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

Instituto Pedro Nunes
Laboratório de Informática e Sistemas
Coimbra

Relatório de Estágio

Mestrado em Engenharia Informática
11-07-2012

SADAC

Sistema de Acompanhamento do Desenvolvimento de Adolescentes e Crianças

Estagiário:

Fernando António Tavares da Rocha
frocha@student.dei.uc.pt
N.º Aluno: 2006129240

Orientadores:

Eng. Alcides Manuel de Almeida Marques
alcides.marques@ipn.pt
Instituto Pedro Nunes

Prof. Doutor Paulo Fernando Pereira de Carvalho
carvalho@dei.uc.pt
Departamento de Engenharia Informática, Universidade de Coimbra

Agradecimentos

A todos os professores que fizeram parte do meu percurso académico pela partilha de conhecimentos, simpatia e inspiração, em especial ao meu Professor e orientador Doutor Paulo Carvalho pela disponibilidade e empenho demonstrados.

À Professora Doutora Marília Curado por desde o início acreditar no meu potencial, me motivar e participar tão proficuamente na minha aprendizagem e construção pessoal.

Ao Engenheiro Alcides Marques pela tão estimada amizade e pelo incomensurável apoio com que desde sempre me presenteou.

A todos os meus amigos e colegas de trabalho pela companhia nas noites mais longas.

À minha família, a quem dedico este trabalho, por todo o apoio e compreensão incondicionais ao longo do meu percurso de vida.

Resumo

Este documento descreve e analisa, numa perspectiva reflexiva, as actividades de concepção, desenvolvimento e gestão referentes à implementação da primeira versão do Sistema de Acompanhamento do Desenvolvimento de Adolescentes e Crianças (SADAC). Esta primeira versão do SADAC, projecto de Engenharia de Software desenvolvido sobre a plataforma Microsoft .NET C#, consiste em três módulos centrais: Raciocínio Baseado em Regras (RBR), Entidades Dinâmicas e Portal do Utente. O módulo de RBR consiste num componente complexo que permite correr regras de negócio altamente configuráveis sobre utentes e respectivos dados, sendo o principal objectivo permitir a codificação dos Planos Nacionais de Saúde, nomeadamente o de Vacinação, e despoletar os devidos alertas. O módulo de Entidades Dinâmicas, desenhado e construído de raiz, permite dar resposta à dinâmica dos modelos de dados clínicos, uma vez que estes evoluem naturalmente com o passar do tempo e dependem do sistema clínico utilizado na unidade hospitalar alvo. O Portal do Utente permite que, por um lado, os profissionais e auxiliares de saúde configurarem o sistema e interajam com os utentes, e por outro, que os utentes acedam aos serviços disponibilizados pelo sistema. O objectivo do sistema é a melhoria da qualidade de vida dos utentes, aumentando o grau de satisfação e proporcionando uma maior fidelização com a unidade hospitalar.

Palavras-Chave

“Entidades Dinâmicas”; “Microsoft .NET”; “Portal do Utente”; “Raciocínio Baseado em Regras”; “Saúde”

Índice

Índice de Ilustrações

Índice de Tabelas

Tabela de Acrónimos

Acrónimo	Descrição
ACID	Atomicity, Consistency, Isolation, and Durability
API	Application Programming Interface
ASP	Active Server Pages
B2B	Business to Business
BOND	Building On Network Dynamics
BRMS	Business Rules Management System
C#	C Sharp
CBR	Case-Based Reasoning
CIL	Common Intermediate Language
CPU	Central Processing Unit
CRM	Customer Relationship Management
CSRF	Cross-Site Request Forgery
DBMS	DataBase Management System
DICOM	Digital Imaging and Communications in Medicine
DSL	Domain Specific Language
DTD	Document Type Definition
EHR	Electronic Health Record
ETL	Extract, Transform and Load
F#	F Sharp
FAST	Facilitated Application Specification Techniques
HL7	Health Level 7
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
I&DT	Investigação e Desenvolvimento Tecnológico
I/O	Input / Output
IDE	Integrated Development Environment
IPN	Instituto Pedro Nunes
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
KBS	Knowledge-Based Systems
LABGEO	Laboratório de Geotecnia

Acrónimo	**Descrição**
LABPHARM	Laboratório de Estudos Farmacêuticos
LAS	Laboratório de Automação e Sistemas
LEC	Laboratório de Electroanálise e Corrosão
LED&MAT	Laboratório de Ensaios, Desgaste e Materiais
LIS	Laboratório de Informática e Sistemas
LOB	Line Of Business
MVC	Model-View-Controller
MVEL	MVFLEX Expression Language
NLB	Network Load Balancing
ORM	Object-Relational Mapping
OWASP	Open Web Application Security Project
PHR	Personal Health Record
PNS	Plano Nacional de Saúde
PNV	Plano Nacional de Vacinação
RBR	Rule-Based Reasoning
RIM	Reference Information Model
RUP	Rational Unified Process
SADAC	Sistema de Acompanhamento do Desenvolvimento de Adolescentes e Crianças
SEO	Search-Engine Optimization
SHA	Secure Hash Algorithm
SIs	Sistemas de Informação
SNS	Serviço Nacional de Saúde
SQL	Structured Query Language
SSL	Secure Sockets Layer
TIs	Tecnologias da Informação
TLS	Transport Layer Security
VB	Visual Basic
WDSL	Web Services Definition Language
WPF	Windows Presentation Foundation
XML	eXtensible Markup Language
XP	Extreme Programming
XSD	XML Schema Definition
XSS	Cross-Site Scripting

Enquadramento

O presente relatório de estágio foi elaborado no âmbito da disciplina de dissertação/estágio, presente no último ano de Mestrado em Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra. O estágio decorre no período compreendido entre o 1 de Setembro de 2011 a 29 de Junho de 2012.

A instituição de acolhimento do meu estágio é o Instituto Pedro Nunes (IPN), uma instituição de direito privado sem fins lucrativos localizada em Coimbra. Estive integrado, no Laboratório de Informática e Sistemas (LIS), um dos seus laboratórios de desenvolvimento tecnológico.

O IPN é amplamente conhecido pela comunidade académica, principalmente pela sua Incubadora de Empresas e Ideias, considerada recentemente a melhor incubadora de empresas do mundo. A sua missão é de estabelecer a ligação entre o meio científico e o tecido empresarial. O berço de centenas de empresas com uma taxa de sucesso acima dos 80%, onde 60% das empresas que passaram são *spin-off* académicos que transformam em negócio o conhecimento gerado em instituições de Investigação e Desenvolvimento Tecnológico (I&DT) de Coimbra. Da incubadora nasceram empresas de sucesso como a Critical Software, Crioestaminal, WIT Software, Active Space Technologies, entre muitas outras. [1]

O IPN dispõe também dos seus próprios laboratórios de desenvolvimento tecnológico que investigam, desenvolvem e prestam serviços de consultadoria (entre outros serviços especializados) nas mais diversas áreas tecnológicas:

- LIS: Laboratório de Informática e Sistemas, onde o meu estágio foi integrado;
- LAS: Laboratório de Automação e Sistemas, integrando uma Unidade de Instrumentação Industrial;
- LED&MAT: O Laboratório de Ensaios, Desgaste e Materiais, que possui duas unidades: Unidade de Modificação de Superfícies e Unidade de Materiais Granulares;
- LEC: Laboratório de Electroanálise e Corrosão;
- LABGEO: Laboratório de Geotecnia;
- LABPHARM: Laboratório de Estudos Farmacêuticos.

No domínio da Formação, o IPN concentra os seus esforços na formação contínua de alto nível dirigida à gama crescente de quadros com necessidade de actualização nos domínios que surgiram ou evoluíram após a sua formação inicial, à preparação de técnicos especializados, à qualificação de licenciados desempregados, e ainda ligada à sua actividade de criação e incubação de empresas de base tecnológica, ou seja, a formação dirigida a jovens empreendedores.

