· U C ·

Universidade de Coimbra

Relatório de Estágio

Módulo de agenda para a aplicação MedicineOne iPad

Autor:
João Monteiro
jram@student.dei.uc.pt

Professor Orientador:
Prof. Doutor Jorge Henriques
jh@dei.uc.pt

Orientador na Empresa:
Eng. Edgar Lopes
Edgar.Lopes@medicineone.net

Faculdade de Ciências e Tecnologias
Departamento de Engenharia Informática

Junho 2014

Resumo

O MyMedicineOne é uma aplicação que permite a um dado médico fazer a gestão clínica dos seus pacientes. Atualmente existem milhares de profissionais de saúde que usam esta aplicação, revelando-se esta um instrumento bastante importante no seu dia a dia.

Esta aplicação permite aos médicos trabalhar com informações como por exemplo os dados pessoais de um dado doente, as suas patologias, as suas alergias e gerir os seus dados clínicos.

O objetivo deste estágio é acrescentar uma nova funcionalidade à aplicação MyMedicineOne para iPad. Esta nova funcionalidade consiste num sistema de agendamento de consultas que será incorporada na aplicação através de um calendário. Este calendário terá vários tipos de vistas para facilitar a sua consulta e a sua navegação por parte do utilizador. Nestas vistas será possível ao médico marcar as horas que estão disponíveis para marcações de consultas, possibilitando que os utentes possam fazer marcações nessas mesmas horas, ou até pode ser mesmo o próprio médico a marcar logo uma consulta com um determinado utente.

Toda esta informação será sincronizada, através de um servidor CalDAV, com a agenda nativa do iOS para que as consultas também lá apareçam. Este servidor CalDAV também será implementado neste projeto, e também serão feitos testes para garantir os seus requisitos de funcionamento.

Palavras-Chave

Agenda, Calendário, iOS, iPad, Informática Médica, Sistema de Agendamento de Consultas, Aplicação para Gestão de Utentes, Servidor CalDAV

Agradecimentos

Em primeiro lugar gostaria de agradecer ao orientador do DEI, Professor Jorge Henriques e aos orientadores da MedicineOne, Edgar Lopes e Bruno Coelho, pois para além de me orientarem também me apoiaram e mostraram grande disponibilidade sempre que precisei. Gostaria também de agradecer à MedicineOne que me acolheu durante este ano e que sempre fez questão de dar importância e prioridade ao bom desempenho do estágio. Em particular agradeço também ao Marco Tinoco, Nuno Pereira, João Sousa, Audrey Silva, Pedro Faustino, Lara Coelho, Nádia Costa, João Miguel e Carlos Graça por tudo o que me ensinaram e pelo apoio que sempre me deram.

Em último lugar mas não menos importante gostaria de agradecer aos meus pais, ao meu irmão, a minha namorada e aos meus amigos pelo apoio incondicional e por estarem sempre presentes.

Índice

Lista de Figuras

Lista de Tabelas

Termos e Acrónimos

API	Application Programming Interface
APP	Aplicação
CS	Ciências da Saúde
IDE	Integrated Development Environment
IOM	Institute of Medicine
iOS	iPhone/iPad/iPod Operative System
Mac OS	Macintosh Operative System
SIC	Sistemas de Informação Clínica
SOAP	Simple Object Access Protocol
SQLite	Structured Query Language Lite
TIC	Tecnologias da Informação e da Comunicação
UCI	Unidade de Cuidados Intensivos
UI	User Interface
XML	Extensible Markup Language
CalDAV	Calendaring Extensions to WebDAV
OS	Operating system
OSX	Operating System Ten
CardDAV	Card Extensions to WebDAV
PHP	"Hypertext Preprocessor", originalmente Personal Home Page
BD	Base de Dados
IIS	Internet Information Services
PDO	PHP Data Objects
SHA384	Secure Hash Algorithm 384

MD5	Message-Digest algorithm 5
RAM	Random Access Memory
IP	Internet Protocol
NLB	Network Load Balancing
MAC Address	Media Access Control Address
WebDAV	Web-based Distributed Authoring and Versioning
CPU	Central Processing Unit
WS	Windows Server

Capítulo 1

Introdução

1.1 Enquadramento

Nos dias de hoje a informática está presente em todas as áreas. Graças a este facto foram desenvolvidas aplicações capazes de melhorar e impulsionar tanto as condições de trabalho nestes vários setores como também os resultados obtidos nas várias tarefas.

A informática médica é um excelente exemplo que ilustra esta ligação entre a tecnologia e a medicina, pois graças aos instrumentos tecnológicos que estão ligados a esta área é possível fazer um diagonóstico precoce de doenças, uma monitorização mais assídua dos pacientes, que muitas vezes até pode ser feita à distância, e o auxílio nas intervenções médicas ajudando que estas operações sejam concluídas com uma maior taxa de sucesso. Todos estes fatores ilustram apenas um pequeno exemplo do que realmente é a informática médica e basta tentar imaginar que nenhum hospital ou centro de saúde hoje em dia consegue sobreviver sem recurso a equipamentos tecnológicos.

Atualmente a informática médica é uma área em expansão e que tem muitas cartas a dar no mercado, pois todos os avanços científicos que impulsionam esta área impulsionam também a medicina.

O projeto MyMedicineOne representa um sistema de gestão clínica que permite a um profissional de saúde acompanhar todos os seus pacientes através das ferramentas que estão disponíveis neste sistema. No fundo toda a burocracia paciente/médico passa a ser gerida através deste sistema. Toda a informação relativa a dados pessoais do paciente,

patologias, alergias e dados terapêuticos ficam à distância de um clique. O sistema está sempre atualizado e com toda a informação que os profissionais de saúde necessitam. Isto resulta num melhor acompanhamento médico com menos probabilidades de serem cometidos erros.

1.2 Objetivos

Este estágio tem como objetivo acrescentar uma nova funcionalidade à aplicação já existente. Esta nova funcionalidade será um sistema de agendamento de consultas.

É bastante importante que neste tipo de softwares, um médico, para conseguir gerir todos os seus pacientes, para além de ter acesso a todos os seus dados também consiga marcar consultas futuras e aceder a consultas anteriores.

Esta nova funcionalidade poderá ser acedida no menu do MyMedicineOne e irá dispor de várias visões para facilitar a navegação e a marcação de consultas. Com esta alteração, um médico poderá marcar as horas que está disponível para que os utentes façam marcações nessas horas ou até ele próprio marcar uma hora com um determinado utente.

Também será desenvolvido neste estágio um servidor CalDAV que será responsável por sincronizar toda a informação relativa ao agendamento de um utilizador MedicineOne com qualquer software de agendamento que funcione com o protocolo CalDAV. Este protocolo é utilizado para a transmissão de informação em formato de calendário.

Uma vez que este servidor irá ter de comunicar com uma grande quantidade de médicos que usam esta aplicação será importante também garantir a sua escalabilidade, interoperabilidade e tolerância a falhas.

As várias etapas de trabalho deste projeto podem ser divididas em:

1. contextualização do projeto;
2. formação nas novas tecnologias;
3. identificação do problema;

4. análise de requisitos em que foram definidas as funcionalidades da aplicação e o modelo de interface e usabilidade;
5. arquitetura do sistema;
6. construção de um protótipo da aplicação;
7. integração do módulo desenvolvido na aplicação MyMedicineOne;
8. implementação do servidor CalDAV;
9. testes de escalabilidade;

Esta é a listagem final das etapas do projeto. O Projeto inicial consistiria em desenvolver apenas o cliente iPad. No segundo semestre de estágio para além da contiuação da implementação do cliente iPad, o objectivo do estágio focou-se também na construção do servidor CalDAV e nos testes de escalabilidade.

1.3 Estrutura do Documento

Este relatório apresenta todo o trabalho elaborado neste estágio. Para começar é feita uma contextualização do projeto, no fundo neste capítulo vai ser apresentada a área cientifica em que este projeto se insere, os trabalhos que foram analisados e as tecnologias usadas. Feita esta apresentação seguem-se dois capítulos mais técnicos, em que é feito o relatório de todas as análises e especificações e da arquitetura do sistema. Após esta análise é relatada a parte de implementação tanto da aplicação como do servidor CalDAV. Em último lugar serão apresentados os testes de escalabilidade feitos ao servidor e respectivos resultados.

Capítulo 2

Contextualização do Projeto

Este capítulo é iniciado com uma explicação sobre o que é a informática médica e quais são os principais conceitos desta ciência para que este estágio contribui. Depois serão apresentadas as tecnologias que serão usadas neste trabalho.

Seguidamente será apresentada a aplicação MyMedicineOne em que o modulo de agendamento vai ser integrado e também o trabalho relacionado com sistemas de agendamento para iOS.

Para finalizar serão discriminados alguns servidores CalDAV que foram experimentados para servir de base ao servidor do projeto, que será responsável pela sincronização da agenda clínica com outros softwares de agendamento.

2.1 Informática Médica

A informática médica é uma disciplina que estuda e desenvolve sistemas capazes de gerir e monitorizar informação médica. Esta área foca-se em estudar a forma como os cuidados de saúde são prestados e tentar solucionar os problemas existentes com o auxilio de tecnologia, criando assim um elo de ligação entre as Ciências da Saúde e as TIC.[1, 2]

Existem grandes quantidades de erros cometidos por médicos devido ao grande volume de informação com que eles têm que lidar. De acordo com um relatório do IOM cerca que 17,7% dos erros que ocorrem nas Unidades de Cuidados Intensivos (UCI) resultam

na morte ou invalidez dos pacientes e 46% resultam em efeitos nocivos futuros. A informática médica também pretende aumentar a rapidez e qualidade do tratamento de pacientes, fazendo assim, com que este número de erros seja cada vez menor.[3]

Como é que este tipo de problemas podem ser solucionados?

Para melhorar a gestão da informação clínica o objetivo principal dos sistemas informáticos desenvolvidos para as unidades hospitalares deve ser a gestão e integração de informação clínica, de forma a que seja possível responder às várias necessidades que surgem diariamente nestes locais.

E-Health - depois desta introdução à informática médica já é possível definirmos o conceito de *e-Health*. *E-Health* consiste na fusão do mundo das Ciências das Saúde (CS) com as TIC, representando assim todas as aplicações tecnológicas na saúde.[4]

As principais vantagens da *e-Health*[5] são:

- redução da percentagem de erros hospitalares;
- aumento da qualidade do acompanhamento médico;
- aumento da rapidez de atendimento ao paciente e também do seu tratamento;
- revela um bom investimento a médio/longo prazo;
- permite acompanhamento e monitorização à distância;
- aumento da rapidez na prescrição de receitas médicas.

Sistema de Informação Clinica - um SIC representa toda a informação guardada numa dada organização de saúde que possa ser útil a qualquer funcionário para desenvolver o seu trabalho. Desta informação não fazem só parte os dados clínicos e pessoais dos pacientes, mas também todas as outras informações vitais à organização.[1]

Os SIC podem ser representados por toda a cadeia gestora de informação numa clínica, não sendo implícito o uso de tecnologia, embora com o avanço da informática médica já seja possível representar grande parte destas metodologias de gestão de informação através de software.

Em suma as principais vantagens do investimento em um SIC[3] são:

- fácil comunicação entre todos os membros de uma organização e saúde;
- acesso rápido a toda a informação desejada;
- ajuda nas funções clínicas, administrativas e financeiras;
- o aumento do auxilio nas tomadas de decisões pelos profissionais de saúde;
- monitorização e deteção prévia dos problemas de saúde dos pacientes;
- facilidade e rapidez no armazenamento e processamento de grandes quantidades de informação;
- tratamento estatístico de vários tipos de dados, que podem ser usados para investigação clínica e outros estudos.

Todos os dados relativos a um paciente no SIC fazem parte de um Registo Clínico Eletrónico.

Registo Clínico Eletrónico - de uma forma muito sintetizada um registo clínico eletrónico é um repositório digital com toda a informação do estado de saúde e de todos os tratamentos que um paciente recebeu na sua vida. Com estas informações também serão gerados dados demográficos.

2.2 Tecnologias Envolvidas

É bastante importante quando um médico tem de se deslocar, fora do seu consultório, possa ter uma ferramenta com as funcionalidades do MyMedicineOne ao seu dispor. Por esse motivo é que existe uma versão deste software para iPad. Por isso neste capítulo serão abordadas algumas considerações importantes do seu sistema operativo e da linguagem de programação usada no projeto.

iOS

O que é o iOS?

O iOS é o sistema operativo desenvolvido pela Apple que corre nos dispositivos seus dispositivos móveis, iPhone, iPod e iPad. Este OS não corre em nenhuma outra máquina para além das especificadas, assim como também os dispositivos móveis da Apple não

aceitam outro sistema operativo. Por esse motivo este é o sistema operativo para que será projetada este projeto de estágio, uma vez que o objetivo é que esta corra em iPad.

A arquitetura do iOS é constituída por quatro *layers*: Core OS, Core Services, Media e Cocoa Touch.

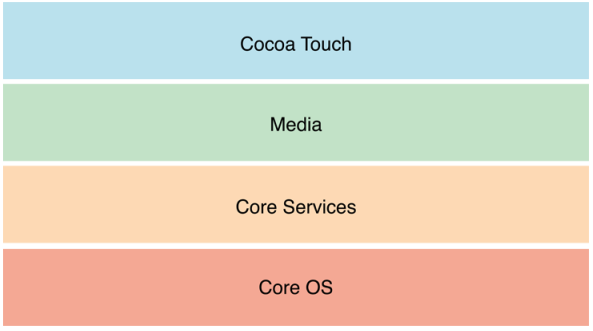

FIGURA 2.1: Arquitetura do iOS.

A construção desta arquitetura em camadas faz com que seja possível encapsular, nas camadas de nível mais superior, *frameworks* relativas à orientação a objetos, uso de *threads*, uso de *sockets*, entre outras. Este simples facto faz com que a quantidade de linhas de código para fazer um dado programa seja muito inferior do que se por exemplo esse mesmo programa fosse desenvolvido na linguagem C. Por este motivo é recomendado aos programadores que usem sempre as funcionalidades que as camadas mais altas lhes proporcionam, programando apenas a um nível mais baixo quando não existir nenhuma *framework* específica aplicável.[6]

Core OS Layer - esta camada é a de mais baixo nível e funciona como um alicerce de todas as outras. Quando o programador não lida com esta camada diretamente, ela vai acaba por ser a raiz das *frameworks* que ele está a utilizar, logo acaba por ser usada na mesma. Muitas vezes quando é necessário implementar algoritmos muito específicos ou quando temos de desenvolver uma APP que interage com algum tipo de *hardware* externo, é necessário programar diretamente nesta camada, uma vez que não existem *frameworks* para estes casos mais especificos de funcionalidades.[7]

Core Services Layer - esta camada encapsula a maior parte das funcionalidades essenciais a uma aplicação, como por exemplo a gestão do tempo de vida dos objetos alocados, armazenamento de dados na base de dados, mantém a integridade dos dados armazenados, e ainda possibilita a interação com a iCloud, sistema de localização e internet.[8]

Media Layer - como o próprio nome indica esta camada é representa toda a estrutura gráfica e de interação com áudio e vídeo do iOS, isto é, sempre que for necessário usar ferramentas de manipulação gráfica ou de áudio podem ser usadas ferramentas desta camada pelo programador.[9]

Cocoa Touch Layer - esta camada é a de nível mais elevado, e também por esse motivo é a que encapsula funcionalidades de mais alto nível como por exemplo todo o sistema de navegação que pode ser usado numa aplicação para iOS bem como funcionalidade de *multitasking*, eventos *touch*, reconhecimento de gestos, notificações. Basicamente toda a aparência da aplicação pode ser trabalhada com as funcionalidades que o Cocoa Touch possui, existe até mesmo a possibilidade de definir todo o *storyboard* da aplicação.[10]

Dentro das quatro camadas descritas existem várias *frameworks*, sendo que cada uma delas oferece um serviço diferente. É importante dar ênfase a uma em específico, que faz parte das *frameworks* da camada Core Services, que é o Core Data:

Core Data - o Core Data é uma *framework* que permite organizar dados para serem serializados em XML, SQLite ou até mesmo código máquina. Esta *framework* permite ao utilizador manipular dados usando representações de alto nível como tabelas, atributos e relações entre eles, também é responsável por assegurar a melhor gestão possível da memória através do controlo do ciclo de vida dos objetos alocados e mantendo a persistência dos dados.[11]

iOS7

Como o iOS7 foi disponibilizado para todos os utilizadores da Apple no dai 18 de Setembro de 2013, este projeto foi desenvolvido para esta versão. Neste momento 74% dos dispositivos Apple correm o iOS7. [12]

Com esta versão do sistema operativo também vieram algumas mudanças que os programadores devem ter em conta como: alterações na UI, alterações na manipulação de texto, melhoria de multitasking e a nova arquitectura é de 64 bits.[13]

Agenda iOS7

Uma vez que este projeto tem como objetivo a construção de uma agenda para uma aplicação do iOS7, é relevante estudarmos como funciona a agenda nativa do sistema.

A agenda do iOS7 é constituída por quatro vistas: anual, mensal, semanal, diária.

Vista Anual

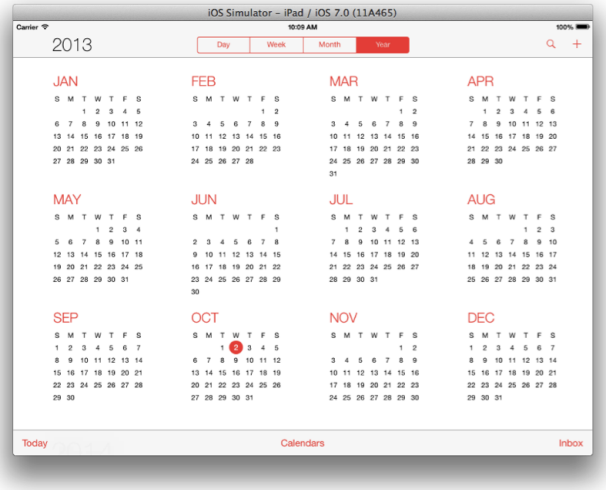

FIGURA 2.2: Vista anual

Nesta vista é possível visualizar o ano todo, isto é, todos os dias de todos os meses, embora não seja possível obter informação sobre os eventos marcados para cada dia. Esta visão é útil para uma navegação em que é preciso dar um grande salto temporal, para o utilizador se deslocar ao longo dos anos basta fazer *scroll* vertical, quando quiser entrar num dado mês basta carregar em cima dele.

Vista Mensal

A vista mensal mostra todos os dias do mês em foco e já permite ver algum detalhe sobre os eventos marcados para cada dia. O sistema de navegação é idêntico ao da vista anual. Neste caso o utilizador ao carregar num dia não vai para a visão diária respetiva, mas se pressionar durante algum tempo pode marcar um evento nesse dia.

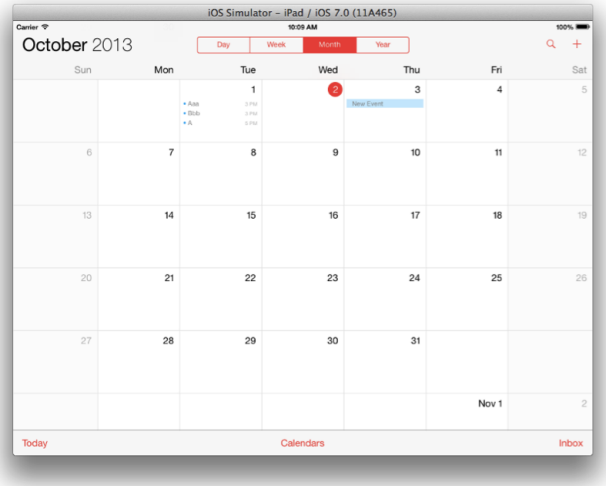

FIGURA 2.3: Vista mensal

Vista Semanal

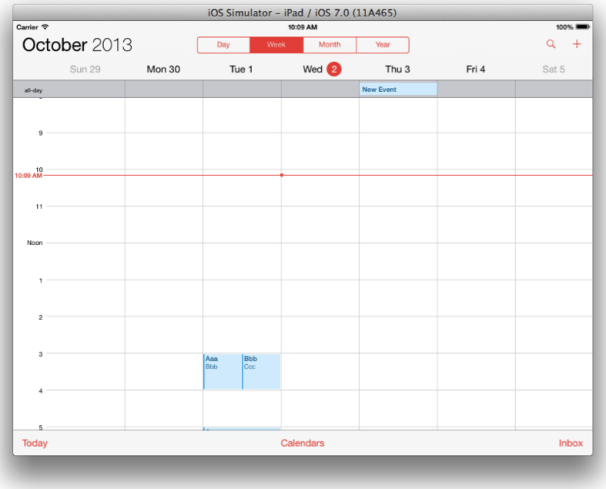

FIGURA 2.4: Vista semanal

Nesta vista a navegação pode ser feita através de um *scroll* horizontal, para mudar de semana, ou um *scroll* vertical para navegar pelas horas dos dias das semana em foco. Nesta visão também existe uma linha que representa a hora atual.

Vista Diária

A visão diária é a visão mais pormenorizada de todas, uma vez que foca um dia de cada vez e permite ver todos os detalhes de um evento quando ele é selecionado na parte

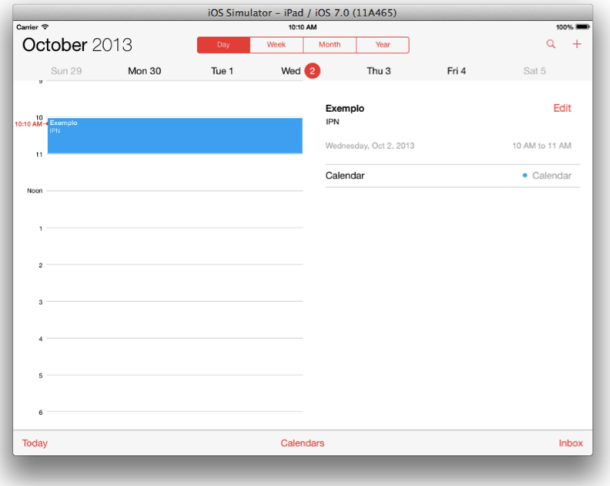

FIGURA 2.5: Vista diária

direita do ecrã. Nesta vista a navegação ao longo do dia é feita pelo *scroll* vertical, e para mudar o dia em foco pode ser através do *scroll* horizontal ou carregando no dia em questão na barra superior.

Em todas as visões é possível posicionar o calendário no dia atual, pressionando o botão "*Today*" e este é representado por um ciclo laranja.

Xcode

O Xcode é o IDE dos sistemas operativos da Apple. Esta ferramenta é bastante utilizada, não só para o desenvolvimento de software para correr nas máquinas da marca, como para desenvolver outros tipos de programas. A razão deste grande uso do Xcode é o fato de este ser uma ferramenta de uso simples e muito completa, acabando por simplificar muito o trabalho do programador, poupando-lhe horas de trabalho.

Na caso específico de desenvolvimento de aplicações para Mac OS e para iOS, o Xcode torna-se ainda mais poderoso, pois toda a estrutura do IDE foi desenvolvida de forma a ter o máximo de funcionalidades que auxiliem este tipos de projetos.

Objective-C

O nome Objective-C é quase auto-explicativo, pois trata-se de uma linguagem de programação orientada a objetos e é um *superset* da linguagem C. Neste momento a maior

parte dos programadores que usam esta linguagem desenvolvem software para OSX ou iOS.

Alguns princípios importantes

Construir uma aplicação com objetos: Para além dos objetos representados pelas classes do sistema, também é possível um utilizador criar as suas próprias classes. Na construção de uma classe podem-se criar atributos públicos, que podem ser acedidos noutras classes, e atributos encapsulados que só podem ser acedidos através de *getters* e *setters*, para além disso também existem os métodos de cada classe. Os métodos são declarados com o tipo do objeto que retornam e com uma lista dos parâmetros de entrada. Todos os métodos são escritos dentro de um bloco de código designado por *implementation*.

Definir categorias: Quando se pretende adicionar um novo método/atributo a uma classe já existente, mesmo que o código fonte desta não esteja disponível, é possível faze-lo definindo uma categoria dessa classe. As categorias funcionam como extensões de classes já existentes e herdam todos os seus métodos e atributos, podendo ser adicionados novos ou alterados os existentes sem qualquer tipo de problema.

Delegates: Em Objective-C para além de se poder usar os métodos de uma classe no objeto dessa mesma classe, também é possível criar métodos que podem ser usados em qualquer classe. Para que isto aconteça basta declarar esses métodos dentro do *delegate* dessa classe, e atribuir o delegate ao objeto na classe em que se pretende usar os métodos.

Os vários tipos de objetos: Esta linguagem vem recheada de objetos que podem ser utilizados, objetos como NSNumber que pode representar qualquer tipo de número e o NSString que é usado para representar strings, estes objetos também têm diversos métodos que se podem revelar muito úteis. O facto de existirem objetos para representarem tipos de variáveis como *int*, *floats* e *chars*, estes também podem ser declarados desta maneira em Objective-C.

Blocos de código: É possível criar blocos que contêm código lá dentro, isto é uma vantagem muito grande numa linguagem, pois o código fica mais organizado e também pode contribuir muito para o aumento da performance de um programa, tanto para correr blocos de código em exemplos de teste, como para por *threads* a usar blocos de código diferentes.

Lidar com erros: É possível fazer um controlo de erros com um tratamento de exceções tal como na linguagem C, mas existem erros que não podem ser contornados desta maneira. O Objective-C temnuma classe chamada NSError responsável por lidar com os erros como por exemplo o mau uso do espaço em disco e problemas em *runtime*. Para além disto o programador é sempre avisado de forma a praticar uma boa programação, focando-se sempre em desenvolver uma aplicação com a melhor performance possível.

2.3 MyMedicineOne

O MyMedicineOne é uma aplicação da MedicineOne que também foi desenvolvida para iPad e na qual será integrada o novo módulo do sistema de agendamento. No entanto para além do sistema de agendamento esta aplicação já tem inúmeras funcionalidades que a tornam valiosa para os profissionais de saúde e em que certamente este projeto vai ajudar a acrescentar valor.

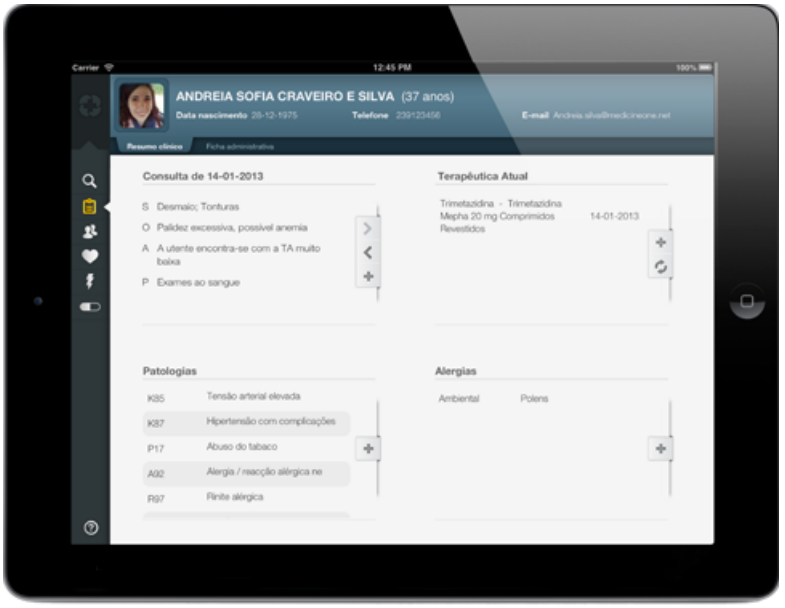

FIGURA 2.6: Aplicação MyMedicineOne.

Lista de funcionalidades[16]: Pesquisa de utentes, Adicionar Utentes, Resumo, Ficha Administrativa, Gestão da Consulta, Gestão de Patologias, Gestão de Alergias, Terapêutica, Interações de medicamentos, Configurações, Notícias, Manual.

2.4 Trabalho Relacionado com Agendamento em iOS

Antes de se iniciar a implementação do projeto, houve uma pesquisa dos softwares open source que existiam a nível de agendas iOS, embora muitas das encontradas tenham sido apenas para iPhone seria possível mais tarde converte-las para iPad[17].

