guarda a informação dos dados pessoais e tem de garantir que apenas o serviço consiga relacionar o identificador do utilizador com os seus dados, isto é, é impossível cruzar informação da base de dados de utilizadores com a base de dados dos parâmetros vitais. É espectável que o serviço realize as seguintes operações: • Gestão de valores de parâmetros vitais; • Verificar se um determinado utilizador, ou serviço, tem permissões para realizar a ação que realiza; • Fornecer a informação de um determinado utilizador; • Fornecer acesso externo para introdução e consulta de dados.
Vital Parameter Analyzer	Este serviço é uma generalização de um serviço de analisador/processador de parâmetros vitais. Como exemplo é previsível que existam os serviços dedicados a pressão arterial, peso, glicémia, etc. Serviço responsável pelo processamento de informação de dados de parâmetros vitais dos utilizadores e gerar alarmes.

	É espectável que o serviço realize as seguintes operações: • Processamento de grandes quantidades de parâmetros vitais; • Gerar alarmes e enviar para o *Alarm M*anager.
Alarm Manager	Este serviço é uma generalização de um serviço de gestor de alarmes. Como exemplo é previsível que existam os serviços dedicados a parâmetros vitais, sensores de ambiente, etc. Serviço responsável pelo processamento e encaminhamento de alarmes enviados pelos serviços de análise de parâmetros vitais ou sensores de ambiente. É espectável que o serviço realize as seguintes operações: • Gerir os alarmes do sistema; • Assegurar o encaminhamento de alarmes para os respetivos sistemas externos e/ou serviços, assim como para os utilizadores finais, nomeadamente cuidadores formais ou informais. Este serviço tem de garantir que não perde a receção de um alarme de qualquer serviço, isto é, tem de garantir alta disponibilidade e uma forte recuperação de falhas.

4.3.3. Módulos

A Figura 15 pretende especificar parte dos pacotes com os dados a transmitir para realizar a comunicação necessária entre serviços.

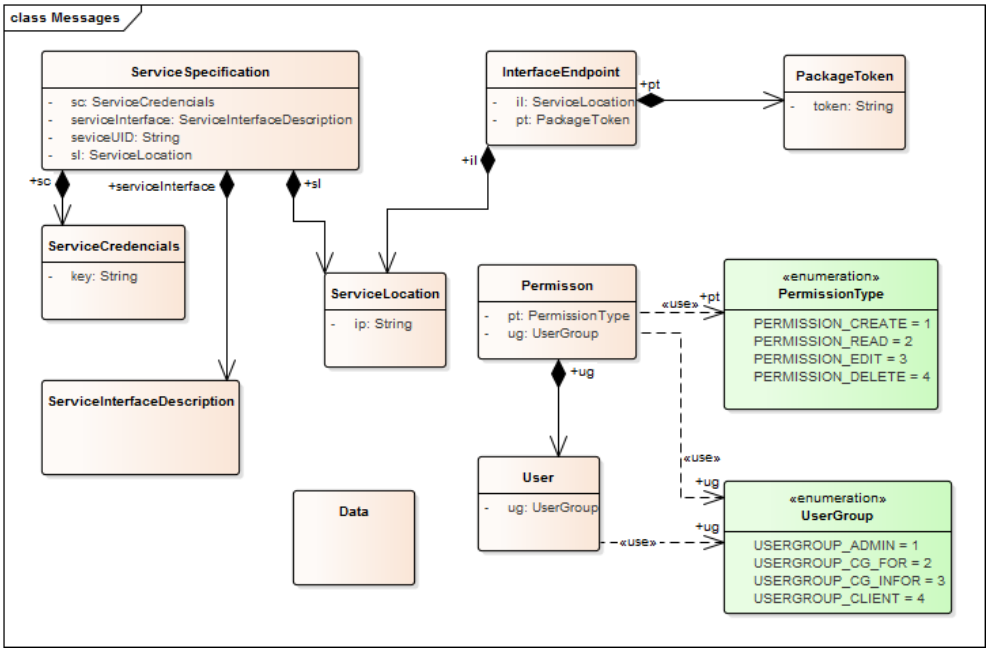


Figura 15 – Vista de módulos com parte das mensagens a transacionar

4.4. Avaliação

Antes de iniciar o processo de desenvolvimento de software de uma solução é boa prática que a arquitetura do sistema esteja suficientemente completa. Considera-se completa uma arquitetura que através de diagramas e prosa seja possível confirmar que os drivers arquiteturais da solução são cumpridos. É então realizada pelos arquitetos uma avaliação à arquitetura.

Uma forma de validar se a arquitetura especificada é suficiente para desenvolver o sistema, é verificar se são descritos componentes com detalhe suficiente que justifique os ASR's presentes na árvore de utilidade. É espectável que a avaliação seja feita com a colaboração uma pessoa diferente da que tenha elaborado os diagramas. Assim, esta atividade foi realizada com a colaboração do AD que com a sua experiência validou os diagramas.

A Tabela 24 apresenta a avaliação feita à arquitetura especificada na secção 4.3 - Descrição Geral de encontra cada ASR identificado como importante na árvore de utilidade da secção 3.6 - Árvore de Utilidade.

Tabela 24 – Avaliação da arquitetura através da justificação de ASR's

ID – ASR	Avaliação
ASR01 – O tempo de testes realizados na adição de um serviço ao sistema não é superior a 8 horas.	Como a comunicação entre serviços é feita através do protocolo HTTP/HTTPS e, ao incluir um mecanismo de registo de um serviço no sistema através do *Discovery Service*, o Administrador de Sistemas é capaz de fazer o *deploy* de um serviço no sistema, em que apenas foram sujeitos a testes os métodos do próprio serviço e a interação com os serviços com quem ele interage. Não sendo necessário testar os restantes serviços, pelo princípio de que não foram feitas alterações neles próprios ou nos dependentes, assume-se que o ASR é cumprido. A monitorização de pedidos ao *Discovery Service* permite identificar de forma automática as dependências entre os serviços.
ASR02 – O tempo de disponibilização de um serviço no sistema não é superior a 2 horas.	Usando a avaliação do ASR01, ao implementar a componente de visualização através do *Interface Service*, em que o serviço recebe dinamicamente os elementos de visualização, garantimos que o ASR é cumprido. Cada serviço fornece um *template* para ser integrado dinamicamente na interface global da aplicação. Assim, para disponibilizar uma determinada fonte de dados ao utilizador não é necessário qualquer desenvolvimento extra na componente web, o que resulta numa redução do tempo total de testes da solução.
ASR28 – O sistema legado do OneCare tem de ficar em funcionamento.	Está previsto, o sistema OneCare continuar a funcionar pelo componente *OneCareV3 Servers*, como indicado anteriormente neste relatório o OneCare tem uma arquitetura baseada em eventos. Ao incluir no sistema o *Integration Service*, é garantido que as mensagens transacionadas no sistema legado são transmitidas para os serviços da arquitetura de micro serviços. Esta integração deve ser desenvolvida a par dos serviços. Os serviços de integração funcionam em paralelo nas duas arquiteturas.
ASR04 – O tempo de disponibilização de uma nova instância de um serviço no sistema não é superior a 8 horas.	A função do *Discovery Service* é reunir informação sobre a localização dos serviços através de um identificador, endereço ip e porto onde os serviços estão à escuta. Permite que dois ou mais serviços com o mesmo identificador sejam registados em servidores diferentes. Por exemplo, podemos ter uma instância do serviço '*bloodpressure_v0.1*' a correr em '192.168.0.1:8080' e outra em '172.18.0.1:9000'. O *Discovery Service* encarregar-se-á de distribuir os pacotes às instâncias disponíveis. É garantida a escalabilidade do sistema uma vez que é possível instanciar serviços em função

	do número de clientes, sem que estes tenham necessidade de saber que serviço estão a aceder.
ASR25 – O sistema tem de estar em conformidade com a Legislação da Comissão de Proteção de Dados.	Este ASR foca na avaliação da arquitetura perante as restrições legais impostas pela Legislação de Proteção de Dados Pessoais relativamente à segurança de dados pessoais através da infraestrutura (RL01, RL02, RL04, RL06, RL08). A implementação do OneCareV3 Servers já teve em conta estas restrições e o Datacenter onde os servidores estão alojados cumprem com as mesmas. Assim, como na arquitetura proposta, os serviços de dados pessoais estão dentro do Datacenter, assume-se que os pressupostos pela Legislação continuam a ser cumpridos.
ASR37 – O modelo de dados tem de estar em conformidade com a Legislação da Comissão de Proteção de Dados no que toca à separação de dados sensíveis.	Cada serviço tem a sua própria base de dados e a arquitetura prevê que a base de dados de utilizadores seja gerida pelo serviço *Users* e as bases de dados com a informação especialmente sensível (ex: dados de pressão arterial) pelos serviços *Vital Parameter Service*. Além desta separação, que evita o cruzamento direto de tabelas, o cruzamento das bases de dados não é possível porque o identificador da pessoa só pode ser correlacionado pelo serviço que o gere.
ASR40 – O sistema tem de estar em conformidade com a Legislação da Comissão de Proteção de Dados no que toca ao acesso limitado aos dados a pessoas autenticadas.	Os métodos de autenticação ficam delegados ao serviço *Authentication* que é responsável pela credenciação dos serviços e dos utilizadores. Qualquer ação no sistema, seja por parte de utilizadores ou serviços, internos ou externos, requer a validação do *token* de sessão por parte do serviço *Authentication*. A legitimidade da ação de um agente sobre um serviço tem, ainda, de ser validada pelo serviço *Permissions*, a fim de validar se o grupo de utilizador que é atribuído ao agente, tem permissões para realizar a ação pretendida. Assegurando assim, a validação do controlo e do acesso aos dados acedidos.

Capítulo 5.
Implementação, Verificação e Validação

Depois da arquitetura, o ciclo de vida de software é a implementação e depois a verificação e validação do produto implementado. Essa validação permite cruzar a solução desenvolvida com os drivers arquiteturais que foram estabelecidos, de forma a saber se cumpre ou não os requisitos e como consequência se resolve o problema inicial.

Para testar a arquitetura é necessário implementar um serviço e verificar se cumpre os principais atributos de qualidade exigidos pela Intellicare, ou seja, a modificabilidade dos serviços, e a testabilidade. Relembrando os capítulos anteriores e resumindo, é extremamente importante que um serviço possa ser incluído no sistema e fique disponível dinamicamente sem alterações significativas, e que o seu tempo de testes seja inferior a um dia de trabalho, como definido pelo atributo de qualidade da arquitetura – AQ01 – presente no documento externo "Requisitos MSA", isto é, que a inclusão de um serviço interfira apena com os serviços que este utiliza.

Para tal, e aproveitando uma oportunidade de negócio da Intellicare, foi inserido no contexto do estágio, o desenvolvimento de um produto ou serviço que serve para testar estas componentes essenciais da nova arquitetura.

5.1. Projeto OneCare Call

O OneCare Call é um serviço de controlo de medições e tomas de medicamentos realizado através de chamadas automáticas de voz. Trata-se de um protótipo proposto pelo orientador AD para justificar a necessidade de migração da arquitetura do OneCare. O processo de elicitação de requisitos e atributos de qualidade adotado seguiu a fase inicial do método ACDM. O resultado deste processo pode ser consultado no anexo externo "Requisitos OCC", assim como uma descrição mais elaborada do serviço.

Para ser identificado o esforço de inclusão do produto na linha OneCare, foi analisado o número de componentes que na atual arquitetura tinham de ser modificados e comparado com o número de componentes a ser testados chega-se à conclusão que na necessidade de produzir frequentemente novos serviços, a arquitetura atual é pouco flexível e extremamente dispendiosa. Pois quanto a componentes modificados, segundo a arquitetura atual, é necessário modificar o componente de base de dados, o core e as interfaces da web e de acesso a dados. Todos estes componentes são responsáveis pelo funcionamento OneCare, ou seja, a dimensão do seu código e o número de dependências é elevada. Ainda, como observado no estado de arte, devido à arquitetura, todos os componentes necessitam de ser testados, resultando num investimento de tempo/esforço maior do que o necessário, uma vez que, os componentes modificados são apenas quatro de um sistema composto por nove, os outro cinco componentes relativos a: gerador de gráficos; importação de dados; sistema de configurações; e gestores de alertas e notificações, apesar de não terem sofrido qualquer alteração continuam a ser sujeitos a testes.

Com esta análise justifica-se a necessidade de alterar a estrutura da aplicação para no futuro a inclusão de novos serviços não estar sujeita a este tipo de impedimento. Então, o pipeline de produção e disponibilização de um qualquer produto ou serviço, tendo em conta que a empresa passa por uma fase de inclusão de novas funcionalidades ativa, deve ser: especificar;

desenvolver; testar e disponibilizar. Esta abordagem só é possível se a arquitetura do sistema deixar de ser monolítica.

Com estes fatores em conta, este protótipo foi desenvolvido como um serviço, respondendo às necessidades e com a qualidade especificada no documento "Requisitos OneCare Call". É então um serviço independente da arquitetura, funcional e passível de ser utilizado por aplicações terceiras.

Note que, o serviço necessita de um sistema externo à empresa. O processo é gerido segundo a metodologia CMMI-DEV reportado na secção 6.3.3 - Gestão de Acordos com Fornecedores e evidenciado pelos anexos externos "Processo de Contratação com a G9".

5.1.1. Drivers Arquiteturais

Como um produto de software, e a semelhança do que se realizou para a nova arquitetura é necessário fazer uma elicitação de drivers arquiteturais. O detalhe dos drivers arquiteturais é confidencial e é apresentado no documento "Requisitos OCC".

Na Tabela 25 são apresentados os requisitos do sistema sob forma de casos de uso.

Tabela 25 – Tabela representativa dos casos de uso do OneCare Call

ID – Titulo	Descrição
UC01 – Configurar controlo	Permite que o Cuidador configure os parâmetros de alerta e de chamada que vão ser gerados e utilizados para controlo da toma de medicamento/realização de medição.
UC02 – Configurar som	Permite que o Cuidador configure o som que vai ser ouvido na chamada gerada pelo controlo respetivo.
UC03 – Escalonar e efetuar chamadas	Permite que as chamadas definidas pelos controlos sejam realizadas.
UC04 – Emitir fatura	Permite que o Suporte emita uma fatura ao cliente com os custos dos serviços de chamadas prestados pela Intellicare
UC05 – Registar resposta	Permite registar a resposta às chamadas efetuadas e gerar alertas para a plataforma OneCare
UC06 – Efetuar chamada	Permite que as chamadas definidas pelos controlos sejam realizadas
UC07 – Comunicação entre o sistema e o VMS	Permite que o sistema comunique com o VMS.

Na Tabela 26 são apresentados os atributos de qualidade.

Tabela 26 – Tabela representativa dos atributos de qualidade do OneCare Call

ID – Titulo	Descrição
AQ01 – Tempo para efetuar chamada de teste	O atributo de qualidade define o tempo que o sistema leva desde que o utilizador submete um som e introduz um número de telefone de teste, até que a chamada chega ao telemóvel. Tem como objetivo reduzir o tempo de espera entre ações por parte do Cuidador. Esta medida existe para levar o Cuidador a uma

	interação mais ativa, reduzir a probabilidade de resistência à tecnologia e excluir a situação de impaciência da espera ativa.
AQ02 – Tempo para processar resposta à chamada de teste	O atributo de qualidade define o tempo que o sistema leva desde que o utilizador responde à chamada de teste através das teclas do equipamento, até que o resultado é apresentado no portal. Tem como objetivo reduzir o tempo de espera entre ações por parte do Cuidador. Esta medida existe para levar o Cuidador a uma interação mais ativa, reduzir a probabilidade de resistência à tecnologia e excluir a situação de impaciência da espera ativa.
AQ03 – Perceção das chamadas por parte do idoso	O atributo de qualidade define a qualidade do som, voz e conjunção de elementos da chamada. Tem como objetivo definir a qualidade da chamada ao estabelecer o número máximo de chamadas não compreendidas pelo idoso. Esta medida existe para levar o idoso à aceitação do controlo e reduzir a probabilidade de uma chamada não ser respondida segundo a verdade. Este atributo de qualidade deve ser testado preferencialmente com um grupo de idosos de tamanho superior a 20 com um total de chamadas superior a 200 a fim de se poder aferir com algum rigor a qualidade da chamada.
AQ04 - Auxilio perante a configuração de um controlo	O atributo de qualidade estabelece a obrigatoriedade de auxiliar o Cuidador no ato de configuração de um controlo no portal OneCare. Tem como objetivo definir os artefactos que são necessários e devem ser disponibilizados pela plataforma. Esta medida existe para levar o Cuidador a uma interação assertiva, reduzir a probabilidade de resistência à tecnologia e reduzir a probabilidade de um Cuidador configurar um controlo erradamente por culpa da disposição de elementos no formulário do portal.
AQ05 – Atraso da chamada de controlo	O atributo de qualidade define o tempo de atraso da chamada tolerado pelo sistema sem ter de avisar o A04 – Suporte. Tem como objetivo avisar o Suporte da necessidade de renegociar o contrato do número máximo de VMS em simultâneo. Esta medida existe para aumentar a disponibilidade do sistema assim como a sua performance devido à sobrecarga do sistema de realização de chamadas.
AQ06 – Time out pedido de chamada	O atributo de qualidade define o tempo máximo de espera de uma confirmação de chamada efetuada tolerado pelo sistema. Tem como objetivo estabelecer o time out de espera pelo sistema VMS. Esta medida existe para aumentar a performance do sistema, avisar o Suporte para eventuais problemas no sistema VMS e evitar que não sejam efetuadas chamadas de controlo sem que os cuidadores saibam.
AQ07 – Time out email com resposta	O atributo de qualidade define o tempo máximo de espera de um email com resposta a um controlo tolerado pelo sistema. Tem como objetivo estabelecer o time out de espera pelo sistema do email de resposta a uma chamada de controlo. Esta medida existe para aumentar a performance e a tolerância a falhas do sistema, avisar o Suporte para eventuais problemas no sistema G9 e evita que fiquem chamadas no sistema sem resolução.
AQ08 – Tempo de notificação	O atributo de qualidade define o tempo que o sistema deve demorar desde que recebe o email de resposta até que é enviado

	para o sistema de alarmes do OneCare. Tem como objetivo assegurar que as respostas dos controlos cheguem aos cuidadores no espaço de tempo definido consoante o tipo de resposta. Esta medida garante que o sistema dê prioridade máxima a respostas do tipo "Preciso de ajuda", alta a respostas "Não" e baixa a "Sim" garantindo que o cuidador recebe notificações logo que o sistema tenha informações.
AQ09 – Tempo de confirmação de alteração de controlo	O atributo de qualidade define o tempo que o sistema leva desde que o utilizador submete uma alteração a um controlo, até que recebe a confirmação de alteração. Tem como objetivo reduzir o tempo de espera entre ações por parte do Cuidador. Esta medida existe para levar o Cuidador a uma interação mais ativa, reduzir a probabilidade de resistência à tecnologia e excluir a situação de impaciência da espera ativa.