O projecto a ser desenvolvido durante o estágio é para um cliente do IPN, a BOND. Com uma vasta experiência e uma equipa distribuída pelas cidades de Lisboa, Luanda e Maputo, a BOND está dotada das mais avançadas competências no domínio das TIs e Sistemas de Informação (SIs), sendo reconhecida no mercado como uma organização de excelência e líder em diversos vectores:

- **Customer Relationship Management (CRM)**

  Produz soluções de automatização, optimização e integração de processos para interacção com clientes, baseadas em produtos próprios.

- **Business Intelligence**

  Possui uma solução tecnológica para optimização, extracção e produção de relatórios de informação para apoio à decisão.

- **Business Technology Consulting**

  Desenho de soluções de negócio tendo como vector direccional a tecnologia.

O sistema a ser implementado tem como cliente-alvo as unidades hospitalares, nomeadamente do sector privado. O sistema consiste numa mais-valia para os departamentos de marketing das unidades hospitalares, uma vez que é direccionado aos utentes, aproximando-os dos serviços da unidade hospitalar, reduzindo deslocações desnecessárias e proporcionando uma melhor qualidade de vida. Por outro lado, funciona como factor de fidelização à unidade hospitalar, uma vez que usufrui dos dados do utente (existentes na unidade) e conterá uma plataforma de gestão de pedidos de marcação de consultas *online*.

1. Introdução

O sector tecnológico tem sofrido uma grande evolução nestas últimas décadas. Cada vez mais existem *smartphones* capazes de executar tarefas mais complexas, actualmente já com capacidades de memória e de processamento superiores a computadores de secretária de há menos de dez anos. Cada vez mais existem redes sociais que nos tentam aproximar tanto de amigos, familiares e colegas como de instituições, fundações, associações e os seus serviços. Apesar dessa abrupta evolução que nos está a tornar virtualmente mais ligados a tudo e a todos, as unidades hospitalares padecem dessa aproximação e utilização de recursos tecnológicos, uma vez que no máximo permitem ver as actuais consultas marcadas e efectuar pedidos de marcação através da sua página de Internet.

Muitas vezes os utentes visitam uma unidade hospitalar para realizarem um exame ou frequentarem uma consulta, e após saírem do edifício, mesmo que sejam dadas instruções para regressarem a uma consulta de forma periódica, para um exame de despistagem, para vacinação, ou até para realizarem um tratamento, os utentes, inconscientes disso ou por esquecimento, não regressam à unidade hospitalar. O motivo pelo que muitas vezes regressam é quando ocorre outro problema de saúde, situação em que ocorre com frequência visitarem outra unidade hospitalar, por acaso ou porque podem sentir que a que frequentavam não foi capaz de os aconselhar e acompanhar devidamente.

Os utentes que realizam consultas de forma periódica, como por exemplo as consultas de pediatria nos primeiros anos de vida, é comum se comprometerem com o médico, e não com a unidade hospitalar, uma vez que é o médico a insistir e a relembrar o utente para voltar a consultá-lo. Isto ocorre porque existe um acompanhamento mais próximo e personalizado.

Este tipo de acompanhamento mais próximo e personalizado torna-se fundamental até à idade adulta, altura em que o utente se deverá tornar independente. Os primeiros anos de vida são cruciais e necessitam de procedimentos importantes que envolvem vacinações, exames e consultas que tornam difícil a sua gestão e organização. Também surgem constantemente dúvidas, como por exemplo "*A temperatura está alta e o bebé não dorme, o que faço?*", que poderiam ser facilmente respondidas. Mesmo após os primeiros anos, agora já criança ou adolescente, continuam a existir muitos procedimentos recomendados e algumas dúvidas que poderiam ser facilmente respondidas e que, normalmente dão origem a uma ida

desnecessária às urgências ou a uma consulta. Esta situação agrava-se ainda mais quando se é responsável por várias crianças ou adolescentes.

Um serviço que tenta colmatar as idas desnecessárias ao médico ou às urgências é a linha Saúde 24 [2], que tal como o nome indica está disponível 24 horas por dia. Este é um serviço de atendimento telefónico, de triagem, aconselhamento e encaminhamento, assistência em saúde pública e de informação geral de saúde, para todos os utentes do Serviço Nacional de Saúde (SNS). Infelizmente grande parte do seu potencial apenas está disponível por telefone e o serviço não tem acesso aos dados do utente do SNS, apenas aos dados que já tenham sido facultados por telefone.

Dados estes cenários descritos, no ponto de vista da unidade hospitalar, identificam-se os seguintes problemas:

- A unidade hospitalar é **incapaz de acompanhar os utentes** uma vez que não os recorda para realizarem uma consulta, exames ou continuação de tratamentos, quando já não se encontram na unidade hospitalar, perdendo assim os utentes, porque estes não regressam;
- O não acompanhamento do utente provoca um sentimento de insatisfação, o que faz com que **não regresse à mesma unidade hospitalar** no futuro, procurando um unidade hospitalar que lhe dê mais acompanhamento e atenção;
- A fraca fidelização de utentes **não permite reunir todas as consultas e exames** (histórico clínico) que o utente realizou, uma vez que o utente muda frequentemente de unidade hospitalar, impossibilitando análises mais exaustivas do histórico clínico, nomeadamente análise de padrões de patologias;
- Apesar da recomendação para os utentes registarem medições de parâmetros de saúde de forma periódicas (pulsação, tensão arterial, entre outros), os utentes frequentemente se esquecem porque não visualizam de imediato o resultado dessas medições, ou esquecem-se de levar para as consultas os registos, **impossibilitando o médico de observar a evolução dessas medidas** e por vezes até ajustar a medicação;

- Mesmo que todas as informações clínicas do utente se encontrassem dentro da unidade hospitalar, a não ser que o médico tenha tido casos idênticos num curto espaço de tempo, o médico **não é capaz de identificar padrões de patologias**.

Observam-se também os seguintes problemas fulcrais para os utentes:

- **O histórico clínico do utente fica espalhado por várias unidades hospitalares** de forma isolada, impedindo um acompanhamento mais preciso por parte da unidade hospitalar, diminuindo a qualidade dos diagnósticos e exigindo a realização de exames redundantes suportados financeiramente pelo utente;
- **O utente não cumpre os passos complementares à consulta**, nomeadamente a marcação periódica de consultas ou exames, tenham sido estes recomendados ou não, uma vez que não existe acompanhamento fora da unidade hospitalar, colocando a saúde do utente em risco;
- Muitas vezes **o utente cria um compromisso com o médico e não com a unidade hospitalar**, uma vez que é ele que lhes recomenda novas consultas e que lhes oferece um acompanhamento mais personalizado;
- **Os utentes não efectuam o registo das suas medições periódicas** quando lhes é pedido, uma vez que não visualizam o resultado de imediato, não possuem local próprio para registar, e muitas esquecem-se de levar os registos para as consultas com o médico.

Uma solução ideal para os problemas descritos, e aquela que foi pedida pelo cliente, consiste em permitir um acompanhamento mais personalizado do utente, fora da unidade hospitalar, que o recorde das suas consultas marcadas, **disponibilize alertas programados que sugiram a marcação de uma consulta, fruto de procedimentos estandardizados em planos de saúde e normas da unidade hospitalar**, permita realizar pedidos de marcação de consultas, permita inserir registos de medições de parâmetros de saúde, permita colocar questões aos auxiliares de saúde, e ainda, **utilize o histórico clínico dos utentes para detectar padrões patológicos** e sugira ao utente a marcação de uma consulta.

Os **alertas programados**, uma das principais funcionalidades da solução, consiste na capacidade de programação (em *runtime*) de alertas através da

codificação de Planos Nacionais de Saúde (PNS), nomeadamente do Plano Nacional de Vacinação (PNV), e de recomendações da própria unidade hospitalar, por exemplo, recomenda-se que as crianças até aos 5 anos de idade tenham uma consulta de pediatria por ano.