Aqui ficam alguma imagens ilustrativas de alguns dos projetos encontrados:

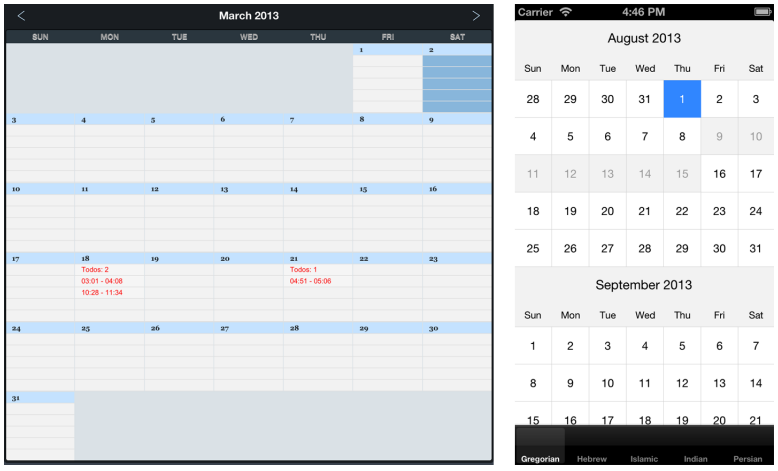

FIGURA 2.7: Soluções encontradas na pesquisa para a visão mensal

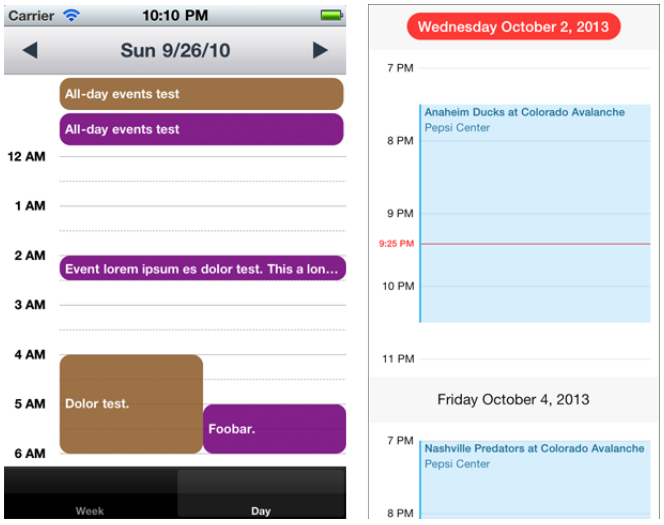

FIGURA 2.8: Soluções encontradas na pesquisa para a visão diária

Estas imagens ilustram os melhores dos exemplos encontrados, de facto existe muito pouco código disponibilizado em termos de agendas para iOS.

Ao fim de um estudo destas soluções, acabaram por ser todas rejeitadas devido aos seguintes motivos:

- existe muito pouco código adaptado para iPad;
- todos os códigos eram bastante diferentes e seria perdido muito tempo para os reprogramar em conformidade;
- toda a interface teria de ser alterada de forma a satisfazer os requisitos do projeto;
- muitos destes projetos não estavam devidamente preparados para que fosse integrado um sistema de agendamento;
- não existe nenhum solução com uma agenda completa e com todas as vistas, diária, semanal, mensal e anual.

Por estas razões seria muito dispendioso em termos de tempo adaptar todas estas estruturas diferentes, de maneira a constituir um projeto minimamente sólido capaz de integrar uma agenda e ser integrado na aplicação da MedicineOne, do que desenvolver todo o sistema de navegação de raiz. Acabou por ser isso que aconteceu.

No entanto esta pesquisa foi preciosa, porque serviu como estudo da linguagem e deu para perceber que estruturas poderia utilizar para se começar a construir um sistema de agendamento do zero.

2.5 Servidores CalDAV Existentes

Foram estudados vários servidores CalDAV open source, para ver se algum deles poderia ser adaptado para este projeto, com o objectivo de sincronizar a agenda da aplicação com outros softwares de agendamento. Dentro de todos estes servidores estes foram os mais relevantes:

- **Radicale** [31]: O projeto Radicale foi desenvolvido em Python e representa um servidor que é capaz de manipular informação relativa a calendários e contactos. Este projeto tem como pontos fortes a sua simplicidade, é um projeto de fácil compreensão e a primeira configuração do servidor é bastante fácil. Toda esta simplicidade também acabou por resultar num factor de exclusão, pois a documentação existente é muito incompleta, a própria organização que desenvolveu este produto classifica-o como o serviço bastante simples que não implementa na

totalidade o standard do protocolo CalDAV e CardDAV. Chegam mesmo a recomendar outros tipos de projetos para servidores de maiores dimensões que têm de lidar com uma grande quantidade de clientes.

- **Davical** [32]: O Davical representa um servidor muito mais complexo do que o Radicale e por sua vez também mais completo, esta solução foi desenvolvida em PHP. Este cenário foi também excluído, porque os seus developers não aconselham a que ele seja instalado em sistemas Windows. Visto que neste caso o software terá de estar obrigatoriamente numa máquina windows esta razão foi suficiente para a exclusão desta hipótese. No entanto o Davical tem uma grande comunidade de developers activa e também possui boa documentação.

- **SabreDAV** [33]: Este projeto foi uma das melhores soluções encontradas, também desenvolvido em PHP. O SabreDAV é um projeto em constante desenvolvimento, com uma grande comunidade de developers online, serviu de base a muitos outros projetos open source e existe muito feedback positivo dos projectos que o usaram como base.

- **Baikal** [34, 35]: O Baikal é uma versão melhorada do SabreDAV, foram feitas bastantes melhorias, sobretudo no que diz respeito à administração do servidor e a facilidade na sua configuração. O seu repositório online é ainda mais activo do que o do SabreDAV, tendo bastantes developers a utilizar este projeto. Para além de toda a documentação do Baikal, também a do SabreDAV pode e deve ser usada como material de apoio aos developers que usem este sistema.

De todos os servidores apresentados o que foi utilizado neste projeto foi o Baikal, pois para além de todas as vantagens já referidas foi possível adaptá-lo para as condições deste sistema garantindo o seu funcionamento como planeado. As outras soluções acabaram por ser excluídas, ou por serem demasiado simplistas, acabando por não satisfazer algumas funcionalidades, ou porque o Baikal acabou por se revelar mais útil e completo, como foi no caso do SabreDAV.

Capítulo 3

Análises e Especificações

3.1 Identificação do Problema

Sendo o MyMedicineOne um sistema de gestão clínica, com o objetivo de ser o mais completo possível a nível de funcionalidades, neste momento não satisfaz dois requisitos importantes:

- **gestão de consultas** - isto é, o médico não tem a possibilidade de programar o seu dia a dia de trabalho de uma maneira facilitada, gerindo o seu tempo para cada paciente de acordo com os recursos e condições especais que cada um destes requer. Para além disso não existe nenhum local na aplicação onde ele possa planear as suas consultas, em que tenha a informação centralizada dos seus pacientes.
- **acesso rápido ao horário de trabalho sempre disponível** - não existe nenhuma maneira de um médico ter acesso rápido ao seu horário de trabalho, em qualquer altura, quer tenha ou não ligação à internet e onde possa comparar o seu horário pessoal com o horário de trabalho.

No fundo estes dois fatores combinados resultam no seguinte problema:

Como é que um software como o MyMedicineOne pode ajudar a facilitar a gestão do trabalho de um profissional de saúde, sendo esta menos burocratizada, mais cómoda e sem perdas de informação clínica?

Solução: É para responder a esta questão que este estágio surgiu, no fundo a criação de uma agenda para o MyMedicineOne pode resolver os problemas referidos nos dois tópicos anteriores da seguinte maneira:

- a informação consulta juntamente informação clínica passa a estar centralizada no sistema, podendo o médico consultar as consultas do seu dia a dia a planear a sua vida profissional em função disso. As consultas agora vão poder ser marcadas, pelos médicos ou por outros funcionários de um posto médico, através das aplicação MedicineOne. Esta marcação pode ser feita a qualquer hora e em qualquer lugar desde que exista ligação à internet.
- a criação de um servidor CalDAV que possibilite ao médico integrar o seu horário de trabalho no software onde tem o seu calendário pessoal, para conseguir organizar o seu dia a dia, e ter sempre acesso a esta informação mesmo que não exista internet.

3.2 Metodologia de Trabalho

A metodologia de trabalho que está a ser seguida neste projeto é o Kanban. O Kanban ao contrário da metodologia de desenvolvimento scrum, não se baseia na realização de *sprints* periódicos com objetivos faseados, no caso deste projeto o desenvolvimento de software está a ser feito com o foco na entrega final e todas as fases do projeto foram planeadas para que todos os requisitos se consigam cumprir até essa data[18].

O Kanban baseia-se em três princípios básicos:

tarefas diárias: o objetivo é olhar para o projeto todo como um puzzle e montar uma peça por dia, tendo sempre em atenção o panorama geral;

tarefas atómicas: esta metodologia defende que uma pessoa não dever estar sobrecarregada com várias tarefas em simultâneo, mas sim cumprir os objetivos um de cada vez, de uma forma faseada.

fluxo de trabalho faseado: quando a tarefa atual acaba a próxima a ser executada é a mais prioritária;

No entanto a consideração do Kanban como uma metodologia de desenvolvimento de software ainda gera muita polémica, uma vez que este é mais usado como uma metodologia de trabalho que teve origem na fábricas da Toyota com o objetivo de aumentar a produção. Por esse motivo este projeto acabou por ser desenrolado seguindo um pouco a ideologia Waterfall misturada um pouco com o modelo de trabalho Kanban. Como se vai poder compreender melhor pelas várias etapas do trabalho.

Na tabela 3.1 é possível consultar o planeamento das etapas principais do projeto. No Anexo B está um diagrama de gantt com etapas mais pormenorizadas (imagem B.5), e um outro com a comparação entre o planeamento e o tempo real que cada etapa demorou (imagem B.6).

Mapeamento do projeto segundo o Kanban:

Definição das funcionalidades da aplicação (23 de Setembro a 11 de Outubro)	Definição de todas as funcionalidades da aplicação de forma a solucionem o problema identificado. Definir os requisitos para essas funcionalidades.
Definições de interface e usabilidade (12 de Outubro a 19 de Outubro)	Definição das funcionalidades de navegação e de design e, mais uma vez, definir os requisitos que elas devem satisfazer.
Definição da bateria de testes da aplicação (20 de Outubro a 26 de Outubro)	Definir todos os testes que vão ser feitos à aplicação elaborando um documento com as condições de entrada, de execução e o resultado esperado.
Construção do protótipo da aplicação (27 de Outubro a 31 de Janeiro)	Construir todo o modelo de navegação num projeto próprio, para mais tarde ser incorporado no projeto final.
Avaliação da interface e usabilidade da aplicação (1 de Fevereiro a 28 de Fevereiro)	Realização de testes aos componentes de navegação desenvolvidos no protótipo.
Adicionar à aplicação MyMedicineOne o módulo da agenda (1 de Março a 8 de Março)	Incorporar o modelo da agenda na aplicação MyMedicineOne.
Análise e criação da estrutura de dados a ser utilizada na comunicação entre a aplicação e o servidor (8 de Março a 31 de Março)	Criação de uma estrutura de dados que suporte todos os dados relativos ao agendamento de consultas.
Alimentação da aplicação com os dados dos serviços (1 de Abril a 31 de Maio)	Preencher a bases de dados da aplicação com dados reais de serviços armazenados no servidor da MedicineOne.
Sincronização com agenda nativa iOS (1 de Junho a 7 de Junho)	Desenvolver serviços que façam a comunicação entre o servidor e a agenda nativa do iOS.
Testes de escalabilidade (8 de Junho até ao final do estágio)	Testar a escalabilidade do servidor CalDAV.

Tabela 3.1: Mapeamento do projeto.

3.3 Análise de Requisitos do Módulo de Agendamento

Neste capítulo vão ser apresentados os requisitos da aplicação. Para isso foi construído um diagrama de níveis de casos de usos, com três níveis, Cloud Level, Kite Level e Sea Level.[19] O Cloud Level só contem um caso de uso que é o mais geral de todos, e à medida que descemos de nível os casos de uso vão sendo decompostos em outros mais específicos. Seguidamente esses casos de uso são dispostos num diagrama de casos de uso e também é feito um diagrama de entidades para ilustrar o contexto em que a aplicação é utilizada. Logo a seguir é feita uma descrição pormenorizada de cada caso de uso e dos requisitos a que devem obedecer.

3.3.1 Casos de Uso

Esta representação foi escolhida, porque consegue-se responder ao problema que o projeto quer resolver num único caso de uso de complexidade muito elevada, para depois se dividir em casos de uso cada vez mais específicos que por sua vez representam problemas mais simples. Desta maneira é mais fácil organizar todos os requisitos e ao mesmo tempo organizar as funcionalidades do projeto.

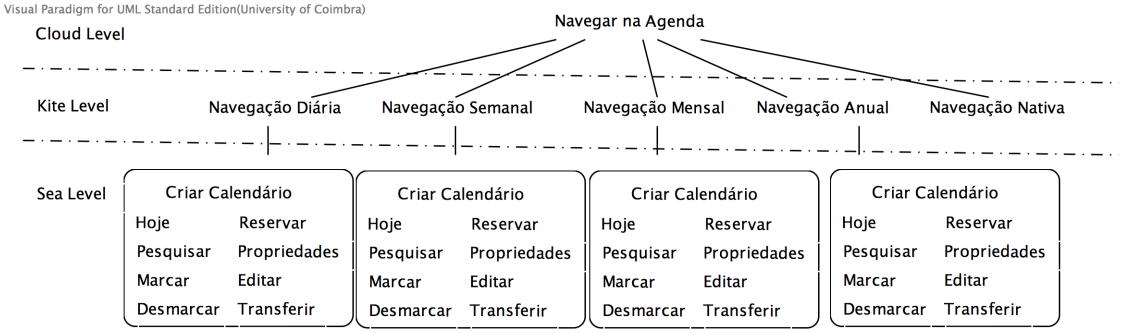

FIGURA 3.1: Diagrama de níveis de casos de uso

O Cloud Level representa o caso de uso mais geral que tem como intuito fornecer ao médico a navegação na agenda, entenda-se aqui que navegar na agenda consiste em consultar e editar informação relativa ao agendamento, como dá para perceber é um caso de uso bastante generalista. Logo no nível seguinte, o Kite Level, o caso de uso anterior é decomposto nas várias navegações que vão ser implementadas, em que cada delas terá de obedecer a requisitos diferentes. No nível mais baixo de todos, o Sea Level é possível ver todos os casos de uso que serão implementados dentro de cada navegação.

Segue-se agora a representação do diagrama de casos de uso:

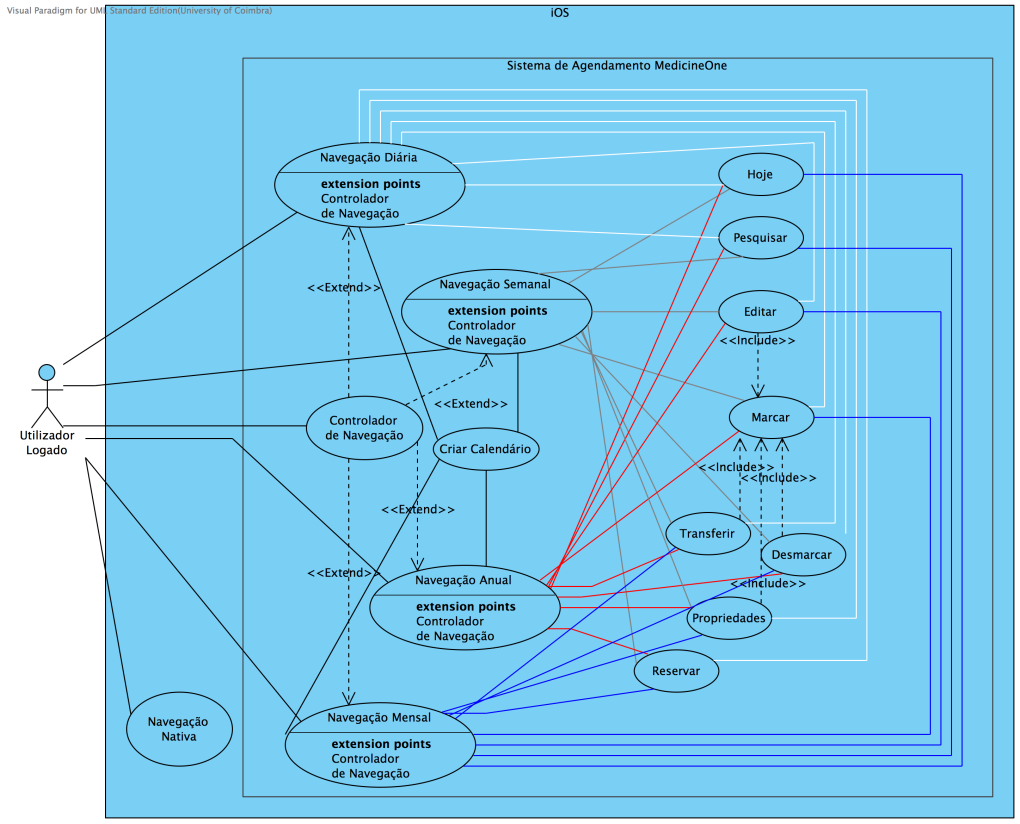

FIGURA 3.2: Diagrama de casos de uso (as ligações foram criadas com cores diferentes apenas para se conseguir ver as ligações das várias navegações para as funcionalidades de agendamento)

Neste diagrama, por convenção, o médico já se encontra logado e pode, através do controlador de navegação escolher o tipo de navegação que quer usar, ou entrar diretamente no último ambiente de navegação utilizado. Dentro de cada ambiente pode usar as funcionalidades: hoje, pesquisar, marcar, reservar e no caso de já ter uma consulta marcada pode também desmarcar, transferir, editar ou aceder às propriedades.

Por fim o diagrama de entidades:

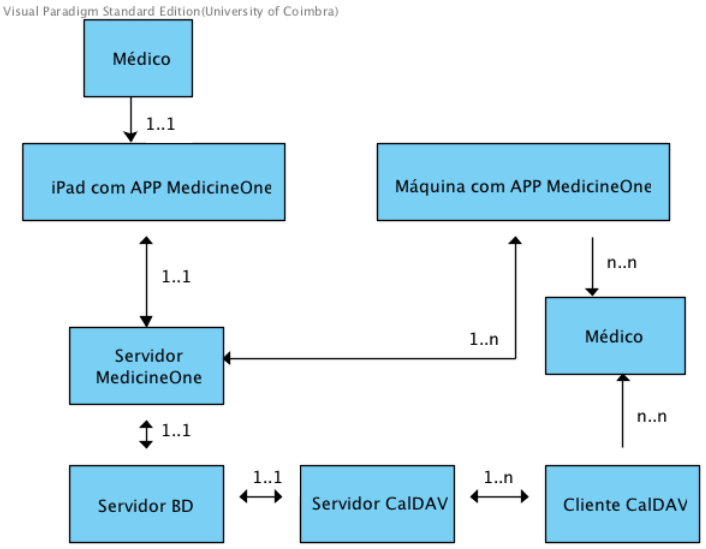

FIGURA 3.3: Diagrama de entidades

O diagrama de entidades mostra-nos que tipo de relação é que as entidades do sistema têm entre si. Neste caso o Médico comunica com o seu iPad que tem a aplicação MyMedicineOne, e esta comunicação é de um para um. Por sua vez a aplicação também comunica com um servidor da MedicineOne. O servidor da MedicineOne comunica com o servidor da Base de Dados e com várias máquinas, estas máquinas podem ser *tablets*, *smartphones* ou computadores de utilizadores da aplicação MyMedicineOne. Cada uma destas máquinas pode ser utilizada por um ou mais utilizadores, assim com cada utilizador pode utilizar várias máquinas.

O servidor CalDAV também é alimentado por dados do Servidor BD e é capaz de comunicar com qualquer máquina que tenha um software que implemente este protocolo.

3.3.2 Descrição dos Casos de Uso

Caso de Uso 1	**Caraterísticas**
Navegar na Agenda	**Nível:** cloud
	Ator: médico
	Objetivo: navegar e executar funções de agendamento
	Pré-condições: estar logado na aplicação MyMedicineOne
	Descrição: o médico entra num ambiente de navegação da agenda e faz qualquer tipo de operação de gestão de consultas
	Garantia de Sucesso: o médico conseguir concluir a operação de agendamento

Tabela 3.2: Tabela caso de uso 1

Caso de Uso 2,3,4,5	**Caraterísticas**
Tipos de Navegação	**Nível:** kite
	Ator: médico
	Objetivo: navegar no calendário com um destaque diário, semanal, mensal ou anual de acordo com a respetiva navegação.
	Pré-condições: selecionar no controlador a navegação desejada. No caso da navegação anual também se poderá mudar para a mensal carregando no mês desejado, e no caso desta também se pode mudar para a visão diária carregando no dia em questão
	Descrição: (1)Navegação Diária: neste ambiente o médico tem acesso a uma tabela com as horas do dia e um espaço reservado para mostrar os detalhes das consultas com grande destaque. A navegação deve ser feita através de *scroll* horizontal para mudar de dia, e vertical para deslizar pelas horas. Para saltar de semana a semana o sistema é exatamente o mesmo mas a interação é feita numa barra superior que representa a semana em foco. Também tem de ser implementada uma linha que desliza sobre a tabela ao longo do dia que representa a hora atual e um circulo no dia em foco.

Tabela 3.3: Tabela caso de uso 2,3,4,5

Caso de Uso 2,3,4,5	**Caraterísticas**
	(2)Navegação Semanal: o médico ao consultar a visão semanal terá acesso a uma matriz em que as colunas representam os dias de uma semana e as linhas as 24 horas. A navegação será feita através do *scroll* horizontal para mudar a semana em foco e vertical para deslizar ao longo das horas. A linha temporal também será implementada, bem como a representação do dia atual com cor diferente (3)Navegação Mensal: no ambiente de visão mensal o médico acede a uma matriz com todos os dias do mês. Neste caso a navegação é feita exclusivamente com *scroll* vertical para mudar de mês. (4)Navegação Anual: na visão anual a navegação é semelhante à mensal, o *scroll* é feito na vertical para navegar ao longo dos anos. Os doze meses em foco são respetivos ao ano em que o médico está posicionado. Geral: Em todos os ambientes anteriores serão implementados todos os requisitos do sea level, existirá também um titulo informativo em todos os ambientes a informar o utilizador em que mês/ano está. O dia atual é sempre representado por um circulo colorido. Se for possível, serão implementada s animações para as navegações e transições entre elas que serão definidas mais tarde de acordo com o tempo disponível.
	Garantia de Sucesso: o médico conseguir navegar para a data que pretende através de qualquer navegação, concluir a operação de agendamento que desejar. O médico deve conseguir ter sempre noção do seu posicionamento temporal atual.

Tabela 3.4: Tabela caso de uso 2,3,4,5

Caso de Uso 6	**Caraterísticas**
Criar Calendário	**Nível:** sea
	Ator: médico
	Objetivo: Criar um novo calendário associado a um médico ou a uma sala de consultas/operações
	Pré-condições: o médico precisa de estar num dos quatro ambientes de navegação descritos anteriormente, em qualquer posição temporal.
	Descrição: o médico pressiona o botão Adicionar Calendário e preenche os campos relativos ao calendário que pretende criar.
	Garantia de Sucesso: é exibida uma notificação a confirmar a criação do calendário.

Tabela 3.5: Tabela caso de uso 6

Caso de Uso 7	**Caraterísticas**
Hoje	**Nível:** sea
	Ator: médico
	Objetivo: ir para o dia actual
	Pré-condições: o médico precisa de estar num dos quatro ambientes de navegação descritos anteriormente, em qualquer posição temporal.
	Descrição: o médico pressiona o botão Hoje.
	Garantia de Sucesso: o posicionamento final é o dia atual.

Tabela 3.6: Tabela caso de uso 6

Caso de Uso 8	**Caraterísticas**
Pesquisar	**Nível:** sea
	Ator: médico
	Objetivo: pesquisar por uma consulta ou pelas consultas de um utente
	Pré-condições: o médico precisa de estar num dos quatro ambientes de navegação descritos anteriormente
	Descrição: o médico usa uma caixa de texto para escrever as *tags* de pesquisa
	Garantia de Sucesso: é apresentada uma lista com todas as consultas correspondentes à procura, caso estas não existam aparecerá uma notificação a informar o utilizador.

Tabela 3.7: Tabela caso de uso 7

Caso de Uso 9	**Caraterísticas**
Marcar	**Nível:** sea
	Ator: médico
	Objetivo: marcar uma consulta
	Pré-condições: o médico precisa de estar num dos quatro ambientes de navegação descritos anteriormente
	Descrição: o médico deve pressionar um botão que é representado por um sinal mais e preencher o formulário relativo à consulta. A não ser que esteja na visão anual, também é possível marcar uma consulta num dado horário pressionando durante alguns segundos o local correspondente.
	Garantia de Sucesso: a consulta fica representada em todas as visões da agenda

Tabela 3.8: Tabela caso de uso 8

Caso de Uso 10	**Caraterísticas**
Desmarcar	**Nível:** sea
	Ator: médico
	Objetivo: desmarcar uma consulta
	Pré-condições: o médico precisa de estar num dos quatro ambientes de navegação descritos anteriormente e a consulta tem de ter sido marcada.
	Descrição: o médico pressiona a consulta e na notificação informativa carrega no botão eliminar
	Garantia de Sucesso: a consulta desaparece em todas as visões da agenda

Tabela 3.9: Tabela caso de uso 9

Caso de Uso 11	**Caraterísticas**
Reservar	**Nível:** sea
	Ator: médico
	Objetivo: reservar tempo disponível para a marcação de consultas
	Pré-condições: o médico precisa de estar num dos quatro ambientes de navegação descritos anteriormente
	Descrição: o médico pressiona o botão reservar e escolhe o tempo que está disponível para atender os seus pacientes. Nessas reservas os utentes poderão marcar consultas através das suas aplicações
	Garantia de Sucesso: a reserva fica representada em todas as visões da agenda

Tabela 3.10: Tabela caso de uso 10

Caso de Uso 12	**Caraterísticas**
Propriedades	**Nível:** sea
	Ator: médico
	Objetivo: consultar as informações de uma consulta
	Pré-condições: o médico precisa de estar num dos quatro ambientes de navegação descritos anteriormente e a consulta tem de ter sido marcada.
	Descrição: o médico pressiona a consulta.
	Garantia de Sucesso: é exibida uma notificação informativa com os dados da consulta.

Tabela 3.11: Tabela caso de uso 11

Caso de Uso 13	**Caraterísticas**
Editar	**Nível:** sea
	Ator: médico
	Objetivo: editar as informações de uma consulta
	Pré-condições: o médico precisa de estar num dos quatro ambientes de navegação descritos anteriormente e a consulta tem de ter sido marcada.
	Descrição: o médico pressiona a consulta, seguidamente carrega no botão editar e pode editar todos os campos informativos. Para finalizar basta carregar no botão concluir.
	Garantia de Sucesso: os dados da consulta estão alterados quando se acede às propriedades.

Tabela 3.12: Tabela caso de uso 12

Caso de Uso 14	**Caraterísticas**
Transferir	**Nível:** sea
	Ator: médico
	Objetivo: alterar a data da consulta
	Pré-condições: o médico precisa de estar num dos quatro ambientes de navegação descritos anteriormente e a consulta tem de ter sido marcada.
	Descrição: o médico pressiona durante algum tempo uma consulta e arrasta-a para outra data.
	Garantia de Sucesso: a consulta fica na data para a qual foi arrastada em todos os ambientes de navegação.

Tabela 3.13: Tabela caso de uso 13

Caso de Uso 15	**Caraterísticas**
Navegação Nativa	**Nível:** kite
	Ator: médico
	Objetivo: consultar a agenda médica no calendário pessoal
	Pré-condições: estar a executar o iCalendar numa máquina da Apple.
	Descrição: ao utilizar o iCalendar o médico terá acesso a toda a informação relativa à sua agenda MedicineOne, porém não irá poder editá-la.
	Garantia de Sucesso: todas as consultas estão representadas no iCalendar.

Tabela 3.14: Tabela caso de uso 14

3.4 Interface da Aplicação

A prototipagem da interface gráfica foi criada usando o software *balsamiq*. Este software permite criar *mockups* para planificar melhor o funcionamento da aplicação.

Foram criados *mockups* para todos os ambientes de navegação, no entanto estas representações são de baixa fidelidade e funcionam apenas como alicerce para o inicio da construção da aplicação, podem e devem sofrer algumas alterações ao longo do desenrolar do projeto com o intuito de melhorar o produto final.

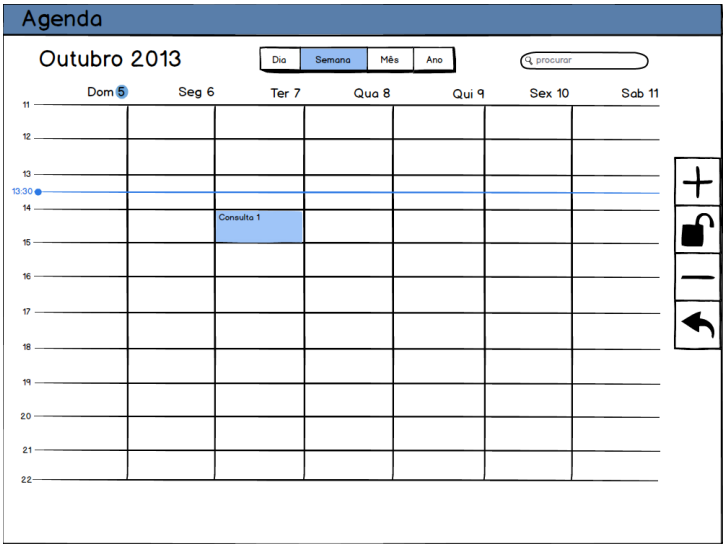

FIGURA 3.4: *Mockup* do ambiente de navegação semanal.

Nota: Os restantes mockups podem ser visualisados no Anexo C.

3.5 Restrições do Projeto

Restrições Técnicas: as restrições técnicas do projeto representam todas as limitações relativas a ferramentas e tecnologias utilizadas.