5.1.2. Arquitetura

Seguindo o ciclo do produto, foi pensada a arquitetura que define a estrutura dos elementos do novo serviço OneCare Call inserindo-os na arquitetura proposta. De seguida são apresentados, à semelhança da arquitetura detalhada anteriormente, diagramas de alocação, componentes e conetores e entidade relação do OneCare Call.

Na Figura 16 é apresentado o diagrama de alocação do OneCare Call.

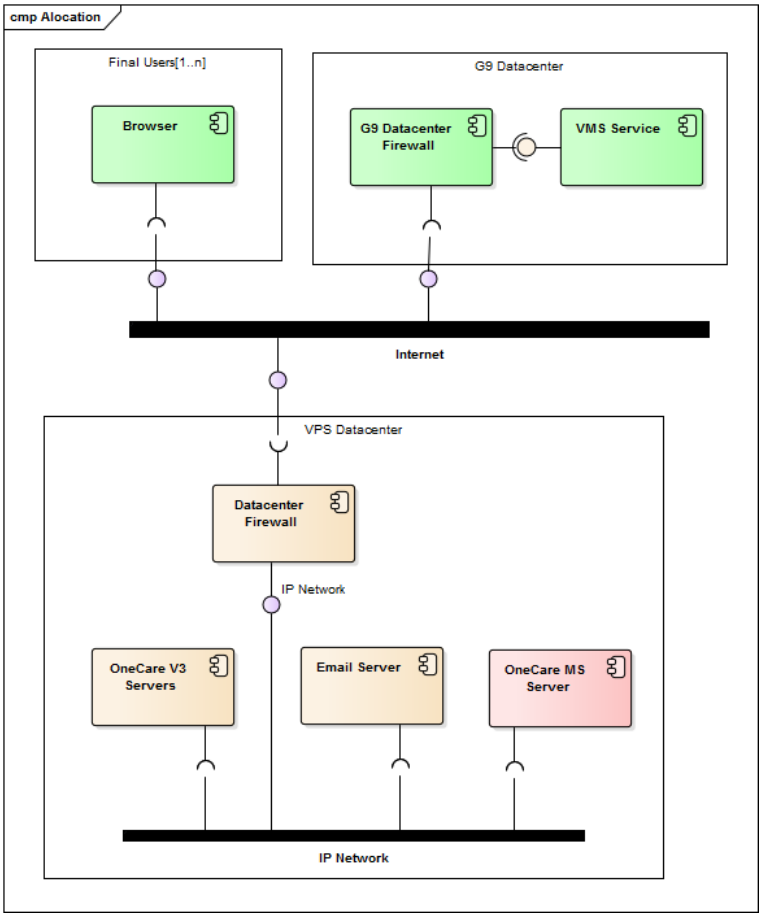


Figura 16 – Diagrama de alocação do OneCare Call

Na Tabela 27 é descrito o propósito de cada componente.

Tabela 27 – Descrição dos componentes do diagrama de alocação do OneCare Call

Componente	Descrição
G9 *Datacenter Firewall*	Componente que tem por objetivo criar restrições de acesso entre as redes existentes.
VMS Service	Servidor responsável por efetuar chamadas automáticas e recolher respostas dos utilizadores finais. É o serviço contratado que faz ponte entre os sistemas OneCare Call e o cliente. É espectável que o serviço realize as seguintes operações: • Efetuar chamadas de voz para o utilizador final; • Enviar a resposta do utilizador por email.
Email Server	Servidor responsável pela gestão de emails da VPS.
OneCare MS Server	Servidor onde estão alojados os micro serviços da Intellicare.

Como podemos observar pelo diagrama anterior, existe a necessidade de fazer uso de um serviço externo à Intellicare e está representado por um limite onde a comunicação com essa empresa é feita através da Internet. Repare-se que o componente alterado, é o componente onde vão ser introduzidos os serviços responsáveis pelo OneCare Call.

Na Figura 17 é apresentado o diagrama de componentes e conetores, no qual se percebe a desmultiplicação de servidores e são evidentes os serviços e aplicações responsáveis pelos métodos e rotinas que tornam possível as funcionalidades especificadas.

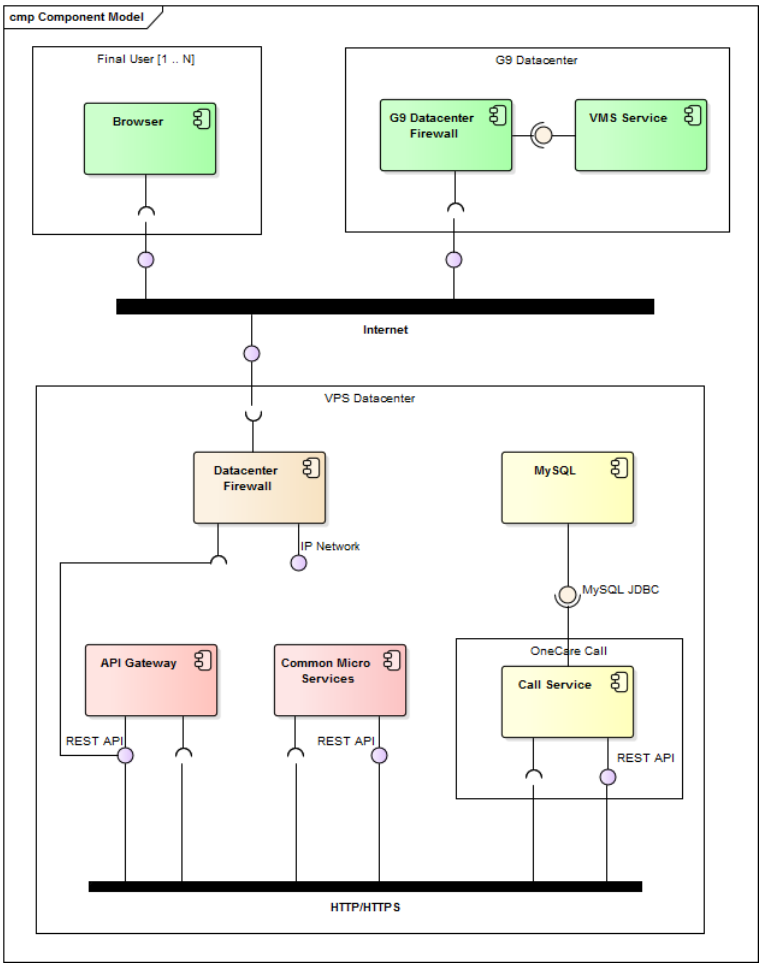


Figura 17 – Diagrama de componentes e conectores especificado para o OneCare Call

Como todos os diagramas devem ser acompanhados de uma prosa descritiva, na Tabela 28 são descritos os novos componentes do diagrama.

Tabela 28 – Descrição dos componentes e conectores especificados do One Care Call

Componente	Descrição
MySQL	Componente responsável por gerir o modelo de dados da aplicação do controlo de chamadas.
Call Service	Serviço responsável pelo core da aplicação, gere os pedidos de dados e faz os pedidos necessários ao *VMS Service*. É espectável que o serviço realize as seguintes operações: • *Create, Read, Update, Delete (CRUD)* dos dados através da interface *MySQL JDBC*; • Gerar pedidos de chamadas automáticas; • Disponibilizar uma *API* para gestão de dados.
Common Micro Services	Este componente é uma representação generalizada dos serviços globais da arquitetura. Neste componente estão incluídos os serviços: • *Discovery Service;* • *Users;* • *Permissions;* • *Interface Service;* • *Authetication;* • *Alert Manager*.

Estas especificações foram validadas com o AD em como a disposição de componentes era suficiente para cumprir com os drivers arquiteturais do OneCare Call.

Criando um desacoplamento dos diversos serviços fazendo uso dos micro serviços de base, entenda-se os micro serviços de suporte nos diagramas acima identificado por "*Common Micro Services*", seguiu-se a especificação de micro serviços dos capítulos anteriores com a intenção de beneficiar das suas vantagens e validar o conceito da arquitetura.

5.1.3. Implementação

Estando definida e validada a arquitetura, o desenvolvimento teve inicio segundo a especificação presente no documento externo "Requisitos OCC". Mas como em muitos outros projetos de software, a especificação sofre alterações e neste caso as alterações foram restrições criadas pela empresa externa.

O meio comunicação, que no inicio estava especificada, de transmissão de respostas dos clientes através de pedidos REST aos serviços do OneCare, representaria um investimento incomportável por parte da Intellicare. Foram feitas renegociações desse mecanismo e ficou definido que o envio das respostas iria ser feito através de email.

Estas alterações tiveram impacto na arquitetura, sendo necessário acrescentar um componente para tratar dos emails. Na Figura 18 é apresentado o diagrama do software implementado para fazer face á restrição.

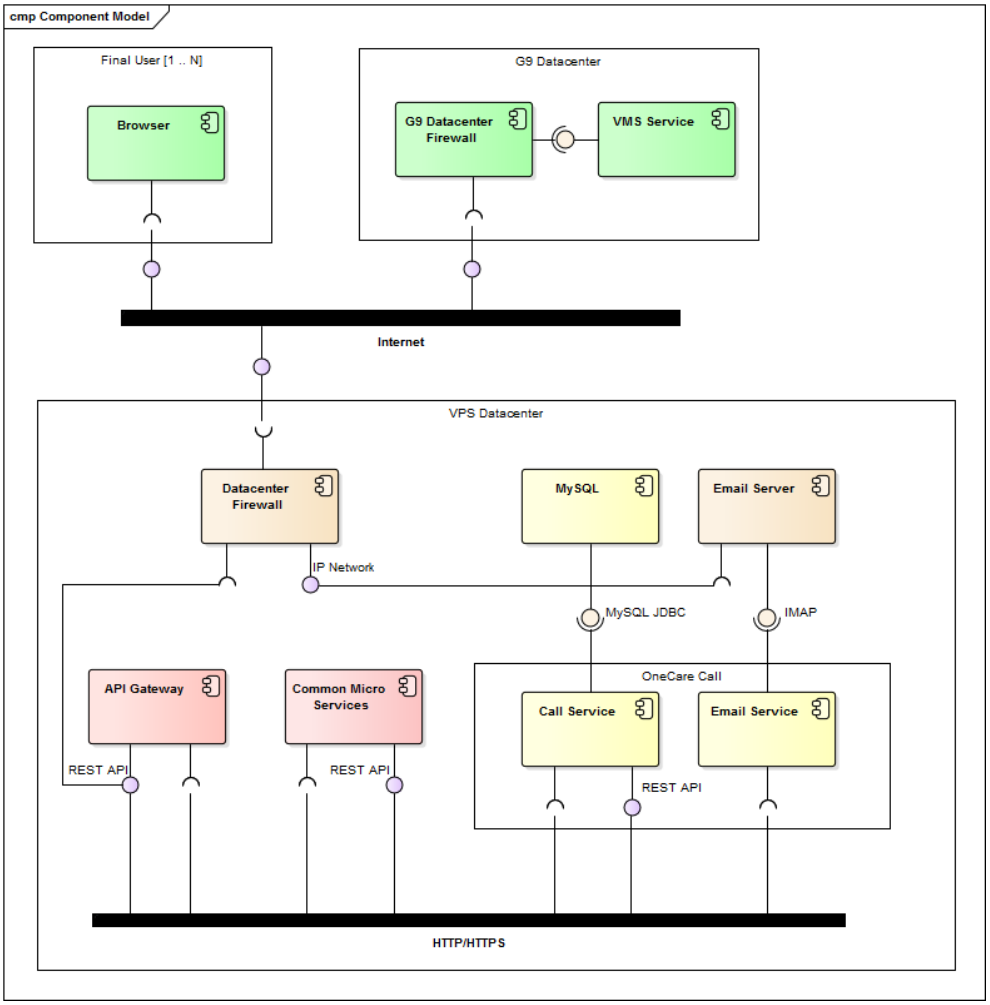


Figura 18 – Diagrama de componentes e conectores implementado

Existe um ponto importante a reter nesta alteração. Uma vez que o *Call Service* já estava implementado e preparado para receber pedidos REST com as respostas às chamadas, foi decidido que esse componente não ia ser alterado e foi criado um novo serviço. O novo serviço é descrito na Tabela 29.

Tabela 29 – Descrição dos componentes e conetores implementados

Componente	Descrição
Email Service	Serviço responsável pela verificação de emails do serviço VMS com as respostas aos pedidos de camada. É espectável que o serviço funcione como um *listener* a uma caixa de email especifica e quando recebe um email da empresa subcontratada com um determinado assunto, processe o conteúdo e faça um pedido REST ao *Call Service*.

O modelo de dados que permite a realização das funcionalidades implementadas está reproduzido na Figura 19.

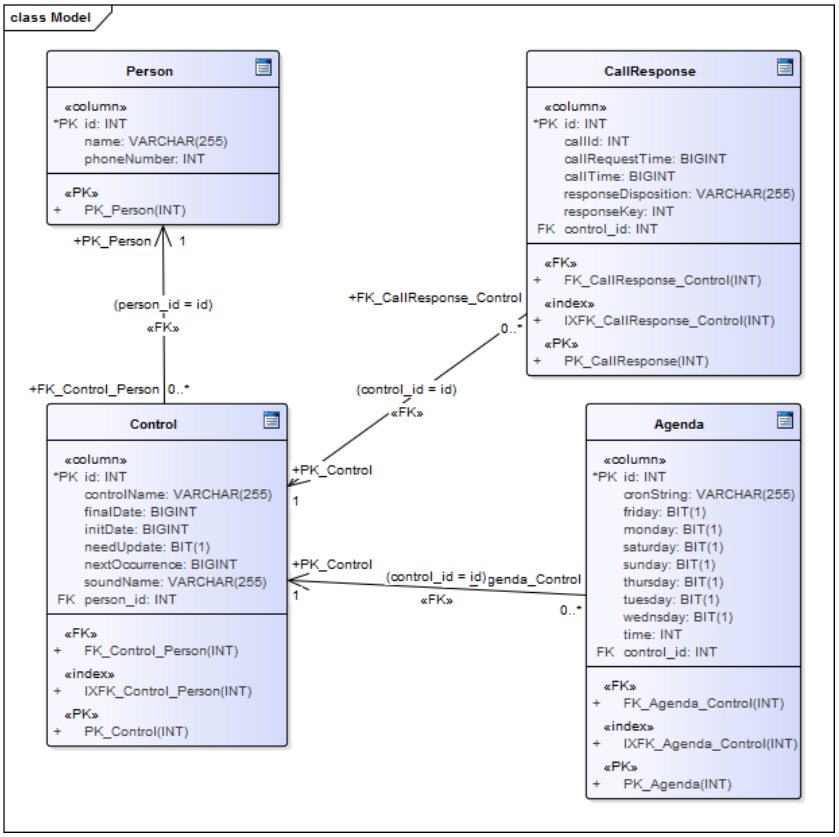


Figura 19 – Diagrama ER do OneCare Call

A prosa do diagrama é descrita na Tabela 30.

Tabela 30 – Descrição das tabelas do diagrama ER do OneCare Call

Tabela	Descrição
Person (Pessoa)	Conjunto de parâmetros que definem uma pessoa, habitualmente um idoso, mas pode ser uma pessoa qualquer que necessite do serviço. Uma pessoa tem um nome; e um número de telemóvel de contacto. Para controlo interno da aplicação é definido um id de identificação único. A uma pessoa, podem ser associados N controlos.
Control (Controlo)	Conjunto de parâmetros que definem uma necessidade de um idoso efetuar uma medição ou tomar uma medicação. Um controlo tem um nome para fácil identificação do cuidador, por exemplo, "Tensão Arterial"; é definida uma data de inicio e fim do controlo; é definido o som a reproduzir na chamada. Para o controlo interno da aplicação, um controlo é associado a uma pessoa; tem um id de identificação único; tem a hora da próxima ocorrência; tem um campo que define se a próxima ocorrência deve ser atualizada. A um controlo podem ser associadas N chamada/resposta e N agendas.
CallResponse (chamada/resposta)	Conjunto de parâmetros que definem uma chamada e a sua resposta. Uma chamada/resposta tem um identificador de chamada; uma hora em que o pedido de chamada foi feito; uma hora em que a chamada foi realizada; o tipo de resposta à chamada; e a tecla que

	foi premida. Para controlo interno uma chamada/resposta é associada a um controlo e tem um id de identificação único.
Agenda (Agenda)	Conjunto de parâmetros que definem uma agenda de um controlo, isto é, a que horas e a que dias da semana o controlo é feito. Uma agenda tem uma hora a que o controlo é realizado; os dias da semana que se realiza; e uma String com a composição destes elementos em formato 'Quartz'. Para controlo interno uma agenda é associada a um controlo e tem um id de identificação único.

Esta entidade relação permite que no futuro um utilizador autenticado no sistema possa ter acesso e gerir os dados dos seus familiares (pessoas).