A codificação de conhecimento pertence ao domínio de Inteligência Artificial, uma vez que se deseja introduzir o conhecimento de profissionais de saúde numa aplicação para que esta possa, autonomamente, reproduzir um comportamento esperado. Esta funcionalidade foi alvo de uma exaustiva investigação de metodologias e produtos que melhor se enquadram para a sua implementação.

A **detecção de padrões**, outra funcionalidade relevante, não pretende ser um sistema de apoio à tomada de decisão, mas sim um sistema pró-activo que identifique, precocemente, casos problemáticos, e que alerte atempadamente o utente para marcar consultas. A detecção de padrões terá como plano de fundo todos os dados do utente, clínicos e pessoais, que se considerem relevantes, e tentará extrapolar, com base na detecção de casos semelhantes, padrões de patologias e de marcações de consultas.

A detecção de padrões implica uma exaustiva análise de metodologias, produtos e provas de conceito, uma vez que o âmbito desta funcionalidade situa-se no limiar do que é tecnologicamente alcançável actualmente. Conforme pedido pelo cliente, esta funcionalidade não existirá na primeira versão do produto, não sendo alvo de estudo durante o período de realização deste estágio, uma vez que é uma funcionalidade complementar do produto e que consiste num risco de implementação.

É importante sublinhar que os alertas enviados aos utentes que sugerem marcações de consultas evitam que o utente se desloque à unidade hospitalar e seja surpreendido com procedimentos recomendados que deveriam ter sucedido e com possíveis patologias cujos diagnósticos já deveriam ter avançado.

É conhecimento comum que na área da saúde, um sistema que não permita a **integração com outros sistemas** está praticamente condenado ao insucesso. Inúmeras investigações têm sido realizadas para tentar melhorar, cada vez mais, a integração entre vários sistemas hospitalares. Fruto dessas investigações surgiram protocolos importantes como o HL7 e o DICOM. É, portanto, necessário considerar ao nível da arquitectura a possibilidade de integração com dados provenientes de outros sistemas clínicos.

A solução a ser implementada terá assim, que ter em consideração a **dinâmica dos modelos de dados clínicos**. O sector da saúde está em constante evolução,

tanto em termos de conhecimento como de estrutura. Por um lado, parâmetros que consideramos hoje irrelevantes podem, no futuro, ser considerados indicadores de determinadas patologias. Por outro lado, são concebidos novos dispositivos capazes de uma maior precisão e de registar novos parâmetros de saúde até então desconhecidos. Apesar dos crescentes avanços de normalização neste sector, por exemplo, o HL7 v3 e o RIM, nem todas as aplicações suportam estas normas, e mesmo dentro das normas existe uma enorme flexibilidade. Para além disso, pode não ser desejado que a aplicação aceda a toda a estrutura de dados clínicos de destino, ou que aceda a alguns dados fora dessa estrutura.

Esta importante funcionalidade necessita de uma investigação que determine a melhor forma de implementação. Não é comum uma aplicação suportar alterações à estrutura de dados de forma dinâmica, permitindo ainda possa ser introduzido novo conhecimento que envolva dados que se situem nessa estrutura dinâmica.

A diversidade de modos de instalação é também um factor atractivo para as unidades hospitalares. A pedido do cliente, **a aplicação deverá poder ser instalada de forma isolada**, com o seu próprio painel de gestão (*BackOffice*), **ou de forma partilhada**, em que outra plataforma, por exemplo a Microsoft Dynamics CRM, serve de repositório de dados e de plataforma de gestão de dados. Este requisito implica uma investigação a nível da arquitectura que suporte ambas as opções de instalação e cumpra, ao mesmo tempo, as outras condições acima especificadas.

Não é o objectivo da aplicação produzir registos clínicos electrónicos (*Electronic Health Record* – EHR). No entanto, para a detecção de padrões e para a programação de alertas é necessário ter acesso a alguns dados dos registos clínicos dos pacientes, podendo até os completar com o registo de medições de parâmetros de saúde, construindo, de forma virtual, um registo clínico pessoal (*Personal Health Record* – PHR). Esta aplicação terá então que ser capaz de se integrar com os dados do sistema clínico instalado, apenas para leitura, não sendo necessário a escrita ou alteração dos dados directamente no modelo de dados do sistema clínico. Será também capaz de guardar os seus registos clínicos pessoais numa Base de Dados.

Não é também objectivo desta solução desenvolver novos dispositivos clínicos, apenas sustentar, tanto quanto possível, a evolução deste sector da saúde, tanto em termos de conhecimento como das métricas registadas por novos equipamentos.

O sistema que contemplará as funcionalidades acima descritas foi denominado Sistema de Acompanhamento do Desenvolvimento de Adolescentes e Crianças (SADAC). A primeira versão do sistema, alvo deste relatório de estágio curricular,

consiste na concepção, estudo e implementação de algumas das suas principais funcionalidades visando colmatar os problemas acima referidos.

Ao nível de estrutura da dissertação, o documento começa por descrever o **estudo bibliográfico e do estado da arte** realizado com o objectivo de construir um conhecimento sólido das vertentes de conhecimento que se aplicam a esta investigação. De seguida são clarificados os **objectivos da investigação e método de abordagem** que define as metodologias e tecnologias utilizadas ao longo da realização do projecto. A secção do **processo de implementação** destaca os principais desafios e métodos de abordagem do SADAC.

A secção de **levantamento e análise de requisitos** descreve os requisitos (funcionais e não funcionais) necessários para cumprir os objectivos do projecto, clarificando os que foram implementados com sucesso durante a realização do estágio. A secção seguinte, **arquitectura**, detalha a estrutura geral do SADAC tendo como pano de fundo a descrição do estado da arte e as respectivas aproximações metodológicas e tecnológicas consideradas.

A secção de **testes** detalha o cenário de testes, os vários tipos de testes e os principais resultados. A seguinte secção do **estado da implementação** resume o estado de implementação dos requisitos, especificando quais foram implementados e devidamente validados pelos *product owner*.

Segue-se a secção de **planeamento** que descreve o plano de trabalho previsto em comparação com o realizado, com a descrição fundamentada das implicações de eventuais desvios. Por fim são apresentadas as **conclusões** finais que exprimem o estado e a experiência do desenvolvimento do projecto, a **bibliografia** e o resumo dos **documentos anexados**.

2. Estudo Bibliográfico e do Estado da Arte

Conforme descrito anteriormente, colocam-se os problemas centrais a serem tecnologicamente analisados:

- É necessário um **módulo que produza alertas consoante o conhecimento inserido por profissionais de saúde**. Deve ser possível codificar Planos Nacionais de Saúde em vigor, nomeadamente o de Vacinação, e outros conhecimentos de natureza semelhante que a unidade hospitalar deseje aplicar;
- É necessário um **módulo que produza alertas extrapolando patologias ou marcações de consultas,** consoante os dados clínicos e pessoais dos utentes da unidade hospitalar, de forma autónoma, numa tentativa de prever futuras patologias com base em casos semelhantes;
- É necessária uma **aplicação acessível pelo utente e pelo corpo hospitalar**, permitindo ao utente o acesso fora da unidade hospitalar;
- A arquitectura deve permitir **instalar a aplicação com um painel de gestão integrado ou com a gestão realizada através de uma aplicação externa**, nomeadamente através da plataforma Microsoft Dynamics CRM;
- A arquitectura deve permitir uma **integração não só directa, mas também indirecta (por importação)** com os dados de sistemas clínicos externos tendo em conta as suas características dinâmicas, suportando a alteração da sua estrutura mesmo após a instalação e permitindo uma fácil adaptação à evolução.