- linguagem Objective-C e C# - a aplicação para iPad será desenvolvida em Objective-C e os serviços implementados do lado do servidor em C#;
- optimizado para iOS7 - a agenda terá de funcionar na versão mais recente do iOS
- uso do Xcode - IDE utilizado para desenvolver a aplicação para iOS

- o design tem de seguir a linha gráfica da aplicação MyMedicineOne
- o sistema de navegação tem de ser de funcionamento semelhante ao sistema de agendamento nativo: tornar o calendário mais familiar para utilizador e facilitar a transição entre este e o nativo
- o servidor CalDAV terá de correr em IIS
- a base de dados do servidor terá de ser em SQL Server
- a base de dados terá de estar na mesma máquina da base de dados geral da aplicação
- o número de máquinas para os testes de escalabilidade e as suas características têm de ser negociados com o administrador do sistema

Restrições de Negócio: estas restrições estão ligadas às limitações obrigatórias que surgem com o desenvolvimento de *software* ligado à saúde.

- respeitar todos os dados privados dos pacientes
- o modelo de dados tem de ser capaz de armazenar toda a informação fundamental para uma consulta
- manter a segurança de todos os dados pessoais dos médicos
- o sistema tem de ser de fácil é rápida utilização

3.6 Atributos de Qualidade

Atributos de qualidade que a aplicação terá de garantir:

Atributo	Descrição
Usabilidade	O objetivo é a navegação ser intuitiva e o médico saber como realizar qualquer tipo de operação de agendamento, sem ter que recorrer a qualquer tipo de manual/ajuda. **Solução**: Criar mockups da aplicação antes da iniciar a implementação, testar a usabilidade através dessa prototipagem. Fazer testes de navegação ao fim da implementação do sistema de navegação.
Integridade no design	A interface deve seguir a linha gráfica da aplicação MyMedicineOne e deve ser constante em todos os ambientes, aliando o design de cada componente à sua função. **Solução**: Submeter a aplicação a uma avaliação do departamento de design da empresa e tomar em conta algumas objeções que possam surgir. Desenvolver a aplicação respeitando as políticas da Apple
Reutilização de componentes	Os ambientes com navegação através de *scroll*, como por exemplo vistas com tabelas, deve ser reutilizada a informação das várias células para que a aplicação não fique sobrecarregada. **Solução**: Implementar um sistema de reutilização de células em Objective-C.
Versatilidade	Os dados relativos às marcações de consultas têm de ser estruturados se forma a que possam ser sincronizados por qualquer dispositivo que corra o MyMedicineOne. **Solução**: As novas tabelas da base de dados têm de estar de acordo com a estrutura de dados da aplicação já existente.
Interoperabilidade	Caso não exista ligação à internet tem de ser possível visualizar a última versão actualziada do calendário. **Solução**: Como sem acesso à internet não é possível entrar na aplicação MedicineOne este requisito terá de ser satisfeito pela última informação enviada do servidor CalDAV.
Tolerância a Falhas	O servidor CalDAV terá de ter um mecanismo de tolerância de falhas que garanta que ele esteja sempre disponível. **Solução**: Ter mais do que uma máquina a correr o servidor e garantir que pelo menos sempre uma delas está operacional. (Esta solução será mais detalhada no capítulo 6)
Escalabilidade	O servidor CalDAV terá de suportar o número de utilizadores e de dados para que está destinado. **Solução**: Serão realizados testes de escalabilidade com o objectivo de estudar o melhor modelo de escalabilidade para o servidor. (Esta solução será mais detalhada no capítulo 6)

Tabela 3.15: Atributos de qualidade.

Capítulo 4

Arquitetura do Sistema

Foram escolhidos dois diagramas para desenhar a arquitetura da aplicação, um diagrama de componentes e conectores e um diagrama de *deployment*.

Na verdade o diagrama de componentes e conectores é bastante poderoso, no que toca à representação de um sistema, acabando por focar praticamente todos os aspetos importantes a considerar na arquitetura, no entanto também será apresentado um diagrama de *deployment* por funcionar um pouco como "resumo" de todo o sistema.

4.1 Representação do Sistema

Num diagrama de componentes e conectores, os componentes comunicam entre si através de interfaces. Estas interfaces são representadas por um circulo na conexão, a parte da conexão que liga ao circulo é a que envia a informação, a outra é a que recebe, como se pode ver na seguinte figura.

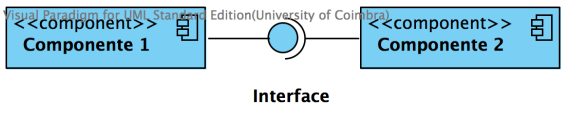

FIGURA 4.1: Componente 1 envia informação para Componente 2 através de uma interface.

Após esta breve introdução segue-se o diagrama representativo desta aplicação.

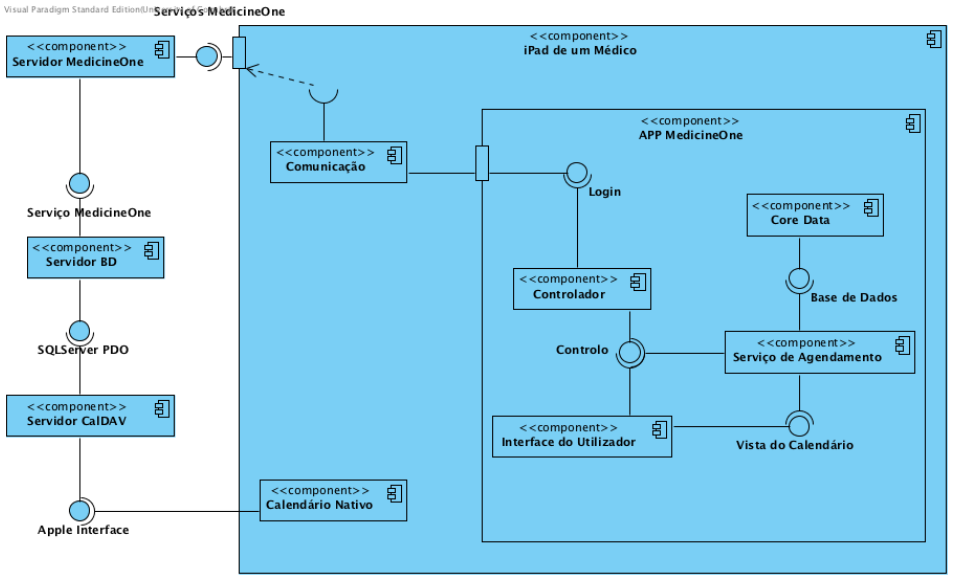

FIGURA 4.2: Diagrama de componentes e conectores.

Servidor MedicineOne: componente que representa o servidor da MedicineOne e todos os serviços que ele fornece.

Servidor CalDAV: representa o servidor que vai buscar a informação ao servidor BD e alimenta o calendario nativo.

Servidor BD: é o servidor que tem armazenadas todas as bases de dados da MedicineOne.

iPad de um Médico: iPad onde vai estar a correr a aplicação MyMedicineOne com o módulo desenvolvido neste estágio.

APP MedicineOne: aplicação MyMedicineOne com o módulo de agendamento integrado.

Controlador: componente responsável por executar os vários ambientes de agendamento.

Interface do Utilizador: componente que representa a parte visual da aplicação, isto é todos os componentes gráficos.

Serviço de Agendamento: componente responsável por executar todas as operações de agendamento.

Core Data: é responsável por fazer a integração da informação de uma consulta na base de dados da aplicação.

Calendário Nativo: representa o calendário nativo do sistema iOS.

Diagrama de Deployment

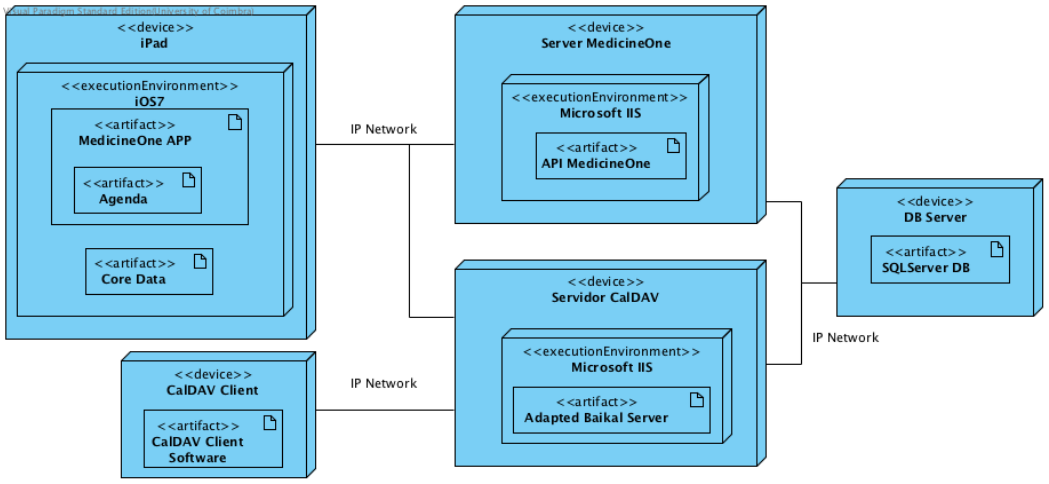

FIGURA 4.3: Diagrama de deployment.

O Diagrama de Deployment ilustra de uma maneira mais simples e resumida os principais intervenientes do sistema.

4.2 Esquema Arquitectural do Servidor CalDAV

Para exemplificar com mais detalhe como funcionará todo o sistema do servidor CalDAV foi criado um diagrama com as interações entre os vários componentes intervenientes, como se pode observar na figura 4.4.

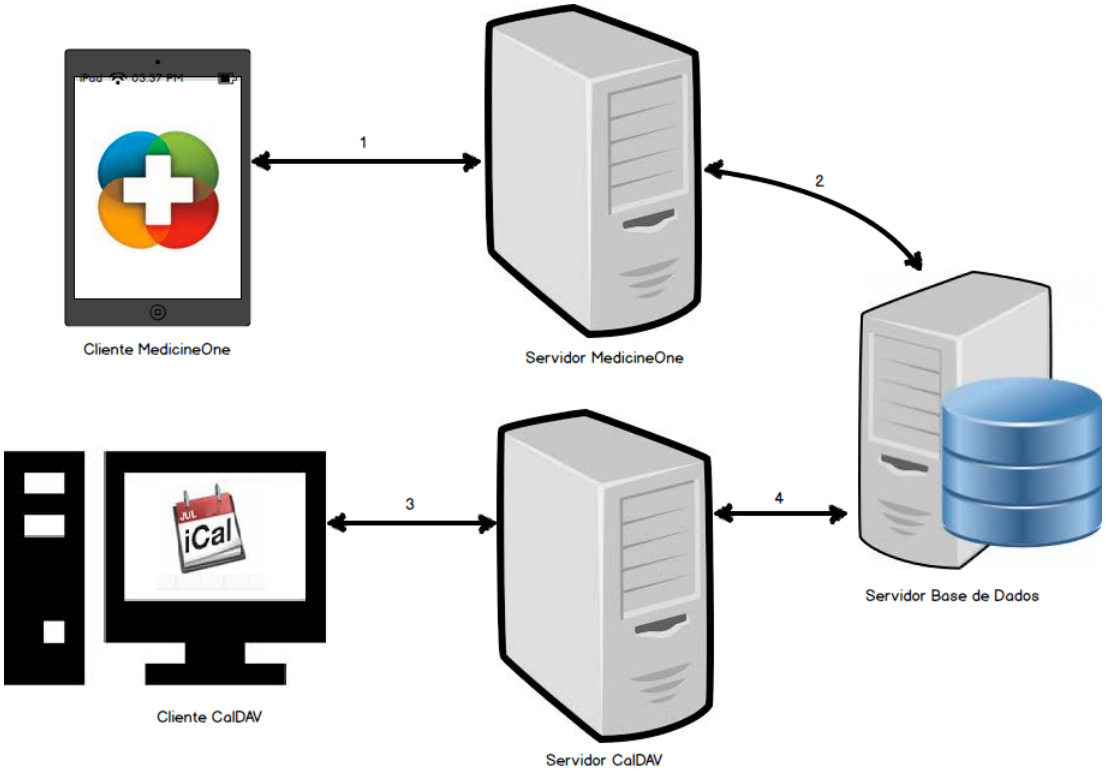

FIGURA 4.4: Arquitectura do Servidor CalDAV.

- O utilizador "Cliente MedicineOne" faz um pedido para sincronizar os seus eventos com o servidor CalDAV. Para isso chama um serviço do Servidor MedicineOne, que corresponde à ligação número 1;
- O serviço que corre no Servidor MedicineOne, executa um procedimento implementado na BD existente no Servidor Base de Dados, esta interacção é ilustrada pela conexão número 2. A partir deste momento a BD que alimenta o servidor CalDAV já tem os dados do novo utilizador;
- O utilizador pode ligar-se através de qualquer software que use o formato CalDAV (ilustrado no diagrama pelo cliente CalDAV), para tal tem de indicar três dados, o endereço do servidor CalDAV e o seu username e password da aplicação MedicineOne. Esta comunicação cliente-servidor é correspondente à ligação número 3.

- O Servidor CalDAV enviará as *querys* respectivas aos dados requeridos para o Servidor Base de Dados, exemplificado pela ligação 4.

Todos os dados mais específicos do desenvolvimento deste sistema serão explicados no capítulo 5 referente à implementação.

Capítulo 5

Implementação

5.1 Modulo de Agendamento

A primeira parte deste capítulo irá abordar a implementação da interface gráfica do ambiente de navegação com as transições e animações principais. Vão ser descritas as quatro vistas da agenda através de um breve resumo e de uma descrição dos principais problemas encontrados durante a implementação.

Seguidamente serão abordadas algumas considerações gerais de toda a implementação da aplicação.

Como é possível ver na figura 5.1, na parte superior do ecrã ao centro está o controlador que permite escolher o tipo de navegação em que o utilizador deseja navegar, neste exemplo foi pressionada a anual.

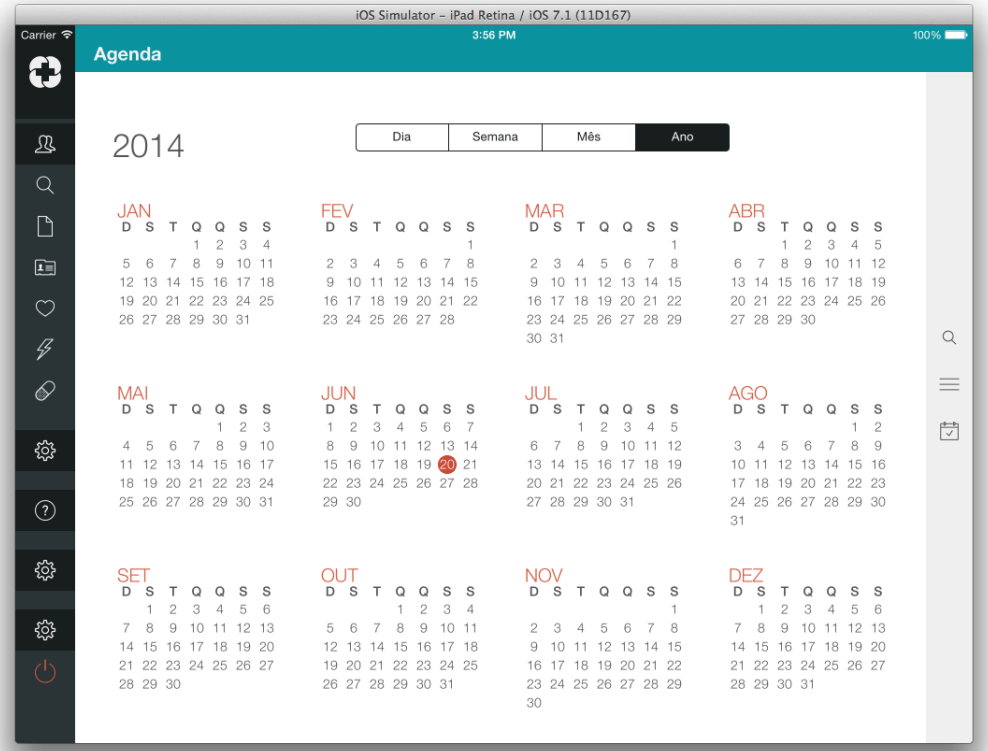

FIGURA 5.1: Implementação da interface do ambiente de navegação anual.

Breve descrição: Quando está selecionado o ano, no canto superior esquerdo é exibido o ano que está em foco que vai alterando à medida que é feito *scroll*. Os vários meses são criados dentro de uma célula de uma CollectionView. Uma CollectionView é um objeto que é constituído por várias secções e cada secção é formada por várias células. Neste caso um ano é uma secção e um mês é uma célula. Dentro de cada mês foram criadas *labels* com o nome do mês, dias da semana, os números dos dias e o circulo que representa o dia atual.

Principais obstáculos: Numa primeira abordagem na implementação da visão anual, quando era feito um *scroll* as *labels* começavam a aparecer sobrepostas umas às outras, isto é, apareciam os dias do novo ano mas também continuavam a ser exibidos o do ano anterior. Nesta altura foi implementada a reutilização de células para que os valores fossem atualizados e não criados novamente uns por cima dos outros.

Para melhorar a performance de navegação e para a aplicação não sobrecarregar tanto o sistema, na transição de um mês para o outro em vez de o valor de todas as *labels* que representam os dias do mês ser alterado, mudam apenas a sua posição de forma a que

coincidam com o dia da semana correspondente naquele mês. No caso de o novo mês ter mais ou menos dias, esse problema é controlado definindo as *labels* como ocultas.

Por uma questão de performance só são carregados em memória os dados relativos aos anos que estão na vizinhança do ano em foco.

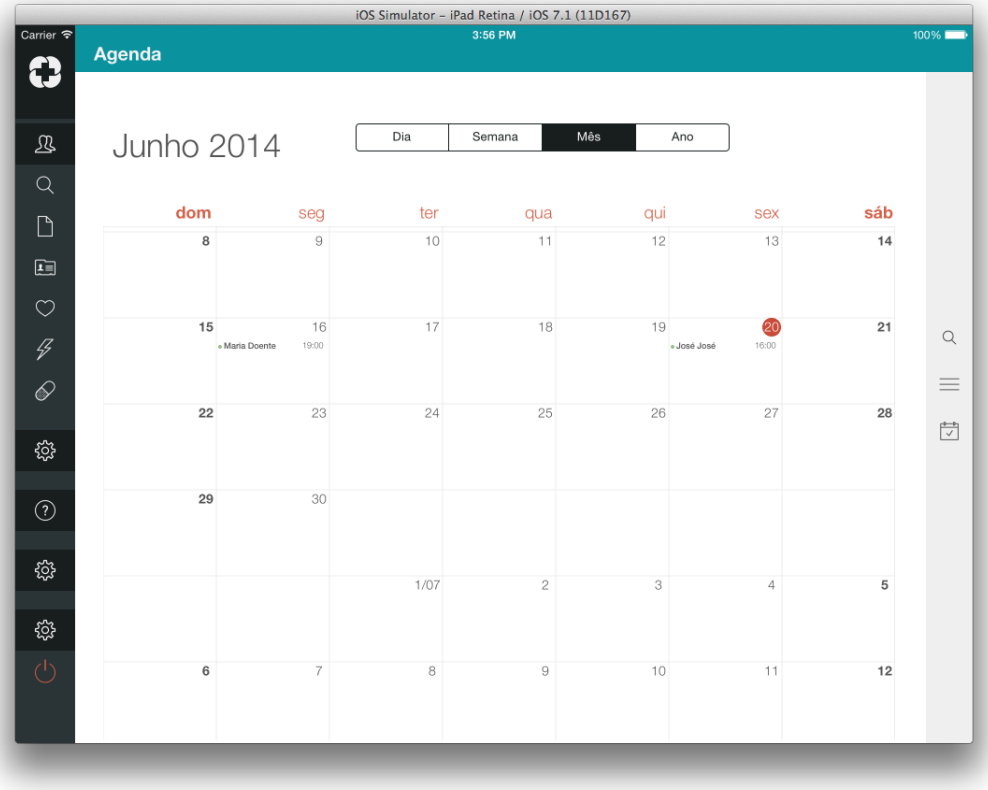

FIGURA 5.2: Implementação da interface do ambiente de navegação mensal.

Breve descrição: No caso da visão mensal, é disposto o mês e o ano na *label* de título. O mês atual é o que está em foco na janela, a não ser que tenha sido feita uma transição de outra visão numa outra data. Nesta visão também é usada uma CollectionView, mas neste caso as secções são os meses e os dias são as células. A navegação também é feita através do *scroll* vertical.

Principais obstáculos: Nesta CollectionView as células também necessitam de ser reutilizadas por o mesmo motivo que as da visão anual. Neste caso em específico também surgiu um problema relativo ao espaçamento entre dois meses, e ao número de células que cada mês ocupa, uma vez que o Objective-C não deixa que cada secção de uma CollectionView tenha um número diferente de células. Por exemplo o mês de Fevereiro

de 2015, começa num Domingo e como tem 28 dias acaba num Sábado, e por esse motivo o mês só ocupa 4 semanas, o mês de Agosto do mesmo ano já começa num Sábado e acaba numa Segunda, e como tem 31 dias vai ocupar 6 semanas. A solução passou por reduzir a dimensão das células das linhas entre dois meses para zero, exceto quando um mês termina a um domingo, sendo assim o outro só pode começar depois de uma linha de intervalo. Esta situação é para se conseguir distinguir bem durante o *scroll* onde acaba um mês e começa o próximo.

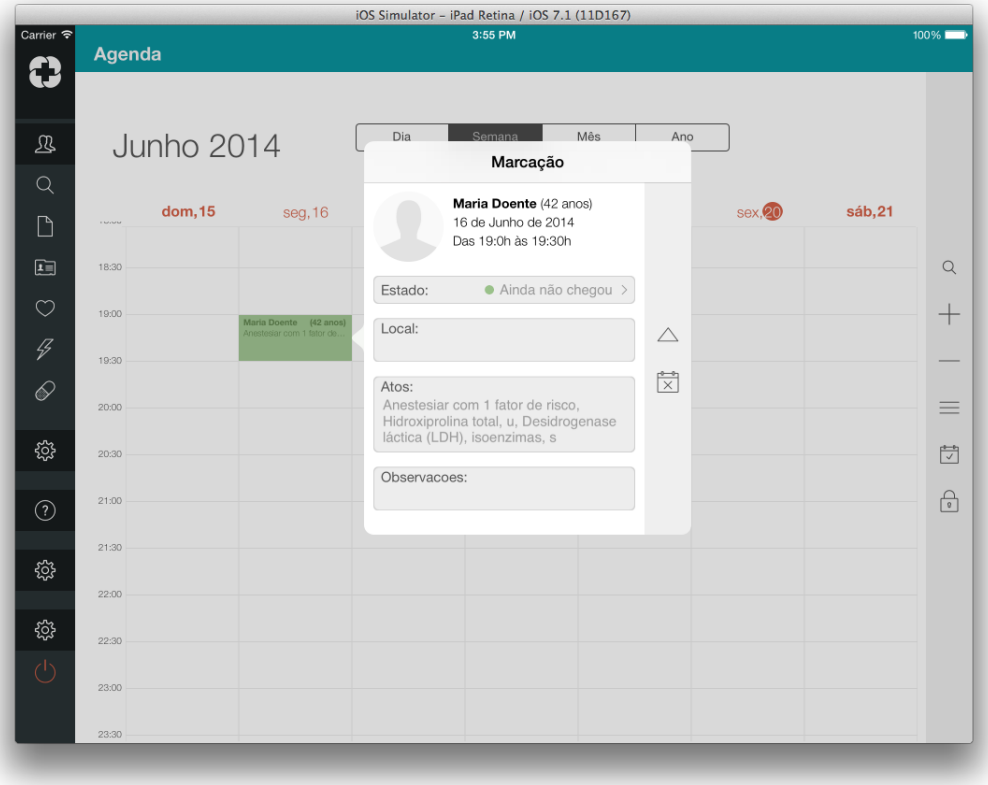

FIGURA 5.3: Implementação da interface do ambiente de navegação semanal.

Breve descrição: No ambiente semanal o título também representa o nome do mês e do ano. Neste caso são usadas duas CollectionViews e uma TableView. Uma das CollectionView é a barra superior onde estão representados os dias da semana mais os dias do mês, a outra é a grelha que representa os blocos horários semanais. As 24 horas diárias são dispostas numa tabela do lado esquerdo, para tal foi usado o objeto TableView.

No caso da visão semanal pode ser feito o *scroll* horizontal para mudar de semana e o vertical para deslizar pelas horas, estes movimentos podem ser feitos em qualquer dos

três componentes anteriormente descritos, pois os movimentos deles estão todos sincronizados. Também foi implementada uma linha e um ponto que representam o tempo atual, a linha vai deslizando ao longo das horas e o ponto para além de acompanhar esse movimento está sempre situado na coluna do dia atual.

Principais obstáculos: Mais uma vez nesta vista também é feita a reutilização de células. Inicialmente a linha temporal era um objecto definido nas células da tabela, mas uma vez que a mudança de posição ao longo do tempo é feita através de um NSTimer, nesta situação era preciso ter vários clocks todos sincronizados para que o movimento da linha fosse também sincronizado. Para melhorar este fator, os objetos que representam a linha e o relógio da hora atual foram criados por cima de todos os outros objetos. Assim só é necessário sincronizar o relógio responsável pelo movimento destes componentes com as animações de fade in e fade out das horas da TableView quando estes se cruzam com elas. Por exemplo entre as 19.50 e as 20.10 na TableView a *label* das 20 horas tem de desaparecer para que não exista uma sobreposição com os objetos que estão em movimento.

Outro problema que surgiu nesta navegação foi com a sincronização do *scroll* horizontal com o vertical, uma vez que não é suposto ser permitido fazer os dois ao mesmo tempo, mas sim quando é feito *scroll* vertical ficar apenas esse ativo e quando é feito o horizontal também, navegando uma semana para a frente ou para trás. Para solucionar este problema foi usada uma implementação disponibilizada online, embora tivessem de ser feitas algumas adaptações neste caso em especifico para que o scroll horizontal fosse sempre ajustado de acordo com o início da semana a não ser quando é feito *drag*.

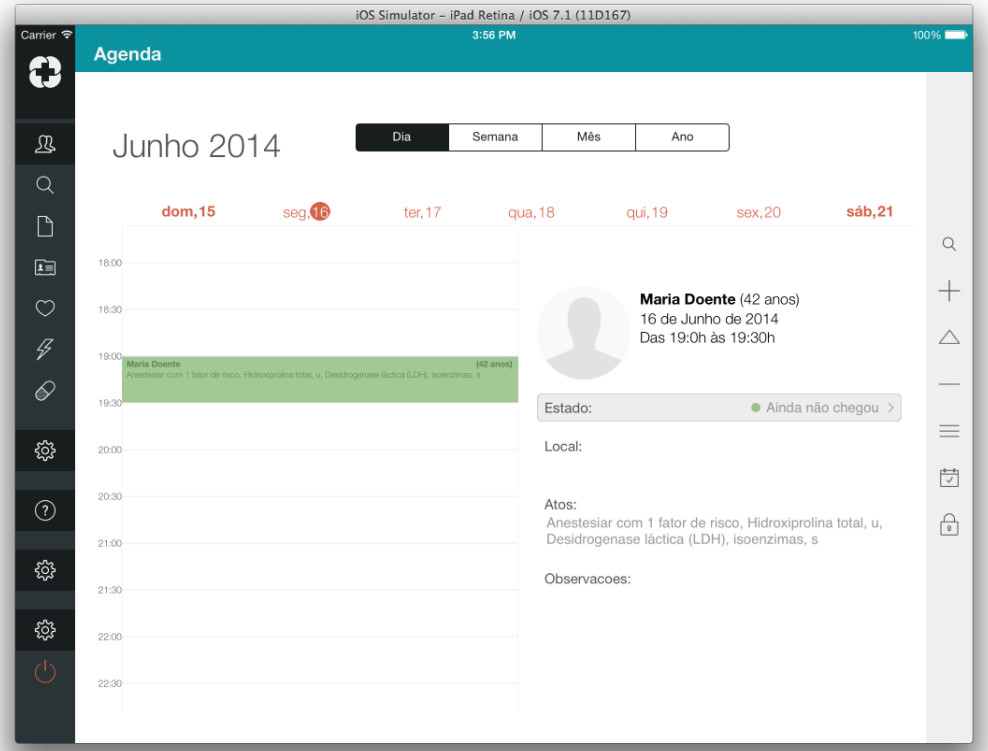

FIGURA 5.4: Implementação da interface do ambiente de navegação diária.

Breve descrição: Na visão diária o funcionamento é muito semelhante à semanal, as diferenças é que neste caso só é focado um dia de cada vez, e a CollectionView da semana só muda quando se transita de sábado para domingo e vice-versa. Também foi adicionada uma *view* que ocupa metade do ecrã com o objetivo de dar destaque aos detalhes de uma consulta selecionada.

Principais obstáculos: O principal obstáculo na implementação desta visão foi a sincronização do movimento da CollectionView diária com a semanal.

Considerações Gerais

A estrutura de dados utilizada para armazenar os dados relativos ao calendário, como os anos, meses, dias do mês, dias da semana foi um dicionário, sendo assim bastante simples aceder à informação que é necessária para preencher cada visão do calendário. Através de uma dada chave do dicionário fica-se com acesso a todos os valores constituintes desse objecto, por exemplo ao consultar o dicionário com a chave *YEAR* para um determinado ano, serão devolvidos todos os meses respectivos a esse ano, que por sua vez contêm

todos os dias referentes a cada um deles. Assim este dicionário só é criado uma vez, no controlador de navegação, e todas as vistas passam a herdá-lo.