5.2. Verificação

A verificação de sistemas é feita por meio de testes, as duas principais razões para se fazer testes são aferir a qualidade ou aceitação e descobrir problemas do produto. Os testes realizam-se porque existem falhas, essa é a verdade maior no que toca ao desenvolvimento de software. (Jorgensen, 2008)

Para a verificação dos serviços implementados foram efetuados testes que serviram para verificar se o software produzido corresponde com o especificado. Foram ainda realizados testes à plataforma da empresa externa. Como exemplos de testes realizados, e pegando no conceito de comportamentos referido na secção 3.1 - , foram elencados e apresentados na Tabela 31, quatro casos de teste que se incluem nas categorias: T – comportamentos testados; 2 – comportamentos especificados e testados; 3 – comportamentos implementados e testados; e 4 comportamentos especificados, implementados e testados.

Tabela 31 – Apresentação de exemplos de testes por categorias de comportamentos

Categoria	Descrição do Teste	Propósito
T	A usabilidade do sistema com equipamentos de telecomunicações de teclas e ecrã tátil.	A resposta às chamadas automáticas é feita por meio de teclas do equipamento telefónico, do 0 ao 9. Como o acesso ao teclado dos equipamentos de teclas é diferente dos de ecrã táctil, é necessário assegurar que a utilização é similar.
2	Perceção de chamadas por parte do idoso.	A resposta do idoso pode gerar alarmes para os seus cuidadores ou no limite pode acionar mecanismos de socorro. É assim importante que o idoso interprete a mensagem de forma acertada.
3	A resposta à chamada é recebida por email e redireciona para o serviço OneCare Call.	Pela impossibilidade de a empresa externa enviar a resposta através da API do serviço Call. É necessário testar se o mecanismo alternativo faz chegar a mensagem corretamente ao serviço.
4	É possível atingir o número máximo contratado de chamadas em simultâneo.	É necessário testar se a plataforma externa suporta o número de chamadas acordadas para que o sistema possa escalonar as chamadas os avisar o suporte de que é necessário aumentar valor máximo.

Dos testes acima referidos, o teste de categoria 2 surge do atributo de qualidade do serviço – AQ02 – este caso de teste, visa testar um comportamento que é impossível de implementar, pois a qualidade do som, quer do telemóvel, quer do ficheiro de áudio, não pode ser alterada por meio de desenvolvimento. No entanto, este comportamento tem de ser tido em conta e caso surjam erros e o teste falhar, é necessário encontrar uma solução, que pode ser por exemplo alterar ou ficheiro de som ou o conteúdo da mensagem, com a mesma finalidade.

O teste da categoria 3, surge da alteração às funcionalidades fornecidas pela empresa externa que na impossibilidade de realizar trabalho de desenvolvimento para enviar a resposta diretamente para a API do serviço OneCare Call. A solução alternativa passou por definir um pacote de informação com os dados da resposta e enviar para uma conta de email, exclusiva para o efeito, da Intellicare. Esta alteração, por ser provisória, mas que serve para o protótipo, não se encontra especificada no documento "Requisitos OCC", daí fazer parte apenas dos comportamentos implementados e testados.

À medida que iam sendo detetados erros nos serviços implementados na Intellicare, estes foram resolvidos a par da implementação que ia sendo realizada. No caso dos erros na detetados na plataforma externa, estes foram reportados por email, e esclarecidos diretamente por telefone ou email com o responsável pela plataforma. Os testes realizados focaram na verificação do core dos serviços e na interação com serviços dependentes, de realçar que neste tipo de solução existem testes que são necessários executar para lá do contexto de desenvolvimento de software

5.3. Validação

Na validação pretende-se demonstrar que foi produzido o software correto, ou seja, que os requisitos estão de acordo com as necessidades do cliente. Tendo o cliente um perfil técnico e estando os requisitos funcionais expressos em casos de uso, decidiu-se que a melhor forma de efetuar a validação seria pela criação de *threads* de execução que percorrem grande parte dos casos de uso e ilustram a forma como o sistema irá ser usado. Esta abordagem é prescrita por Jorgensen em "*Software testing: a craftsman's approach*" (Jorgensen, 2008). Desta forma foram definidas as *threads* de execução definidas abaixo.

5.3.1. *Threads* de Execução

Uma *thread* é definida por um conjunto de ações e pode ter a forma de: um cenário de utilização normal; um caso de teste ao nível do sistema; um par estímulo-resposta; um comportamento que resulte de uma sequência de inputs ao nível do sistema; uma sequência de transição de estados de uma máquina do sistema; uma sequência de interação entre objetos e execução de métodos. Espera-se que as *threads* sejam capazes de ser executadas ou testadas através dos casos de uso especificados e do sistema. É no sentido de testar e validar os requisitos identificados na secção 3.1 - que se optou por utilizar este mecanismo. (Jergensen, 2008)

Pretende-se apresentar duas *threads* que sejam capazes de validar o sistema e garantir uma cobertura dos requisitos vasta, privilegiando os que foram avaliados com prioritários. Este trabalho vai permitir identificar falhas nos requisitos, perceber se o que está especificado está bem escrito. Ajuda a resolver o problema da difícil comunicação com o cliente, uma vez que as *threads* são desenvolvidas numa linguagem familiar, e ainda permitir que identifique através da sua experiência as fases das *threads* que estão menos bem especificadas ou mais especificas da área de negócio. Isto não significa que sejam identificados todos os defeitos da especificação, mas garante pelo menos a integridade dos casos de uso mais prioritários.

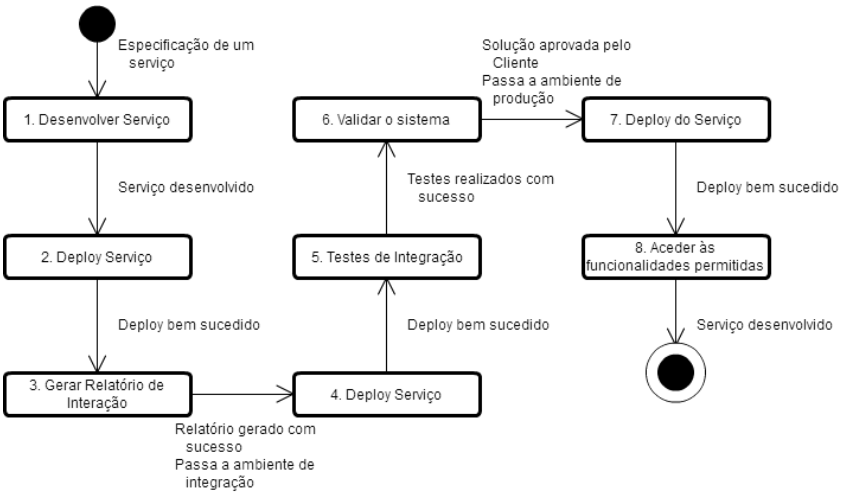


Figura 20 – Thread de execução de transições entre ambientes

Na Figura 20 pretende-se que a TE identifique explicitamente o percurso de *deploy* de um serviço desde o desenvolvimento até ser usado pelo utilizador final. A passagem é feita através dos três ambientes (desenvolvimento, integração e produção) e a TE identifica as diferentes fases e divide as ações possíveis e necessárias para que um serviço avance no seu ciclo de vida. O objetivo é então garantir que os requisitos identificados estão devidamente estruturados e que através deles se consegue garantir a execução do *pipeline*. Com esta TE é possível testar que o sistema segundo os atributos de qualidade AQ01 e AQ02 da arquitetura passando pelos casos de uso UC01, e do UC04 ao UC09.

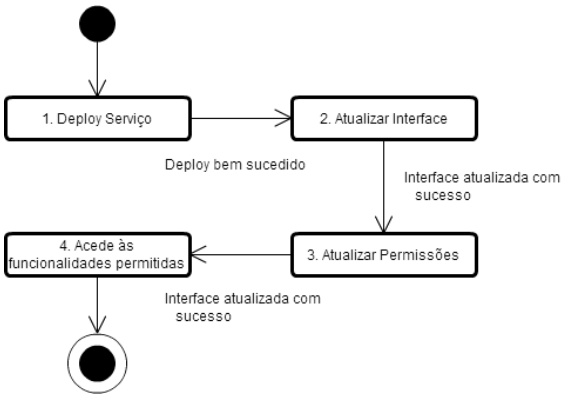


Figura 21 – Thread de execução da atualização automática da interface

Na Figura 21 pretende-se que a TE identifique o percurso da atualização, automática da interface, e manual das permissões do Administrado do Sistema, perante os diferentes grupos de utilizadores. O fluxo de eventos parte no *deploy* de um serviço bem-sucedido no sistema. O serviço regista-se no *Discovery System* e a partir dai tem de haver dois procedimentos independentes: por um lado o serviço responsável pelas permissões é capaz de identificar o novo serviço, atualizado através do *Discovery System*, e permite definir novas regras para esse serviço; por outro, a interface também é capaz de o identificar, atualizado pelo *Discovery System*, e permite que o utilizador final interaja com os dados do novo serviço se este tiver permissões para tal. A Thread de execução é apresentada mais detalhadamente na Figura 22.

A validação positiva da TE é importante para garantir que o sistema responde à necessidade de utilização de um serviço, segundo os atributos de qualidade AQ01 e AQ02 da arquitetura passando pelos casos de uso UC01, UC04, UC05 e UC09.

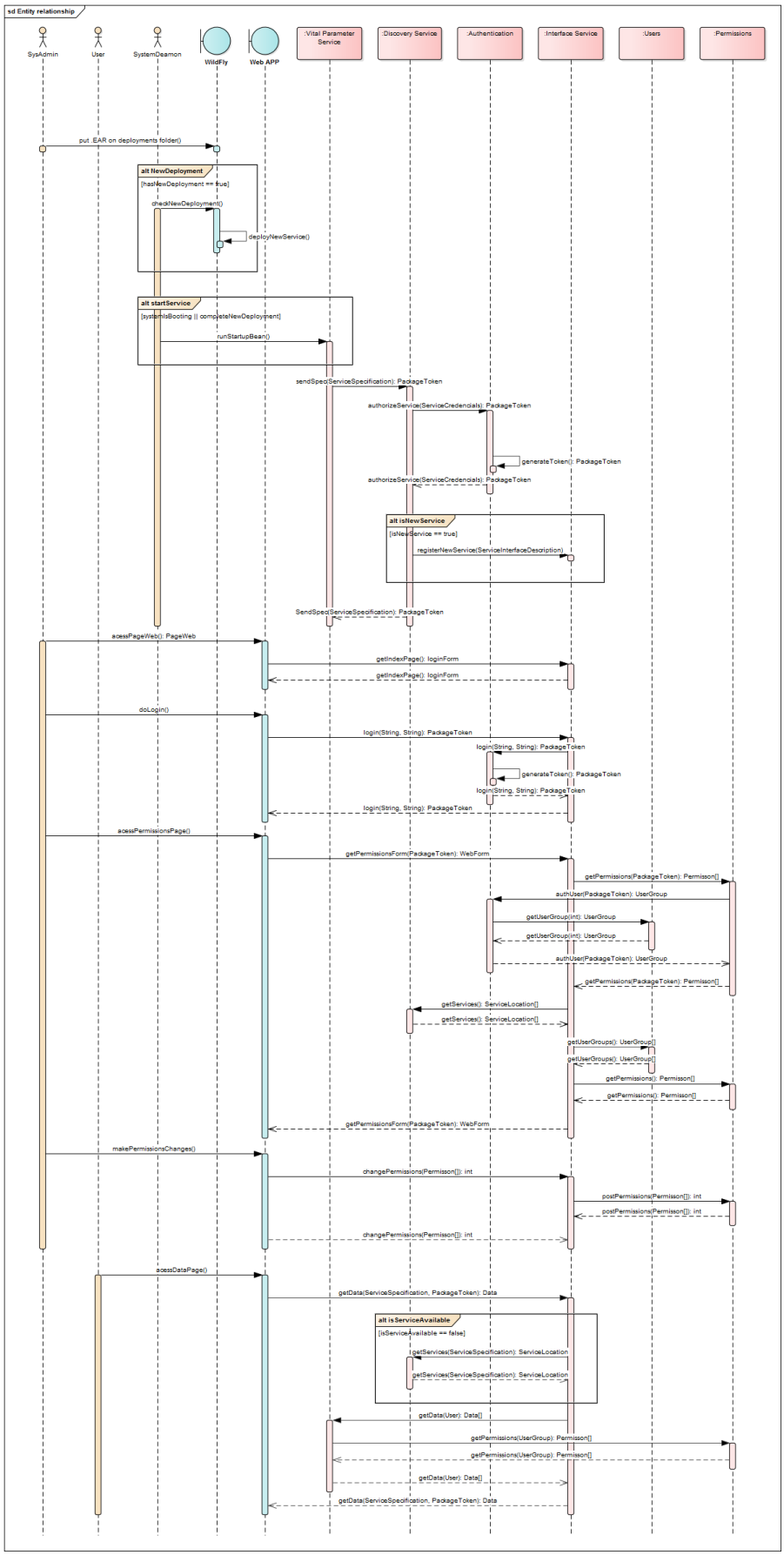


Figura 22 – Thread de execução detalhada da atualização automática da interface

5.3.2. Resultados da Validação

Interessava ainda perceber se as métricas definidas nos atributos de qualidade faziam sentidos. Em termos de testes o produto OneCare Call é integralmente testado com esforço de 4 horas sendo que a duração pode ser bastante superior uma vez que os compassos de espera para *trigger* de chamada podem durar várias horas. Sendo o OneCare Call um componente muito simples, as 8 horas de esforço para 80% dos casos podem ser difíceis de cumprir.

O objetivo de disponibilizar o serviço em 2 horas em 80% dos casos – atributo de qualidade AQ02 do documento externo "Requisitos MSA", no caso do OneCare Call foi conseguido. O tempo necessário à inclusão do serviço na interface é inferior a 10 minutos, e depreende-se que, para um outro serviço mais complexo, em 80% dos casos esse tempo seja possível de cumprir.

O serviço de email, que foi necessário implementar devido à restrição de funcionalidades disponíveis pela empresa externa, é um ótimo exemplo que serve de justificação e validação da arquitetura de micro serviços.

O desenvolvimento inicial e especificado pela Intellicare do OneCare Call, previa a receção de respostas através de uma API REST invocada diretamente pelos serviços da empresa externa. O software foi desenvolvido, testado e inicializado nos servidores da empresa. Realizava a gestão dos controlos, os pedidos de chamadas agendadas eram feitos à plataforma externa, e tinha disponível uma API REST para inserção de respostas. Pelas restrições da empresa externa, as respostas chegavam por email à Intellicare e tinham de ser introduzidas no serviço. Foi decido que, em vez do serviço consultar a caixa de entrada de email, essa ação iria ser realizada por um serviço independente. Esta solução foi realizada por dois motivos:

1. Não é necessário efetuar alterações ao serviço core do OneCare Call, e não é necessário testar as alterações realizadas, uma vez que o serviço irá fazer uso da API REST disponível;
2. Quando a empresa externa realizar o desenvolvimento necessário para enviar a resposta por um pedido REST, o core do OneCare Call continuará preparado para receber esses pedidos. A única alteração ao sistema é a desativação do serviço independente que gere a caixa de email, mas nem essa alteração interfere com o sistema.

Com este exemplo, a necessidade de modificar um componente sem afetar o funcionamento do resto do sistema também é validada. Através do desacoplamento de serviços e da disponibilização de uma API suficientemente completa de gestão da informação do modelo de dados, é possível desenvolver serviços complementares aos existentes que desempenhem outro tipo de papel e disponibilizem funcionalidades com vista a aumentar o valor da solução.

Capítulo 6.
Metodologia e Plano de Trabalhos

Uma atividade importante e essencial da Engenharia de Software é a Gestão de Projeto, neste capítulo são detalhados os processos de gestão e de execução do projeto realizados com base no padrão previsto e detalhado no documento Modelo de Maturidade em Capacitação - Integração (MMCI, sigla original - CMMI, *Capability Maturity Model - Integration*). O documento é um modelo de referência que apresenta uma coleção de boas práticas nas diferentes áreas de engenharia e gestão, procurando melhorar e estabelecer um padrão único na execução de processos empresariais. O CMMI no modelo para desenvolvimento, apesenta um guia de boas práticas para uma organização que foca nos processos de desenvolvimento de produtos ou serviços.

O modelo prevê que as empresas sigam dois tipos de caminhos para a melhoria dos seus processos, continuamente ou por estágios. Na progressão contínua a empresa decide melhorar os processos de acordo com uma área de desenvolvimento especifica em que o nível da empresa é definido por níveis de capacidade. Na avaliação dos processos por estágios, o nível da empresa é estabelecido por níveis de maturidade, onde existem conjuntos de processos que vão sendo melhorados em lote de acordo com a área do processo. Para atingir um certo nível quer de maturidade, quer de capacidade é exigido que a empresa satisfaça os objetivos das áreas de processo daquele nível.

Neste caso é utilizada a versão 1.3 do CMMI para desenvolvimento numa progressão por níveis de maturidade em que os processos executados correspondem ao nível de maturidade 2 (ML2 – *Maturity Level 2*).

6.1. Metodologia

Foram utilizados métodos ou procedimentos para a realização das atividades que definem o ciclo de vida da arquitetura. As secções seguintes pretendem resumir o processo e documentar um aspeto importante no estágio no que toca à realização de experiências.

6.1.1. Modelo ACDM

Como referido anteriormente, o método a utilizar ao longo do estágio será o ACDM. Este método carateriza-se por ser dividido em duas partes, uma de incerteza e outra, logicamente, de certeza em relação ao sistema. Nas fases 1 e 2 concentra-se o esforço na recolha e estabilização dos requisitos funcionais, atributos de qualidade, restrições técnicas e de negócio. Já as fases 3, 4, 5 e 6 são dedicadas a desenvolver e avaliar a arquitetura do sistema. Durante esta fase, vão ser feitas experiencias para garantir que as restrições técnicas e de negócio, assim como os atributos de qualidade, são cumpridos. Por fim, nas fases 7 e 8, a arquitetura já estará bem definida e o nível de certeza é elevado. O foco nesta fase é o planeamento e a execução da arquitetura até aqui elaborada. A Figura 23 mostra os grupos de fases, as fases de desenho, revisão, experimentação e refinação da arquitetura, e ainda a descrição mais ramificada, os estágios.