O módulo que se baseia na base de dados de casos da unidade hospitalar **não será alvo de estudo exaustivo neste relatório**, uma vez que não foi previsto ser implementado dentro da duração do estágio. No entanto, é necessário considerar a sua existência uma vez que o módulo fará parte da aplicação SADAC e deve ser planeada a sua integração futura. Os restantes problemas foram alvo de uma análise exaustiva e metódica ao longo desta secção.

2.1. Análise Metodológica

Esta secção apresenta o estudo metodológico com o objectivo de determinar as metodologias indicadas para a resolução dos problemas acima referidos.

2.1.1. Produção de Alertas Baseados em Conhecimento

Esta secção foca-se na resolução do problema da produção de alertas baseados em conhecimento, com o objectivo de identificar a melhor metodologia a adoptar para a codificação, persistência e aplicação do conhecimento.

Pretende-se encontrar um método que permita codificar o conhecimento de profissionais de saúde da unidade hospitalar e guardá-lo numa Base de Dados de Conhecimento (*Knowledge Base*), para que possa ser lido, tanto por um ser humano como interpretado e aplicado pela máquina. Estes tipos de sistemas denominam-se **Sistemas Baseados em Conhecimento** (*Knowledge-Based Systems* – KBS).

Como o sistema pretende simular as sugestões de um profissional num dado domínio (perito ou profissional de saúde da unidade hospitalar), trata-se mais especificamente de um **Sistema Especialista**. Neste caso pretende-se codificar o conhecimento de profissionais de saúde da unidade hospitalar de forma a ser aplicado utente a utente, e consoante as informações do utente, gerar ou não determinados alertas.

O Plano Nacional de Saúde apresenta uma estrutura que já é baseada em regras, as quais os profissionais e auxiliares executam por vezes até mentalmente. Este processo baseado em regras reflecte-se no computador através do **Raciocínio Baseado em Regras**. Este mecanismo de raciocínio baseado em regras está presente nos **Sistemas de Gestão de Regras de Negócio** (BRMS).

Um BRMS necessita de um **Motor de Regras** que é, no seu núcleo, um mecanismo de execução de regras de negócio. As regras de negócio são simples afirmações orientadas ao negócio que de alguma forma codificam decisões e são normalmente escritas de forma condicional. O motor de regras executa determinadas acções caso as respectivas condições das regras se observem [3].

Segue-se um exemplo de uma regra numa linguagem praticamente natural, conforme o que está descrito no Plano Nacional de Saúde para as Vacinas:

- **SE** ((o utente tiver 18 meses de idade ou mais) **E** (ainda não tiver recebido a 4.ª dose de DTPa e a 4.ª dose de Hib))

  ENTÃO alertar para marcar consulta de pediatria para receber as vacinas DTPa (4.ª dose) e Hib (4.ª dose).

A implementação de um motor de regras é um processo exaustivo, complexo e demasiado demorado. Assim sendo, a solução mais natural será a **utilização de um**

motor de regras já implementado, reutilizando trabalho já feito e aumentado a compatibilidade com outros sistemas de regras já existentes.

Em termos de metodologia, o **Raciocínio Baseado em Regras** (através de um motor de regras), adequa-se mais às necessidades apresentadas de codificação de conhecimentos por parte de profissionais de saúde da unidade hospitalar (pessoas não técnicas), numa linguagem que tanto o profissional como o sistema consigam interpretar.

O sistema de regras a implementar visa codificar apenas regras simples, explícitas e formalmente definidas, nomeadamente as que estão presentes nos Planos Nacionais de Saúde (PNS), por exemplo o Plano Nacional de Vacinação (PNV). Poderão ser adicionadas outras regras que a unidade hospitalar deseje, mas não se prevê regras de grande complexidade. Por estes motivos, as limitações da utilização de Raciocínio Baseado em Regras pouco afectarão este projecto.

2.1.2. Aplicação Acessível pelo Utente

As principais características tecnológicas que a aplicação a ser acedida pelo utente deve respeitar são:

- A aplicação deve funcionar dentro e fora da unidade hospitalar, ou seja, para os profissionais de saúde e auxiliares da unidade hospitalar e para os utentes, sem configurações adicionais. A aplicação deve suportar os processos de autenticação e autorização e o conjunto de permissões a que o utilizador tem acesso;
- Compatibilidade com o *hardware* e *software* que os utentes possam possuir, uma vez que a aplicação cliente vai correr num dispositivo do utente;
- Deve ser possível cumprir os objectivos de alto-nível inicialmente propostos para a aplicação do utente.

Existem três possibilidades de criação de uma aplicação acessível pelo utente dentro e fora da unidade hospitalar:

- Uma aplicação *standalone cliente* que comunica com uma aplicação no servidor;
- Um portal (*WebSite*) dinâmico acessível por um *Browser* (Explorador de Internet);
- Uma aplicação para um dispositivo móvel.

Das três opções acima referidas, tendo em conta a preocupação da compatibilidade da aplicação com o *hardware* e *software* do utilizador, o tipo de aplicação que mais se ajusta é sem dúvida o portal. O portal consegue ser transversal aos tipos de dispositivos utilizados – é acessível através de um *Browser*, e todos os sistemas operativos mais utilizados actualmente (Microsoft Windows, Mac OS e Linux) permitem visualizar páginas de Internet através de um *Browser* [4]. Existem ainda telemóveis e outros dispositivos móveis como o iPad e Tablets, que também suportam a visualização de *WebSites*. O portal permite ainda a disponibilização de serviços web (*WebServices*) que podem ser acedidos por outras aplicações.

A implementação do portal pode ser complementada por uma ou várias aplicações para dispositivos móveis, uma vez que nos encontramos numa explosão tecnológica de dispositivos móveis e os utentes das unidades hospitalares privadas normalmente possuem maior poder de compra que no sector público. No entanto, não considero esta uma opção principal, uma vez que se estaria a restringir muito a quantidade de pessoas que poderia ter acesso à aplicação.

Caso o cliente deseje uma aplicação móvel será necessário uma análise de mercado por parte do departamento de marketing para investigar qual a tecnologia que teria mais impacto que se desenvolvesse a aplicação. Em termos técnicos, a aplicação poderia ser desenvolvida para qualquer sistema operativo móvel recente. Poderá ser, portanto, importante considerar parâmetros como a actual tendência de mercado.

A escolha da tecnologia para a implementação do portal é também indiferente a nível técnico, uma vez que é possível cumprir todos os objectivos acima referidos em qualquer uma das plataformas actualmente mais usadas para o desenvolvimento de páginas de Internet dinâmicas (PHP, ASP .NET, JSP, Ruby on Rails e Perl) [4]. Esta escolha tecnológica pode ser então feita em função da tecnologia que os módulos serão implementados, por razões de compatibilidade, ou na tecnologia que o cliente preferir.

2.1.3. Instalação Isolada ou Partilhada

A aplicação SADAC deve suportar dois tipos de instalação:

- **Isolada**, sem o auxílio de configuração de uma plataforma externa de gestão de dados. Será portanto necessário um *BackOffice* integrado e uma integração apenas a nível de dados clínicos;

- **Partilhada**, em que a configuração e o *BackOffice* da aplicação são realizados por uma aplicação externa, nomeadamente pela plataforma Microsoft Dynamics CRM. Será portanto necessária uma integração de todos os dados, clínicos e aplicacionais.

O *BackOffice* consiste na capacidade de gestão de pedidos de marcação dos utentes, gestão de mensagens, gestão de registos de medições de parâmetros de saúde, gestão de utilizadores, gestão de alertas e configuração do módulo de raciocínio baseado em regras (RBR).

Caso a aplicação seja instalada de forma **isolada**, terá que ser desenvolvido um *BackOffice* e terá que ser realizada uma integração apenas a nível de dados clínicos com a plataforma externa de gestão de dados clínicos. Os dados aplicacionais poderão ser persistidos numa base de dados própria.