Foi criada uma estrutura de componentes finita em todas as visões, pois os objectos em Objective-C necessitam sempre de um valor que descreva o seu comprimento, por exemplo no caso da visão anual, foi criada uma CollectionView com 5 secções que se iam reutilizando e reposicionando de forma a ser possível navegar infinitamente ao longo dos anos. A posição inicial era na célula central e sempre que era alcançada a primeira ou a última célula, a CollectionView automaticamente se reposiciona na célula central novamente e actualiza os anos que estão a ser exibidos. Tudo isto mantendo a animação de scroll dando a ideia que é uma lista continua. Em cada visão foi usado um número de secções diferentes de maneira a se verificar que os dados carregassem rapidamente e a fluidez do movimento não seja afectada.

As funcionalidades de agendamento finalizadas foram as de navegar para o dia de hoje, editar consulta, desmarcar consulta, pesquisar consulta e propriedades da consulta. As funcionalidades de transferir e reservar não foram implementadas por o servidor não ter nenhum serviço específico para elas. E a marcação ainda não está 100% funcional pois o modelo de negócios para esta operação ainda não foi estipulado.

5.2 Servidor CalDAV

Este subcapítulo fará um apanhado de como é que o servidor CalDAV foi instalado e configurado no sistema. Na realidade este servidor também suporta o protocolo CardDAV, que é usado tal como o CalDAV mas para partilha de informação relativa a contactos e listas de contactos, caso no futuro a empresa queira também integrar algum serviço com este protocolo poderá usar este servidor bem como a sua estrutura de dados.

Como todo o projecto Baikal foi criado para correr num servidor Apache com ligação a uma base de dados SQLitse ou MySQL, foi feita uma migração para um servidor IIS e alterações para ele comunicar com uma base de dados em SQL Server.

Base de Dados SQL Server

Primeiramente serão analisadas as alterações feitas na Base de Dados MedicineOne. Esta base de dados que alimenta a aplicação MedicineOne tem toda a informação necessária

para o sistema de agendamento, no entanto foi necessário criar novas tabelas que embora estejam na mesma base de dados da aplicação são específicas para a integração com clientes CalDAV. Esta decisão foi tomada devido ao facto deste módulo da aplicação ser um extra a que só os médicos que requererem este serviço terão acesso, assim se uma clínica não quiser este módulo as tabelas nem são preenchidas com os dados desses médicos. Esta base de dados da aplicação também já tem bastantes colunas, que são todas as essenciais para o funcionamento normal da APP. Também está a ser usada por todos os developers dos departamentos mobile e de qualidade da empresa. Devido a todos estes factores foram criadas novas tabelas para funcionar com o serviço CalDAV.

Apenas quando um utilizador requer o serviço CalDAV, na aplicação MedicineOne, é que estas novas tabelas recebem os dados, ou seja, quando esse request é feito vão ser iniciados vários procedimentos que selecionarão dados das tabelas gerais da aplicação e posteriormente os irão pôr nos formatos indicados para serem inseridos nas novas tabelas usadas pelo servidor CalDAV.

Uma vez que o cliente CalDAV não terá permissões de escrita, não há o risco de existirem alterações do lado dele. Como no caso contrário já é possível, foram também programados Triggers para actualizar os dados das novas tabelas, sempre que são feitas alterações, remoções ou adições de consultas na agenda da aplicação MedicioneOne.

Servidor CalDAV

Em relação ao servidor CalDAV, primeiramente foi necessário instalar e configurar o PHP no IIS Manager Service. Uma parte importantíssima desta configuração consiste em instalar os drivers PDO para SQL Server. Também é importante fazer algumas configurações para um melhor desempenho do servidor e também por questões de segurança, como por exemplo configurar o https, actualizar a lista de verbos que são usados na comunicação nas Request Restrictions do IIS e configurar também o número de *Maximum Worker Processes* de acordo com o processador da máquina, por exemplo este deverá ser igual a 2 se estiver a usar um *dual core*, para estarem sempre os 2 processadores a satisfazer pedidos em simultâneo. Também foi necessário alterar o código do servidor para a comunicação com a nova base de dados. Estas alterações consistiram em criar uma nova ligação PDO, mas desta vez que aceda a uma BD em SQL Server, em vez de MySQL, alteração de todas as *querys*, também para SQL Server, e também foi necessário alterar as configurações de acesso ao servidor da BD. As permissões foram mudadas para

que os clientes CalDAV apenas tenham permissões de leitura sobre os eventos em todos os calendários externos à aplicação.

Servidor MedicineOne

Também tiveram de ser feitas mudanças no servidor MedicineOne. Foi criado um novo serviço que quando for requerido, iniciará um procedimento na base de dados que desencadeará o arranque de todos os outros procedimentos relativos ao preenchimento das novas tabelas. Este novo serviço será invocado quando um utilizador desejar integrar o seu calendário profissional com um cliente CalDAV.

Aplicação MedicineOne

Como já deu para perceber será através da aplicação MedicineOne que um utilizador poderá solicitar a integração com o servidor CalDAV, através de um novo serviço. O serviço recebe como input, o seu nome de utilizador e uma hash que será gerada através do algoritmo MD5. Como a hash usada pela aplicação MedicineOne é gerada usando o algoritmo SHA384 e esta hash não é compatível com os clientes CalDAV. Visto que também não existe nenhuma password armazenada em plaintext, ainda se ponderou gerar uma nova password aleatória de acesso ao servidor CalDAV, mas também não seria a maneira mais segura e mais prática para o utilizador. A solução que acabou por ser implementada, consiste em quando um utilizador faz o request do serviço CalDAV tem de digitar novamente a sua password. Desta vez serão geradas duas hashes usando os dois algoritmos, a hash gerada em SHA384 servirá para fazer a validação da password e caso essa seja validada a outra hash será enviada como input do serviço junto com o username. Assim o utilizador poderá agora ligar-se ao seu calendário com qualquer software de agendamento que tenha integração com CalDAV, tendo de inserir apenas o endereço do servidor e os seus dados de login da MedicineOne.

Capítulo 6

Escalabilidade do servidor CalDAV

No final do servidor CalDAV estar funcional e devidamente configurado qualquer cliente CalDAV já conseguiria comunicar com ele sem qualquer problema, mas isso não é o suficiente para que este vá para produção. Embora a comunicação já seja possível entre vários clientes e o servidor criado é necessário perceber se este está preparado para satisfazer todos os pedidos que lhe vão ser feitos por clientes reais da MedicineOne. Por este motivo a análise da escalabilidade é uma etapa importantíssima para este projeto.

6.1 Noções de Escalabilidade

O que é a Escalabilidade?

A escalabilidade muitas vezes é definida como "the ease with which a system or component can be modified to fit the problem area", no entanto esta definição pode ser simplificada através destas três características [27]:

- Capacidade de lidar com um aumento de utilizadores;
- Capacidade de lidar com um aumento da base de dados;
- Facilidade na manutenção do sistema;

Ou seja um sistema escalável é um sistema hábil, capaz de lidar com um aumento significativo de trabalho de uma forma uniforme.

Escalabilidade Vertical VS Escalabilidade Horizontal

Para tornar um sistema escalável a consideração do hardware a usar é um ponto essencial. Embora muitas vezes o hardware pareça ser um investimento elevado, o custo do software acaba por se tornar ainda maior e a longo prazo torna-se muito menos sustentável. Muitas vezes faz-se inicialmente um grande investimento em software e passado alguns anos terá na mesma de se adquirir hardware novo.

Mas de que maneira é que deve ser feita esta consideração de hardware? Quantas máquinas são precisas para um sistema ser escalável? Cada caso é um caso e para responder a este tipo de perguntas existem duas maneiras diferentes de escalar um sistema: Escalabilidade Vertical e Escalabilidade Horizontal [27].

Ecalabilidade Vertical

Para escalar um sistema verticalmente pode-se começar por exemplo por um servidor bastante simples para o serviço que se pretende fornecer e outro para o servidor da base de dados. Quando alguma destas máquinas chega ao limite da sua capacidade substitui-se por uma máquina melhor, e será sempre assim sucessivamente até existir uma máquina com hardware suficientemente poderoso para responder satisfatoriamente a todos os testes a que foi submetido.

O ponto fraco da escalabilidade vertical é o custo associado a esta vertente, como podemos ver na figura 6.1, o aumento do custo não é proporcional ao aumento de processamento , RAM, disco e outros hardwares.

Conclui-se, com isto, que é muito mais barato escalar um sistema com várias máquinas menos poderosas do que ir aumentando as capacidade de uma só máquina.

O que torna os sistemas verticais bastantes apetecíveis é a facilidade da sua implementação, pois basta implementar o software numa máquina vulgar e assim que ele estiver pronto pode-se passar para uma "super máquina" e está pronto para entrar em produção. Assim cada vez que for necessário mais capacidade basta aumentar a capacidade da máquina e nunca mais vai ser necessário mexer no software. No entanto

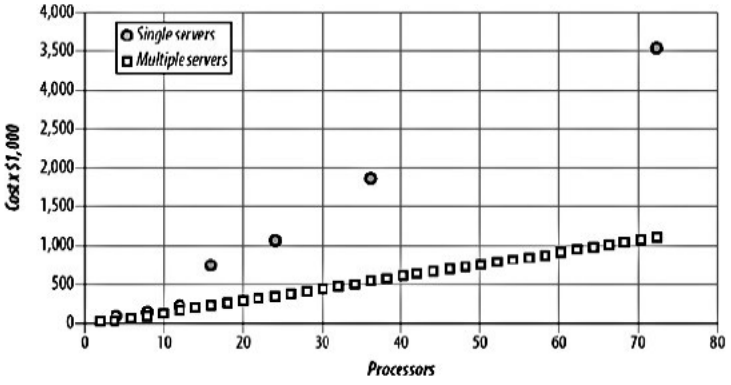

FIGURA 6.1: Escalar com máquinas pequenas VS escalar com máquinas grandes.

como já vimos na figura 6.1 o aumento do preço para sustentar estes sistemas torna-se exponencial.

Escalabilidade Horizontal

Tal como no modelo vertical, neste modelo também é feito um aumento de hardware. Embora neste caso em vez de aumentar a qualidade do hardware de uma só máquina, vão-se aumentando o número de máquinas que correm um dado serviço, assim são adicionadas máquinas até se garantir um resultado satisfatório.

No caso de se optar por escalar um sistema horizontalmente é importante considerar o tipo de máquinas que vão ser adquiridas tendo em conta a qualidade do hardware, o seu preço e o software que vão correr. Se estes pequenos detalhes forem tomados em conta é possível construir sistemas escaláveis a um relativo baixo custo.

Escalabilidade do PHP

Há alguns anos atrás o PHP não era uma linguagem que entrasse na equação quando se tratava de construir um sistema escalável, mas com o passar dos anos este tem ganho mais reputação e hoje em dia já são inúmeros os serviços que correm em servidores PHP.

Vejamos então como é que esta linguagem obedece aos três princípios fundamentais da escalabilidade [27]:

- **aumento de utilizadores**: o aumento de utilizadores pode ser facilmente satisfeito com o aumento de máquinas a correr o mesmo servidor PHP. Uma vez que

enquanto um processo de PHP está a processar requests não pode estar a comunicar com outros processos que estão a executar outras funções. Assumindo o exemplo em que é feito um request dos dados de um utilizador e estes dados vão ser enviados várias vezes nos próximos pedidos, então é usada a camada de dados para armazenar essa informação. Ou seja todos os processos são independentes dos requests que estão a ser feitos, podendo estes estar a receber respostas de vários servidores que prestam o mesmo serviço.

- **aumento dos dados do sistema**: o armazenamento de dados é independente do processamento de um serviço em PHP. É indiferente o crescimento de uma base de dados, uma vez que estes vão ser acedidos exactamente com as mesmas *querys*. A responsabilidade de lidar com este tipo de crescimento passa para a camada de dados.

- **facilidade na manutenção**: um servidor PHP é bastante fácil de manter, uma vez que é uma linguagem de fácil compreensão, existe bastante documentação, software disponível online e é uma linguagem bastante usada o que faz com que exista muita mão de obra capaz de trabalhar com este tipo de tecnologia.

Em suma, com o PHP é possível criar um servidor não só com bastante liberdade arquitetônica bem como liberdade na escolha e configuração ao nível da base de dados, de acordo com o número de utilizadores, dados armazenados e a política de manutenção mais adequada a um serviço.

Load Balancing

Load Balancing como o próprio nome indica é um método para distribuir os pedidos feitos a um sistema pelas várias máquinas que o constituem. Posto isto, é fulcral a existência de um Load Balancer quando se opta por seguir um modelo de escalabilidade horizontal, uma vez que neste modelo não existe nenhum sistema operativo capaz de distribuir os vários pedidos entre todas as máquinas do sistema.

Como podemos ver na figura 6.2, um load balancer pode ser representado por uma máquina que recebe todos os requests de todos os utilizadores do servidor e os distribui pelas máquinas do sistema de acordo com o algoritmo em uso. Assim todos os clientes comunicam com uma só máquina, isto é um único endereço de IP, mas têm ao seu dispor várias máquinas que podem responder ao seu pedido.

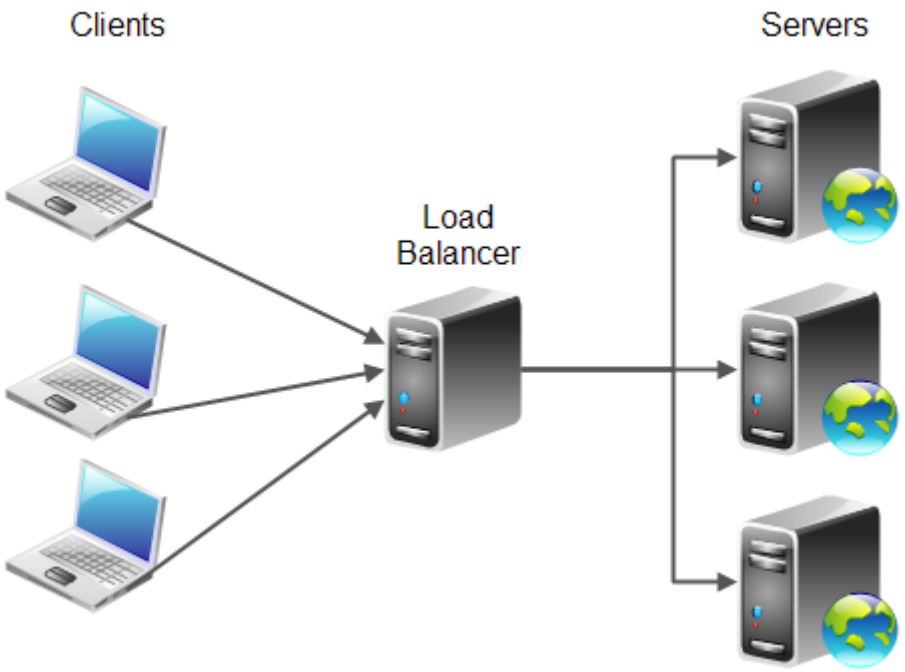

FIGURA 6.2: Esquema de funcionamento do Load Balancing.

É possível fazer load balancing com hardware ou com software [27].

- **Load Balancing com hardware:** quando se opta para por fazer load balancing com hardware é necessário adquirir um dispositivo de load balancing. Para além de estes dispositivos serem caros, são também difíceis de configurar em relação às alternativas existentes em software. A grande vantagem de optar por esta solução é que se torna muito mais fácil detectar o estado actual de uma máquina, se uma máquina morrer o load balancer tem automaticamente conhecimento disso e deixa de lhe enviar tráfego.

- **Load Balancing com software:** o load balancing com software é uma solução que pode e deve ser tomada em conta antes de se decidir investir num hardware de load balancing. Neste caso em vez de existir um dispositivo a correr um sistema operativo de load balancing, instala-se um software com a mesma finalidade numa máquina comum. Para além desta solução ficar muito mais em conta do que adquirir um load balancer físico existem muitos software open-source, desenvolvidos nas mais variadas linguagens que pode servir como solução para muitos sistemas.

Algoritmos de Load Balancing [23, 24]:

- **Round Robin:** este é o algoritmo de escalonamento mais utilizado, limita-se a reencaminhar cada um dos novos requests para o próximo servidor que estiver na sequência de execução. Este algoritmo deve ser principalmente utilizado quando as máquinas que estão a fazer de servidor têm todas igual capacidade de memória e de processamento.

- **Ratio:** neste algoritmo é definido um ratio para cada servidor. O número de conexões que cada servidor vai receber ao longo do tempo será proporcional ao seu nível de ratio. Esta solução é útil quando existem várias máquinas no sistema com diferentes características, pois nesse caso o ratio tem de ser ajustado de acordo com a capacidade de cada uma delas.

- **Dynamic Ratio:** este algoritmo tem um funcionamento semelhante ao anterior, só que neste caso o ratio é actualizado em tempo real pelo sistema, e tem em atenção as características de cada maquina, tal como a sua sobrecarga em cada instante. Este algoritmo é usado apenas por alguns softwares existentes no mercado, dentro dos quais está inserido o Windows Management Instrumentation.

- **Observed:** tal como no Ratio e no Dynamic Ratio, no uso deste algoritmo também é calculado um ratio mas neste caso é proporcional ao número de conexões concorrentes a cada maquina. Este algoritmo é útil para quando existe uma grande pool de servidores.

- **Least Response Time:** neste algoritmo cada pedido é reencaminhado para a máquina que tem o menor tempo de resposta por request. Este algoritmo deve-se utilizar quando existem máquinas com mais capacidades que outras no sistema.

Estes são apenas alguns exemplos de algoritmos para fazer Load Balancing, pois existem muitos mais. Estes cinco exemplos pretendem exemplificar várias abordagens de acordo com as características das máquinas pertencentes à pool de servidores.

6.2 Definição dos Cenários de Teste e Indicadores a Considerar

Ao longo deste capítulo vão ser apresentadas todas as características do sistema, desde o Load Balancer utilizado, a ferramenta de Benchmarking, as características dos servidores e apresentação da bateria dos testes que foram corridos.

Optou-se por usar o modelo de escalabilidade horizontal e usar um software de Load Balancing que não fosse dispendioso, com o objectivo de construir um sistema escalável com o menor custo possível.

Network Load Balancing Manager

O NLB Manager é um software da Microsoft que é compatível com o sistema Microsoft Windows Server. O algoritmo usado pelo NLB é semelhante ao tradicional Round Robin, mas acaba por ter um funcionamento mais completo uma vez que o Round Robin não tem nenhum mecanismo que verifique a disponibilidade de um dado servidor. Assim, se um servidor falhar o Load Balancer vai continuar a enviar-lhe trafego até que o administrador da rede remova aquele servidor da pool. No caso do uso do NLB Manager este problema deixa de existir, uma vez que este contem um mecanismo de controlo de falhas e bottlenecks.

Uma outra vantagem do Network Load Blancing é o facto deste software não ter especificado nenhum tipo de exigências mínimas de hardware, podendo ser instalado em qualquer máquina. Este factor faz com que os custos associados se tornem mais reduzidos na construção e configuração de um servidor [25, 26].

Os motivos mencionados anteriormente acrescidos ao facto de os servidores da empresa serem Microsoft Windows Server, e por isso este software ser gratuito, foram as razões por as quais este Load Balancer ser escolhido para este projeto.

Na configuração deste software inicialmente é criado um novo cluster na rede, ao qual vão pertencer as máquinas que irão correr o software do servidor. Os servidores são adicionados ao cluster através do seu IP e também é definido um IP virtual que representará o cluster. Este novo IP será o que os clientes usarão para comunicar com o servidor. Um outro campo que é necessário configurar é o Cluster Operation Mode. Este parâmetro pode ser configurado em Unicast ou Multicast, em ambos os casos apenas uma máquina

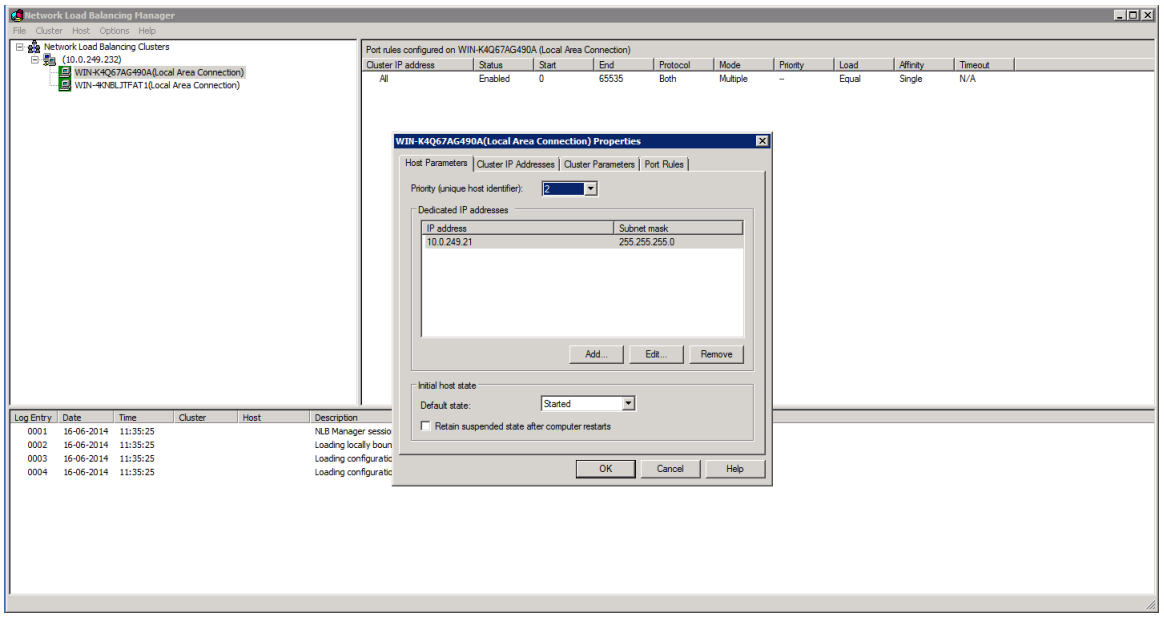

FIGURA 6.3: Network Load Balancing

responderá ao request feito por um cliente. No entanto, no modo Unicast é criado um novo MAC Address que representa os hosts pertencentes a um cluster e quando é feito um pedido todas as máquinas irão receber esse pedido. No caso do modo Multicast este novo MAC Address não é criado e o modo de reencaminhamento dos requests é o habitual. Também podem ser controlados os acessos às portas do cluster através das Port Rules.

Ferramentas de Benchmarking

Existem alguns softwares de benchmarking que foram desenvolvidos com o intuito de avaliar a performance de servidores WebDAV. Para este projeto foram testados alguns deles, os mais robustos foram o Tsung [28] e o Prestan [29]. Verificou-se que estas duas ferramentas de benchmarking não eram suficientemente específicas para o servidor CalDAV deste projeto e não mostravam muita capacidade de serem configuradas para este caso especifico. Algumas não implementavam a autenticação digest e outras eram bastante incompletas na construção dos requests.

Devido aos motivos mencionados anteriormente foi alterado um projecto referente a uma biblioteca para um cliente CalDAV. Este projeto chama-se Twisted Caldav [30] e foi desenvolvido na linguagem Ruby.

O Twisted Caldav foi alterado de forma a ter dois modos de execução, um dos modos em que é criada uma pool de threads e são todas executadas em simultâneo, cada uma delas representando um cliente que pretende interagir com o servidor. No outro modo é possível programar um conjunto de threads que são criadas e executadas de n em n segundos. Este cliente também foi alterado para medir o tempo que cada request de cada thread demora a obter resposta do servidor.

Apresentação das maquinas do sistema

Na tabela 6.1 irão ser apresentadas as máquinas que fizeram parte dos testes de escalabilidade do sistema:

Máquina	**Papel**	**Sistema Operativo**	**Memória**	**Processador**
C_A	Cliente	Ubuntu 12.04.4	512 Mb	Intel(R) Xeon(R) CPU E5-2670 0 @ 2,60GHz
C_B	Cliente	Ubuntu 12.04.4	512 Mb	Intel(R) Xeon(R) CPU E5-2670 0 @ 2,60GHz
S_A	Servidor	Windows Server 2008 R2 Enterprise	4,00 Gb	4 x Intel(R) Xeon(R) CPU E5-2670 0 @ 2,60GHz
S_B	Servidor	Windows Server 2008 R2 Enterprise	4,00 Gb	4 x Intel(R) Xeon(R) CPU E5-2670 0 @ 2,60GHz
S_BD	Servidor (BD)	Windows Server 2008 R2 Enterprise	24,00 Gb	8 x Intel(R) Xeon(R) CPU X5675 @ 3,07GHz

Tabela 6.1: Máquinas do Sistema.

Apresentação dos testes

Antes de serem apresentados os testes a que os servidores vão ser sujeitos é importante explicar o panorama de utilização dos servidores em geral da MedicineOne.

Quando uma organização de saúde pretende aderir ao software da MedicineOne, pode optar por usar um servidor interno à organização e configurar todos os clientes MedicineOne para acederem a esse servidor, ou então aderir ao MedicineOne Cloud em que nesse caso a aplicação usa como servidor a Cloud da MedicineOne.

O servidor CalDAV desenvolvido neste estágio quando entrar em produção vai ter de ser inserido nos servidores particulares dos clientes MedicineOne e também na Cloud. Uma

vez que a Cloud da MedicineOne é o servidor que tem mais clientes e acessos, estes foi o servidor padrão usado para definir os números de clientes que vão ser utilizados nestes testes.

Existem dois tipos de cenários de utilização em que um cliente do servidor CalDAV se pode inserir:

- o primeiro request que é feito ao servidor, em que este tem de transferir todos os eventos do calendário daquele utilizador;
- sempre que há uma alteração, edição ou remoção de um evento a informação desse evento vai ser actualizada;

Como podemos ver, a quantidade de dados que têm de ser transferidos no primeiro caso de utilização será muito superior ao do segundo, por esse motivo nos testes de escalabilidade o cenário de utilização que vai ser utilizada será o primeiro, ou seja, todos os utilizadores que se vão ligar ao servidor vão fazer o download de todos os eventos do seu calendário.

O número máximo de utilizadores da aplicação MedicineOne que comunica com a Cloud tem diariamente picos de navegação que se aproximam dos 2000 utilizadores simultâneos e a média de eventos que cada utilizador têm nos seus agendamentos é de 600 (aproximadamente 300 Kb).

Será com base nestes números que os testes vão ser construídos.

Para por em execução estes testes vão ser usados dois workflows de utilização:

- **Big-Bang**: todas as threads representativas dos clientes enviarão o request para o servidor ao mesmo tempo;
- **Ramp-Up**: de n em n segundos serão iniciadas várias threads havendo assim um número de utilizadores concorrentes crescente;

Estes testes serão corridos para vários tipos de cenários, com diferentes números de threads, num cenário single server, outro dual server, e dual server em que existe uma falha num dos servidores (Figura 6.4).

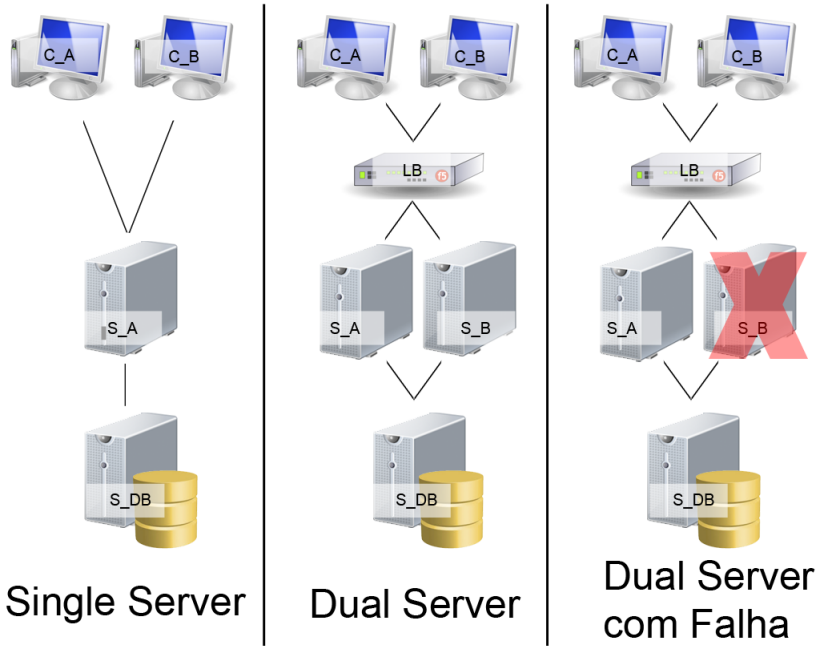

FIGURA 6.4: Esquema dos servidores em uso nos testes

workflow	**Número de Threads**
Big-Bang	500
	1000
	2000
Ramp-Up	500 (10 Th de 2 em 2s)
	500 (50 Th de 20 em 20s)
	1000 (50 Th de 2 em 2s)
	1000 (200 Th de 20 em 20s)
	2000 (100 Th de 2 em 2s)
	2000 (400 Th de 20 em 20s)

TABELA 6.2: Valores de input para os testes.