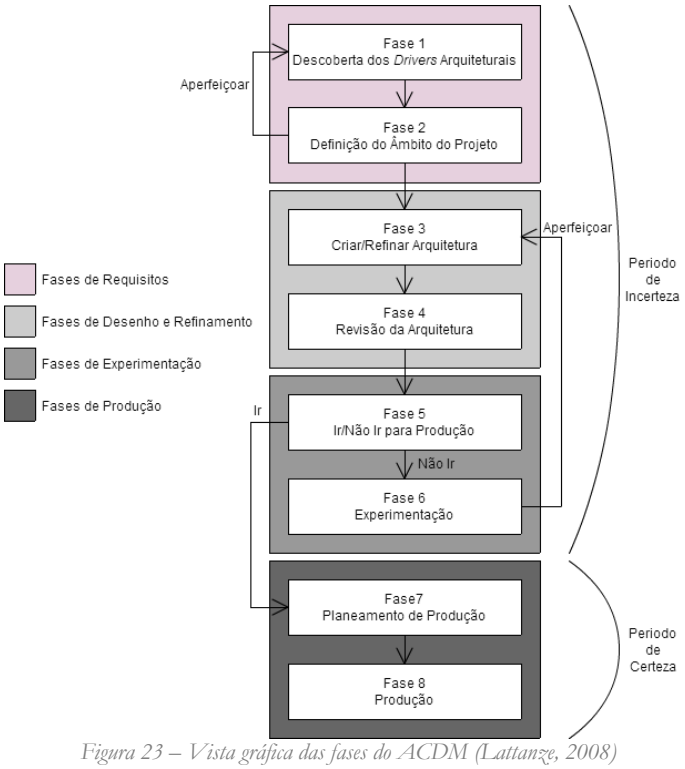


Figura 23 – Vista gráfica das fases do ACDM (Lattanze, 2008)

O propósito da fase 1 do ACDM é que a equipa realize reuniões com os interessados pelo projeto para descobrir e documentar os *driver's* arquiteturais que incluem requisitos funcionais de alto nível, restrições técnicas, restrições de negócio e atributos de qualidade. Para que as reuniões sejam produtivas e não uma perda de tempo para quem está presente (os *stakeholders)*, o ACDM providência ao gestor de projeto ideias e técnicas de estruturação da reunião, a que dá o nome de *workshop's*. Estas reuniões têm, normalmente, a seguinte sequência de acontecimentos: Introdução do sistema ou produto aos participantes; Recolha de contextos operacionais; Identificação de atributos de qualidade a garantir pela solução; Esclarecimento de restrições de negócio e restrições técnicas. E, o seu objetivo é recolher a maior quantidade de informação possível e relevante para o problema e evitar, ao máximo, discutir pormenores de "afinação", uma vez que isso só irá ser realizado na fase seguinte.

A escolha dos stakeholders, presentes nessas reuniões, deve ter em conta a sua experiência na área e necessidade que temos da mesma com vista à existência de um equilíbrio entre quantidade e qualidade de informação fornecida. Diferentes grupos de *stakeholders* têm diferentes necessidades e perspetivas do sistema e, consequentemente, os resultados das reuniões reproduziram essa diferença. No caso de serem efetuadas várias reuniões com os diferentes tipos de *stakeholders* é necessário que a equipa de arquitetura do projeto reúna, também, para consolidar a informação recolhida num só documento.

Na segunda fase, o ACDM prevê que sejam analisados os documentos resultantes da fase anterior, identificando e documentando os aspetos mais relevantes e que tragam valor para a solução. Esta atividade vai permitir que a equipa consiga identificar e ordenar os requisitos mais importantes da solução. Espera-se que sejam produzidos, nesta fase, documentos formais de *drivers* arquiteturais com diferentes níveis de visibilidade para os respetivos *stakeholders*. Estes documentos de especificação de requisitos devem definir e clarificar o âmbito do projeto e, consequentemente, a formalização dos requisitos inerentes ao sistema.

Realizadas as fases de grupo da elicitação de requisitos chega a altura em que o nível de certeza do projeto começa a crescer. Com os requisitos e âmbito do projeto definidos é, agora, possível começar a pensar e a elaborar uma solução arquitetural. As fases seguintes, como vamos

perceber, são fases pelas quais a definição de uma arquitetura passa pelo menos mais que uma vez. A fase 3 destina-se, então, à definição e construção da arquitetura da solução. Dependendo da fase em que estamos, seja a fase 2 ou a fase 6, existem objetivos diferentes para cada uma das situações. Se for a primeira vez que se estiver a passar nesta fase, o objetivo passa pela construção de uma arquitetura composta por componentes e conectores que satisfaçam os requisitos do cliente e que se organizem de forma a garantir que os atributos de qualidade são alcançados. Caso já exista uma arquitetura, ou seja na fase 6, existem documentos ou outros artefactos que indiquem as alterações necessárias a realizar à arquitetura até que a solução esteja pronta para produção.

A fase 4 é a fase mais importante do método do ACDM. É nesta etapa que a equipa analisa e reflete sobre os problemas da arquitetura. Tendo, então, como objetivo procurar, discutir e identificar os *issues* encontrados e avaliá-los segundo a Tabela 32.

Tabela 32 – Classificação de "issues" (Lattanze, 2008)

Classificação	Identificação	Descrição
1	Não necessita de ação	Quando um problema é identificado mas, não é necessário haver esforço adicional por parte da equipa
2	Necessita de reformular a documentação	Esta ação requer esforço por parte da equipa. No entanto, são problemas que podem ser resolvidos sem ser necessário o redesenho da arquitetura.
3	Necessita documentação técnica	Este tipo de *issues* são resolvidos com a leitura de livros ou artigos que comprovem que as decisões tomadas respondem acertadamente aos drivers arquiteturais.
4	Necessita de informação dos drivers arquiteturais	Esta ação envolve a interação com os stakeholders do projeto na medida em que é necessário recolher mais informação sobre um assunto relacionado com a especificação.
5	Necessita experimentação	Esta ação é necessária quando se acha necessário elaborar protótipos que garantam que o desenho da arquitetura responde como desejado.

É esperado que surjam dúvidas e que se sinta a necessidade de refazer de imediato certos aspetos da arquitetura, mas é importante, nesta fase, que a equipa de análise se foque apenas em encontrar problemas e incertezas, mais do que em encontrar soluções. Resultará, então, uma avaliação em formato de documento à arquitetura. Esse documento servirá de condutor e apoiará o trabalho realizado nas fases seguintes.

A fase 5 é aquela em que se toma a decisão de avançar com a solução elaborada para produção, ou nos apercebemos da necessidade de repetir novamente o ciclo de refinamento da arquitetura. A avaliação da decisão é auxiliada pelo documento produzido na fase 4. Cabe à equipa decidir quantos e quais os tipos de issues permitidos para que a arquitetura passe para a fase de produção. É aconselhável que os problemas existentes não sejam de nível superior a 3, pois isso pode causar incerteza na veracidade da solução e no comprometimento do arquiteto perante o projeto. Como referido anteriormente, é espectável que a arquitetura não vá para produção à primeira iteração passando assim para a fase 6.

Denominada de fase de experimentação, a fase 6 do modelo ACDM, tem como objetivo mitigar os *issues* identificados na fase 4. Nesta fase, a equipa deve ser dividida em equipas menores

e devem ser-lhes atribuídas tarefas que resolvam os problemas da arquitetura, isto é, que recolham informação necessária para garantir que na fase seguinte (novamente a fase 3) se consigam realizar as alterações suficientes e necessárias para alcançar a arquitetura desejada.

Depois de alguns ciclos de afinação da arquitetura, e alcançado o patamar de certeza desejado, a equipa pode tomar a decisão de avançar na fase 5 do modelo, passando, então, para as fases de produção da solução. Estas dividem-se em planeamento e produção, espera-se que a equipa de produção use os artefactos criados durante a fase de incerteza do ACDM a fim de ser produzida uma solução que possa ser validada pelo cliente. Para que esta validação seja possível a solução deve, logicamente, responder aos requisitos funcionais e com a qualidade exigida pelos drivers arquiteturais.

6.1.2. Arquitetura Conduzida pelos Riscos (*Risk Driven Architecture*)

Os riscos identificados, na Tabela 33, incidem na reutilização de código da solução atual do OneCare. Estes foram avaliados e considerados de nível 5 na escala de prioridades de avaliação, do modelo ACDM. As derivações que o método de trabalho tem relativamente ao ACDM standard é que a arquitetura é conduzida pelos riscos, ou seja, a arquitetura foca o crescimento ou a tomada de decisões os riscos identificados.

Tabela 33 – Tabela com identificação dos riscos da arquitetura

ID	Descrição	Probabilidade	Impacto
EXP01	O servidor *WildFly* (WF) pode não conseguir disponibilizar ao utilizador uma página no formato .xhtml gerada automaticamente em *runtime*	Média	Médio
EXP02	O servidor WF na versão 10 pode não suportar o código legado da solução devido a usar a CDI (*Contexts* and *Dependency Injection*), *Deltaspike*	Média	Médio
EXP03	Pode ser possível utilizar o *Discovery Service* da famosa Spring que reduzirá o tempo de desenvolvimento necessário ao projeto.	Alto	Alto
EXP04	É mais vantajosa a transformação do código legado da linha de produtos OneCare, desenvolvido em JBoss6, para WF10.	Média	Alto
EXP05	É necessário verificar se os mecanismos de *routing* dos servidores da empresa funcionam em WF10	Média	Médio

Foram realizadas experiencias para mitigar os riscos, e são apresentadas nas tabelas seguintes.

Tabela 34 – Tabela descritiva da experiência EXP01

Pré-Experimentação	
Titulo: WF10 gerar ficheiros .xhtml	**ID**: EXP01
Descrição: saber como aceder a páginas geradas dinamicamente no WF 10	**Responsável**: Mário Pereira
Deposição do problema:	

• Experimentação necessária
Propósito: pretende-se que seja esclarecido e garantido que o servidor consegue gerar ficheiros .xhtml e disponibiliza-los ao cliente.
Descrição da experiência: a experiência vai ser realizada no ambiente de desenvolvimento Eclipse; vai ser criada uma aplicação web disponibilizando persistência, controladores e vista, para simular uma estrutura de um serviço, para a vista recorre-se ao uso de Java Server Faces. A aplicação vai gerar um ficheiro .xhtml e disponibiliza-lo ao cliente. Vamos aproveitar para por à disposição uma API que servirá para receber e guardar um pedaço de código, do tipo *String*, de um serviço externo.
Artefactos criados: vai ser criado um serviço que disponibiliza uma API para que um serviço externo possa enviar pedaços de código, e ainda que gere um ficheiro .xhmtl com os pedaços de código guardados.
Critério de aceitação: a experiência dá-se por concluída quando for verificado o comportamento de interação com a API por parte de um serviço externo, e quando for possível o cliente visualizar a página gerada
Recursos necessários: computador, IDE Eclipse, Java EE, WF 10.0 (ultima versão estável à data)
Duração estimada e *milestones*: 30 horas para o objetivo total.
Pós-Experimentação
Sumário das descobertas: é possível, num ambiente composto por uma aplicação web a correr num servidor WF ver. 9, gerar páginas .XHTML dinamicamente e disponibiliza-las ao utilizador.
Duração real: 29h30
Recursos finais: nada a acrescentar aos recursos iniciais.
Recomendações: a solução é de certo modo rudimentar e pouco dinâmica, devem ser considerados métodos de *templating* que consigam aplicar algum dinamismo à solução, privilegiando o acesso à informação em vez da criação de elementos estáticos.

Tabela 35 – Tabela descritiva da experiência EXP02

Pre-Experimentação:	
Titulo: WF 9 com *DeltaSpike*	**ID**: EXP02
Descrição: risco do servidor WF 9 não suportar as funcionalidades do *Deltaspike*	**Responsável**: Mário Pereira
Deposição do problema: • Experimentação necessária	
Propósito: pretende-se que seja esclarecido e garantido que o servidor consegue fazer uso das componentes do *Deltaspike*, ou seja, se é possível reutilizar código da solução atual.	

Descrição da experiência: a experiência vai ser realizada no ambiente de desenvolvimento Eclipse; vai ser criada uma aplicação web disponibilizando persistência, controladores e vista, para simular uma estrutura de um serviço e ainda disponibilizadas componentes de CDI (*Deltaspike*) para a vista recorre-se ao uso de Java Server Faces. A aplicação vai ser um *helloworld* com notações do *Deltaspike* a correr no servidor WF. É necessário recorrer à documentação online do *Deltaspike* para saber como incluir a extensão à aplicação.
Artefactos criados: não é espetável que esta experiência gere artefactos para serem usados posteriormente. Restringindo-se apenas à verificação do funcionamento e integração de componentes.
Critério de aceitação: a experiência dá-se por concluída quando for testado e verificado o comportamento esperado de pelo menos uma anotação com sucesso.
Recursos necessários: computador, IDE *Eclipse*, Java EE, WF 10.0 (ultima versão estável à data), *Deltaspike* 1.5.2 (ultima versão à data)
Duração estimada e *milestones*: 20 horas para o objetivo total
Pós-experimentação
Sumário das descobertas: está garantido que o servidor WF na versão 10 consegue executar código com anotações de *Deltaspike* proveniente do sistema legado.
Duração total: 31h00
Recursos finais: nada a acrescentar dos recursos iniciais.
Recomendações: é recomendado que se procurem soluções mais recentes e não dependentes do *Deltaspike* por ser uma tecnologia ultrapassada evidenciado pelas funcionalidades que fornece. Em titulo de exemplo, não justifica usar uma CDI que permita mascarar Java EE 7 num projeto nativamente desenvolvido em Java EE 6, quando já estão a ser desenvolvidos serviços em Java EE 8.

Tabela 36 – Tabela descritiva da experiência EXP03

Pre-Experimentação:	
Titulo: *Discovery Service Spring*	**ID**: EXP03
Descrição: verificar se o *Discovery Service* da *Spring* pode ser aplicada ao projeto	**Responsável**: Mário Pereira
Deposição do problema: • Experimentação necessária	
Propósito: é de interesse do projeto verificar se a solução já existente providenciada pela Spring pode ser aplicada ao projeto uma vez que é uma implementação que vai ter de ser desenvolvida e neste caso já estaria pronta e testada.	
Descrição da experiência: a experiência vai ser realizada no ambiente de desenvolvimento Eclipse; vão ser seguidos tutoriais da plataforma Spring para que a curva de aprendizagem tenha tendência a ser mais acentuada em menos tempo. Vai ser testado um ambiente de	

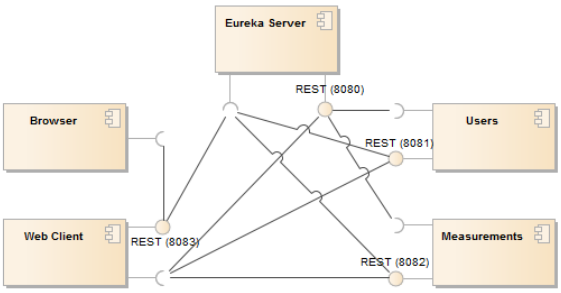
dois serviços que simularão fontes de dados, um serviço que funcionará como servidor *Discovery* e um outro que irá fazer a recolha dos serviços disponíveis.
Artefactos criados: os serviços gerados serão meramente para teste da tecnologia, assim, nem para protótipo são considerados.
Critério de aceitação: a experiência dá-se por concluída quando se conseguir obter a configuração e a passagem de dados da Figura 24.    *Figura 24 – Composição de serviços para a experiência EXP03*
Recursos necessários: computador, IDE *Eclipse*, Java EE, WF 10.0 (ultima versão estável à data), tutoriais *Spring*; Documentação online de *Spring*.
Duração estimada e *milestones*: 48 horas para o objetivo total
Pós-experimentação
Sumário das descobertas: foram desenvolvidos os serviços propostos pela experiência recorrendo aos tutoriais online da plataforma *Spring*. Foi testada a comunicação entre os serviços em várias máquinas. Os serviços foram duplicados e foi testado o balanceamento da carga entre eles. Os serviços tiveram o comportamento esperado, o que levou a querer que a solução poderia reduzir os custos de desenvolvimento. Contudo, a solução implica que sejam desenvolvidos serviços dependentes da tecnologia *Spring* e do sistema de *Discovery Eureka*, ou seja, é um risco adotar esta alternativa pois poderá limitar a solução em termos de tecnologias (*Spring* é baseado em Java) e funcionalidades.
Duração total: 64h00
Recursos finais: foi utilizado mais uma máquina para testar como o sistema reagia em termos de *routing* e escalabilidade.
Recomendações: é recomendado que o sistema *Discovery Eureka* não seja adotado uma vez que as limitações podem ser sobrepostas às vantagens. Além disso a confiança que se tem no produto já feito é obrigatoriamente baixa e piora aquando necessidade de implementar novas funcionalidades ou realizar alterações às existentes.