A instalação de forma **partilhada** assume a directa integração da aplicação com dados de outras plataformas externas, nomeadamente, com a plataforma Microsoft Dynamics CRM. Este é um ponto muito importante e o qual teve de ser cuidadosamente estudado para avaliar o método mais adequado, as vantagens, desvantagens, compromissos e potenciais problemas de segurança que a integração envolve. É importante salientar que neste caso tanto os dados clínicos como aplicacionais deverão ser integrados.

Uma vez que deve ser possível integrar o SADAC com outras plataformas, e não só com a Microsoft Dynamics CRM, a arquitectura da aplicação deve estar preparada para isso. Uma arquitectura que permita definir qual o componente a utilizar para comunicar, sem ter conhecimento concreto do componente quando a aplicação foi compilada, é uma arquitectura com suporte de *plugins*, neste caso, *plugins* de acesso a dados. Deve ser realizado um estudo metodológico com o objectivo de determinar as melhores escolhas tecnológicas.

Por esse motivo, torna-se necessário utilizar uma metodologia a nível da arquitectura que permita desenvolver vários tipos de acesso a dados com o mínimo de alterações e menor impacto possíveis.

Arquitectura por Camadas

Independentemente do tipo de aplicação a desenvolver, e se tem uma interface com o utilizador ou apenas expõe serviços, é possível decompor o desenho da arquitectura em grupos lógicos de componentes de *software*. Estes grupos lógicos, denominados por *layers* ou camadas, diferenciam-se pelo tipo de tarefas

desempenhadas pelos componentes e permitem construir um desenho da arquitectura que facilita a reutilização de componentes. [5] As camadas lógicas podem também ser novamente subdivididas pelo tipo de tarefas que desempenham, resultando numa arquitectura por camadas. No caso de arquitecturas *N-Tiers* ou *3-Tiers*, a ideia base é a mesma, divisão e subdivisão por tipo de tarefas desempenhadas, mas a implementação é orientada, desde o início, para que as camadas (agora *tiers*) possam ser executadas em servidores diferentes, permitindo, por exemplo, *load-balancing* (balanceamento de carga) a vários níveis. [6]

Tipicamente uma aplicação (ou componente de *software*) é dividida em três camadas: apresentação, lógica de negócio (ou apenas negócio) e acesso a dados (ou apenas dados). Existem várias abordagens de desenho a considerar no que diz respeito à comunicação entre camadas [5]:

- ***Abstract Interface***

  Esta abordagem passa pela declaração de uma classe abstracta ou interface que serve como definição para a concretização de classes. Estas definições servem de API para as camadas que acedem, as camadas que acedem não necessitam de saber do tipo específico do objecto nem de todas as suas funcionalidades.

- ***Common Design Type***

  Muitas *design patterns* definem tipos de objectos concretos que servem de API, em vez de interfaces abstractas. É caso disso a conhecida *pattern Table Data Gateway*, que consiste numa classe concreta de acesso a dados que permite inserir, actualizar e eliminar registos abstraindo os mecanismos internos da camada de acesso a dados.

- ***Dependency Inversion***

  Esta opção de desenho consiste na separação das declarações de interfaces num componente separado dos componentes que concretizam as interfaces e do componente que constrói as concretizações. Esta abordagem evita referências circulares, em vez de um camada referenciar a outra, ambas as camadas referenciam o componente comum.

- ***Message-based***

  Em vez de interagir com componentes das outras camadas através de métodos ou aceder a propriedades é possível utilizar uma comunicação baseada em mensagens. Existem várias tecnologias que suportam esta forma de comunicação, por exemplo: *Windows Communication Foundation*, *Web Services* e *Microsoft Message Queuing*. É possível construir interfaces que concretizam esta comunicação através de

mensagens, o conceito chave é que o acesso entre camadas seja realizado através de mensagens.

Os *plugins* podem comunicar utilizando precisamente as mesmas abordagens, uma vez que se trata de mais uma camada de abstracção. Um *plugin* é constituído por um grupo de classes abstractas que definem o modelo que a concretização do *plugin* deve respeitar e a respectiva fábrica (*plugin factory*) que instancia o *plugin*. [7]

É possível integrar aplicações em cada uma das suas camadas em grande parte das aplicações. A integração em cada camada resulta num tipo distinto de integração, com as suas próprias especificidades:

- Integração a nível da apresentação
- Integração funcional
- Integração de dados

A seguinte ilustração apresenta os três tipos de integração aplicacional possíveis e as metodologias utilizadas:

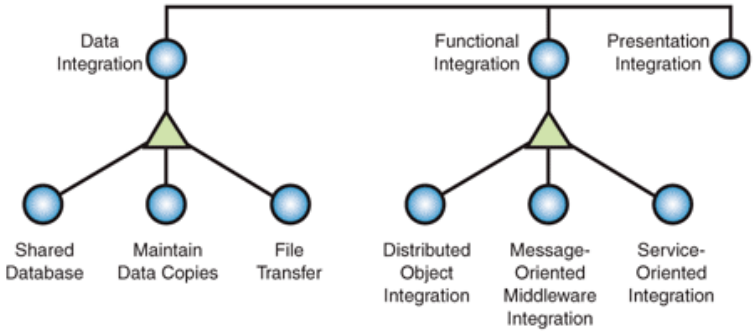

Ilustração 1 – Tipos de Integração [8]

Os componentes que permitem a integração de aplicações, independentemente da camada a que se destinem, denominam-se muitas vezes por conectores (*connectors*) ou adaptadores (*adapters*).

O primeiro tipo de integração a considerar é a integração a nível de dados [8], em seguida, a funcional. De uma forma geral, a integração a nível da apresentação apenas deve ser utilizada em último recurso, caso a aplicação seja um silo funcional, ou seja, não disponibilize interfaces de comunicação com a camada de negócio, e seja demasiado arriscado aceder directamente aos dados e ignorar a lógica da camada de negócio, tanto por questões de complexidade como de segurança. [9]

Neste caso específico, não é desejado integrar um sistema com outro com o objectivo de interagir com o sistema. A necessidade de integração foca-se nos dados, e, por esse motivo, a integração deverá ocorrer sempre ao nível dos dados, mesmo

que se utilize outro tipo de integração, o único objectivo será sempre o de manipular dados. O anexo "*Análise de Integração de Sistemas*" apresenta uma descrição mais completa das metodologias de integração de sistemas, os seus benefícios, compromissos e potenciais problemas de segurança.

O SADAC deve integrar com a Microsoft Dynamics CRM ao nível de dados clínicos e aplicacionais. Para isso, é necessário compreender as formas de integração que esta plataforma disponibiliza e qual a mais indicada.

Microsoft Dynamics CRM

A plataforma Microsoft Dynamics CRM permite a criação de produtos direccionados a uma linha de negócio (*Line of Business* – LOB), com características para aumentar e monitorizar a produtividade. É muitas vezes utilizado por departamentos de Marketing, Vendas e Apoio ao Cliente. [10] Apesar de conter as siglas CRM (*Customer Relationship Manager*), o que sugere gestão de relação com o cliente, é na verdade um produto muito mais amplo, sendo possível desenvolver aplicações para qualquer tipo de transacções e actividades, com uma fantástica capacidade de extensibilidade. Pelas palavras de Townsend, na altura presidente de um Gold Partner da Microsoft, e com mais de vinte anos de experiência em customizações "*It is a truism of software development that anything is possible with enough time and money, but you probably do not want to prove this with your own time or money*". [11]

A Dynamics CRM encontra-se preparada para uma integração vertical, em todas as suas camadas, com outros sistemas. Ao nível de apresentação, permite o desenvolvimento de listas, formulários personalizado, IFrames e ainda aplicações Silverlight completas. A nível da camada de negócio permite a integração com a Microsoft Dynamics ERP, Microsoft Excel, Microsoft SharePoint e outras aplicações através de *Web Services*. Permite também a criação de *plugins* de negócio.