Na tabela 6.2 estão os inputs de cada teste que serão executados para os vários cenários:

Para conseguir classificar a escalabilidade do sistema serão medidas e consideradas as seguintes grandezas:

- Número de bytes transitados por segundo entre o cliente e o servidor;
- Número de utilizadores concorrentes no servidor;
- Tempo médio por request medido do lado do cliente;
- Utilização de CPU nos servidores;

- Utilização da memória nos servidores;
- Utilização de CPU no servidor da Base de Dados;
- Utilização da memória no servidor da Base de Dados;

6.3 Resultados dos Cenários Considerados

Resultados dos testes em Single Server:

Single Server									
Workflow	Nº Threads		Bytes/s	Tempo		Servidor		Base de Dados	
				Calendário (s)	Evento (ms)	CPU (%)	Memória (%)	CPU (%)	Memória (%)
Big-Bang	500		2172381	53.1	89	77.3	11.6	2.6	76.9
	1000		2668563	96.03	160	92.3	10.6	5.4	73.9
	2000		2756545	188.3	314	94.14	11.17	7.93	74.14
Ramp-Up	500	10/2s md=10	2193855	2.2	4	75.3	10.9	5.7	73.8
		50/20s md=11	1323312	7.13	12	46.2	13.8	9.3	73.9
	1000	50/2s md=395	2751313	79.82	133	96.11	10.6	16.95	73.57
		200/20s md=280	2499816	58.65	98	88.6	10.69	7.36	73.33
	2000	100/2s md=858	2699419	172	287	93.6	10.59	6.98	72.79
		400/20s md=767	2685948	152.1	254	93.42	10.55	6.46	72.07

Tabela 6.3: Resultado dos testes para o Single Server.

Antes de analisar os resultados é importante referir que a máquina que corre a base de dados está a correr todas as bases de dados de testes da MedicineOne, mas tendo em conta as capacidades da mesma a nível de memória e CPU as oscilações em relação aos valores habituais não foram nada significativas, uma vez que os recursos desta máquina sem o servidor CalDAV estar a correr variam entre os 70% e os 74% de utilização da memória e entre os 0% e os 1,6% de consumo do CPU.

Começando agora por analisar os resultados do Big-Bang, pode-se concluir que o número médio de bytes por segundo não variou muito, o que é natural pois as condições da rede nos três testes são as mesmas e os requests foram todos satisfeitos sem interrupções nem outro tipo de perturbações. Por outro lado o tempo médio da transmissão de um calendário (600 eventos), aumentou para praticamente o dobro de Big-Bang 500 para o Big-Bang 1000 e voltou a duplicar para o Big-Bang 2000. Também é possível perceber este aumento de carga ao analisar o servidor, pois os valores do CPU no primeiro teste rondam a média de 77,3% e nos outros dois casos já passam os 90%. Já a nível de memória os valores não são nada preocupantes uma vez que se mantêm sempre baixos e sem grandes variações.

Ao analisar os três gráficos dos testes Big-Bang, presentes no Anexo A, figura A.1 , figura A.2 e figura A.3 fica a ideia de que são gráficos muito semelhantes, mas tendo em conta as diferenças das escalas dá para perceber o aumento do tempo a satisfazer os pedidos com a observação da diminuição dos utilizadores concorrentes. Também é possível constatar que nos três cenários os picos atingidos pelo numero de bytes/s transferidos não variam muito na ordem de grandeza,

Analisando agora os resultados do Ramp-Up, para os dois testes com 500 threads, no primeiro caso com 10 threads de 2 em 2 segundos o tempo que um cliente demora a ter resposta é de 2,2s, o que é bastante bom, dá uma média de 4 milisegundos por evento de um calendário. Também é possível concluir que os requests são despachados do servidor com sensivelmente a mesma cadencia que vão chegando, pois o número médio de utilizadores concorrentes é 10. O processamento do servidor teve um valor médio de utilização de 75,3%, melhorando apenas 2% em relação ao Big-Bang 500.

No cenário de 50 utilizadores de 20 em 20 segundos, o número médio de utilizadores por segundo manteve-se na mesma ordem de grandeza, neste caso foram 11. O valor do tempo por calendário aumentou um pouco, isto deve-se ao facto de neste caso existirem picos com mais utilizadores concorrentes. Ao nível do CPU do servidor o valor médio de utilização diminuiu para 46,2% devido aos requests terem sido mais pausados.

Analisando agora os gráficos dos testes Ramp-Up 500 do Anexo A, figura A.4 e figura A.5, correspondendo ao primeiro e segundo teste respectivamente. Reparamos que no primeiro gráfico de facto o número de utilizadores concorrentes é mais ou menos constante, mas estes nunca chegam a desaparecer antes de chegarem mais pedidos. Pelo contrário no segundo caso de teste surgem 50 utilizadores concorrentes e são rapidamente atendidos fazendo com que existam pausas entre as próximas 50 threads a iniciar em que não são atendidos clientes nenhuns. Por este motivo é que reparamos que o número médio de bytes por segundo diminuiu em relação aos valores dos restantes testes.

Tanto nos testes Ramp-Up para 1000 como para 2000 utilizadores, percebemos que o servidor lida melhor com um aumento de carga maior mas com um pausa mais alargada, do que com um aumento de carga menor e uma pausa também menor. Também é possível ver que os aumentos dos tempos médios de transferencia foram muito agravados e o processamento do servidor já voltou a rondar os 90%.

Com a comparação dos gráficos do Anexo A, figura A.6, figura A.7, figura A.8 e figura A.9 conclui-se de facto que uma pausa maior entre o arranque das threads vai de facto fazer com que o número médio de utilizadores concorrentes seja menor e por esse motivo o servidor irá ter uma menor carga de trabalho.

Ao longo de todos os testes, a máquina que corre a Base de Dados só sofreu variações significativas na utilização do CPU em relação aos valores padrão, mas mesmo assim não foram valores muito elevados.

Resultados dos testes em Dual Server:

Dual Server													
Workflow	Nº Threads			Bytes/s		Tempo		Servidor				Base de Dados	
						Calendário (s)	Evento (ms)	CPU (%)		Memória (%)		CPU (%)	Memória (%)
Big-Bang	500			1821197	2005320	26.51	44	64.3	72.3	10.7	12.48	5.96	72.44
	1000			1760869	1335288	49.08	82	60.2	47.81	16.95	11.69	6.8	73.04
	2000			2318640	2125478	98.31	164	84.3	75.5	13.6	11.18	12.44	72.67
Ramp-Up	500	10/2s	md=2|2	1128687	1025570	1.1	2	37.1	36.42	10.13	10.3	6.4	72.23
		50/20s	md=2|2	659850	585770	4.32	7	22.9	20.6	11.8	13.14	4.53	72.2
	1000	50/2s	md=140|141	2408230	2156775	31.5	53	79.52	74.13	13.94	13.98	9.96	72.23
		200/20s	md=50|46	1655116	1556441	12.21	20	55.12	54.54	13.4	12.86	7.86	72.24
	2000	100/2s	md=371|379	2192530	2021211	74.18	124	72.88	69.62	12.7	11.55	9.59	72.26
		400/20s	md=270|265	2580581	2355511	54.63	91	85.47	82.3	11.61	10.78	10.88	72.23

Tabela 6.4: Resultado dos testes para o Dual Server.

Nos testes com Dual Server, tabela 6.4 , com o cenário Big-Bang os resultados melhoraram significativamente, o tempo de resposta reduziu para cerca de metade em relação ao single server. Os níveis de processamento dos dois servidores também reduziram, demonstrando assim também a diminuição de carga em cada um deles.

Através da análise dos gráficos do Anexo A, da figura A.10 até à figura A.15, concluí-se que cada servidor recebeu cerca de metade dos requests e acabam por ter uma performance semelhante, algo que também era expectável pelo facto das máquinas terem as mesmas características.

Passando para os resultados com o Ram-Up também se conseguem observar bastantes melhorias, o nível máximo de processamento que uma máquina atinge é de 85,47% isto novamente mostra uma melhoria na sobrecarga das máquinas. Os tempos também reduziram bastante em relação aos resultados em Single Server, nos testes de carga mais baixa desceram para cerca de metade e nos de carga mais elevada para cerca de um terço. Os gráficos referentes a estes testes estão no Anexo A desde a figura A.16 até à figura A.27. Ao comparar os gráficos dos testes em Dual Server com os do Single Server concluísse que o número médio de utilizadores concorrentes é muito menor e que todas as threads são atendidas mais rapidamente.

Mais uma vez os valores medidos na máquina que corre a Base de Dados não sofreram muitas alterações, o processamento aumentou ligeiramente devido ao facto de estarem dois servidores a consumirem dados da mesma máquina.

Falha de um servidor

Para testar como se comportaria o sistema caso existisse uma falha num servidor foi usado o cenário Big-Bang com 1000 threads em Dual-Server. Com este cenário foram feitos 2 testes:

- matar um servidor durante a execução dos pedidos para verificar se o Load Balancer reencaminha todo o trafego para o outro servidor;
- correr novamente os testes com apenas um servidor para registar os resultados;

Os resultados do primeiro teste são possíveis observar nos gráficos do Anexo A, figura A.28 e figura A.29, onde se vê que passado poucos segundos de um servidor ir a baixo o outro tem imediatamente um aumento significativo no número de utilizadores concorrentes.

No caso do segundo teste os resultados foram registados no gráfico figura A.30 e os valores médios de tempo por request de calendário foram de 99,65s e o numero de bytes por segundo foi de 2535627. Tanto o gráfico como estes valores são parecidos com os resultados Big-Bang 1000 em Single Server

6.4 Conclusões dos Testes

Depois de uma análise cuidada dos valores e dos gráficos de todos os testes de escalabilidade é possível concluir qual será a melhor solução a implementar num servidor de CalDAV MedicineOne, no entanto não existe uma solução óptima para todas as situações em que este servidor terá de ser instalado.

O exemplo de estudo foi o da Cloud da MedicineOne, e mesmo que sejam atingidos picos diários de utilização de 2000 utilizadores estes nunca estarão todos a realizar a mesma operação, muito menos a pedirem todos o seu calendário pela primeira vez, isso daria um total de 1.200.000 consultas pedidas no mesmo instante o que será praticamente impossível de atingir. Na realidade até o cenário dos 500 requests simultâneos parece bastante improvável de acontecer. Mas uma vez que a MedicineOne está a entrar no mercado brasileiro e isso está a resultar num aumento significativo no número de utilizadores, os testes foram desenrolados na mesma com esta ordem de grandeza.

Com este exemplo de estudo conclui-se que a solução Dual Server apresentou sem dúvida melhores resultados do que a Single Server e para além das melhorias nos resultados também resultou na construção de um sistema com controlo de falhas caso um servidor fique indisponível. No entanto se o servidor CalDAV for instalado numa clinica pequena, ou até mesmo num hospital com apenas algumas dezenas de médicos uma solução Single Server com uma máquina com as mesmas características das deste projeto chegará perfeitamente para responder a todos os pedidos daqueles utilizadores. No caso da organização crescer pode então utilizar-se a mais valia da escalabilidade horizontal e acrescentar mais uma máquina e configurar novamente o Load Balancer.

Capítulo 7

Conclusão

Trabalhar num projeto real numa empresa revelou-se ser uma ótima maneira de realizar um estágio curricular, pois o estagiário pode trabalhar numa empresa de uma área do seu gosto e ao mesmo tempo estar a aperfeiçoar as suas competências em engenharia de software. Para além disto, este tipo de estágio, também é útil para melhorar a perceção de como é que um projeto de software realmente é estruturado e de todos os processos essenciais para o seu desenvolvimento.

Existiram várias alterações em relação ao estágio inicial, e por esse motivo houve certas funcionalidades que eram suposto serem obrigatórias implementar segundo plano inicial que passaram a ser secundárias, acabando assim este estágio por se focar não só no cliente MedicineOne mas também na construção do servidor CalDAV e no estudo da sua escalabilidade. Estas alterações acabaram por aumentar um pouco mais o âmbito do projecto, mas também o tornaram mais interessante, com conteúdos mais diversificados o que resultou na construção de um sistema bastante mais completo.

Embora a aplicação iPad não tenha ficado completa, acabaram por ficar poucas funcionalidades por implementar que mal existam as condições necessárias para o seu funcionamento serão facilmente concluídas. Já em relação ao servidor CalDAV todos os requisitos foram satisfeitos.

O problema inicial que moveu este projecto foi combatido, pois agora já existe um módulo de agendamento que será lançado brevemente e permitirá a um médico gerir todas as suas consultas na sua aplicação MedicineOne para iPad. E também existe um servidor CalDAV que garante os requisitos de escalabilidade, interoperabilidade e

tolerante a falhas para que seja possível ter acesso à agenda profissional em qualquer software que suporte o protocolo CalDAV.

Como trabalho futuro deste projeto fica a implementação das três funcionalidades incompletas na aplicação móvel, e a realização dos testes finais de desempenho da aplicação e de usabilidade, quando esta estiver concluída.

Anexo A

Gráficos dos Testes de Escalabilidade

Neste anexo estão representados os testes exportados do Performance Monitor do WS. A azul os utilizadores concorrentes (escala 1:1) e a verde os bytes/s (escala 10000:1).

A.1 Testes em Single Server

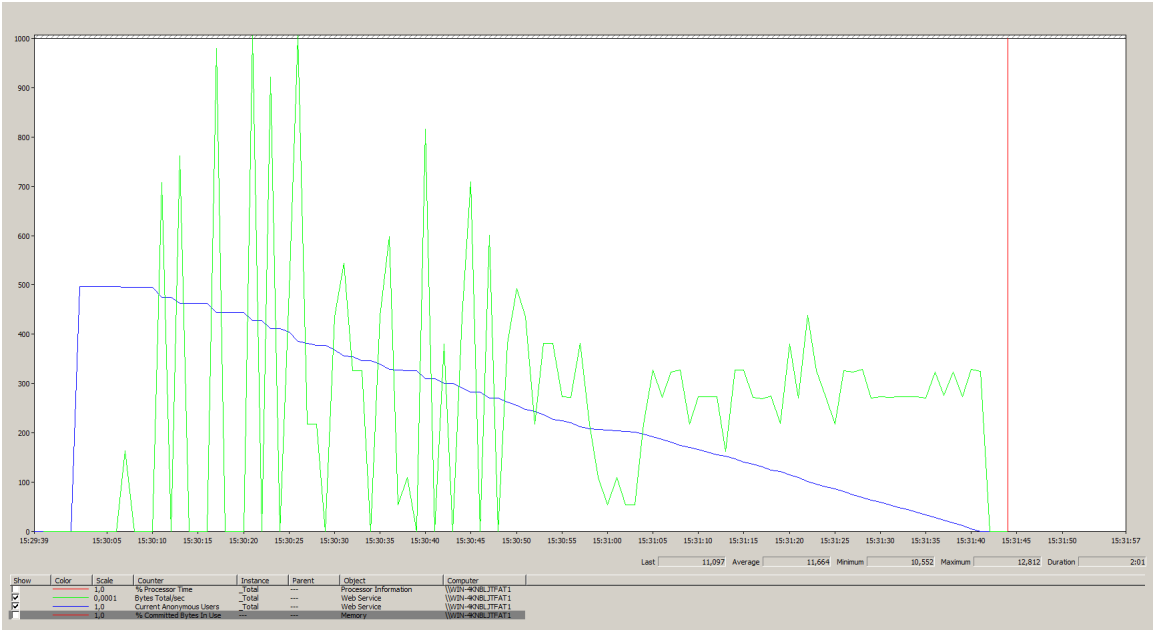

FIGURA A.1: Gráfico do teste Big-Bang 500 em Single Server

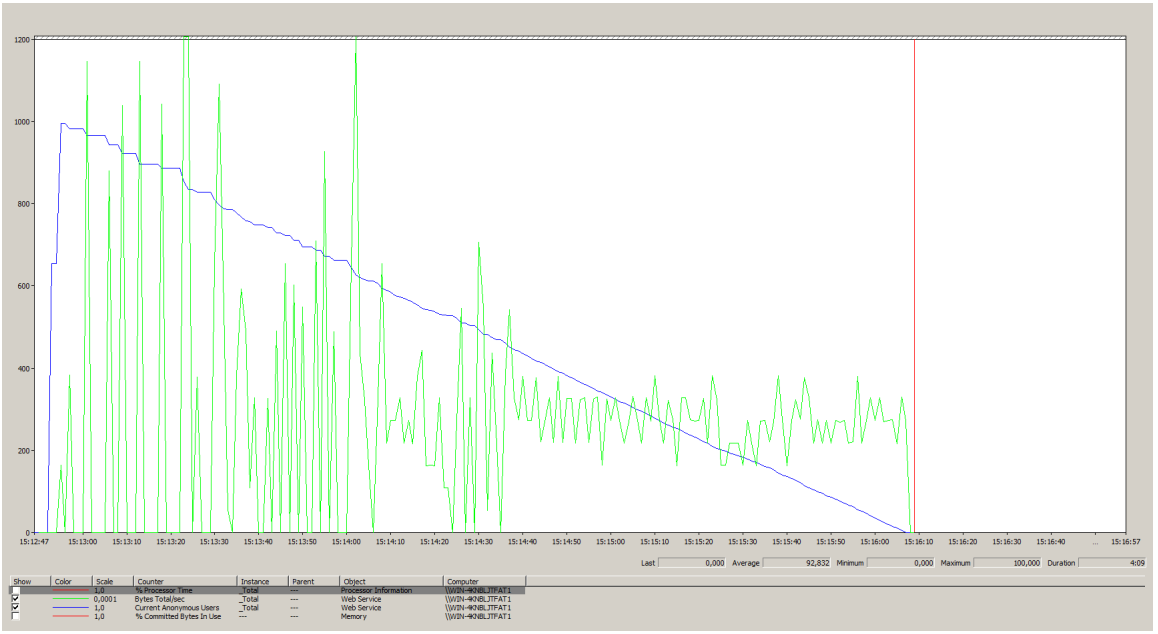

FIGURA A.2: Gráfico do teste Big-Bang 1000 em Single Server

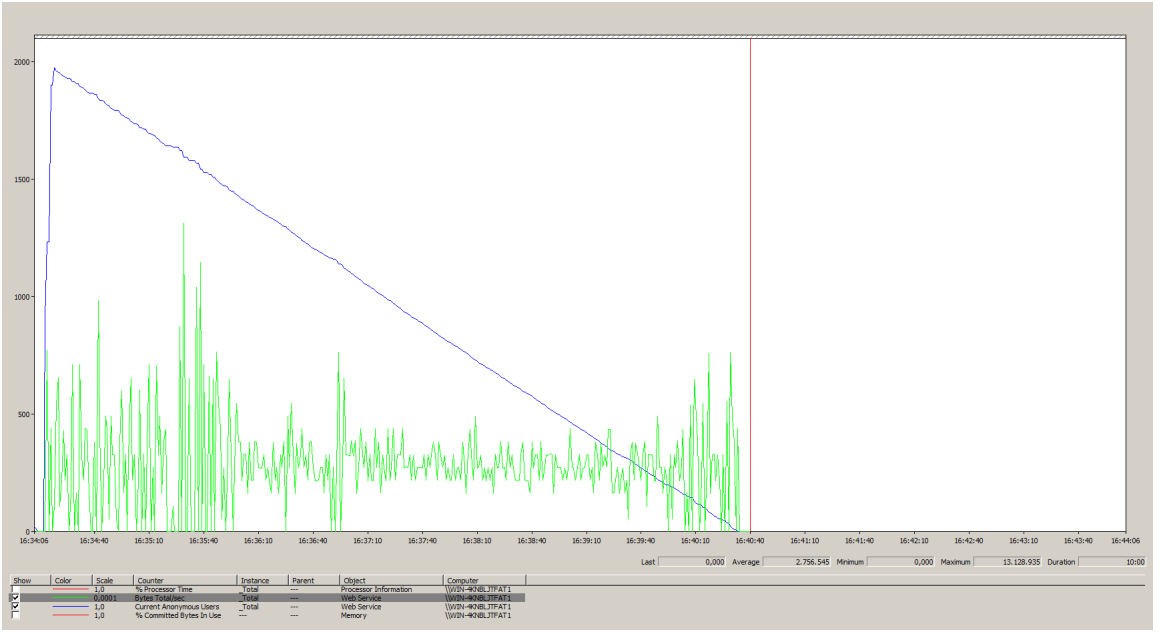

FIGURA A.3: Gráfico do teste Big-Bang 2000 em Single Server

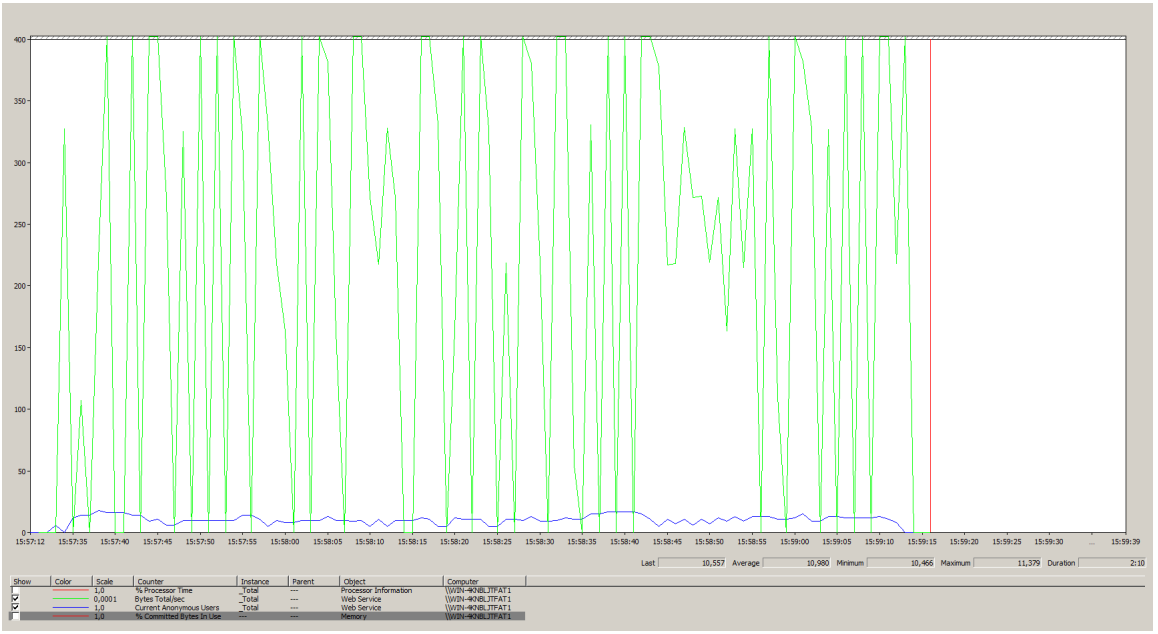

FIGURA A.4: Gráfico do teste Ramp-Up 500 em Single Server com 10 Threads de 2 em 2 segundos

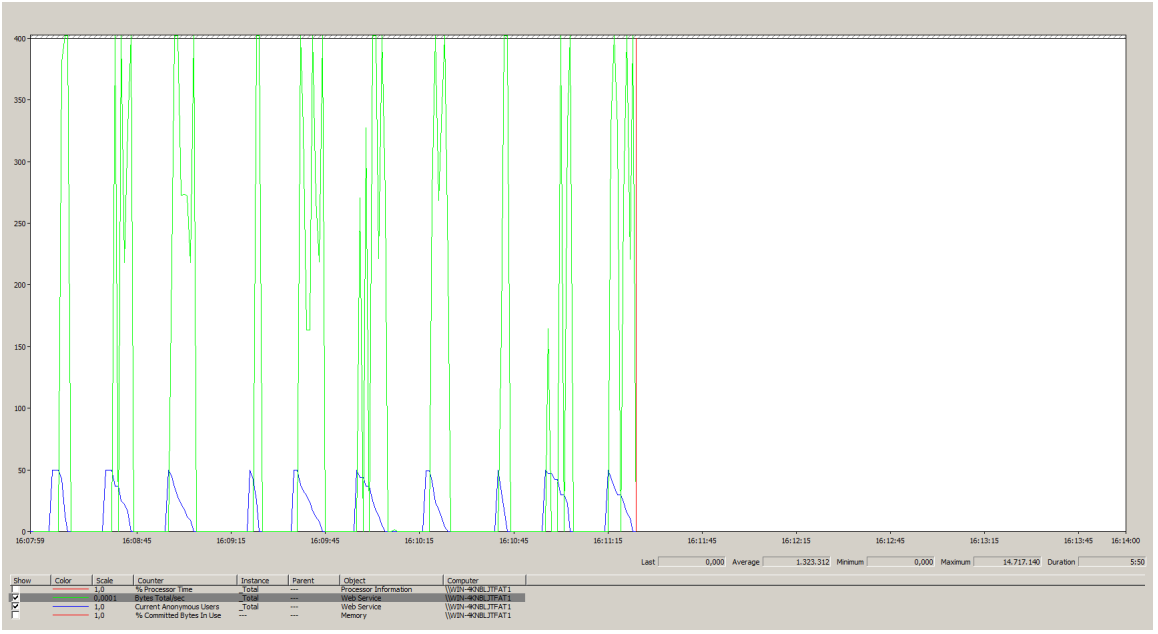

FIGURA A.5: Gráfico do teste Ramp-Up 500 em Single Server com 50 Threads de 20 em 20 segundos

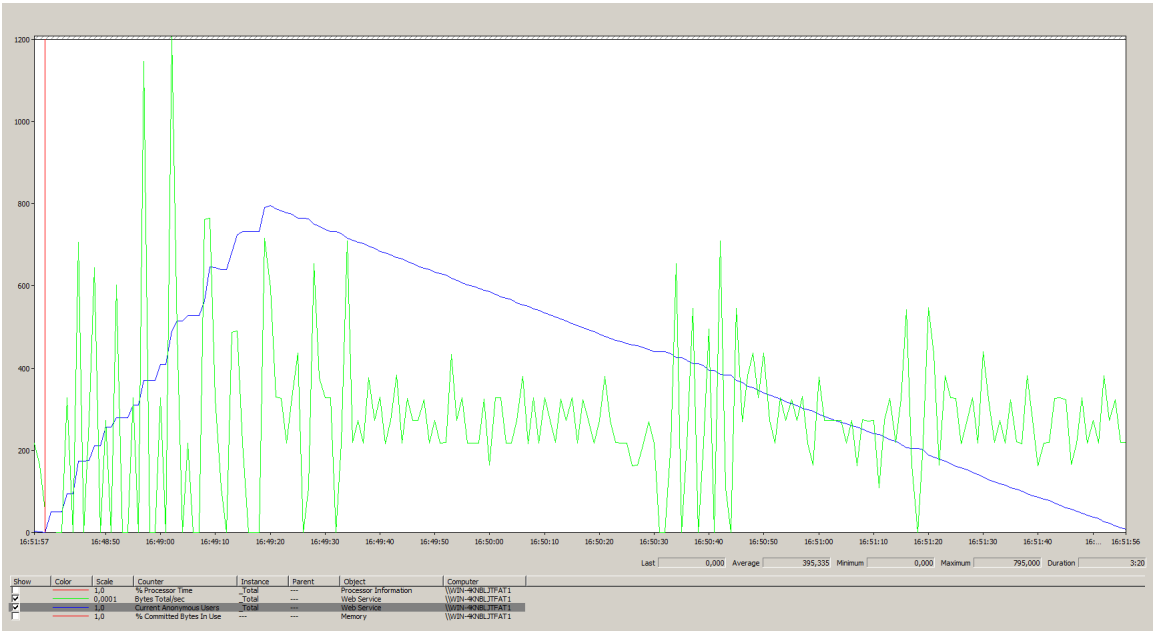

FIGURA A.6: Gráfico do teste Ramp-Up 1000 em Single Server com 50 Threads de 2 em 2 segundos

FIGURA A.7: Gráfico do teste Ramp-Up 1000 em Single Server com 200 Threads de 20 em 20 segundos

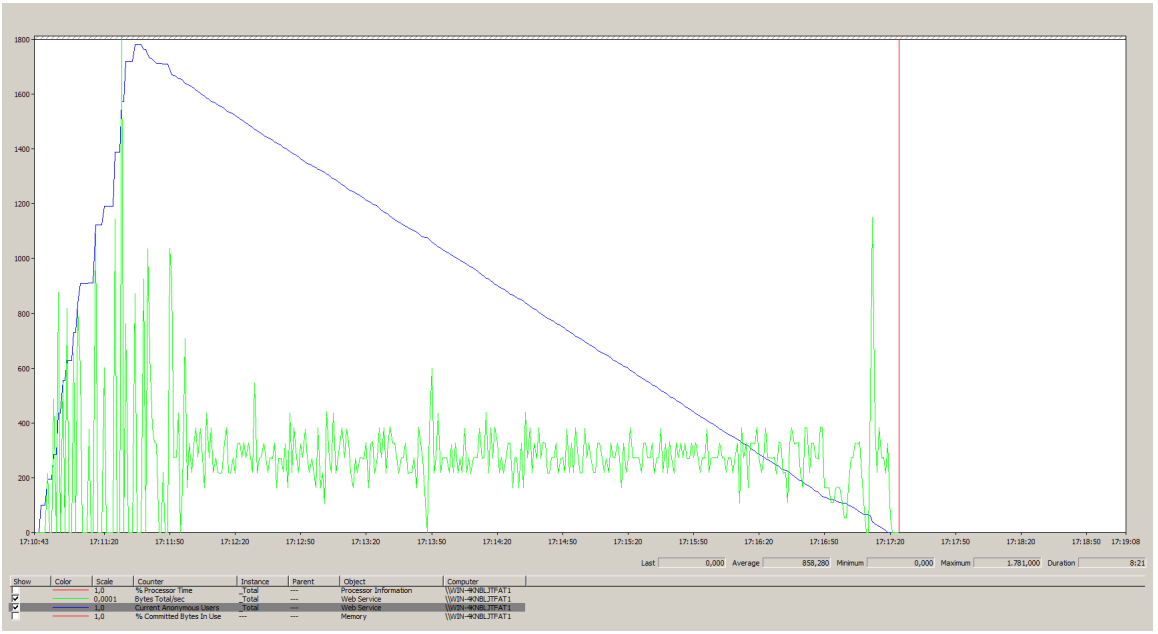

FIGURA A.8: Gráfico do teste Ramp-Up 2000 em Single Server com 100 Threads de 2 em 2 segundos