Tabela 37 – Tabela descritiva da experiência EXP04

Pre-Experimentação:	
Titulo: Passagem de JBoss6 para WF10	**ID**: EXP04
Descrição: tentativa de *refractoring* do código legado de JBoss6 para WF10	**Responsável**: Mário Pereira

Deposição do problema: • Experimentação necessária
Propósito: é necessário verificar se a transformação do código legado é possível de ser realizada no tempo disponível do estágio. Esta experiencia tem influência no desenrolar do estágio.
Descrição da experiência: esta experiência conta com a colaboração do Eng. AD e da colaboradora MPt pela experiencia e conhecimento do código legado. Vai ser utilizado o código da linha de produtos disponível na SVN da empresa recorrendo ao IDE *Eclipse*. Os projetos têm dependências entre si, pelo que a transformação tem de ser feita gradualmente dos modelos para as vistas. O objetivo é usar três jar's de modelo de dados que não necessitam de transformação e efetuar o refractoring e controlo de dependências de onze projetos.
Artefactos criados: os projetos resultantes desta experiencia vão ser incluídos como elementos externos na nova arquitetura.
Critério de aceitação: a experiência dá-se por concluída quando se conseguir fazer o *deploy* dos projetos por inteiro e o sistema ficar disponível através do servidor WF10.
Recursos necessários: computador, IDE *Eclipse*, Java EE, WF 10.0 (ultima versão estável à data), Dependências do projeto legado; Código dos projetos; Documentação dos projetos.
Duração estimada e *milestones*: 20 horas para o objetivo total
Pós-experimentação
Sumário das descobertas: a experiência foi levada a cabo por três intervenientes que tentaram transformar o código com sucesso, isto é, aceite sem qualquer erro por parte da ferramenta Eclipse. No entanto, o objetivo não foi alcançado, entre outros fatores, por não se conseguir que os projetos identificassem as dependências de projetos no servidor. A experiencia foi terminada ainda que inacabada pois estima-se que a sua conclusão necessita de recursos indisponíveis para o tempo disponível no presente estágio.
Duração total: atingidas as 48h, apesar de não se ter atingido o objetivo, foi dado por concluída a experiência.
Recursos finais: nada a acrescentar aos recursos iniciais
Recomendações: é recomendado que seja encontrado um mecanismo de comunicação entre o sistema antigo e o novo deixando de lado a possibilidade de transformar o código, como pretendido por esta experiencia, e que seja realizada uma adaptação progressiva das funcionalidades numa arquitetura alternativa que tenha como base a decomposição dos componentes mais bem definida e que sejam independentes entre si a fim desta situação não ser um problema no futuro.

Tabela 38 – Tabela descritiva da experiência EXP05

Pre-Experimentação:	
Titulo: Roteamento dos servidores legados	**ID**: EXP05

Descrição: verificar se o sistema de roteamento funciona com o *WilfFly10*	**Responsável**: Mário Pereira
Deposição do problema: • Experimentação necessária	
Propósito: é necessário verificar se o WF10 se comporta da mesma forma que o JBoss6 usando o modo de cluster usando o atual sistema de roteamento da Apache.	
Descrição da experiência: esta experiência vai ser realizada através do *deploy* de um serviço básico, já produzido para outras experiências, em dois servidores WF10 iniciados em modo cluster e um serviço de Apache de roteamento (httpd) com as configurações do servidor atual em produção. O objetivo é que os pedidos ao sistema sejam reencaminhados pelo serviço Apache para os servidores disponíveis. Os testes serão feitos com os dois ativos e com o primeiro em baixo, em ambos os casos o pedido tem de ser respondido.	
Artefactos criados: os projetos resultantes desta experiencia não são novos; pretende-se que as modificações dos ficheiros existentes de configuração do httpd sejam apenas em termos de ip's e portos de comunicação.	
Critério de aceitação: a experiência dá-se por concluída quando se conseguir que os testes pressupostos na descrição da experiencia sejam satisfeitos.	
Recursos necessários: computador; Java EE; *Wildfly* 10.0 (ultima versão estável à data); serviço da experiência EXP03; *Apache* httpd 2-2; configuração *Apache* httpd do servidor atual.	
Duração estimada e *milestones*: 16 horas para o objetivo total	
Pós-experimentação	
Sumário das descobertas: a experiência foi realizada como indicado na descrição, onde os testes foram realizados com sucesso. Ficou garantido que o serviço de roteamento atual comporta-se como esperado mesmo com a atualização do servidor.	
Duração total: 8h00 com os objetivos alcançados	
Recursos finais: nada a acrescentar aos recursos iniciais	
Recomendações: não existem recomendações de relevo provenientes desta experiencia além da confirmação do uso da tecnologia.	

A realização destas experiencias permitiu tomar decisões acertadas em relação ao desenho da arquitetura alternativa para suportar as necessidades do sistema OneCare.

6.2. Planeamento Detalhado

Para a elaboração de um planeamento de trabalhos estruturado, utilizou-se um processo de subdivisão das entregas e tarefas em componentes menores e mais fáceis de gerir. Em formato de árvore, uma Estrutura Analítica do Projeto (*WBS* – original: *Work Breakdown Schedule*) é uma ferramenta de controlo de tarefas com o objetivo de identificar o caminho crítico que uma equipa tem de percorrer para concluir o trabalho e completar o projeto.

Estes WBS foram discutidos e estruturados no início de cada semestre e com o desenrolar das atividades foram feitas as devidas alterações.

6.2.1. Primeiro Semestre

No início do estágio foi definido que o primeiro semestre iria ser dividido em duas grandes partes: uma que compreendia o período entre setembro e outubro; e outra de novembro a janeiro. Para a primeira fase estava prevista uma primeira versão do documento de requisitos onde fossem formalizados apenas os requisitos relativos à alteração da arquitetura. Já na segunda fase, tendo em conta a experiencia da Intellicare com utilização do ACDM e a carga horária que deveria cumprir, previa-se que fosse possível efetuar 4 a 6 iterações na especificação da arquitetura incluindo as experiências necessárias para documentar e provar que as decisões de arquitetura cumprem os requisitos necessários.

Assim, na Tabela 39 é apresentada um WBS com o plano macro inicialmente estabelecido para o 1º semestre de estágio onde são apresentados os parâmetros mais relevantes.

Tabela 39 – WBS das atividades planeadas do 1º semestre

Número WBS	Nome da Tarefa / Titulo	Inicio Planeado (m/d/a)	Final Planeado (m/d/a)	Dependências	Duração (dias)
1	Upgrade OneCare	09/14/2015	01/29/2016		100
1.1	Gestão de projeto	09/14/2015	01/29/2016		100
1.1.1	Planeamento	09/14/2015	09/14/2015		1
1.1.2	Monitorização	01/29/2016	01/29/2016		1
1.2	Relatório Intermédio	09/14/2015	01/29/2016		100
1.2.1	Capitulo 1 - Introdução	09/14/2015	09/21/2015		5
1.2.2	Capitulo 2 - Estado da Arte	09/21/2015	10/09/2015		14
1.2.3	Capitulo 3 - Requisitos	10/26/2015	10/30/2015		5
1.2.4	Capitulo 4 - Arquitetura	01/18/2016	01/29/2016		10
1.3	Documento de Requisitos	10/12/2015	10/30/2015		14
1.3.1	Reunião com stakeholders	10/12/2015	10/13/2015		1
1.3.2	Definição de Requisitos	10/13/2015	10/30/2015	1.3.1	13
1.4	Documento de Arquitetura	11/02/2015	01/29/2016		65
1.4.1	1ª Iteração	11/02/2015	11/18/2015		13
1.4.2	2ª Iteração	11/19/2015	12/07/2015	1.4.1	13
1.4.3	3ª Iteração	12/08/2015	12/24/2015	1.4.2	13
1.4.4	4ª Iteração	12/25/2015	01/12/2016	1.4.3	13
1.4.5	5ª Iteração	01/13/2016	01/29/2016	1.4.4	13

A Figura 25 traduz o WBS anterior num diagrama de *Gantt* que permite uma visualização temporal da organização das tarefas.

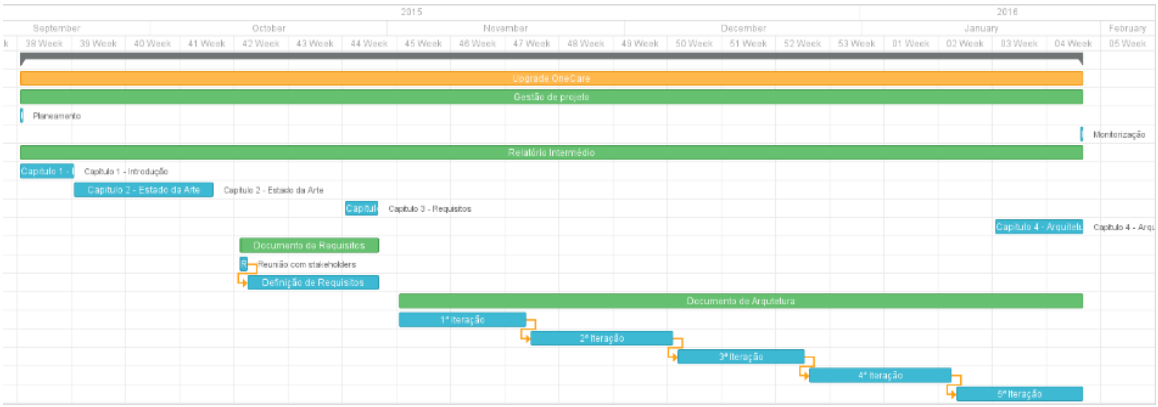


Figura 25 – Diagrama de Gantt da calendarização das tarefas planeadas para o 1º semestre

No presente estágio, depois de completada a definição dos requisitos da arqutetura, surgiu uma oportunidade de incorporar um novo produto/serviço à plataforma OneCare, o One-Care Call. Neste sentido, o eng. AD sugeriu que concentrasse esforços na elicitação de requisitos e atributos de qualidade desse serviço e, posteriormente, na análise do esforço de inclusão do produto na linha OneCare. A necessidade de validar a nova arquitetura esteve sempre presente pelo que o OneCare Call apareceu na altura certa tendo sido decidido usá-lo para efetuar a referida validação. Este processo é reportado no relatório pelo 5.1 - Projeto OneCare Call.

Posto isto, a Tabela 40 é apresentada um WBS com as tarefas realizadas durante o 1º semestre de estágio.

Tabela 40 – WBS das atividades realizadas do 1º semestre

Número WBS	Nome da Tarefa / Titulo	Inicio Planeado (m/d/a)	Final Planeado (m/d/a)	Dependências	Duração (dias)
1	Upgrade OneCare	09/14/2015	01/29/2016		100
1.1	Relatório Intermédio	09/14/2015	01/29/2016		100
1.1.1	Estado da Arte	09/14/2015	09/28/2015		11
1.1.1.1	Pesquisa de Artigos sobre MS	09/14/2015	09/22/2015		7
1.1.1.1.1	Leitura de Artigos	09/14/2015	09/18/2015		5
1.1.1.1.2	Visualização de apresentações	09/21/2015	09/22/2015		2
1.1.1.2	Escrita	09/23/2015	09/28/2015		4
1.1.2	Introdução	10/02/2015	10/05/2015		2
1.1.2.1	Inicio da escrita	10/02/2015	10/05/2015		2
1.1.3	Planeamento e Metodologia	10/20/2015	10/29/2015	1.1.2.3	8
1.1.3.1	Leitura livro sobre ACDM	10/20/2015	10/29/2015		8
1.1.4	Requisitos e Atributos de Qualidade	11/30/2015	12/01/2015	1.1.3	2
1.1.4.1	Escrita	11/30/2015	12/01/2015		2
1.1.5	Formulação Geral	01/04/2016	01/18/2016		11
1.1.6	Apresentação	01/25/2016	01/29/2016		5
1.2	Gestão de Projeto	09/14/2015	10/19/2015		26

1.2.1	Reunião AD	09/14/2015	09/14/2015		1
1.2.2	Reunião BC	10/01/2015	10/02/2015		1
1.2.3	Exploração ferramentas de gestão de projeto	10/01/2015	10/19/2015		13
1.2.3.1	Registo e exploração SCRAIM	10/01/2015	10/05/2015		3
1.2.3.2	Exploração e reporte de melhorias	10/06/2015	10/07/2015		2
1.2.3.3	Criação da folha de excel	10/19/2015	10/19/2015		1
1.2.4	Planeamento com AD	10/08/2015	10/09/2015		2
1.2.5	Preenchimento dos processos de qualidade	10/12/2015	10/15/2015		4
1.3	Documento de Requisitos	10/30/2015	11/27/2015		21
1.3.1	Análise arquitetura em produção	10/30/2015	11/02/2015		2
1.3.2	Reunião com stakeholders (OneCare Call)	11/03/2015	11/03/2015		1
1.3.3	Especificação de Requisitos (OneCare Call)	11/04/2015	11/23/2015	1.3.2	14
1.3.4	Documentação da integração com empresa subcontratada (OneCare Call)	11/20/2015	11/27/2015	1.3.3	6
1.4	Documento de Arquitetura	11/30/2015	12/28/2015		21
1.4.1	Estudo Arquitetura Atual	11/30/2015	12/01/2015		2
1.4.2	Experiência com WildFly e Deltaspike	12/02/2015	12/14/2015		8
1.4.3	Experiências com JSF	12/17/2015	12/28/2015		8

A Figura 26 traduz o WBS anterior num diagrama de *Gantt* que permite uma visualização temporal da organização das tarefas.

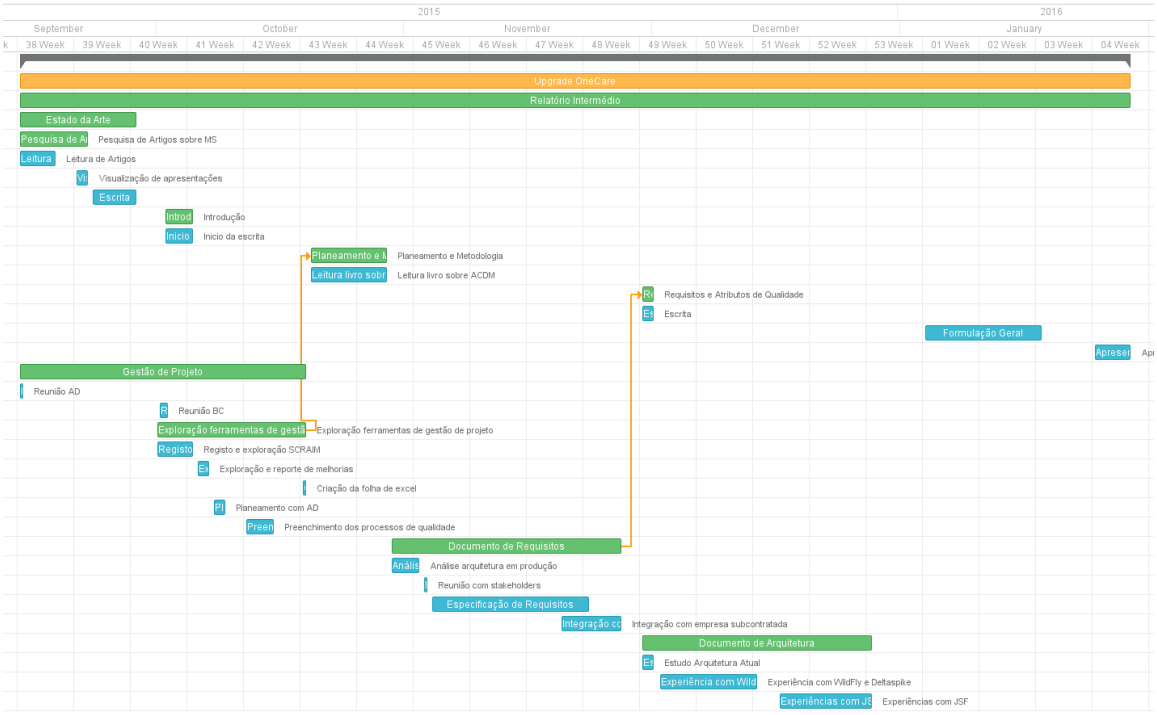


Figura 26 – Diagrama de Gantt da calendarização das tarefas realizadas no 1º semestre

Ao concluir os trabalhos do 1º semestre fico visível que havia uma dificuldade acrescida na elaboração do relatório de estágio e que esse era o maior risco para o cumprimento dos objetivos do estágio pois é necessário alocar mais tempo na sua redação o que teria impacto na realização das outras tarefas.

6.2.2. Segundo Semestre

O planeamento do 2º semestre teve em conta o risco identificado no final do 1º semestre, onde se previa alocar horas de trabalho ao desenvolvimento do relatório de estágio, ao longo do semestre e cada vez de forma mais acentuada, isto é, com o passar do tempo, o esforço dedicado à componente académica aumenta. Na Figura 27 é apresentada a distribuição de esforço pelo tipo de tarefa ao longo dos Sprints planeados.

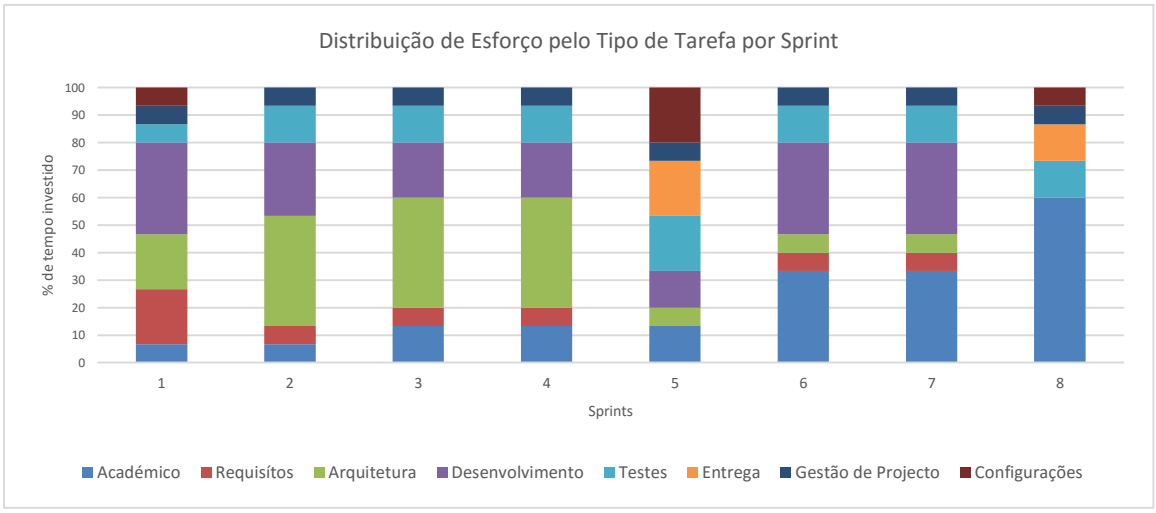


Figura 27 – Distribuição de Esforço pelo Tipo de Tarefa por Sprint

A ideia de planear o 2º semestre a partir de *Sprints*, ao contrario do primeiro semestre que o planeamento estava orientado aos documentos, teve o impacto de dividir as tarefas por blocos

com vista a definir um objetivo principal. Na Tabela 41 é apresentada uma tabela com os Sprints e seus objetivos a realizar durante o 2º semestre de estágio.