A nível da camada de dados, a Dynamics CRM é construída tendo por base a Microsoft xRM Framework. Esta plataforma base permite o acesso a dados por aplicações externas com suporte nativo de permissões sobre a estrutura de dados, registo a registo, ao contrário de um simples DBMS.

Uma vez que o objectivo da integração com a Microsoft Dynamics CRM centra-se na necessidade de acesso aos dados, terá que ser implementado um *plugin* que permita a comunicação directa com os dados através da Microsoft xRM Framework. No caso de outras plataformas, a comunicação com os dados poderá ser realizada

utilizando outro tipo de integração ou protocolo, por exemplo: Hibernate, SQL, Web Services e REST.

2.1.4. Integração Directa e Indirecta de Dados Clínicos

A integração directa dos dados consiste na anteriormente referida *design pattern* de integração de dados *Shared Databases*, em que ambas as aplicações acedem à mesma base de dados. A integração indirecta consiste na *design pattern* de integração de dados *Maintain Data Copies*, em que existe um procedimento de sincronização de dados entre uma base de dados de origem e outra de destino.

O problema da integração com dados clínicos deve-se essencialmente à sua dinâmica. Se apenas fosse necessário integrar com um sistema de gestão de dados clínicos e esse sistema fosse gerido pelo cliente (como é o caso da aplicação para a Microsoft Dynamics CRM), a estrutura de dados poderia ser estática. No entanto, como existe uma vasta diversidade de sistemas de gestão de dados clínicos, todos com estruturas de dados diferentes, torna-se impossível prever a estrutura de dados de destino. Ao mesmo tempo, não é desejado que sempre que seja necessária uma integração com um sistema de dados clínicos o SADAC tenha que ser reconstruído, recompilado e lançada uma nova versão.

É, portanto, necessário desenvolver um *plugin* para acesso a dados clínicos com uma estrutura dinâmica de dados. Este *plugin*, no caso de uma integração directa, teria uma estrutura de dados semelhante à do sistema de gestão de dados clínicos de destino. No entanto, se a estrutura de dados fosse alterada por causa de uma actualização ou se fosse necessário ter em consideração mais uma entidade, seria necessário realizar modificações ao *plugin*. Numa integração indirecta, onde os dados do sistema clínico de destino são importados para a aplicação SADAC, o mesmo acontece. Não é possível evitar a recompilação do SADAC ou do *plugin* a não ser que seja possível configurar a estrutura de dados do *plugin* em *runtime*, sem recompilação.

Para resolver o problema é necessário desenvolver um componente que permita gerir entidades dinâmicas, realizar algum negócio básico e permita a sua apresentação ao utilizador. Este tipo de componente é utilizado pelas plataformas de gestão de dados dinâmicos, por exemplo, a Microsoft Dynamics CRM e o Microsoft Sharepoint. Será necessário realizar um estudo tecnológico mais detalhado de como estas plataformas implementaram a gestão de entidades dinâmicas.

Este componente de entidades dinâmicas deve permitir construir a estrutura de dados clínicos compatível com o sistema de gestão de dados clínicos existente. A integração neste caso passará pela conversão dos dados do sistema de gestão de dados clínicos para a estrutura de dados configurada pelo módulo de entidades dinâmicas. Se existir uma modificação no sistema de dados clínicos ou for desejado incluir mais alguma informação, é possível configurar o módulo de entidades dinâmicas para ter essas novas informações em consideração.

2.2. Enquadramento Tecnológico

O anexo "*Enquadramento do Estudo do Estado da Arte*" apresenta um enquadramento mais exaustivo do Raciocínio Baseado em Regras, das Regras de Negócio e dos Motores de Regras de Negócio. Esses tópicos serão abordados no relatório de uma forma mais sucinta.

2.2.1. Raciocínio Baseado em Regras

A capacidade de resolver problemas com base em regras denomina-se Raciocínio Baseado em Regras (*Rule-Based Reasoning* – RBR), e é um tipo de Sistema Especialista mais comum [12], pertencente à área de Sistemas Baseados em Conhecimento da Inteligência Artificial.

2.2.2. Regras de Negócio

O conceito de regras de negócio não detém uma definição única formal e unanimemente aceite. De acordo com a definição mais frequente, uma regra de negócio é uma afirmação que caracteriza um aspecto específico de um modelo operacional de negócio, assente no pressuposto que a estratégia de negócio pode ser definida como o encadeamento de processos de negócio concretos. Esta caracterização pode incidir sobre diferentes aspectos do conhecimento sobre o qual assenta a lógica de negócio, podendo reflectir-se, por exemplo, num condicionamento de fluxo, na detecção de eventos, dedução de informação, entre outros.

As regras são normalmente compostas por duas partes: as condições (*Left-Hand Side* – LHS) e as acções ou consequências (*Right-Hand Side* – RHS). É comum apresentarem-se na forma "SE <Condições> ENTÃO <Acções>", onde as acções são despoletadas de forma condicionada.

2.2.3. Motores de Regras de Negócio

Os motores de regras de negócio são ferramentas que executam determinado conjunto de regras de negócio (ou seja, determinam as consequências lógicas) com base em factos ou axiomas. [13] [14]

Consoante o modo como operam, os motores de regras podem ser classificados em três tipos: *forward-chaining*, *backward-chaining* e híbridos (suportam ambas as estratégias anteriores) [15]. Normalmente o próprio problema define qual a estratégia a utilizar, por exemplo, se se deseja partir de factos (*forward-chaining*) ou de objectivos (*backward-chaining*). Também pode ser automaticamente estimado o melhor método comparando o número de condições face ao número de acções determinando a estratégia que em média será mais eficaz [16].

Um motor de regras, de uma forma geral, avalia as regras da sua base de dados verificando se as condições se cumprem de acordo com os factos presentes na sua memória de trabalho e executa as respectivas acções. Caso o motor suporte a reavaliação de regras, este processo acontece até que nenhuma regra possa ser despoletada.

Apesar de na teoria um motor de regras implementar uma técnica simples, os grandes desafios encontram-se na definição das regras e na escalabilidade do algoritmo.

Definição das Regras

Se a definição das regras for específica do motor de regras os profissionais de saúde (responsáveis pela sua configuração) terão que aprender tantas linguagens quantos os sistemas com que tiverem que lidar. Se for demasiado genérica, poderá ser tão técnica e complexa que poderão ser necessários conhecimentos de linguagens de programação. Em qualquer dos casos, se não for implementada segundo uma norma, a base de dados de regras perderá a capacidade de poder ser integrada com outros sistemas.

Para evitar que as bases de dados de regras fiquem restritas a um determinado *software* ou empresa (*vendor lock-in*), existem algumas normas que permitem facilitar a integração com outras aplicações. Desta forma, escolhendo um motor de regras que suporte determinada norma possibilita a integração com outros sistemas de regras que também a suportem, ou até mesma a substituição de um motor de regras por outro.

Optimizações

A execução sequencial de regras, no caso de *forward-chaining* (em termos de desempenho a *backward-chaining* é semelhante), consiste na avaliação das condições de cada regra, face aos factos que cada registo de dados define. Isto é, numa base de dados de 10 regras, e de 10 registos de dados, a primeira regra é escolhida (de acordo com um determinado método de escolha) e as suas condições são avaliadas conforme os factos do primeiro registo de dados. Caso as condições se cumpram, as acções dessa regra são despoletadas. Se essas acções modificarem os próprios factos, todas as regras poderão ter que ser novamente avaliadas, caso contrário, procede-se ao mesmo algoritmo de execução para a segunda regra sobre o primeiro registo de dados. Este processo implica a avaliação mínima de 100 condições (10 regras sobre 10 registos) na base de dados especificada, facilmente se conclui que apresenta um desempenho polinomial ($O(n \times p)$) assumindo que não existe alteração dos factos.