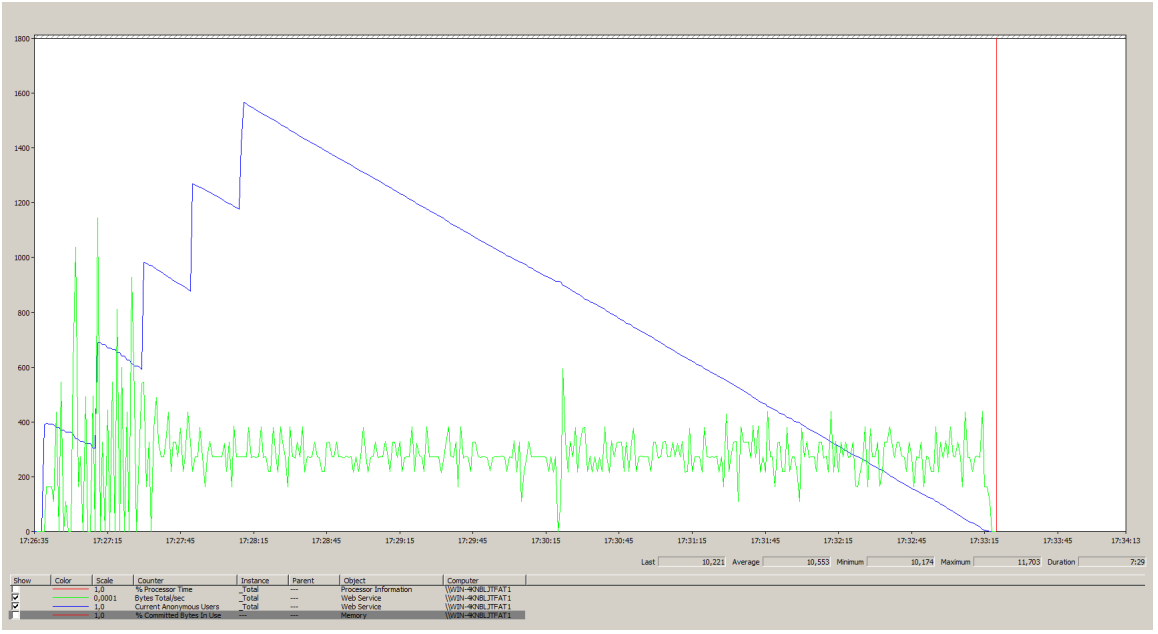

FIGURA A.9: Gráfico do teste Ramp-Up 2000 em Single Server com 400 Threads de 20 em 20 segundos

A.2 Testes em Dual Server

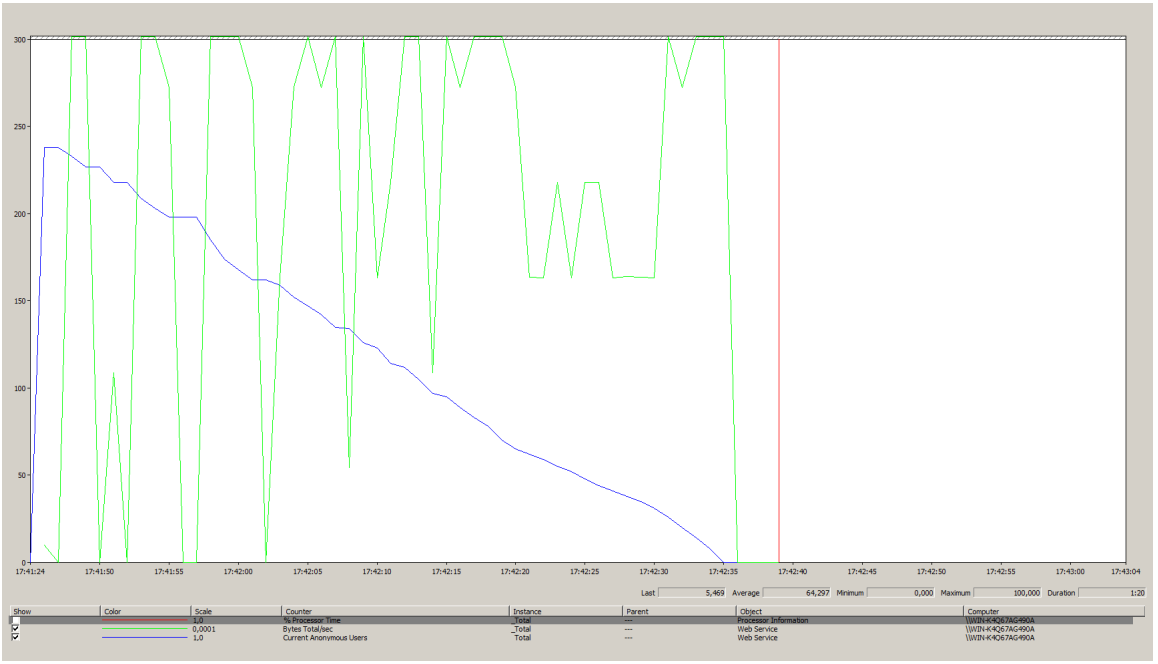

FIGURA A.10: Gráfico do teste Big-Bang 500 em Dual Server (Servidor S_A)

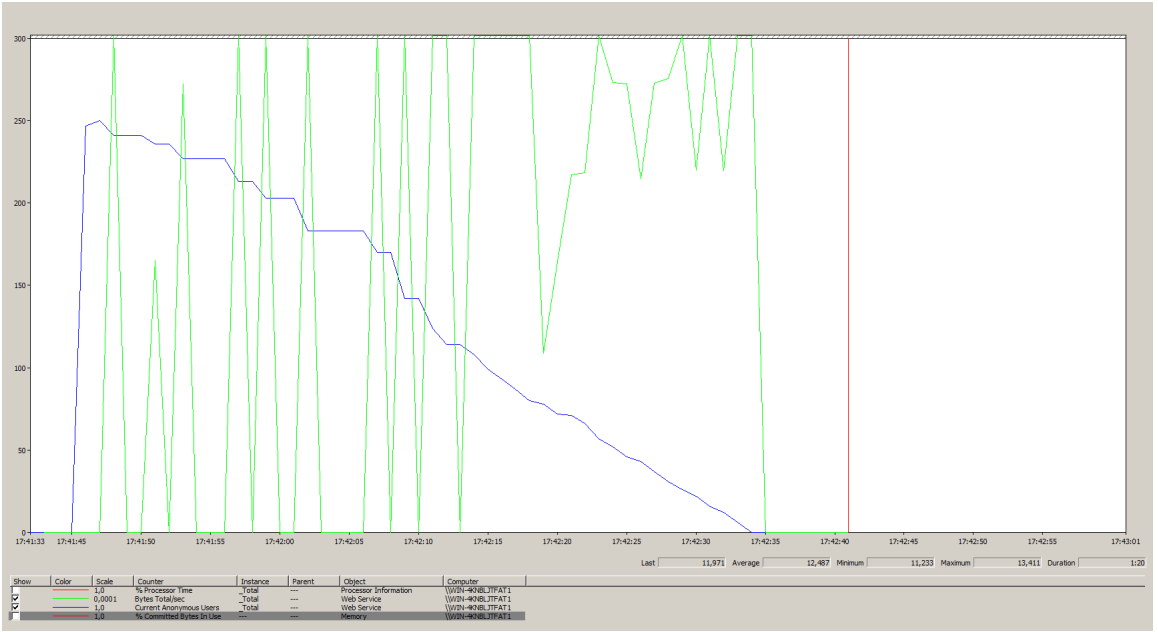

FIGURA A.11: Gráfico do teste Big-Bang 500 em Dual Server (Servidor S_B)

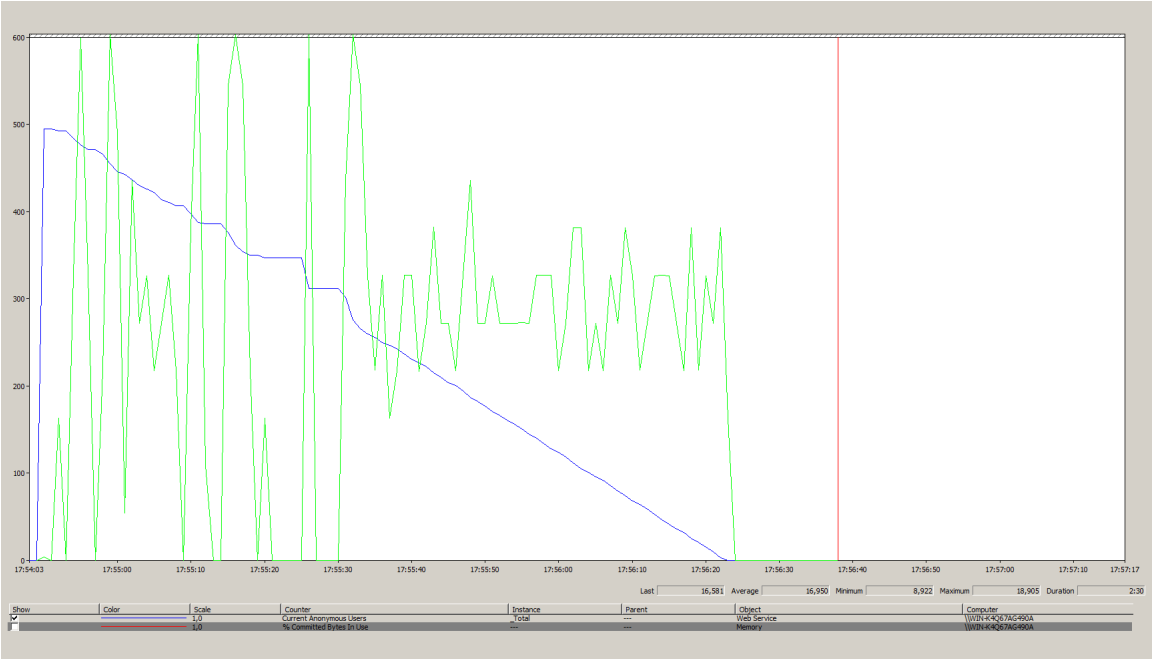

FIGURA A.12: Gráfico do teste Big-Bang 1000 em Dual Server (Servidor S_A)

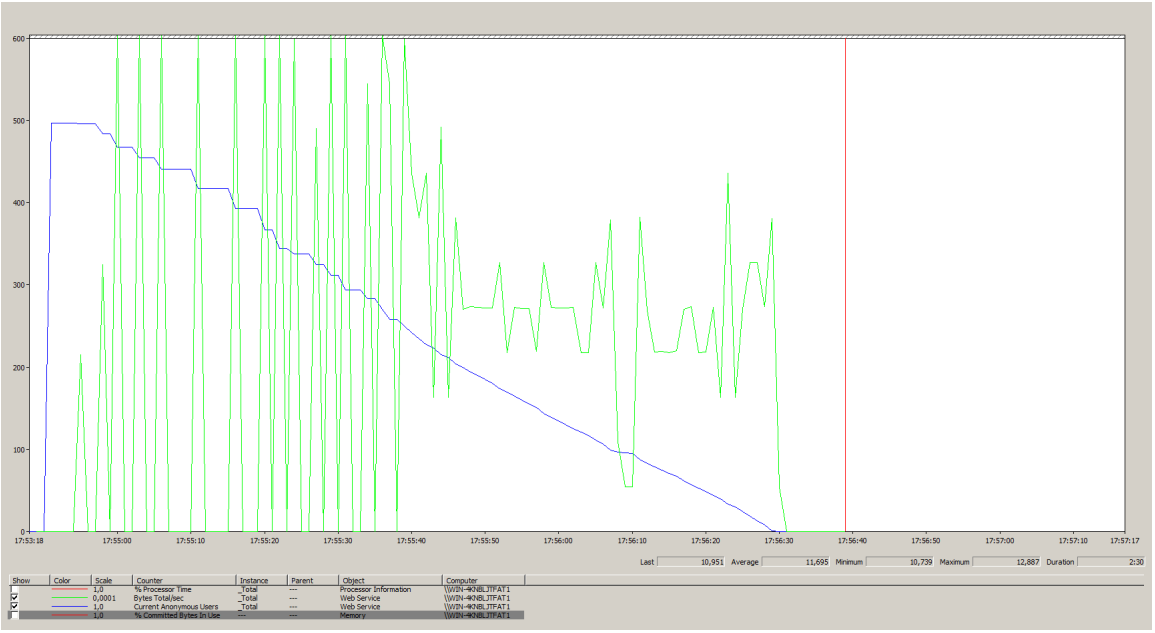

FIGURA A.13: Gráfico do teste Big-Bang 1000 em Dual Server (Servidor S_B)

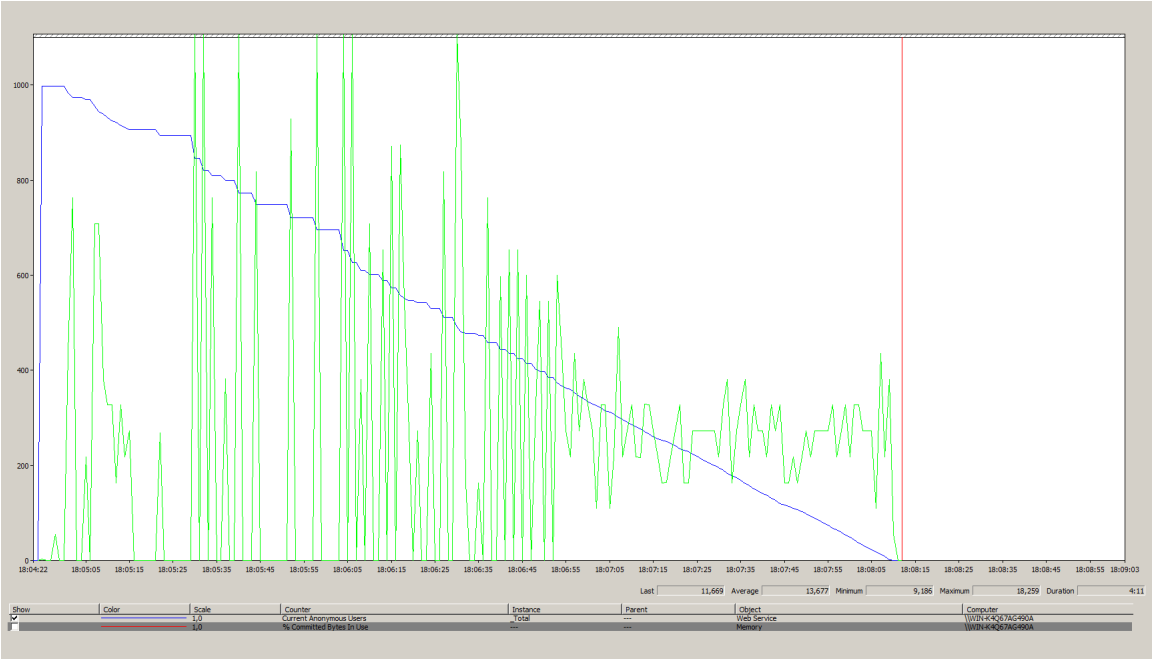

FIGURA A.14: Gráfico do teste Big-Bang 2000 em Dual Server (Servidor S_A)

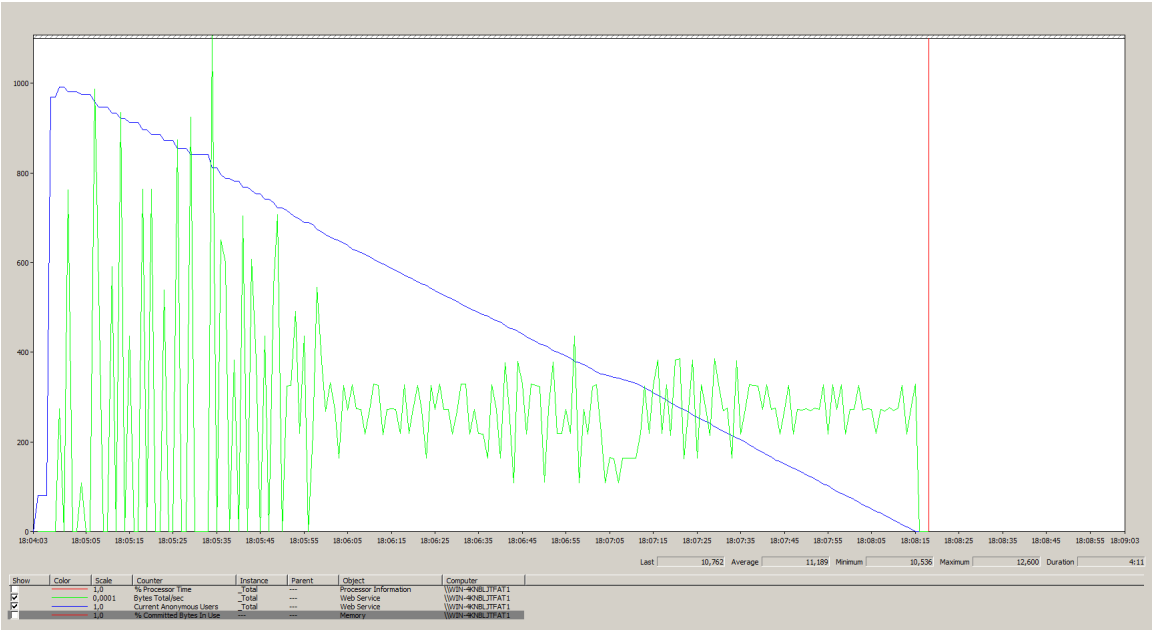

FIGURA A.15: Gráfico do teste Big-Bang 2000 em Dual Server (Servidor S_B)

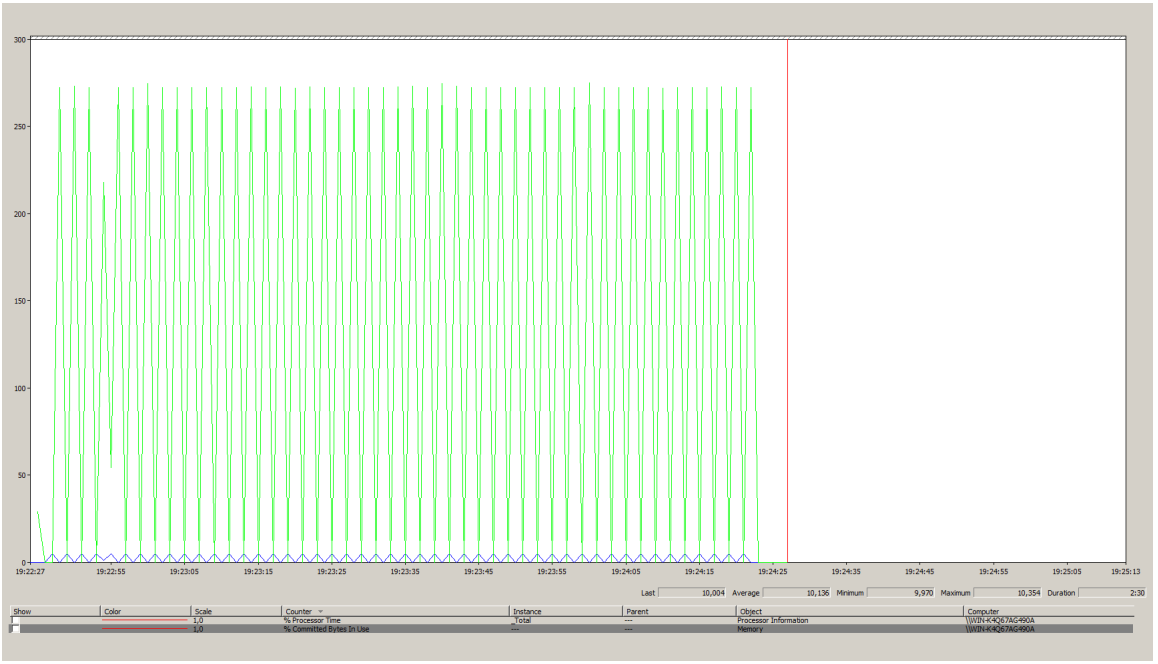

FIGURA A.16: Gráfico do teste Ramp-Up 500 em Dual Server com 10 Threads de 2 em 2 segundos (Servidor S_A)

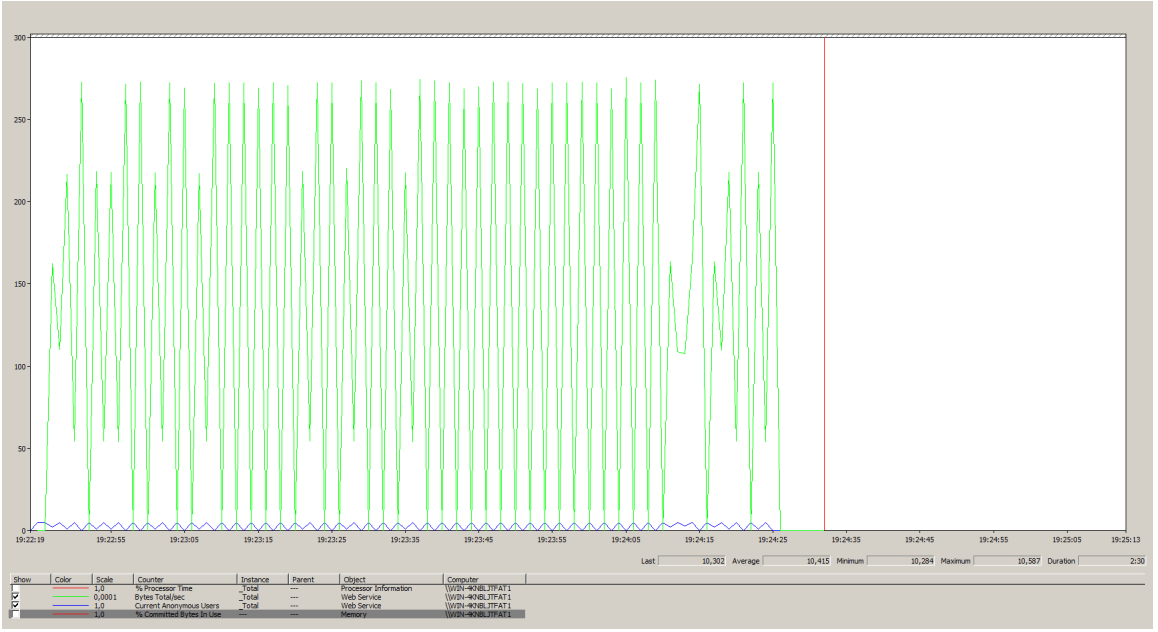

FIGURA A.17: Gráfico do teste Ramp-Up 500 em Dual Server com 10 Threads de 2 em 2 segundos (Servidor S_B)

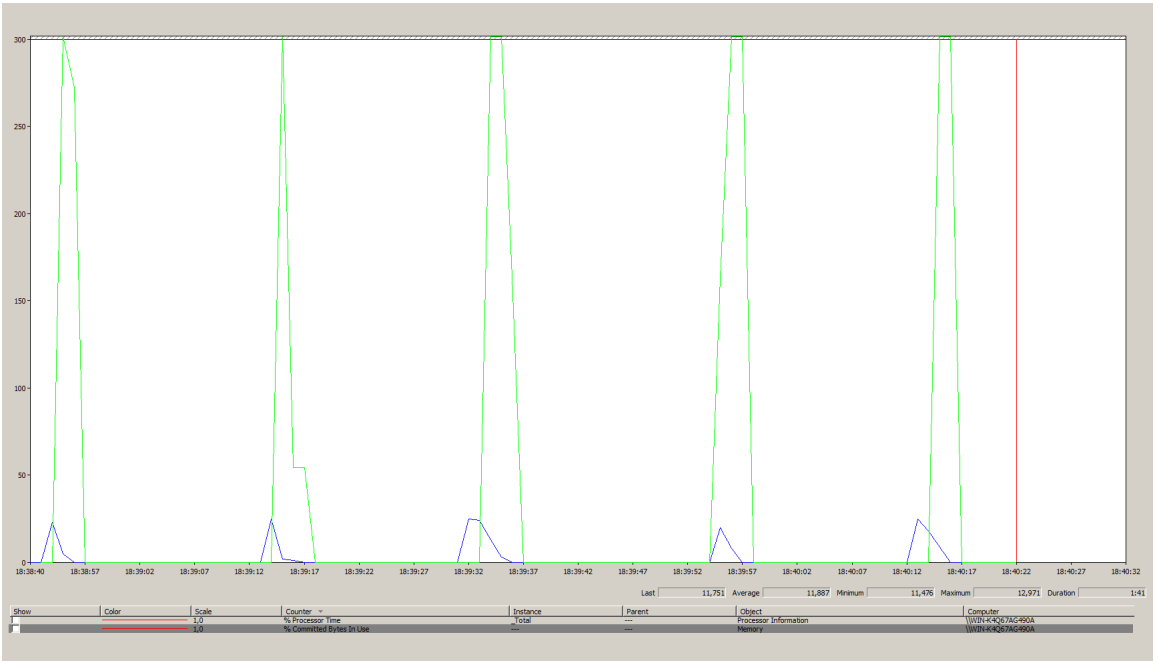

FIGURA A.18: Gráfico do teste Ramp-Up 500 em Dual Server com 50 Threads de 20 em 20 segundos (Servidor S_A)

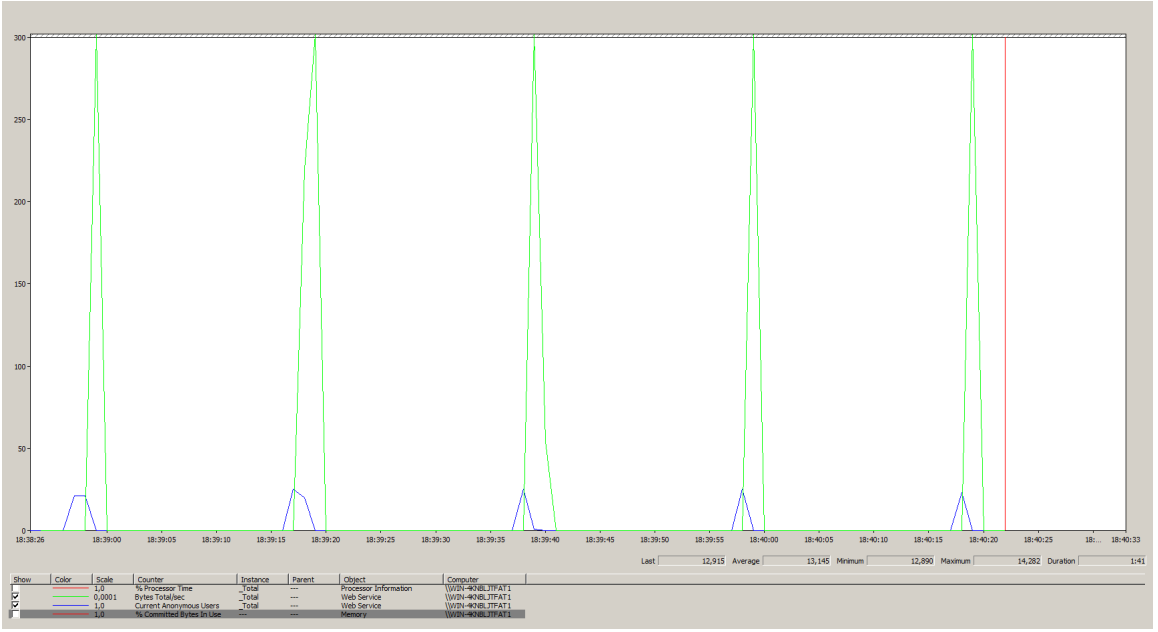

FIGURA A.19: Gráfico do teste Ramp-Up 500 em Dual Server com 50 Threads de 20 em 20 segundos (Servidor S_B)

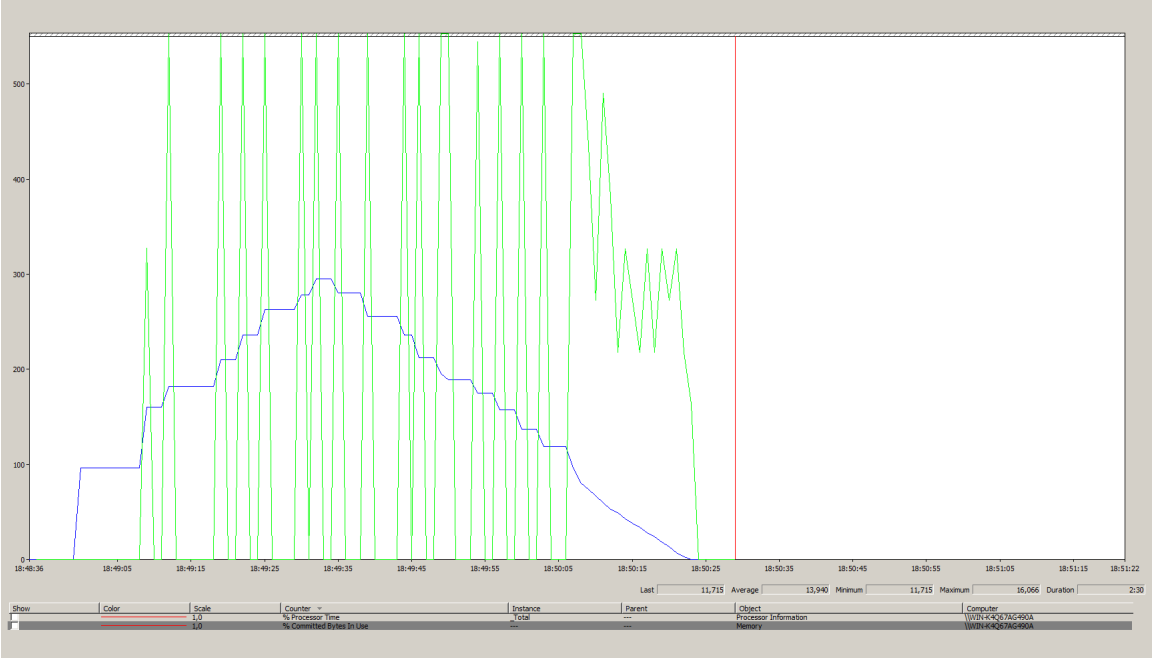

FIGURA A.20: Gráfico do teste Ramp-Up 1000 em Dual Server com 50 Threads de 2 em 2 segundos (Servidor S_A)