Tabela 41 – WBS das atividades planeadas do 2º semestre

Número WBS	Nome da Tarefa / Titulo	Inicio Planeado (m/d/a)	Final Planeado (m/d/a)	Dependências	Duração (dias)
1	Upgrade OneCare	09/14/2015	06/29/2016		100,44
1.1	Sprint 1 – Experiências	02/10/2016	02/26/2016		12,56
1.2	Sprint 2 – Implementação OC Call	02/26/2016	03/16/2016	1.2.1.1	12,56
1.3	Sprint 3 – Protótipo Arquitetura	03/16/2016	04/01/2016	1.2.1.2	12,56
1.4	Sprint 4 – Serviço de Permissões	04/01/2016	04/20/2016	1.2.1.3	12,56
1.5	Sprint 5 – Pré produção	04/20/2016	05/06/2016	1.2.1.4	12,56
1.6	Sprint 6 – Serviços bio marcadores	05/06/2016	05/25/2016	1.2.1.5	12,56
1.7	Sprint 7 – Serviços ambiente	05/25/2016	06/10/2016	1.2.1.6	12,56
1.8	Sprint 8 – Plataforma de Testes	06/10/2016	06/29/2016	1.2.1.7	12,56

A Figura 28 traduz o WBS anterior num diagrama de *Gantt* que permite uma visualização temporal da organização das tarefas.

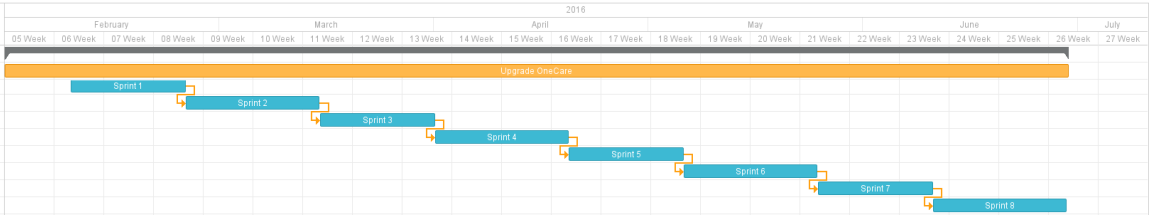


Figura 28 – Diagrama de Gantt da calendarização das tarefas planeadas para o 2º semestre

À semelhança do primeiro semestre, as tarefas planeadas sofreram alterações. O primeiro Sprint correu como planeado, o segundo foi divido em dois onde foi desenvolvido o protótipo do OneCare Call. No segundo Sprint o desenvolvimento foi feito externamente onde foram negociadas e testadas as funcionalidades da plataforma e no terceiro o desenvolvimento interno com a finalização dos requisitos e implementação. No quarto Sprint tentou-se compensar o tempo investido nos anteriores com a experiencia EX05. Com a estimativa do tempo que restara para os objetivos dos outros Sprints a colidir com a elaboração do relatório final, optou-se por realizar apenas mais um Sprint com o objetivo de finalizar o relatório. Ficando assim o desenvolvimento da documentação da arquitetura virada para o esquema do relatório. Na Tabela 42 é apresentado o WBS das atividades realizadas no segundo semestre.

Tabela 42 – WBS das atividades realizadas do 2º semestre

Número WBS	Nome da Tarefa / Titulo	Inicio Planeado (m/d/a)	Final Planeado (m/d/a)	Dependências	Duração (dias)
1	Upgrade OneCare	09/14/2015	01/29/2016		101
1.1	Sprint 1	02/10/2016	02/25/2016		12

1.1.1	Experiência OSGi	02/10/2016	02/11/2016		2
1.1.2	Experiência Spring Docker	02/12/2016	02/12/2016	1.1.1	1
1.1.3	Experiência com Spring	02/15/2016	02/25/2016	1.1.2	9
1.2	Sprint 2	02/26/2016	03/25/2016		21
1.2.1	Negociação da plataforma	02/26/2016	03/07/2016		7
1.2.2	Testes da plataforma	03/09/2016	03/25/2016	1.2.2	13
1.3	Sprint 3	03/28/2016	04/13/2016		13
1.3.1	Atributos de Qualidade OC Call	03/28/2016	03/30/2016		3
1.3.2	Desenvolvimento OC Call	03/31/2016	04/13/2016	1.3.1	10
1.4	Sprint 4	04/14/2016	04/28/2016		11
1.4.1	Documento Requisitos Arquitetura	04/14/2016	04/21/2016		6
1.4.2	Experiência Migração OC	04/22/2016	04/28/2016		5
1.5	Sprint 5	04/29/2016	06/29/2016		44
1.5.1	Relatório	04/29/2016	06/29/2016		44

E na Figura 29, a representação gráfica do WBS anterior como forma de diagrama de *Gantt*.

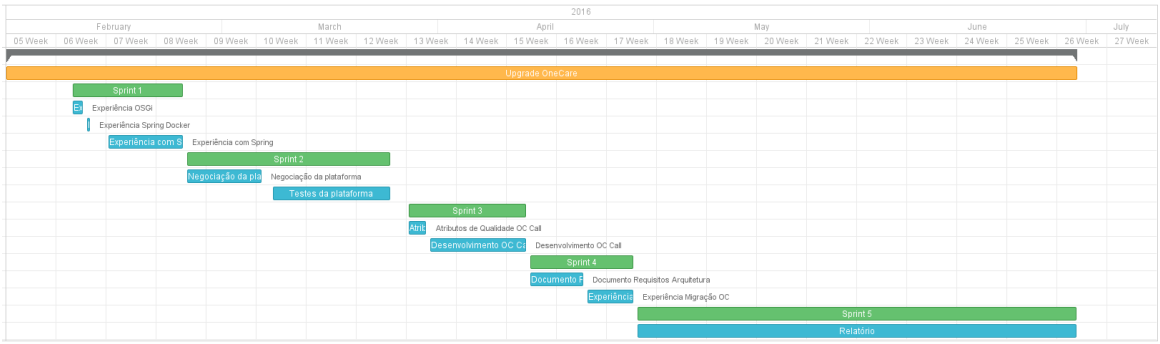


Figura 29 – Diagrama de Gantt da calendarização das tarefas realizadas no 2º semestre

Pelo fim do segundo semestre, a experiência ganha com o decorrer do estágio na empresa Intellicare é de que, o desenrolar das operações de uma empresa sofrem bastantes alterações. Neste caso, devido à dimensão reduzida da empresa e da necessidade de restruturação por falta de meios ou de novas oportunidades de negócio. Também fica a sensação de que por vezes a não realização de determinadas tarefas ou o desfasamento da sua duração face à estimativa, aconteceram devido ao estagiário fazer parte do plano operacional da empresa, e colaborar com ela, em certas tarefas que em nada tiveram a ver com o estágio. Esta observação é interpretada no bom sentido, na medida em que, os conhecimentos do estagiário foram aplicados nas tarefas do dia a dia da empresa, como se fosse um colaborador da mesma.

6.2.3. Ferramentas para Gestão de Tempo

A gestão de tempo nas diferentes fases do projeto foi feita através de uma folha de cálculo onde são registadas as horas planeadas e as executadas, através das quais serão produzidos gráficos representados abaixo. São apresentados gráficos referentes ao esforço cumulativo – gráfico onde se pode acompanhar as horas acumuladas do projeto – e, o outro, de planea-

mento versus execução – gráfico onde podemos analisar em que tipo de atividade foram investidas as horas executadas ao longo do estágio e dar uma ideia de como o tempo está a ser planeado e gerido.

Numa primeira fase o planeamento das horas foi realizado semanalmente e, mais tarde, quinzenalmente, de preferência no início da semana. O registo de horas executadas foi reportado diariamente na ferramenta *online* SCRAIM. A folha de cálculo é atualizada, no fim de cada ciclo de planeamento, segundo o relatório disponibilizado pela ferramenta SCRAIM com o número de horas realizadas, separadas por atividade.

Na Figura 30 é apresentado um gráfico que representa o esforço acumulado por semana ao longo do 1º semestre de trabalho.

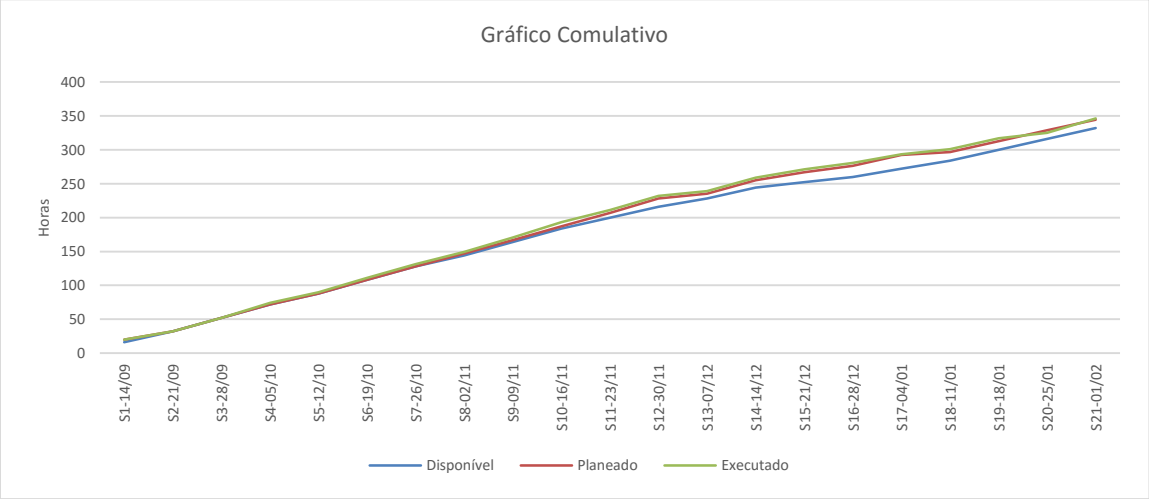


Figura 30 – Gráfico de esforço cumulativo do 1º semestre

A Figura 31 representa o esforço acumulado por semana ao longo do 2º semestre de trabalho.

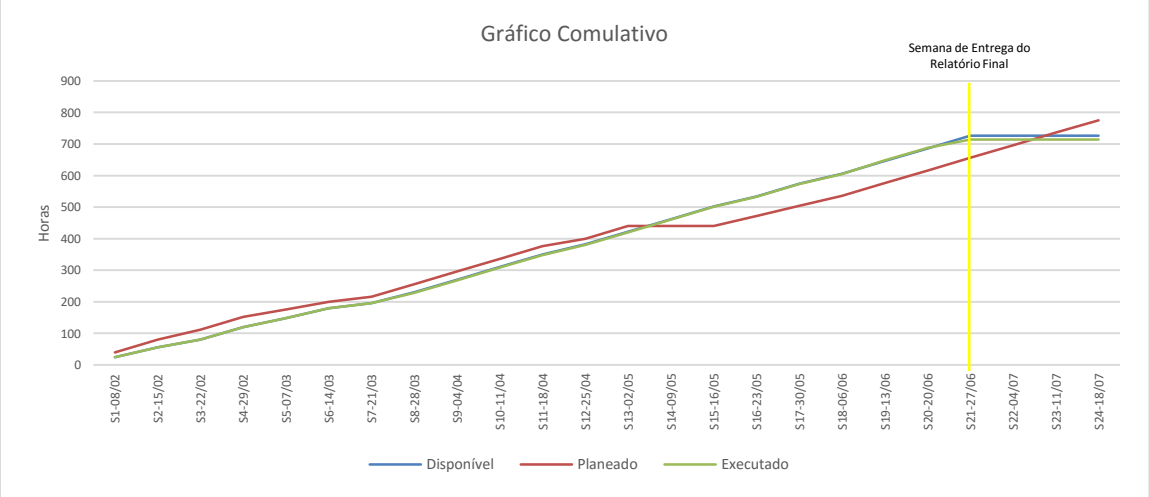


Figura 31 – Gráfico de esforço cumulativo do 2º semestre

A utilidade destes gráficos está em perceber a discrepância em termos de tempo de esforço disponível, planeado e executado. Sendo que o tempo disponível é calculado por "$\sum_{i=1}^{n}(5 - f_i) * 8$" onde *i* é o número da semana, *n* o número da semana a calcular o tempo cumulativo e f_i é o número de feriados na semana *i*. O tempo planeado é a acumulação das horas disponíveis pelo estagiário para a realização do estágio, e o tempo executado é a acumulação das horas realizadas e reportadas na ferramenta SCRAIM.

Uma das formas utilizadas para acompanhar o trabalho, foi colocar lado a lado o esforço aplicado a cada tarefa por semana, planeado e executado. Este objeto tem como propósito a monitorização e acompanhamento da estimativa de planeamento, comparando com a efetividade do esforço utilizado na realização das tarefas. Na Figura 32 é apresentado o gráfico de comparação entre o esforço planeado versus o executado distribuído por semanas.

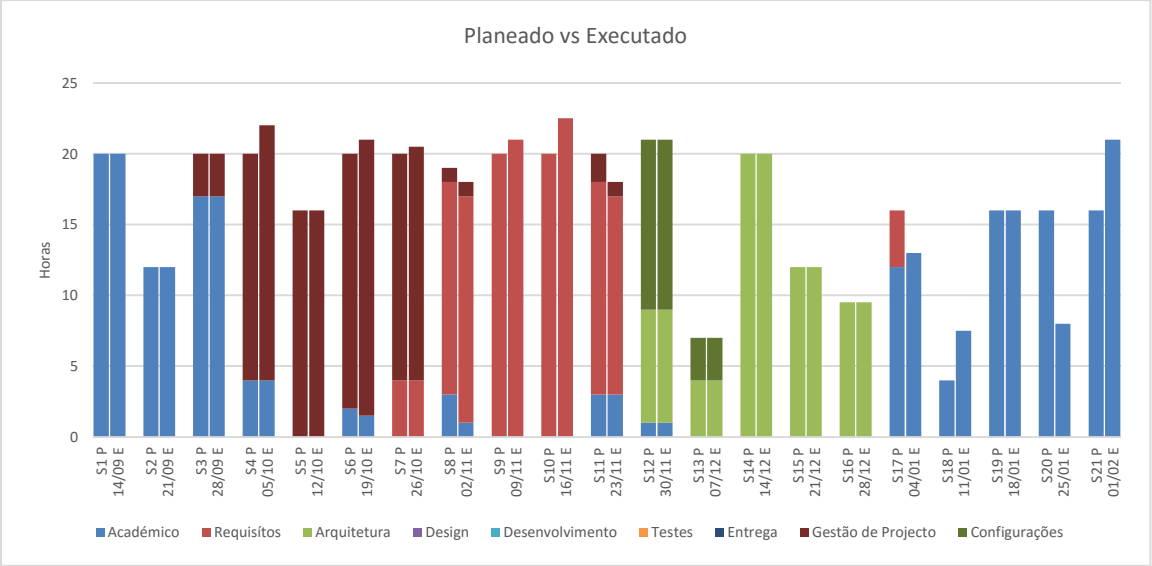


Figura 32 – Gráfico da distribuição de esforço por tipo de tarefa planeada e executada do 1º semestre

Durante o 1º semestre o planeamento das atividades foi realizado ao inicio da semana e monitorizado no final, daí a relação planeado/executado ser mais constante do que no segundo semestre como representado na Figura 33.

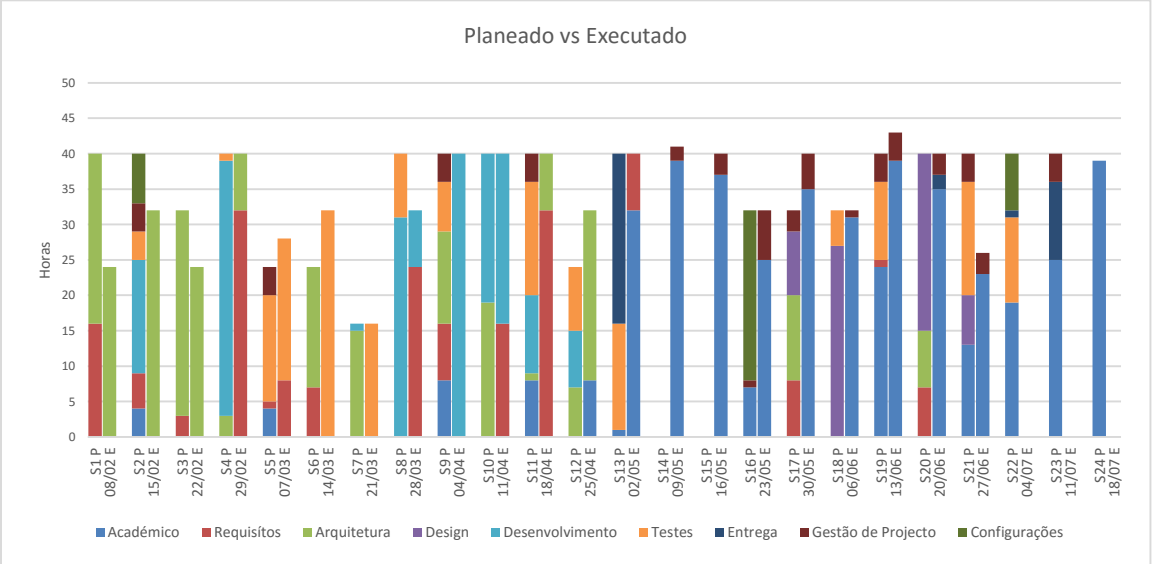


Figura 33 – Gráfico da distribuição de esforço por tipo de tarefa planeada e executada do 2º semestre

Desta vez, o planeamento teve por base a divisão de tarefas relativamente aos 8 sprints planeados já apresentados anteriormente na Figura 27, aos quais não foram feitas alterações ao longo do tempo. Resultando numa maior discrepância entre o planeado e executado a partir do quarto sprint. Este facto também é mais acentuado pela não granularidade das tarefas executadas, isto é, aquando a especificação de uma tarefa realizada, a separação do tipo de tarefa não foi feia com o rigor que o planeamento previa, daí haver uma heterogeneidade nas atividades das tarefas planeadas que não se verifica na sua execução.