Para bases de dados com centenas de milhares de registos e com centenas ou milhares de regras, um desempenho polinomial pode não ser o suficiente, principalmente se estiver integrado num sistema em tempo-real. Em vez de efectuar uma execução sequencial, alguns motores de regras estruturam as suas bases de dados de regras e de registos com o intuito de reduzir a redundância, melhorando assim o desempenho. As metodologias mais utilizadas para esse efeito são: Rete I, Rete II, Rete III, Treat, Rete* e Leaps, descritas no anexo “Enquadramento do Estudo do Estado da Arte”.

2.2.4. Sistemas de Gestão de Regras de Negócio

Os Sistemas de Gestão de Regras de Negócio (*Business Rules Management System* - BRMS) são um tipo de *software* orientado à definição, gestão e execução da lógica operacional de um sistema através de regras. [17]

Ao nível do funcionamento, um BRMS típico tem três componentes [17]:

- Um **repositório de armazenamento** de regras;
- Ferramentas que permitam a **criação, edição e formalização de regras** (*rule authoring*) por parte de analistas de negócio – expressão de conceitos de negócio em linguagem natural – e que possibilitem aos especialistas em tecnologias de informação o mapeamento entre conceitos de negócio e estruturas programáticas;

- Um ambiente de execução e de interacção, que permita a entidades externas aceder e invocar as regras de negócio a nível aplicacional, ou seja, um **motor de regras de negócio**.

As principais vantagens da utilização de sistemas assentes em regras são a separação de papéis e a autonomia das unidades organizacionais. O departamento responsável pela orientação estratégica assume o controlo directo sobre o comportamento operacional dos sistemas empresariais, sem estar dependente do departamento tecnológico para a implementação dos mesmos [18]. Estes factores conferem autonomia às tarefas relacionadas com a elaboração e edição de regras, o que resulta, a curto/médio prazo, em menores custos de manutenção operacional. Esta abordagem traduz-se ainda numa maior eficiência na gestão estratégica e num aumento do potencial de automatização da gestão operacional. A separação da camada lógica do restante código é facilitadora da criação de mecanismos de actualização automatizada por parte de agentes externos [17].

2.3. Análise Tecnológica

Esta secção representa os estudos tecnológicos realizados para determinarem os produtos ou tecnologias mais adequadas para a resolução dos problemas inicialmente propostos, de acordo com as metodologias anteriormente identificadas.

2.3.1. Análise de Motores de Regras de Negócio

Esta secção descreve resumidamente a análise exaustiva realizada a 33 motores de regras detalhada no documento anexo "*Análise de Sistemas de Gestão de Regras de Negócio*". Foram analisados sistemas tanto gratuitos como comerciais.

Inicialmente foram reunidos todos os produtos cuja descrição pertencesse à categoria de "motores de regras" através de artigos e pesquisas na Internet (gratuitos, de código aberto e comerciais), resultando num total de trinta e três (33) produtos. Desses, foram excluídos todos os produtos que constituem *abandonware* (ou seja, cujo desenvolvimento tenha sido descontinuado e não exista suporte do produto), que não sejam focados na gestão de regras de negócio ou que não tenham maturidade suficiente. Após esta exclusão, restaram seis (6) produtos a serem analisados com mais detalhe, três comerciais e três gratuitos.

Estes seis produtos foram objecto de um processo de análise e comparação mais exaustivo, tendo em vista o seu enquadramento no cenário operacional do cliente

– nomeadamente, ao nível da declaração, edição, validação e execução de regras de negócio.

2.3.1.1. Processo de Exclusão Preliminar

Esta secção resume o processo de exclusão de alguns motores de regras de negócio, segundo critérios bem definidos. Foram utilizados os seguintes critérios de exclusão:

- **Natureza do Projecto**

  Não orientados para gestão de regras de negócio.

- **Abandonware**

  Identificados como abandonados.

- **Falta de Maturidade**

  Não foram encontrados estudos de casos reais.

A aplicação destes critérios de exclusão sobre o total de 33 motores de regras de negócio identificados resultou na exclusão dos seguintes 27 motores de regras:

- **Excluídos pela Natureza do Projecto**

  Esper, Jena, Prova, OFBiz Rule Engine, Microsoft BizTalk Server Business Rule Engine (MS-BRE), Hammurapi Event Bus.

- **Excluídos por Abandonware**

  Jamocha, Rools, InfoSapient, CommonRules, Zilonis, JRuleEngine, JEOPS, jDREW, Bossam, jLisa, SweetRules, Algernon, Mandarax, OpenLexicon, Hammurapi Rules, CLIPS, Take, TermWare, NxBRE.

- **Excluídos por Falta de Maturidade**

  DTRules, Ruleby.

Desta exclusão restaram 6 motores de regras a serem analisados com maior detalhe, uma vez que os 6 representam potenciais escolhas para serem incorporados.

2.3.1.2. Processo de Análise

Foram estipulados os seguintes critérios de análise com vista o estudo objectivo das funcionalidades e características relevantes de cada ferramenta.

- **Grau de Actualidade**

Este critério pretende avaliar até que ponto um projecto está activo e em contínuo desenvolvimento. É relevante na medida em que evita adoptar um produto descontinuado e desactualizado.

- **Tipo de Licença**

A subsecção relativa ao tipo de licença pretende identificar a licença ou conjunto de licenças associadas à utilização do *software* em análise.

- **Documentação**

Este critério pretende analisar a documentação do produto, tanto a nível da quantidade como da qualidade, apresentação de exemplos, estruturação e outros factores relevantes.

- **Posição de Mercado**

O parâmetro posição de mercado visa identificar o grau de adopção do produto no meio académico, empresarial e a popularidade.

- **Requisitos**

Este critério pretende avaliar as dependências que as ferramentas apresentam em termos de instalação e execução, ou seja, necessárias para o ambiente de execução.

- **Tecnologias, Tipos de Regras e Formatos Suportados**

O objectivo deste critério é avaliar o nível de compatibilidade da plataforma com as várias tecnologias, formatos e quasi-standards disponíveis actualmente.

- **Integração com Outros Produtos**

O parâmetro de integração com outros produtos visa analisar as potencialidades de cada solução ao nível da interoperabilidade funcional com sistemas e ferramentas externas.

- **Ferramentas de Apoio**

Este critério identifica ferramentas indicadas para apoiar o desenvolvimento com o produto.

Segue-se o resumo do estudo efectuado da avaliação das características dos produtos de acordo com os critérios de análise previamente estipulados. O anexo "*Análise de Sistemas de Gestão de Regras de Negócio*" contempla o estudo completo.

Foram analisados com maior detalhe os seguintes motores de regras:

- **Drools (JBoss Rules)**
- **OpenRules**
- **WF Rules**
- **Blaze Advisor**
- **ILOG JRules**
- **Orcale Business Rules**

A descrição dos motores de regras permitiu compreender os respectivos vectores funcionais, o contexto envolvente de cada motor de regras, o enquadramento empresarial e permitiu a identificação de critérios de comparação importantes que deram origem ao processo de comparação resumido na próxima subsecção.

2.3.1.3. Processo de Comparação

Esta secção resume a análise comparativa realizada sobre os 6 motores de regras anteriormente referidos. Seguem-se os critérios utilizados:

- **Metodologias**

  Tipo de produto e metodologias de execução das regras suportados.

- **Forma de Definição de Regras**

  Enumera as diversas formas que o produto dispõe para definir regras.

- **Ambiente de Desenvolvimento**

  Descreve os ambientes de desenvolvimento suportados de forma nativa.

- **Documentação**

  Descreve a quantidade e qualidade da documentação do produto.

- **Comunidade**

  Este critério avalia a popularidade do produto quantificando entradas em *blogs* e fóruns.

- **Tipo de Licença**

  Especifica o tipo de licença (ou licenças) sob o qual o produto é distribuído.