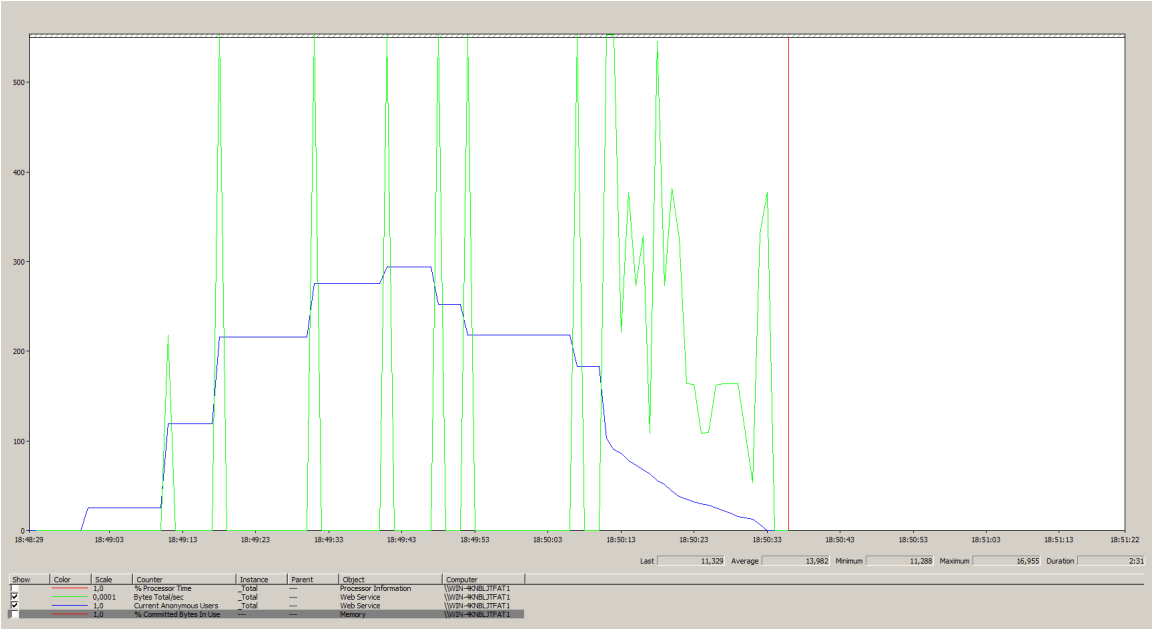

FIGURA A.21: Gráfico do teste Ramp-Up 1000 em Dual Server com 50 Threads de 2 em 2 segundos (Servidor S_B)

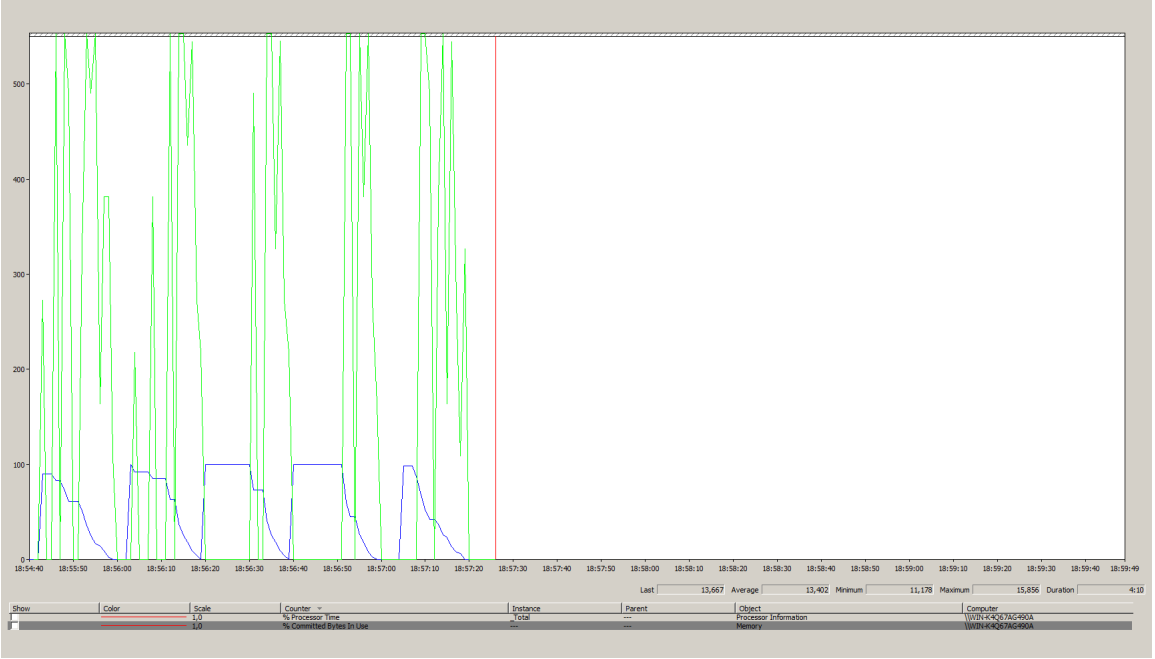

FIGURA A.22: Gráfico do teste Ramp-Up 1000 em Dual Server com 200 Threads de 20 em 20 segundos (Servidor S_A)

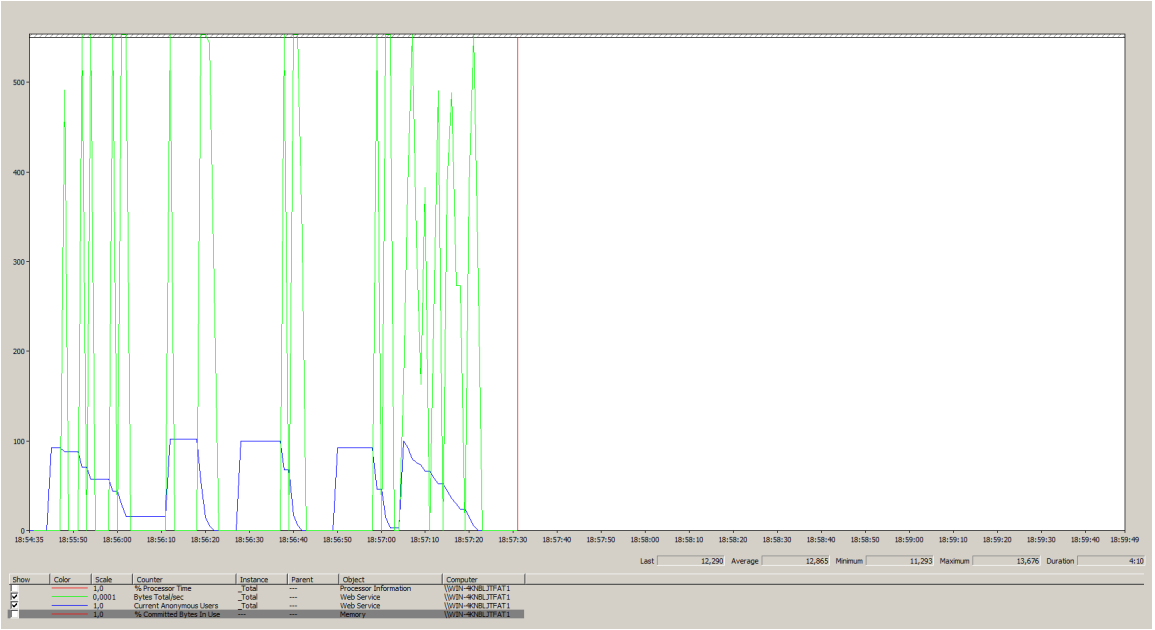

FIGURA A.23: Gráfico do teste Ramp-Up 1000 em Dual Server com 200 Threads de 20 em 20 segundos (Servidor S_B)

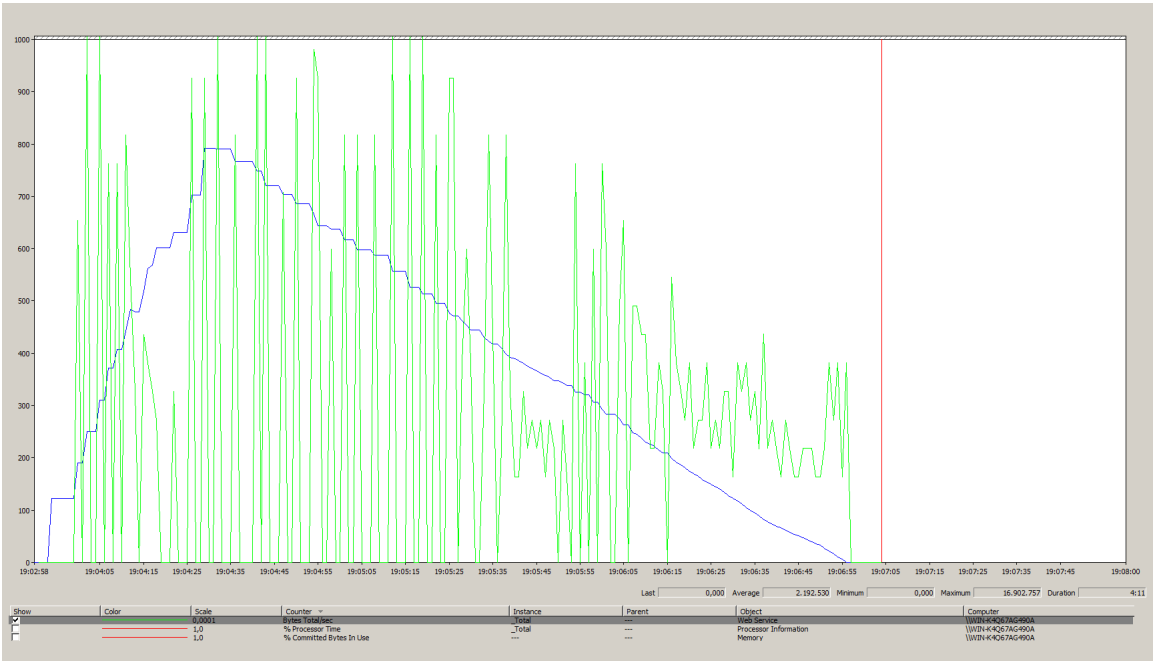

FIGURA A.24: Gráfico do teste Ramp-Up 2000 em Dual Server com 100 Threads de 2 em 2 segundos (Servidor S_A)

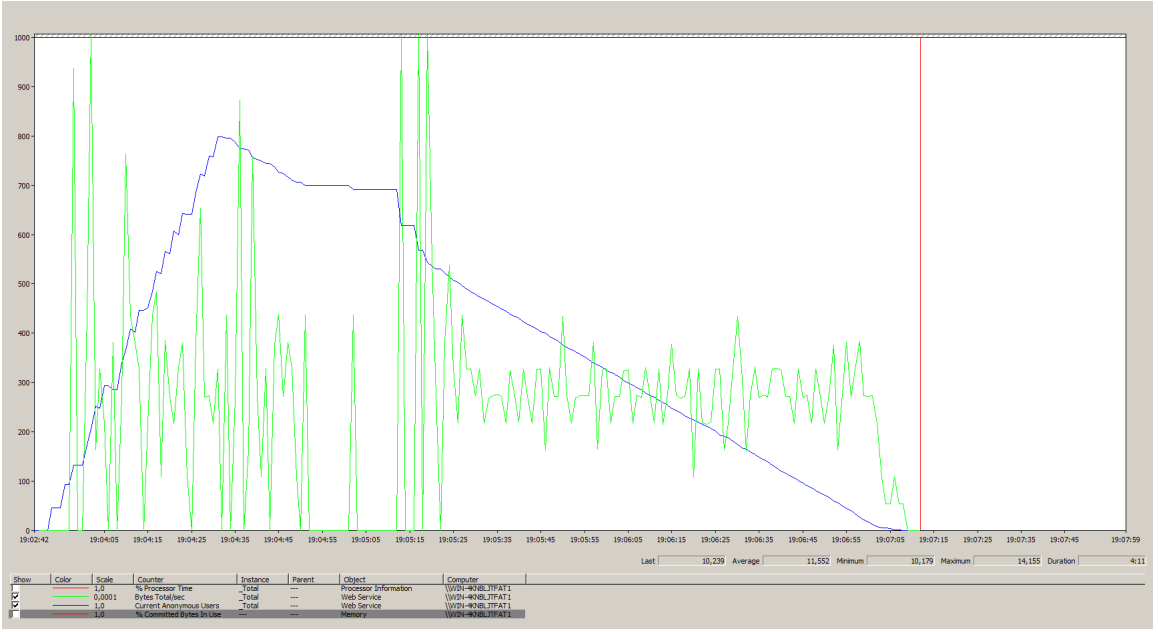

FIGURA A.25: Gráfico do teste Ramp-Up 2000 em Dual Server com 100 Threads de 2 em 2 segundos (Servidor S_B)

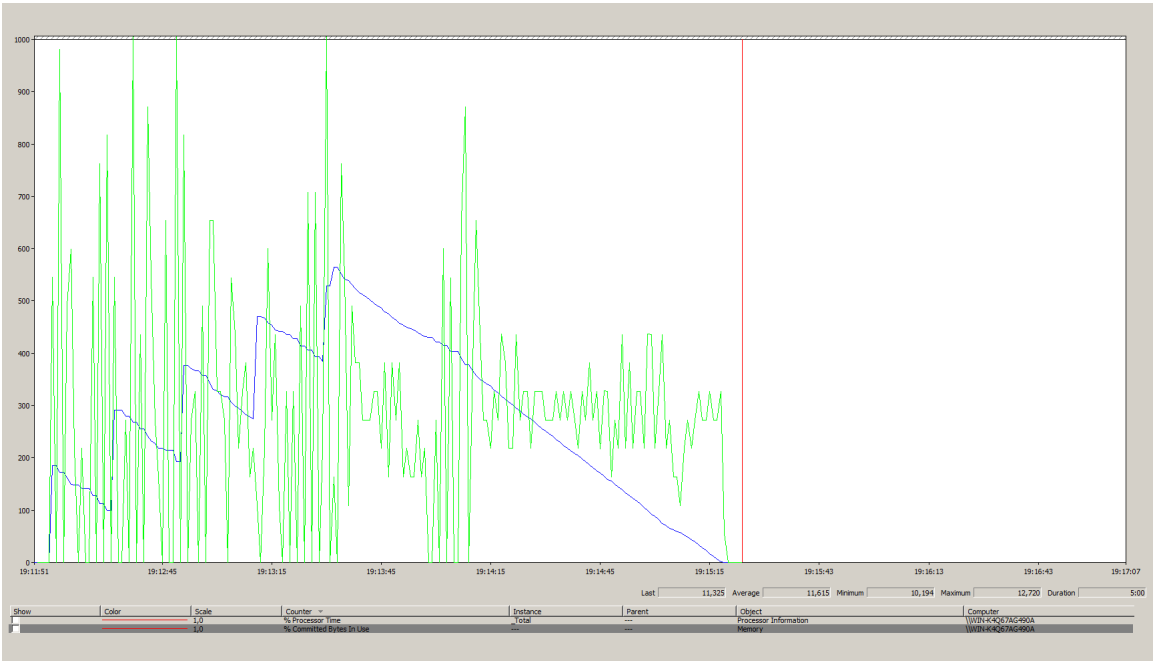

FIGURA A.26: Gráfico do teste Ramp-Up 2000 em Dual Server com 400 Threads de 20 em 20 segundos (Servidor S_A)

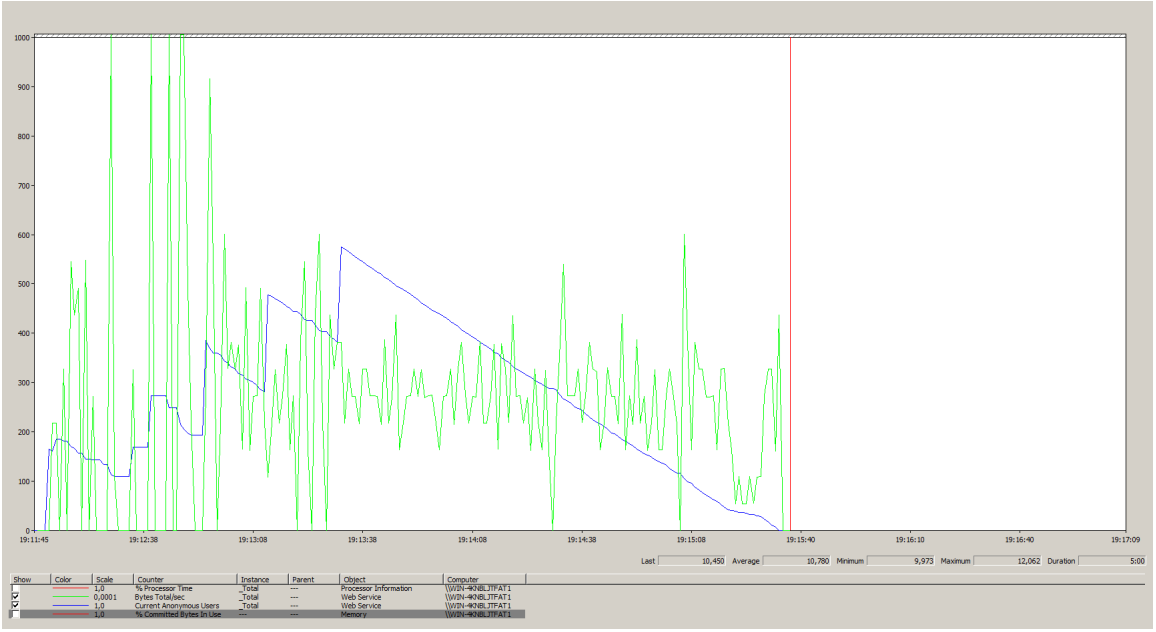

FIGURA A.27: Gráfico do teste Ramp-Up 2000 em Dual Server com 400 Threads de 20 em 20 segundos (Servidor S_B)

A.3 Testes com Ocorrência de Falhas

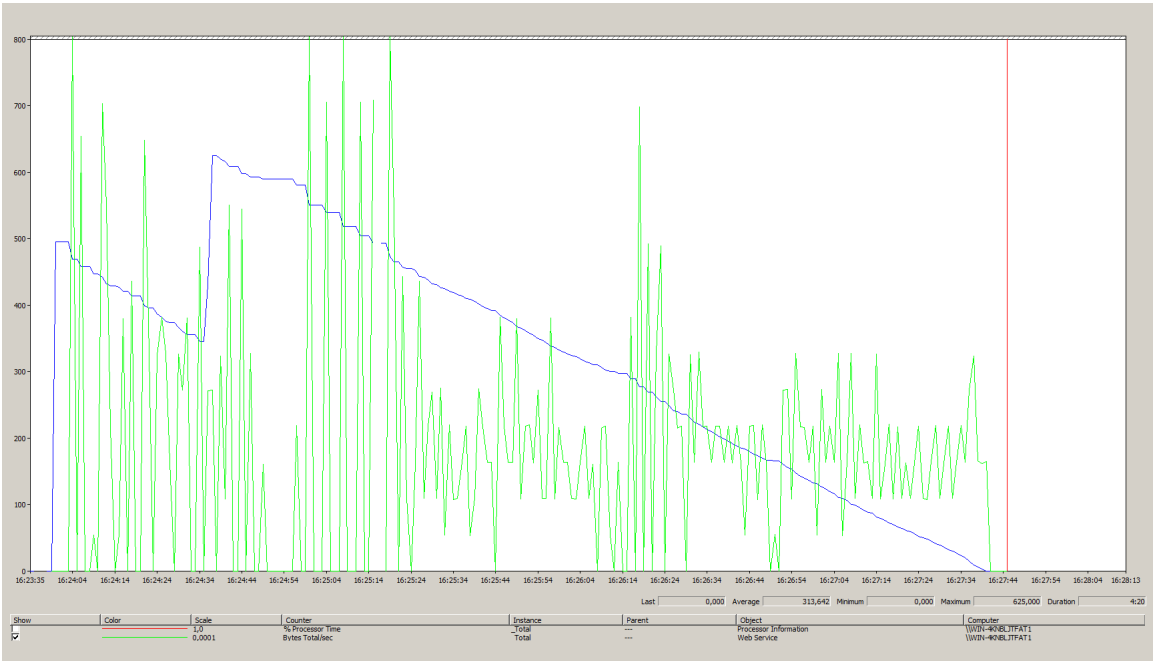

FIGURA A.28: Big-Bang 1000 em Dual Server com Ocorrência de Falha Durante a Execução (Servidor S_A)

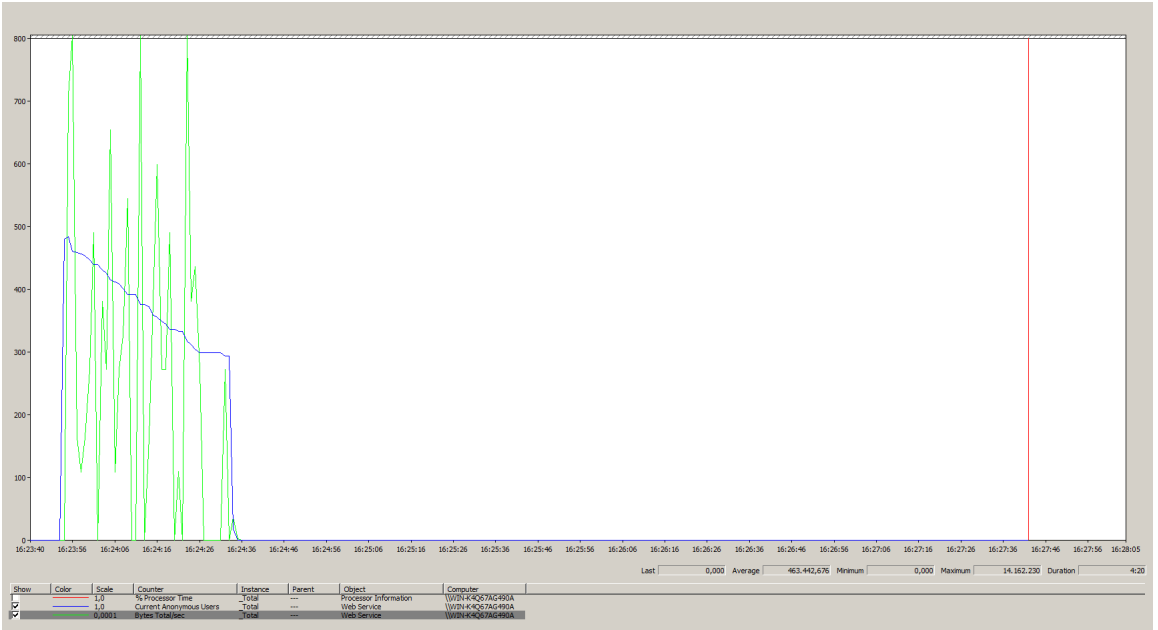

FIGURA A.29: Big-Bang 1000 em Dual Server com Ocorrência de Falha Durante a Execução (Servidor S_B)

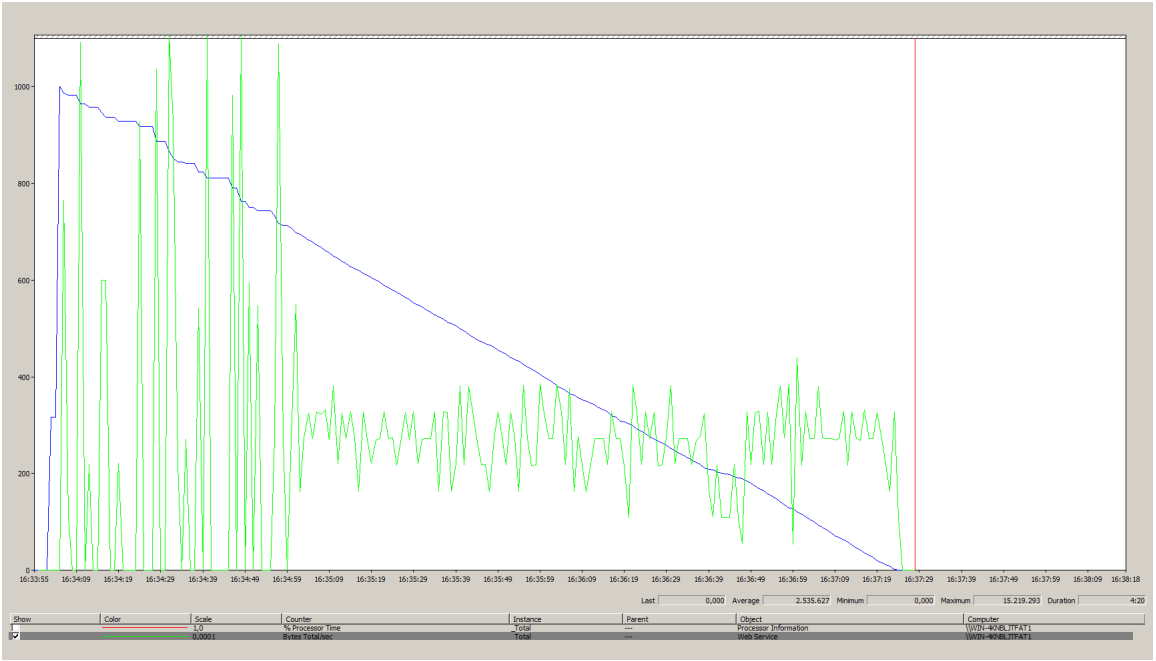

FIGURA A.30: Big-Bang 1000 em Dual Server com Apenas o Servidor S_A Disponível

Anexo B

Diagramas Extra

B.1 Diagramas de Sequência da Aplicação

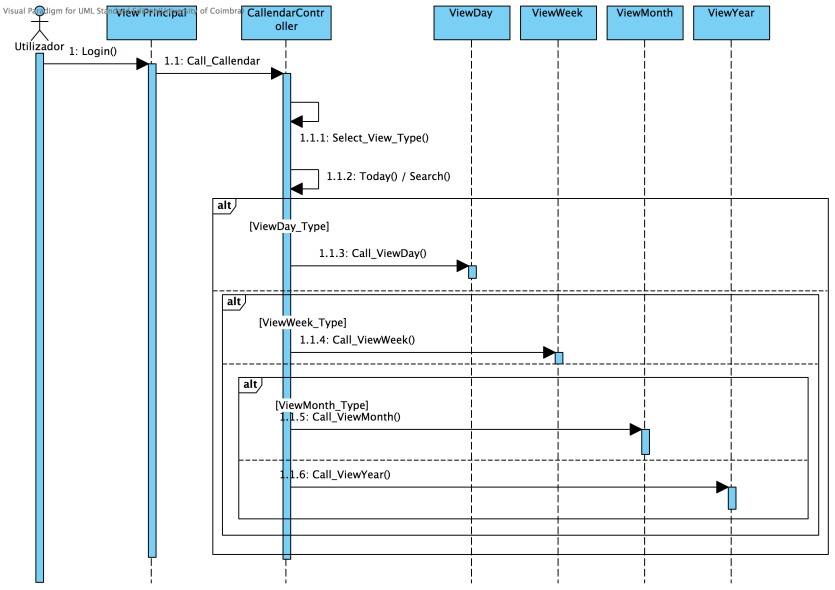

FIGURA B.1: Diagrama de Sequência Ilustrativo da Funcionalidade Hoje

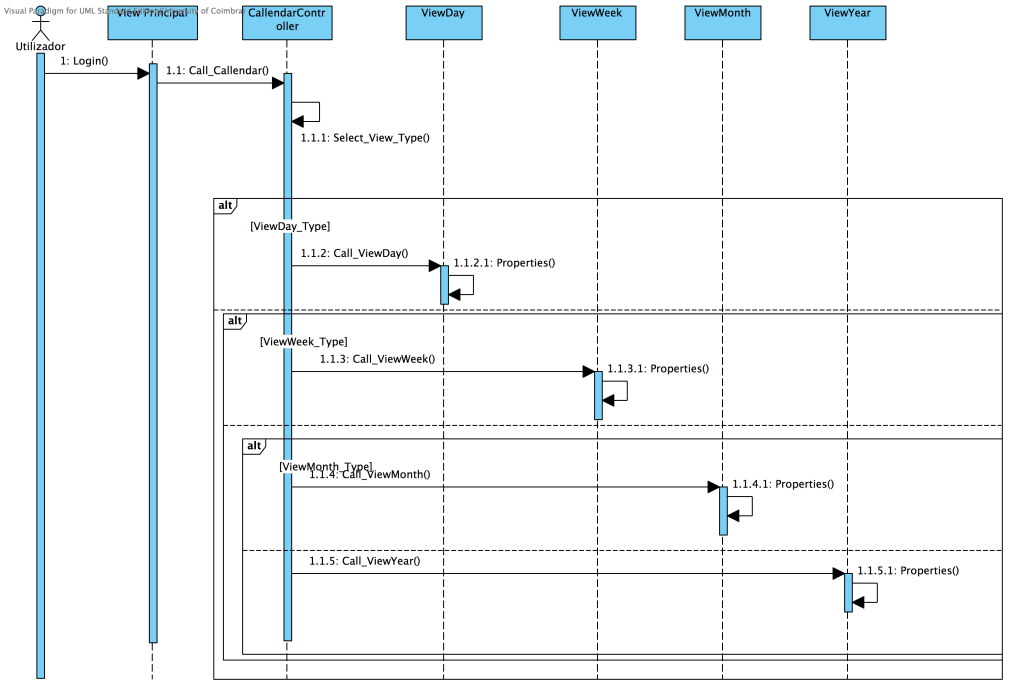

FIGURA B.2: Diagrama de Sequência Ilustrativo da Funcionalidade Propriedades

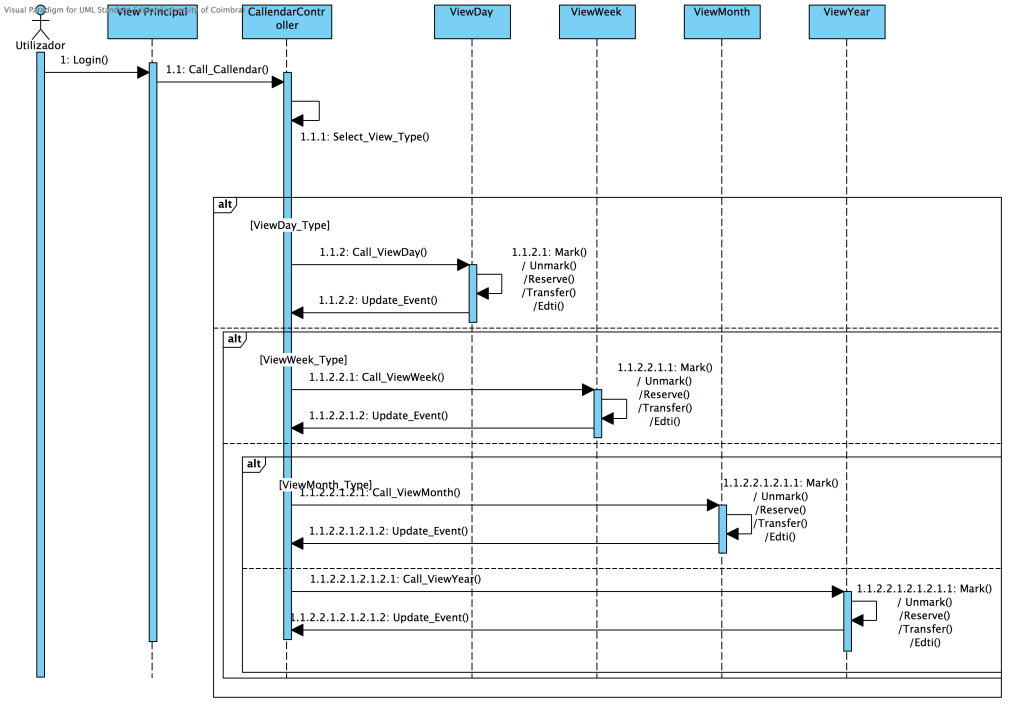

FIGURA B.3: Diagrama de Sequência Ilustrativo das Funcionalidades Marcar, Desmarcar, Trasnferir e Editar