6.3. CMMI-DEV Nível 2

O nível utilizado justifica-se sendo o nível em que a empresa está e deseja manter a certificação da sua qualidade de processos. Numa perspetiva global de desenvolvimento ou otimização de processos. o nível 2 assegura a gestão do projeto enquanto que o nível 3 trabalha a gestão da empresa. De seguida, são apresentadas as diferentes áreas de processos do segundo nível de maturidade, identificados os objetivos e as tarefas que as satisfazem e confrontado com o trabalho realizado durante o estágio. Neste nível existem sete áreas de processo enunciadas nas subsecções seguintes, dentro de cada área existem objetivos específicos (OE) e sugestão de prática especifica (PE) para os alcançar. Procura-se com as descrições seguintes identificar as atividades que satisfazem exigências do modelo, sendo que este serviu de guia para melhorar o acompanhamento das tarefas de estágio e foram objeto de suporte para os objetivos alcançados.

6.3.1. Planeamento de Projeto

O propósito desta atividade é estabelecer e gerir planos que definem as atividades do projeto e envolve atividades de desenvolvimento do plano de projeto, interação apropriada com as partes interessadas, criar comprometimento com o plano e dar manutenção às suas atividades. As atividades realizadas para justificar as PE estão representadas na Tabela 43.

Tabela 43 – Tabela de relação prática/execução relativo ao Planeamento de Projeto

OE1 – Definir e atualizar as estimativas dos parâmetros do planeamento do projeto	
PE1.1 – Definir um plano de trabalho macro para estimar o âmbito do projeto	Foi definido no inicio do 1º e 2º semestres, em reunião com AD, tendo em conta as horas e os recursos disponíveis.
PE1.2 – Definir e atualizar estimativas de trabalho e atribuição de tarefas	Atividade realizada através da plataforma SCRAIM, e monitorizado através do excel de gestão de tempo.
PE1.3 – Definir o ciclo de vida do projeto para projetar o esforço do planeamento	O ciclo de vida é restrito à especificação do estágio, tendo sido aplicado o processo técnico ACDM.
PE1.4 – Estimar o esforço do projeto e das tarefas baseado na estimação racional	Foi usado um modelo de custo fixo uma vez que o esforço disponível está definido pelo estágio. Foram adaptados os objetivos ao longo do estágio para responder a esse custo fixo.
OE2 – O plano de projeto é definido e atualizado como uma tarefa base na gestão de projeto	
PE2.1 – Definir e atualizar o orçamento e a calendarização do projeto	O orçamento do projeto é definido pelo tempo e esforço do estágio.
PE2.2 – Identificar e analisar os riscos do projeto	Foi realizada a identificação e análise dos riscos iniciais do projeto e documentados no documento de qualidade IP05 disponível em anexo externo.
PE2.3 – Planear e a gestão de dados do projeto	Atividade realizada através da plataforma SCRAIM e consolidada na folha de excel de gestão de tempo.

PE2.4 – Planear os recursos para concluir o projeto	O projeto tem recursos fixos, tendo sido efetuado o planeamento dos mesmos.
PE2.5 – Planear o conhecimento e as capacidades necessárias para concluir o projeto	Foram realizadas atividades de estudo sempre que identificada a limitação para a conclusão de um objetivo.
PE2.6 – Planear o envolvimento das partes interessadas	Foram realizadas reuniões periódicas com AD.
PE2.7 – Definir e atualizar o plano geral do projeto	Foram realizadas reuniões periódicas com AD e BC. O progresso do trabalho foi monitorizado através da plataforma SCRAIM para as tarefas, e através da folha de excel para gestão de tempo para os recursos.
OE3 – São definidos e atualizados os comprometimentos com o plano do projeto	
PE3.1 – Rever os planos que afetam o projeto para perceber o eventual comprometimento do mesmo	Decorreram atividades de avaliação, nomeadamente a avaliação intermédia, que exigiu um replaneamento dos objetivos, foco e metodologia do projeto.
PE3.2 – Ajustar o plano do projeto para conciliar os recursos estimados com os disponíveis	Foi feito um replaneamento do projeto aquando a identificação da limitação de recursos que implicaram o aumento do custo em certo tipos de atividades.
PE3.3 – Obter comprometimento de partes interessadas relevantes para suporte e realização do plano	Foram consultados os orientadores de estágio através de reuniões periódicas.

6.3.2. Gestão de Requisitos

O propósito deste conjunto de processos é gerir os requisitos dos componentes e dos produtos do projeto e garantir o alinhamento dos mesmos com o plano do projeto. Os processos desta área estão previstos no ACDM e a descrição do modelo utilizado é descrita e especificada no anexo "Modelo ACDM". As atividades realizadas para justificar as PE estão representadas na Tabela 44.

Tabela 44 – Tabela de relação prática/execução relativo à Gestão de Requisitos

OE1 – Os requisitos são geridos e as inconsistências com o plano do projeto são identificados	
PE1.1 – Desenvolver e perceber com os produtores de requisitos estratégias de comunicação de requisitos	Atividade prevista e realizada por estar prevista no modelo ACDM fases um e dois.
PE1.2 – Obter o comprometimento dos participantes no projeto sobre os requisitos	Foram realizadas reuniões periódicas com o AD e BC.
PE1.3 – Gerir as alterações de requisitos à medida que estes evoluem com o tempo	Foram elaboradas alterações ao documento de requisitos do protótipo OneCare Call à medida que foram identificados novos requisitos ou restrições através da negociação com a entidade externa.

PE1.4 – Manter uma conexão bidirecional entre requisitos e trabalho executado	Não foi realizado por não ser disponibilizada por parte da empresa a ferramenta *Enterprise Architect,* uma vez que o número de licenças da aplicação não o permitia.
PE1.5 – Assegurar que os planos do projeto e o trabalho realizado vão de encontro com os requisitos	Foi avaliado o desenvolvimento do protótipo em função com o documento de requisitos.

6.3.3. Gestão de Acordos com Fornecedores

O âmbito desta área de processo é responsável pela aquisição de produtos ou serviços que acrescentem valor ao projeto. O propósito da gestão de acordos com fornecedores é gerir a aquisição de produtos e serviços dos fornecedores. Estes contratos são referentes a modificações do orçamento que representem um risco significante para o projeto. As atividades realizadas para justificar as PE estão representadas na Tabela 45.

Para fazer face à necessidade de incluir um serviço fora do contexto e domínio da empresa, tomou-se a decisão de subcontratar serviços externos. Para este processo usou-se a Gestão de Acordos com Fornecedores segundo CMMI para Desenvolvimento (CMMI Product Team, 2010). O processo prevê: planeamento, acordo, monitorização e aceitação dos serviços contratados.

Tabela 45 – Tabela de relação prática/execução relativo à Gestão de Acordos com Fornecedores

OE1 – Acordos com fornecedores são estabelecidos e geridos	
PE1.1 – Determinar o tipo de aquisição para cada produto ou componente adquirido	A aquisição de serviço perante a empresa subcontratada foi de um serviço PaaS.
PE1.2 – Selecionar fornecedores baseado na avaliação das capacidades de encontro com requisitos específicos e critérios estabelecidos	Atividade realizada através do processo de aprovisionamento da empresa que está certificado em termos de ISO 9001.
PE1.3 – Estabelecer e gerir os acordos realizados	Para estabelecer o acordo com o fornecedor, foram feitas reuniões e produzido o documento *OCC RiquerementsSpecification v1.0.pdf* disponível em anexo externo. A proposta de trabalho surge do documento de requisitos em formato Javadoc com a especificação dos métodos da interação entre sistemas. Trocas de emails com condições comerciais
OE2 – Acordos com fornecedores são cumpridos por ambos os lados, quer do projeto quer do fornecedor	
PE2.1 – Realizar as atividades com os fornecedores como especificado nos acordos	Foram realizadas reuniões e trocas de email com o responsável do fornecedor.
PE2.2 – Garantir que o acordo é cumprido antes de aceitar o produto adquirido	Foram realizados testes de ambos os lados enquanto o desenvolvimento do produto.
PE2.3 – Garantir a receção dos produtos adquiridos	N/A. O produto não foi concluído até à data.

6.3.4. Monitorização e Controlo de Projeto

O propósito da monitorização e controlo do projeto é providenciar ferramentas de controlo do progresso do projeto para que possam ser tomadas ações corretivas apropriadas quando o desempenho do projeto desviar significativamente do planeado. Um plano de projeto é um artefacto base da monitorização de um projeto, comunica com os interessados no projeto o estado atual do projeto ao longo da sua execução pela verificação da relação entre o trabalho executado e o estimado. É através do plano que se tomam as ações assertivas quando essa relação é significativamente negativa. Essa significância mede-se através do diagnóstico de não ser possível cumprir os objetivos e metas definidas. As atividades realizadas para justificar as PE estão representadas na Tabela 46.

Tabela 46 – Tabela de relação prática/execução relativo à Monitorização e Controlo de Projeto

OE1 – É gerido o estado atual do projeto em termos de desempenho e progresso em relação ao plano do projeto	
PE1.1 – Monitorização dos parâmetros atuais do planeamento do projeto de acordo com o plano de projeto	Realizado através do controlo de tarefas e horas estimadas.
PE1.2 – Monitorizar os compromissos em relação ao previsto no plano de projeto	Realizado através de reuniões periódicas com AD
PE1.3 – Monitorizar riscos de encontro com os identificados no plano do projeto	Realizado de acordo com o prescrito pelo processo técnico do ACDM
PE1.4 – Monitorizar a gestão da informação de acordo com o plano de projeto	Estabelecidos pela empresa modelos de documentos e repositórios para a sua alocação e divulgação.
PE1.5 – Monitorizar o envolvimento das pessoas interessadas de acordo com o plano de projeto	N/A.
PE1.6 – Rever periodicamente o progresso e os problemas do projeto	Foram realizadas reuniões com BC e AD .
PE1.7 – Rever as metas alcançadas e os resultados do projeto na sua fase atual	Foram realizadas reuniões com BC e AD .
OE2 – Ações corretivas são executadas e geridas quando o desempenho do projeto desvia significativamente do planeado	
PE2.1 – Recolher e analisar problemas e determinar ações para os corrigir	Após a recolha e identificação de problemas foram realizadas reuniões com AD ou BC para determinar as ações corretivas que melhor se adaptavam aos recursos disponíveis e ao estágio em si.
PE2.2 – Executar ações corretivas nos problemas identificados	Ao aplicar as medidas corretivas, foram novamente estimadas as tarefas e aplicadas ao planeamento.
PE2.3 – Gerir de perto as ações tomadas	O plano atualizado foi novamente gerido pelas atividades desta área de processo.

6.3.5. Controlo de Qualidade de Processos e Produto

O propósito do controlo de qualidade de processos e produto é providenciar a gestão a uma equipa com a visão objetiva dos processos e do trabalho associado. Os processos desta área focam no objetivo de alcançar produtos e serviços produzidos com alta qualidade fornecendo à equipa ferramentas de controlo de qualidade de processo ao longo do ciclo de vida do projeto. As atividades realizadas para justificar as PE estão representadas na Tabela 47.

Tabela 47 – Tabela de relação prática/execução relativo ao Controlo de Qualidade de Processos e Produtos

OE1 – A adesão dos processos e produtos executados em relação ás descrições de processo, padrões e procedimentos são objetivamente avaliados.	
PE1.1 – Avaliar os processos de execução selecionados de acordo com os processos, padrões e procedimentos aplicáveis.	O processo de qualidade da empresa prevê uma auditoria formal ao projeto em fases posteriores às atingidas. Foi efetuado um *self-assessment* ao cumprimento dos processos.
PE1.2 – Avaliar o trabalho realizado de acordo com os processos, padrões e procedimentos aplicáveis.	O material produzido foi revisto e avaliado pelas partes interessadas da empresa.
OE2 – Os problemas ao longo do projeto são identificados e são tomadas medidas de correção.	
PE2.1 – Comunicar os problemas identificados e assegurar a resolução perante a equipa e os gestores.	A tarefa foi realizada recorrendo à ferramenta SCRAIM.
PE2.2 – Estabelecer e manter registo das atividades de controlo de qualidade.	A tarefa foi realizada recorrendo à ferramenta SCRAIM.

6.3.6. Gestão de Configurações

O propósito da gestão de configurações é estabelecer e manter a integridade do material produzido usando identificadores, controlo, auditorias e acompanhar os estados das configurações. As atividades realizadas para justificar as PE estão representadas na Tabela 48.

Tabela 48 – Tabela de relação prática/execução relativo à Gestão de Configurações

OE1 – São estabelecidas as *baselines* do material produzido	
PE1.1 – Identificar os itens e os componentes do material produzido para ser monitorizado.	Realizado segundo as práticas da empresa. Tendo sido colocado no sistema os documentos de requisitos, arquitetura e os artefactos de código.
PE1.2 – Estabelecer e manter o sistema de gestão de configurações assim como a própria gestão para controlar o material produzido	Atividade assegurada pela utilização do repositório da empresa.
PE1.3 – Criar patamares para uso interno e entrega ao cliente.	Atividade realizada apenas para especificação do OneCare Call.
OE2 – Alterações ao material produzido controlado é identificado e gerido	

PE2.1 – Identificar pedidos de alterações aos itens geridos.	O repositório da empresa garante o controlo de versões dos documentos.
PE2.2 – Controlar as alterações dos itens geridos.	Na atualização dos documentos no repositório é prática da empresa deixar o comentário da submissão.
OE3 – A integridade das *baselines* é estabelecida e gerida	
PE3.1 – Estabelecer e gerir registos que descrevem o material produzido que e gerido.	N/A. Não foram formalmente definidas *baselines*.
PE3.2 – Realizar auditoria às configurações para manter a integridade das *baselines*.	N/A. Não foram formalmente definidas *baselines*.

6.3.7. Medidas e Análise

O propósito da área de processo de medidas e análise é desenvolver e gerir a capacidade de medida para dar suporte às necessidades de gestão de informação. As atividades realizadas para justificar as PE estão representadas na Tabela 49.

Tabela 49 – Tabela de relação prática/execução relativo às Medidas e Análise

OE1 – As medidas dos objetivos e das atividades estão alinhadas com os objetivos e as necessidades identificadas.	
PE1.1 – Estabelecer e gerir objetivos de medidas derivados dos objetivos e necessidades identificados.	Atividade realizada com AD.
PE1.2 – Especificar medidas para relacionar os objetivos de medição	Número de horas por tipo de tarefa.
PE1.3 – Especificar como são obtidas e geridas as informações sobre medidas.	O controlo é feito pelo registo de horas de trabalho na ferramenta SCRAIM.
PE1.4 – Especificar como é que as medidas são analisadas e comunicadas.	Os dados são retirados da ferramenta SCRAIM e analisados através de gráficos na folha de Excel de gestão de tempo.
OE2 – São providenciados resultados de medição provenientes das necessidades e objetivos do projeto	
PE2.1 – Obter informação de medidas especificas	Os dados são retirados como forma de relatório da ferramenta SCRAIM.
PE2.2 – Analisar e interpretar a informação de medidas especificas	É analisado periodicamente a evolução das tarefas e do esforço realizado através da folha de Excel de gestão de tempo.
PE2.3 – Gerir e guardar informação e especificação de medidas e análises de resultados	Os dados são guardados na folha de Excel de gestão de tempo.
PE2.4 – Comunicar os resultados da medição e atividades de análise a todas as partes interessadas do projeto.	Foram realizadas reuniões periódicas com AD e reportadas no presente relatório.

Capítulo 7.
Conclusões

O desafio de encontrar uma arquitetura alternativa para a linha de produtos OneCare que permita agilizar as mudanças resultantes de novos desafios tecnológicos ou de negócio foi resolvido. Com a aplicação do OneCare Call, ficou provado que uma arquitetura de micro serviços é uma solução para os problemas identificados no início do estágio, pelo que a solução encontrada permite a migração gradual do produto.

Para estruturar o trabalho do projeto foi seguido o processo técnico ACDM. Durante as fases 1 e 2 foi identificado o problema, recolhidas as necessidades do sistema atual e criada documentação de requisitos para a linha de produtos OneCare e para o produto OneCare Call. Durante as fases 3 a 6, foi efetuado o estudo dos padrões de arquitetura e das soluções com mais semelhanças, Oracle IoT *reference architecture*, AAL4ALL, FIWARE e ainda a arquitetura atual do produto. Foi criada iterativamente a nova versão da solução e no final o produto OneCare Call que permitiu validar a nova arquitetura ao satisfazer os drivers arquiteturais.

A utilização de micro serviços faz sentido para a linha de produtos OneCare, pela partilha de informação entre soluções, tais como dados de utilizadores; pela necessidade de certificar serviços independentes, por exemplo os serviços considerados dispositivos médicos de software; e para agilizar a inclusão de novas pessoas no desenvolvimento do sistema ao reduzir a complexidade dos serviços, passando a beneficiar da modificabilidade, testabilidade, portabilidade e interoperabilidade que a nova arquitetura oferece.