B.2 Diagrama ER da Base de Dados do Servidor CalDAV

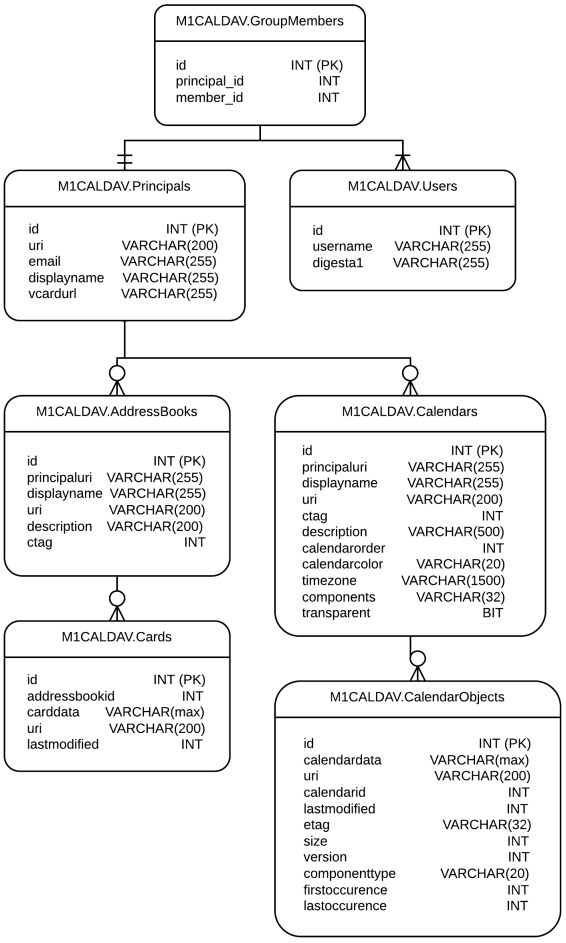

FIGURA B.4: Diagrama ER da Base de Dados do Servidor CalDAV

B.3 Diagrama de Gantt

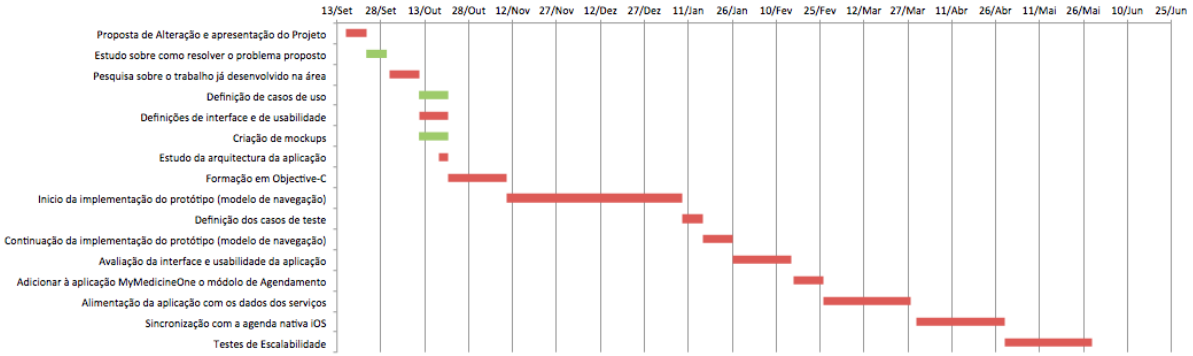

FIGURA B.5: Diagrama de Gantt com o Planeamento do Projecto

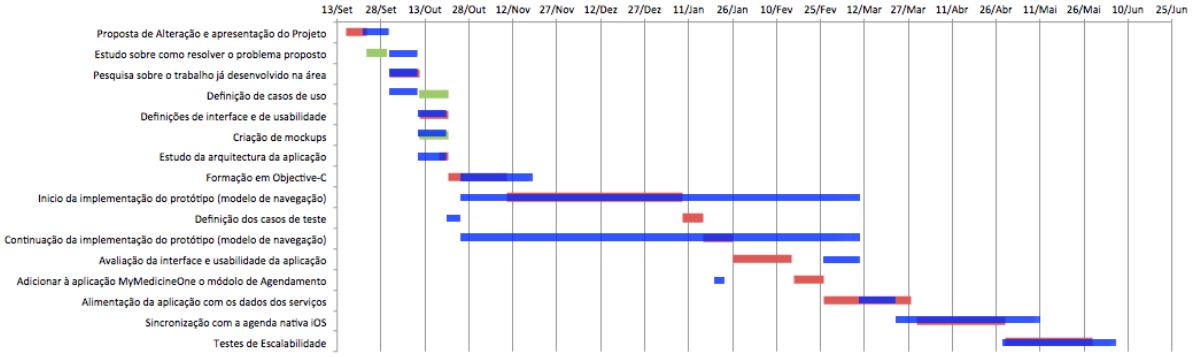

FIGURA B.6: Diagrama de Gantt de Comparação do Planeamento com o Tempo Real

Anexo C

Mockups

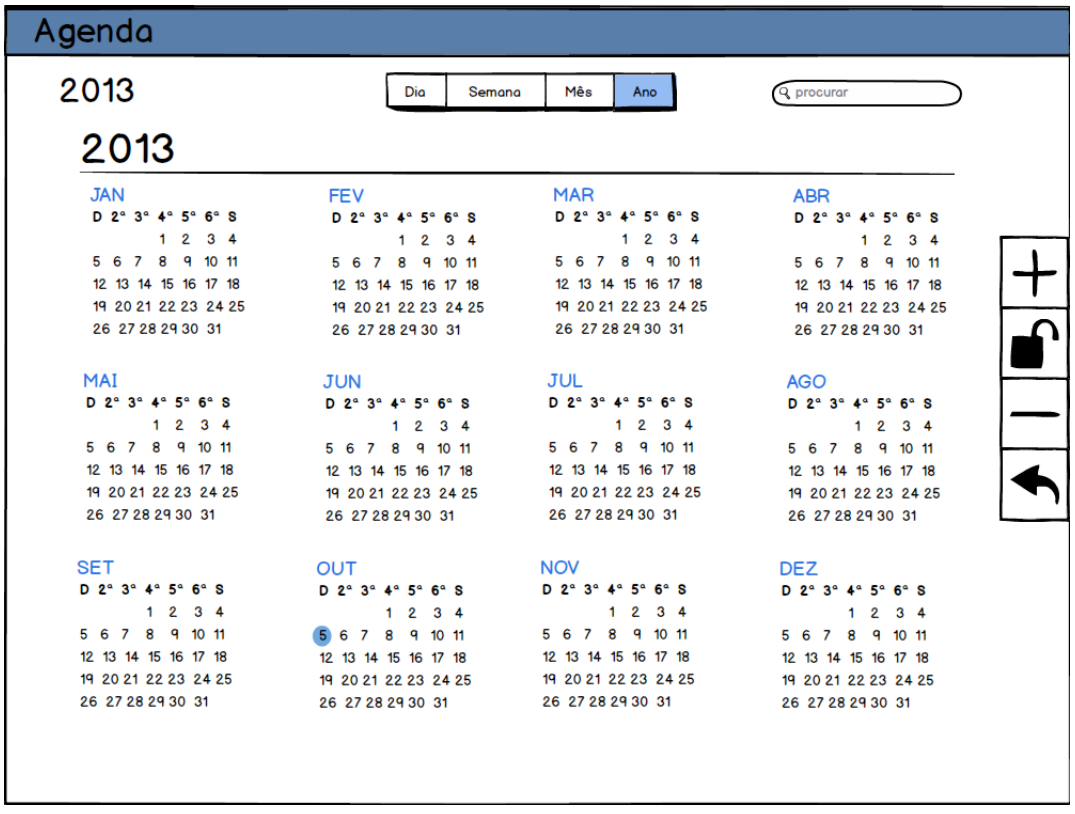

FIGURA C.1: Mockup do ambiente de navegação anual

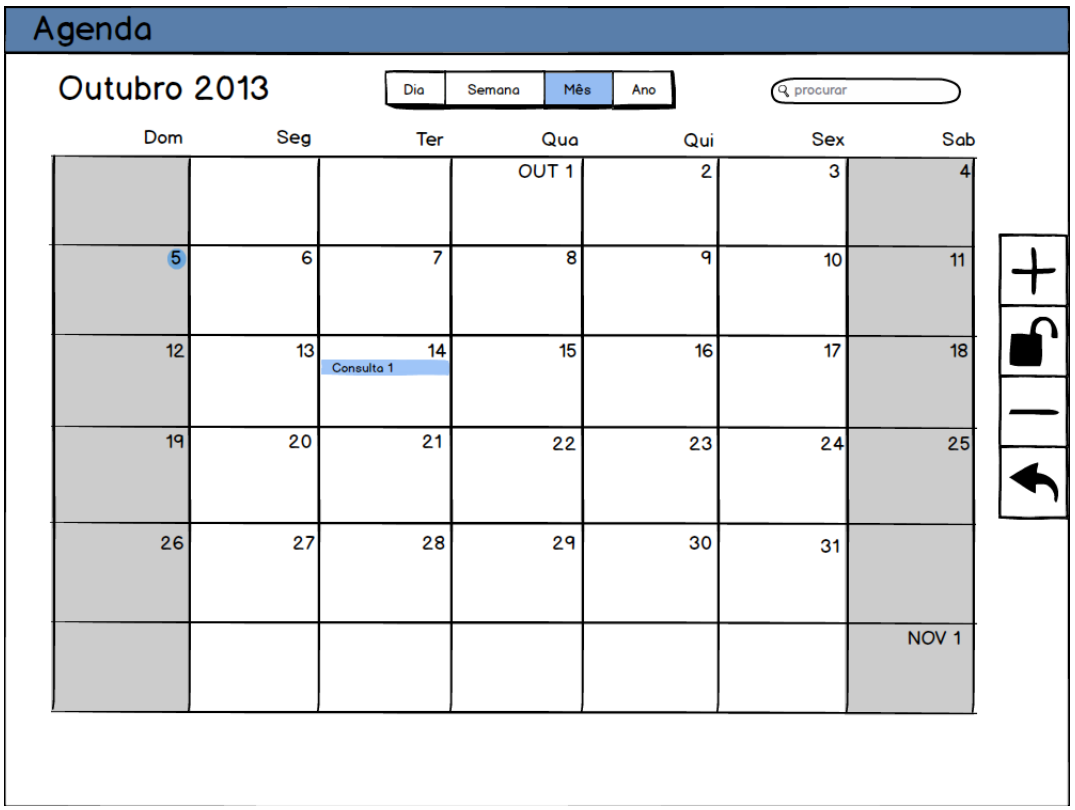

FIGURA C.2: Mockup do ambiente de navegação mensal

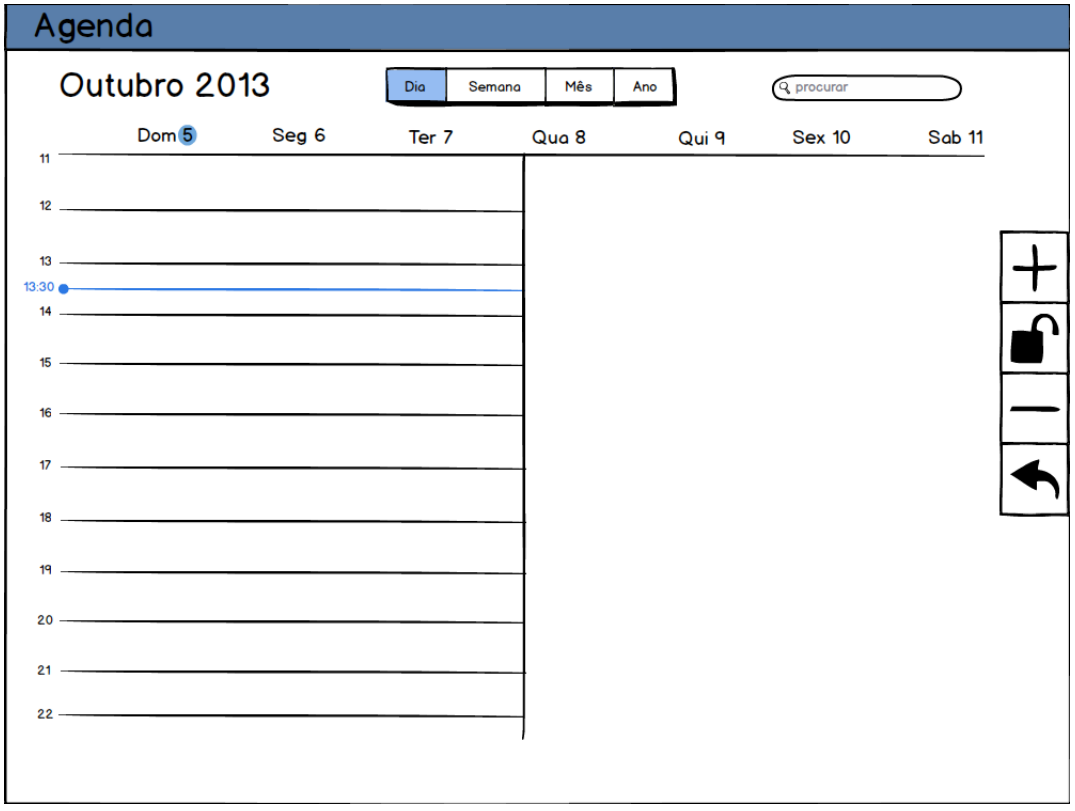

FIGURA C.3: Mockup do ambiente de navegação diária

Anexo D

Cenários de Teste da Aplicação

Os cenários de testes foram construídos com base em **várias histórias de utilização**, sendo que cada história **não tem necessariamente só um cenário**. Por sua vez **cada cenário pode variar** dependendo do ambiente de navegação em que seja executado. Por esse motivo dentro do fluxo de teste de um cenário também existirão **alíneas para designar as diferenças de um ambiente para o outro**.

História 1: O médico quer programar o seu dia de trabalho, ver informação das consultas e reservar o tempo de atendimento

Cenário 1.1: Ver informações de uma consulta no mesmo ambiente de navegação

Fluxo de teste:

1. selecionar no controlador de navegação o tipo de visão;
    (a) no caso de ser o navegador anual é necessário selecionar o mês;
2. carregar numa consulta previamente marcada;
3. confirmar as propriedades da consulta;

Cenário 1.2: Ver informações de uma consulta no ambiente de navegação diária e mudar para o semanal

Fluxo de teste:

1. selecionar no controlador de navegação a navegação diária;

2. carregar numa consulta previamente marcada;
3. confirmar as propriedades da consulta;
4. selecionar no controlador de navegação a navegação semanal;
5. carregar novamente na mesma consulta;
6. confirmar mais uma vez as propriedades;

Cenário 1.3: Ver informações de uma consulta numa data longínqua e voltar para o dia de hoje

Fluxo de teste:

1. navegar usando o *scroll* para uma data longínqua;
2. selecionar no controlador de navegação o tipo de visão;
   (a) no caso de ser o navegador anual é necessário selecionar o mês;
3. carregar numa consulta previamente marcada;
4. confirmar as propriedades da consulta;
5. carregar no botão hoje;
6. verificar se o posicionamento corresponde ao dia atual;

Cenário 1.4: Reservar o tempo disponível para a marcação de consultas

Fluxo de teste:

1. carregar no botão reservar;
2. escolher o intervalo de tempo que quer reservar para marcação de consultas;
3. concluir a reserva;
4. verificar na visão diária, semanal e mensal se a reserva foi concluída;

História 2: O médico quer marcar uma consulta

Cenário 2.1: Fazer a marcação usando o botão de marcação

Fluxo de teste:

1. carregar no botão marcar;
2. preencher o formulário com os dados relativos à consulta;
3. concluir a marcação;
4. verificar na visão diária, semanal e mensal se a marcação foi efetuada;

Cenário 2.2: Fazer a marcação através de um *long press*

Fluxo de teste:

1. selecionar no controlador de navegação o tipo de visão;
   (a) no caso de ser o navegador anual é necessário selecionar o mês;
2. pressionar durante algum tempo a célula em que quer inserir a consulta;
3. preencher o formulário com os dados relativos à consulta;
4. concluir a marcação;
5. verificar na visão diária, semanal e mensal se a marcação foi efetuada;

História 3: O médico quer transferir uma consulta para outra data

Cenário 3.1: Transferir uma consulta para uma data próxima por arrastamento e mudar de ambiente de navegação

Fluxo de teste:

1. selecionar no controlador de navegação o tipo de visão;
   (a) no caso de ser o navegador anual é necessário selecionar o mês;
2. pressionar durante algum tempo uma consulta previamente marcada;
3. arrastar para uma outra data

4. verificar na visão diária, semanal e mensal se a transferência foi efetuada;

Cenário 3.2: Transferir uma consulta para outra data através do botão transferir e mudar de ambiente de navegação

Fluxo de teste:

1. carregar no botão transferir;
2. indicar uma nova data para a consulta;
3. concluir transferência;
4. verificar na visão diária, semanal e mensal se a transferência foi efetuada;

História 4: O médico quer editar os dados relativos a uma consulta

Cenário 4.1: Tentar editar as informações de uma consulta e cancelar

Fluxo de teste:

1. selecionar no controlador de navegação o tipo de visão;
   (a) no caso de ser o navegador anual é necessário selecionar o mês;
2. selecionar uma consulta previamente marcada;
3. na janela informativa pressionar o botão editar;
4. editar os dados da consulta;
5. carregar em cancelar;
6. verificar na visão diária, semanal e mensal se a consulta não foi alterada;

Cenário 4.2: Editar as informações de uma consulta e mudar de ambiente de navegação

Fluxo de teste:

1. selecionar no controlador de navegação o tipo de visão;
   (a) no caso de ser o navegador anual é necessário selecionar o mês;

2. selecionar uma consulta previamente marcada;
3. na janela informativa pressionar o botão editar;
4. editar os dados da consulta;
5. concluir a edição;
6. verificar na visão diária, semanal e mensal se a consulta foi atualizada;

História 5: O médico quer desmarcar uma consulta

Cenário 5.1: Desmarcar uma consulta e mudar de navegação

Fluxo de teste:

1. selecionar no controlador de navegação o tipo de visão;
   (a) no caso de ser o navegador anual é necessário selecionar o mês;
2. selecionar uma consulta previamente marcada;
3. na janela informativa pressionar o botão editar;
4. editar os dados da consulta;
5. concluir a edição;
6. verificar na visão diária, semanal e mensal se a consulta foi atualizada;

História 6: O médico quer encontrar a última consulta que teve com um dos seus pacientes

Cenário 6.1: Pesquisar pelas consultas do paciente

Fluxo de teste:

1. carregar no botão de pesquisa;
2. carregar na caixa de texto de pesquisa;
3. escrever o nome do paciente;
4. aguardar resultados;

5. escolher consulta;
    (a) se existir alguma consulta carregar na consulta;
        i. a visão em foco é localizada na data da consulta;
        ii. as propriedades da consulta são exibidas;
    (b) caso não exista aparecerá uma notificação;

Referências

[1] Jorge Henriques e Paulo Carvalho. Gestão e integração de informacão clínica, universidade de coimbra. Setembro 2012. Última Consulta: Dezembro de 2013.

[2] GIMED. Grupo de investigação e desenvolvimento em informática médica da universidade fernando pessoa. 2001. URL http://link.aip.org/link/?RSI/72/4477/1. Última Consulta: Dezembro de 2013.

[3] Álvaro Rocha. Sistemas e tecnologias de informação na saúde e qualidade assistencial nas uci, universidade fernando pessoa, gimed. Janeiro 2012. URL http://www.slideshare.net/amrrocha/sistemas-e-tecnologias-de-informao-na-sade-e-qualidade-assistencial-nas-uci. Última Consulta: Dezembro de 2013.

[4] AISTI – Associação Ibérica de Sistemas e Tecnologias de Informação. Revista ibérica de sistemas e tecnologias de informação. Dezembro 2009. URL http://www.aisti.eu/risti/RISTI%20N4.pdf. Última Consulta: Dezembro de 2013.

[5] Álvaro Rocha. e-saúde - conceito, evolução e aplicações móveis, universidade fernando pessoa, gimed. Janeiro 2012. URL http://www.slideshare.net/amrrocha/e-saude-conceito-evoluo-e-aplicaes-mveis-palestra-ip-guarda-28-de-jan-2012. Última Consulta: Dezembro de 2013.

[6] Apple Inc. About the ios technologies. 2013. URL https://developer.apple.com/library/IOs/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007898-CH1-SW1. Última Consulta: Dezembro de 2013.

[7] Apple Inc. Core os layer. 2013. URL https://developer.apple.com/library/IOs/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/

CoreOSLayer/CoreOSLayer.html#//apple_ref/doc/uid/TP40007898-CH11-SW1. Última Consulta: Dezembro de 2013.

[8] Apple Inc. Core services layer. 2013. URL https://developer.apple.com/library/IOs/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreServicesLayer/CoreServicesLayer.html#//apple_ref/doc/uid/TP40007898-CH10-SW5. Última Consulta: Dezembro de 2013.

[9] Apple Inc. Media layer. 2013. URL https://developer.apple.com/library/IOs/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/MediaLayer/MediaLayer.html#//apple_ref/doc/uid/TP40007898-CH9-SW4. Última Consulta: Dezembro de 2013.

[10] Apple Inc. Cocoa touch layer. 2013. URL https://developer.apple.com/library/IOs/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html#//apple_ref/doc/uid/TP40007898-CH3-SW1. Última Consulta: Dezembro de 2013.

[11] Apple Inc. Coredata. 2013. URL https://developer.apple.com/library/ios/documentation/DataManagement/Devpedia-CoreData/coreDataOverview.html. Última Consulta: Dezembro de 2013.

[12] T3. 74% dos dispositivos apple corre o ios 7. Dezembro 2013. URL http://t3.sapo.pt/noticias/74-dos-dispositivos-apple-corre-o-ios-7. Última Consulta: Dezembro de 2013.

[13] Tom Thompson. New in ios 7: What developers need to know, infoworld. Outubro 2013. URL http://www.infoworld.com/d/application-development/new-in-ios-7-what-developers-need-know-227683?page=0,0. Última Consulta: Dezembro de 2013.

[14] Apple Inc. What's new in xcode 5, apple inc. 2013. URL https://developer.apple.com/technologies/tools/whats-new.html. Última Consulta: Dezembro de 2013.

[15] Apptite. Xcode tuturial. 2012. URL http://www.apptite.be/tutorial_xcode.php. Última Consulta: Dezembro de 2013.

[16] MedicineOne. Aceder ao mymedicineone. 2013. URL `http://mobile.medicineone.net/howitworksipad/index`. Última Consulta: Dezembro de 2013.

[17] Cocoa Controls. Cocoa controls search. 2013. URL `https://www.cocoacontrols.com/search?q=calendar`. Última Consulta: Dezembro de 2013.

[18] Version One. What is kanban? kanban software tools. 2013. URL `http://www.versionone.com/what-is-kanban/`. Última Consulta: Dezembro de 2013.

[19] Sander Hoogendoorn. Smart use cases. Fevereiro 2011. URL `http://www.slideshare.net/aahoogendoorn/an-introduction-to-smart-use-cases`. Última Consulta: Dezembro de 2013.

[20] Apple Inc. O sistema operativo móvel mais avançado do mundo. na sua melhor forma. 2013. URL `http://www.apple.com/pt/ios/what-is/`. Última Consulta: Dezembro de 2013.

[21] Apple Inc. Novidades do ios7. 2013. URL `https://www.apple.com/pt/ios/whats-new/`. Última Consulta: Dezembro de 2013.

[22] Apple Inc. About objective-c. 2012. URL `https://developer.apple.com/library/mac/documentation/cocoa/conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html`. Última Consulta: Dezembro de 2013.

[23] OIT Office of Information Technology. What load balancing methods are available? 2014. URL `http://oit2.utk.edu/helpdesk/kb/entry/1699/`. Última Consulta: Junho de 2014.

[24] Citrix Product Documentation. Load balancing algorithms. Agosto 2013. URL `http://support.citrix.com/proddocs/topic/netscaler-load-balancing-93/ns-lb-customizing-lbalgorithms-wrapper-con.html`. Última Consulta: Junho de 2014.

[25] Microsoft. Network load balancing manager. Janeiro 2005. URL `http://technet.microsoft.com/en-us/library/cc776931(v=ws.10).aspx`. Última Consulta: Junho de 2014.

[26] Microsoft. Network load balancing technical overview. 2014. URL `http://technet.microsoft.com/en-us/library/bb742455.aspx`. Última Consulta: Junho de 2014.

[27] Cal Henderson. Building scalable web sites, capítulo 9. Maio 2006. Última Consulta: Junho de 2014.

[28] Tsung. Tsung documentation. 2013. URL `http://tsung.erlang-projects.org/user_manual/introduction.html#what-is-tsung`. Última Consulta: Junho de 2014.

[29] Tim Olsen. Repositório github do prestan. 2013. URL `https://github.com/tolsen/Prestan`. Última Consulta: Junho de 2014.

[30] lemoncodes. Repositório github do twisted-caldav. 2014. URL `https://github.com/lemoncodes/twisted-caldav`. Última Consulta: Junho de 2014.

[31] Kozea. Radicale official page. 2013. URL `http://radicale.org/`. Última Consulta: Junho de 2014.

[32] Andrew McMillan. Davical official page. 2011. URL `http://www.davical.org/`. Última Consulta: Junho de 2014.

[33] SabreDAV. Sabredav official page. 2014. URL `http://sabre.io/`. Última Consulta: Junho de 2014.

[34] Jérôme Schneider. Baikal official page. 2014. URL `http://baikal-server.com/`. Última Consulta: Junho de 2014.

[35] Jérôme Schneider. Repositório github do baikal. 2014. URL `https://github.com/jeromeschneider/Baikal`. Última Consulta: Junho de 2014.