Estando tecnicamente provado que a solução encontrada resolve o problema, a Intellicare terá de decidir se o valor obtido com a migração da solução é vantajoso em termos de negócio a curto e longo prazo. Será necessário desacoplar os diferentes serviços dos produtos do OneCare e progressivamente migrar o sistema para a arquitetura de micro serviços. É uma alteração à arquitetura e ao método de desenvolvimento de serviços, em os obstáculos podem ser a resistência ao modelo de trabalho que é imposto pela arquitetura e a necessidade de manter transitoriamente uma estrutura de servidores mais complexa.

No decorrer do projeto foram seguidas boas práticas associadas ao desenvolvimento de software e fundamentados por processos e padrões. As boas práticas foram identificadas e avaliadas de acordo com o *CMMI Dev* 1.3, nível 2. Os processos usados foram os em vigor na Intellicare e partilhados com VPS, em especial os de controlo de qualidade e relação com fornecedores e o processo técnico ACDM. Em termos de utilização de standards, foram usados REST e JSON para garantir de interoperabilidade, e encriptação para garantir confidencialidade das comunicações.

Do ponto de vista pessoal, ganhei novas capacidades pelo contacto com o mundo profissional pelo desenrolar do projeto e consegui aplicar a formação adquirida ao longo do curso de Engenharia Informática. Um facto que me deixa seguro de que posso ser um bom profissional é ter conseguido realizar tarefas expostas a um processo imposto pela empresa, e ter conseguido integrar-me bem com a equipa de trabalho.

Vou melhor preparado enquanto Engenheiro, e a sensação de que fico é que quero fazer carreira numa profissão de constante atualização com as novas tecnologias. O que ambiciono,

é realizar um trabalho a gerir equipas e realizar o trabalho de arquitetura de software, onde a minha vontade é alargar a minha cultura em termos de sistemas de informação e acompanhar a evolução da tecnologia, procurando aprender sobre as novas soluções no mercado para poder tomar melhores decisões na escolha das tecnologias dos projetos futuros.

Tenho confiança no resultado e na qualidade e do trabalho produzido consolidados pela formação adquirida.

Capítulo 8.
Referências

Abbott, M. L., & Fisher, M. T. (2009). The art of scalability: Scalable web architecture, processes, and organizations for the modern enterprise. Pearson Education.

Alagarasan, V. (2015). Seven Microservices Anti-patterns. Retirado de: http://www.infoq.com/articles/seven-uservices-antipatterns.

CMMI Product Team. (2010) "CMMI for Development", version 1.3, Software. Carnegie Mellon, 363-372.

Conway, M. E. (1968). How do committees invent. Datamation, 14(4), 28-31.

Cubo, J., Nieto, A., & Pimentel, E. (2014). A cloud-based Internet of Things platform for ambient assisted living. Sensors, 14(8), 14070-14105.

Dohr, A., Modre-Osprian, R., Drobics, M., Hayn, D., & Schreier, G. (2010). The Internet of Things for Ambient Assisted Living. ITNG, 10, 804-809.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645-1660.

Jorgensen, P. C. (2008). Software testing: a craftsman's approach. CRC press.

Lattanze, A. J. (2008). Architecting Software Intensive Systems: A Practitioners Guide. CRC Press.

Lewis, J. & Fowler, M. (2014). Microservices: a definition of this new architectural term. Retirado de: http://www.martinfowler.com/articles/microservices.html.

Namiot, D., & Sneps-Sneppe, M. (2014). On micro-services architecture. International Journal of Open Information Technologies, 2(9), 24-27.

Newman, S. (2015). Building Microservices. " O'Reilly Media, Inc.".

Oracle (2014). Internet of Things: Role of Oracle Fusion Middleware.

Ramparany, F., Marquez, F. G., Soriano, J., & Elsaleh, T. (2014, October). Handling smart environment devices, data and services at the semantic level with the FI-WARE core platform. In Big Data (Big Data), 2014 IEEE International Conference on (pp. 14-20). IEEE.

Richardson, C. (2014). Microservice architecture patterns and best practices. Retirado de: http://www.microservices.io/.

Rogers, Rich. (2005). Reuse engineering for SOA. Retirado de: http://www.ibm.com/developerworks/webservices/library/ws-reuse-soa/index.html.

Rumbaugh, J., Jacobson, I., & Booch, G. (2004). Unified Modeling Language Reference Manual, The. Pearson Higher Education.

Pereira, A., Machado, R. J., Fernandes, J. E., Teixeira, J., Santos, N., & Lima, A. (2014). Using the NIST reference model for refining logical architectures. In Computational Science and Its Applications–ICCSA 2014 (pp. 185-199). Springer International Publishing.

Preston-Werner, T. (s.d.). Semantic Versioning 2.0.0. Retirado de: http://semver.org/.

SmartBear Software (2015) Why You Can't Talk About Microservices Without Mentioning Netflix. Retirado de: http://blog.smartbear.com/microservices/why-you-cant-talk-about-microservices-without-mentioning-netflix/

Vogels, W. (2009). Eventually consistent. Communications of the Acm, 52(1), 40–44. http://doi.org/10.1145/1435417.1435432

Michelson, B. M. (2006). Event-driven architecture overview. *Patricia Seybold Group, 2*.

Capítulo 9.
ANEXOS

9.1. Descrição dos casos de uso legados da linha de produtos OneCare

Tabela 50 – Descrições dos casos de uso legados

ID	Descrição
UC01	Os utilizadores ou agentes de comunicação do OneCare deverão aceder ao sistema apenas após a introdução de um login e password. Na versão 3.0 será necessário integrar o sistema de autenticação do OneCareMais-Perto com o sistema de autenticação utilizado para a receção de dados do dispositivo Android. Na fase 3.3 será necessário incluir a verificação da validação do e-mail do utente. Na fase 3.5 será necessário incluir uma chamada da validação do pin de um utente.
UC02	Os utilizadores configuram os equipamentos de notificação de alertas que comunicam com o sistema. Na fase 3.4 será necessário adicionar uma interface simplificada para utentes finais poderem associar e configurar equipamentos de notificação de localização.
UC03	Os utilizadores configuram os equipamentos de medições da saúde.
UC04	Os utilizadores configuram os valores de disparo dos alertas de saúde. Os limites de alerta a serem contemplados correspondem a verificações simples de ultrapassagem de um valor mínimo ou máximo.
UC05	O utilizador configura a informação básica da pessoa a ser monitorizada no dispositivo. Este caso de uso irá tornar-se obsoleto na fase 3.2 quando o sistema permitir a obtenção de pessoas monitorizadas do servidor.
UC06	O utilizador configura as horas limites da receção das provas de vida.
UC07	O utilizador configura os utentes do sistema. Na fase 3.3 será necessário incluir a validação e reenvio do número de telemóvel utente.
UC08	O utente efetua uma medição com o dispositivo e após a sua receção e seleção da pessoa monitorizada a que corresponde o sistema envia-a para o servidor. A medição deverá poder ser realizada tanto com a aplicação aberta como com a adição de um widget ao homescreen Android. Na versão 3.8 será adicionado à aplicação Android atual o processamento das medições de temperatura.
UC09	O sistema envia mensagens de maior prioridade caso um alerta esteja sem tratamento mais tempo do que o devido.
UC10	O sistema insere alertas de falha de prova de vida caso não sejam recebidos contactos dentro dos intervalos limites configurados nas provas de vida.
UC11	O sistema insere alertas de saúde caso os valores armazenados nos sistemas ultrapassem os limites configurados.

UC12	O sistema obtém, do Portal da Saúde, os dados dos utentes configurados para este efeito.
UC13	O sistema processa as notificações recebidas dos dispositivos de notificação de localização.
UC14	O sistema publica interna e/ou externamente os dados recebidos. A publicação dos dados deverá depender das configurações do utente. Na versão 3.0 será necessário adicionar a publicação dos dados internamente. Na versão 3.5 será necessário adicionar a publicação dos dados no portal da saúde.
UC15	O sistema recebe e armazena os dados originados tanto na aplicação, como dados originados no "Router OneCareV1". Na versão 3.2 será necessário adicionar processar medições utilizando a identificação dos utentes em vez do botão. Na versão 3.8 será necessário adicionar a receção de dados de temperaturas.
UC16	O sistema recebe as notificações enviadas pelos dispositivos de notificação de localização.
UC17	O utilizador acede aos alertas e faz o tratamento do seu estado.
UC18	O utilizador acede aos alertas e faz o tratamento do seu estado.
UC19	O sistema verifica as medições de saúde armazenadas desde a última vez que correu.
UC20	O sistema verifica as notificações inseridas desde a última vez que correu.
UC21	O utilizador visualiza os dados biométricos armazenados no sistema. A visualização pode ser feita em gráfico e/ou tabela.
UC22	Os sistemas externos que precisam de obter dados (como o Giraff+) obtêm a informação armazenada e podem obter gráficos com esta informação.
UC23	O sistema envia as mensagens necessárias para a configuração dos equipamentos.
UC24	O utilizador configura as operações a realizar quando os utentes pressionarem nos botões do equipamento. Para cada botão: · Alarme; · Vermelho; · Verde; · Mais; · Menos. O utilizador escolhe se o equipamento se comportará do mesmo modo que atualmente ou se executará uma das seguintes operações: · Telefonar; · Introduzir alarme; · Introduzir prova de vida. E para qual/quais números as operações escolhidas deverão ser realizadas. Os números a utilizar serão obtidos das configurações da organização ou das relações do utente associado ao equipamento (as relações que estejam configuradas para isto).
UC25	As tablets que suportem a apresentação de utilizadores à partir dos dados do servidor poderão obter a lista de pessoas monitorizadas configurados para o IMEI da tablet. Na fase 3.4 o caso de uso passará a depender da execução do da associação do equipamento ao utente.
UC26	O sistema obtém a lista de pessoas monitorizadas do servidor e armazena-a internamente.
UC27	O utilizador acede à página de ativação de conta com uma chave de ativação válida, o sistema valida o seu e-mail e, caso a chave de ativação se refira à recuperação de

	password, permite ao utilizador digitar as novas credenciais de modo a atualizar a password.
UC28	Os utilizadores digitam o e-mail que usaram no "Registo de utentes" confirmam a recuperação dos dados. O sistema envia um e-mail de recuperação de acesso. O e-mail conterá um link para a recuperação da password e opcionalmente, caso o endereço de e-mail não tenha sido validado, um link para a validação do e-mail.
UC29	O utente regista valores que mediu manualmente. O sistema processa a informação da mesma forma que faria para os dados recebidos dos equipamentos
UC30	Os utilizadores registam-se no sistema indicando para isso os seus dados pessoais, o e-mail e telemóvel. O registo na plataforma implica o envio de um e-mail e um SMS de validação para os contactos definidos no registo. A autenticação dos utentes registados desta forma apenas será possível após a validação do e-mail. Os dados a preencher pelos utilizadores no registo de utentes é o seguinte (todos os campos são obrigatórios): · Login; · Nome; · Password; · Confirmação da password; · E-mail · Telemóvel; · Morada; · Código postal.
UC31	O sistema, deteta que o equipamento não existe ou existe e não está associado ao utente autenticado. O sistema associa o equipamento ao utente criando-o caso necessário.
UC32	O utilizador insere as suas credenciais e confirma o seu armazenamento. O sistema verifica o acesso ao sistema e em caso de sucesso armazena as credenciais. O sistema não deverá permitir o acesso às funcionalidades da aplicação sem ter sido realizado o armazenamento das credenciais do utilizador.
UC33	O utilizador escolhe a pessoa a monitorizar, digita o seu pin (caso necessário), escolhe o tipo de dados a adquirir, escolhe o equipamento de medição (caso necessário) e inicia a recolha de dados. A informação recebida é enviada para o servidor à medida que esta estiver disponível (mas em blocos de 1 segundo). A paragem da monitorização (após inicio com sucesso) implica o retorno à janela de escolha do tipo de medição. O utilizador deverá poder alterar o equipamento caso exista algum erro na sua escolha anterior. Os equipamentos a suportar serão os equipamentos de Oximetria Nonin 4100 e os equipamentos genéricos da Plux (com sensores de ECG, Temperatura, Respiração e Acelerómetro).
UC34	O sistema recebe medições de alta frequência armazenando-as num ficheiro de dados. Caso o ficheiro não exista o sistema cria um novo ficheiro na pasta de armazenamento.

9.2. Descrição de atributos de qualidade legados da linha de produtos OneCare

Tabela 51 – Descrições dos atributos de qualidade legados

ID	Descrição
AQ01	Uma pessoa pretende descobrir a password de um equipamento GH3000 enviando comandos de SMS para o equipamento. A probabilidade de o utilizador acertar na password corrente do equipamento tem de ser inferior a 1/1 000 000.

AQ02	Cinco utilizadores interagem com o sistema e avaliam cada funcionalidade de acordo com os dez princípios da heurística de usabilidade como definidos por Jakob Nielsen. Todas as ações da aplicação devem pontuar, em média, acima de 3,5 numa escala de 1 a 5. Escala 1 - Não cumprimento 2 - As questões encontradas degradam seriamente a avaliação 3 - Algumas questões são encontradas com mais impacto negligenciável 4 - Apenas pequenas questões podem ser encontradas com um impacto negligenciável 5 - Totalmente Compatível ou não aplicável
AQ03	O utilizador pressiona o botão vermelho ou quedas, o dispositivo envia com sucesso um SMS de alarme. Em 95% dos casos, o operador é notificado em menos de 2 minutos.
AQ04	Um utilizador usa a aplicação, não efetua *logout* e não tem qualquer tipo de interação no *tablet* durante um longo período de tempo. E considerando que é possível que alguém, sem as devidas permissões, consiga ter acesso ao *tablet* e consulte dados dos utentes. A aplicação efetua *logout* automático após detetar 10 minutos seguidos de inatividade.
AQ05	Um utilizador pretende efetuar uma medição usando a aplicação. O utilizador identifica a funcionalidade que pretende usar facilmente. 80% dos utilizadores não necessitam de consultar o manual de utilizador ou contactar um utilizador experiente.
AQ06	Um utilizador recebe uma medição na aplicação, tendo ativado a funcionalidade de visualização de alertas. O utilizador identifica facilmente se foi gerado um alerta no sistema. sempre que seja detetada uma situação anómala pelo sistema o alerta deve ser representado no sistema com um símbolo de cor amarela. Quando o valor da medição se encontra dentro de um intervalo correto, então é sempre mostrado um símbolo de cor verde.
AQ07	O utilizador está a usar a aplicação no *tablet*. O utilizador é servido por outro servidor. O utilizador não tem conhecimento que o servidor falhou.
AQ08	O utilizador está a usar a aplicação no *tablet*. O utilizador usa a aplicação onde lhe é fornecida a informação de que se encontra em modo offline. O utilizador consegue efetuar a monitorização de uma medição e este sabe que se encontra em modo offline.
AQ09	O equipamento, depois de ativado o sistema de *bluetooth*, envia para a aplicação os dados de uma medição e em seguida a aplicação envia para a API ISA server. A API ISA server deverá registar entrada de novos dados sempre que o utilizador efetue uma operação na aplicação que inclua geração de novos dados.
AQ10	O equipamento, depois de ativado o sistema de *bluetooth*, envia para a aplicação os dados de uma medição e em seguida a aplicação envia para a API ISA server. O utilizador efetua a medição com sucesso e os dados são guardados no *tablet* até a rede estar disponível e poder haver sincronização de dados com a API ISA server. Todos os dados captados pela aplicação em modo offline são guardados internamente no *tablet*.
AQ11	O equipamento, depois de ativado o sistema de *bluetooth*, envia para a aplicação os dados de uma medição e em seguida a aplicação envia para a API ISA server. Quando a rede passa ao estado disponível, a aplicação sincroniza os dados com a API ISA server. Todos os dados captados pela aplicação enquanto a rede se encontrava indisponível, são enviados para a API ISA server.

AQ12	O equipamento, depois de ativado o sistema de *bluetooth*, envia para a aplicação os dados de uma medição. O sistema recebe a medição enviada pelo equipamento. A aplicação deverá apresentar no *tablet* o valor obtido na medição num tempo máximo de 10 segundos.
AQ13	O equipamento, depois de ativado o sistema de *bluetooth*, envia para a aplicação os dados de uma medição e em seguida a aplicação envia para a API ISA server. O sistema recebe a medição enviada pelo equipamento e envia os dados para a API ISA server. A API ISA server deverá registar a informação recebida no máximo de 10 segundos.
AQ14	A aplicação tenta encaminhar informação para o servidor. A mensagem fica em fila à espera que o servidor responda. O utilizador não tem conhecimento que o servidor falhou.
AQ15	O serviço está em execução no *tablet*. A aplicação alerta do utilizador que se encontra em modo total ou parcialmente offline. O utilizador consegue aperceber-se que o *tablet* perdeu conectividade a alguma das redes (GPRS ou Internet).
AQ16	Foi recebida uma SMS de um número sinalizado que deve ser encaminhada para o servidor. O conteúdo da SMS deve ser tratado e encaminhado para o servidor. O servidor deve receber a informação encaminhada pela aplicação de *gateway* SMS com o mínimo de atraso possível.
AQ17	Foi recebida uma SMS de um número sinalizado que deve ser encaminhada para o servidor. A aplicação de *gateway* SMS deve guardar os dados *tablet* até a rede estar disponível e ser possível encaminhá-los. Todos os dados processados pela aplicação em modo offline são guardados internamente no *tablet*.
AQ18	O equipamento deteta uma ligação à Internet e tem dados guardados localmente, que devem ser encaminhados para o servidor. Quando a rede passa ao estado disponível a aplicação encaminha todos os dados que possa ter armazenados localmente. Depois do envio a cópia local dos dados deve ser eliminada. Todos os dados guardados localmente no *tablet* durante o período de indisponibilidade são encaminhados para o servidor.