A seguinte tabela sintetiza a informação recolhida considerando os critérios acima especificados:

	Metodologias	Definição de Regras	Ambiente de Desenvolvimento	Documentação	Comunidade	Licença
Drools	BRMS – RETE v1 Melhorado	Sintaxes Próprias (2); DSL	Java	Excelente	Fórum: 10.006 tópicos; *Blog*: 685 entradas;	Apache License v2
OpenRules	BRMS – Tabelas de Decisão	Tabelas de Decisão	Java	Excelente	*Blog*: 20 entradas.	GPL v2 / Comercial
WF Rules	BRMS – Sequencial; Por Prioridade	Graficamente; Sintaxe Própria	.NET	Excelente	Fórum: 13.321 tópicos.	BCL; Ms-RSL
Blaze Advisor	BRMS – RETE v3	Sintaxe Própria; Árvores de Decisão; Tabelas de Decisão; *Scorecards*	Java / .NET / Web Services	Excelente	Fórum: 2.968 *posts*; *Blog*: 905 entradas.	Comercial
ILOG JRules	BRMS – Baseado em RETE	Árvores de Decisão; Tabelas de Decisão; Sintaxes de Linguagem Natural	Java / .NET / Web Services	Excelente	Fórum: 16.241 tópicos.	Comercial
Oracle Business Rules	BRMS – RETE v1	Graficamente; Tabelas de Decisão	Java / Web Services	Excelente	Fórum: 12.405 tópicos.	Comercial

Tabela 1 – Comparação das Características dos BRMS

2.3.1.4. Conclusões

As soluções comerciais analisadas constituem produtos completos e orientados à integração em arquitecturas pré-existentes orientadas a serviços (SOA). São, como seria de esperar, produtos mais robustos, comtemplado os seus próprios servidores aplicacionais, ferramentas de apoio, mecanismos de integração e suporte técnico especializado. Uma vez que são produtos naturalmente mais completos, justifica-se a separação da análise entre os sistemas comerciais dos sistemas gratuitos.

A nível comercial, o Oracle Business Rules é claramente inferior, com uma menor diversidade de declaração de regras e não suporta a plataforma Microsoft .NET nativamente. O Blaze Advisor, líder a nível de desempenho, dispõe de uma fraca comunidade. O **ILOG JRules**, segundo os critérios especificados, é o melhor produto comercial e que melhor se enquadraria na implementação do produto desejado.

De entre os produtos gratuitos analisados o OpenRules é o que tem uma menor comunidade, com apenas 20 entradas no *blog* oficial, e suporta apenas a

declaração de regras através de tabelas de decisão. Torna-se portanto o produto mais fraco. O WF Rules, apesar de ter uma forte comunidade e permitir a declaração de regras graficamente ou através de uma sintaxe própria, não suporta nenhuma optimização de execução de regras para além da prioridade, colocando-o atrás do Drools que dispõe de uma versão própria e melhorada do algoritmo Rete I com um desempenho claramente superior aos outros produtos comerciais.

O **Drools**, desenvolvido em Java e com um desempenho superior a todos os sistemas gratuitos estudados, comparado a sistemas comerciais e com vários testemunhos de empresas satisfeitas com a sua adopção, com suporte da declaração de regras através de várias sintaxes próprias ou construção de sintaxes adaptadas ao domínio do problema, e com uma comunidade rica e activa, é o BRMS de eleição para o desenvolvimento do SADAC. Suporta também as normas JSR-94 e JSR-331 que facilitam a integração com outros sistemas.

2.3.2. Análise de Entidades Dinâmicas

Não foi possível encontrar produtos que permitam a criação de entidades dinâmicas embutidas numa aplicação. Existem sim produtos que permitem a criação de formulários dinâmicos, muitas vezes *online*, como é o caso do Wufoo[1], ou Google Docs[2] (ou recentemente Google Drive), que permitem a criação e partilha de formulários dinâmicos.

Como anteriormente referido, tanto o Microsoft Sharepoint como a Microsoft Dynamics CRM contém um componente que gere internamente entidades dinâmicas. Estes produtos são instaláveis num servidor permitindo uma análise mais detalhada da abordagem que utilizam para a organização e persistência dos dados dinâmicos.

O Microsoft Sharepoint é um produto colaborativo que permite administrar e personalizar *Web Sites*, gerir listas e bibliotecas e gerir e publicar documentos e relatórios. A Microsoft Dynamics CRM, como anteriormente referido no enquadramento tecnológico, é uma plataforma que permite a criação de produtos direccionados a uma linha de negócio (*Line of Business* – LOB), com características para aumentar e monitorizar a produtividade.

No que diz respeito a entidades dinâmicas, ambos os produtos apresentam soluções bastante semelhantes, permitem a criação de entidades com a definição de campos e respectivos tipos através da construção de formulários e construção de

[1] Disponível em http://wufoo.com/
[2] Disponível em https://drive.google.com/

vistas que servem para listar as entidades. Esta será então a base para a criação de entidades dinâmicas.

A criação de uma nova entidade despoleta a criação de uma, ou várias, novas tabelas de base de dados, com os respectivos campos configurados. Cada vez que o modelo é alterado, é realizada uma validação da consistência entre o modelo e a estrutura da base de dados.

No que diz respeito a persistir a **definição de campos** e entidades, ou seja o modelo, existem várias alternativas:

1. Tabelas de bases de dados;
2. Definição em formato binário;
3. Definição em texto/XML.

Tabelas de Bases de Dados

A seguinte imagem ilustra o modelo conceptual (apenas para a concretização das definições) segundo esta abordagem.

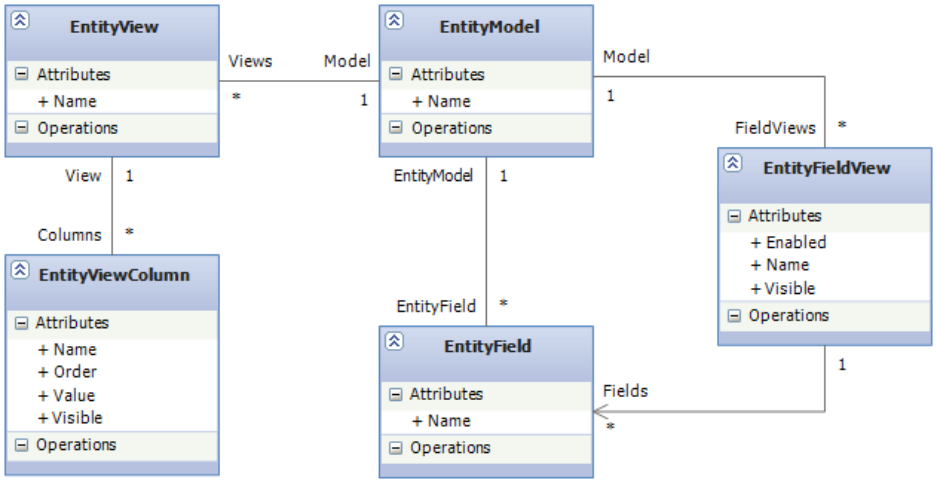

Ilustração 2 – Modelo DynamicEntity em Tabelas de Bases de Dados

Na abordagem das várias tabelas de bases de dados, em termos de estrutura da base de dados, existe uma tabela para os modelos, uma tabela de campos (para os formulários) relacionada com a de modelos e outra de vistas relacionada com os campos (para as listagens de entidades). Ao distribuir a definição das entidades e respectivos campos por várias tabelas de bases de dados trás a vantagem de permitir alterações colaborativas com menor granularidade. Desta forma seria possível um utilizador adicionar um campo enquanto outro utilizador modifica uma vista.

Esta abordagem traz, no entanto, uma grande desvantagem a nível de desempenho. Na maior parte do tempo de vida das aplicações a operação mais

comum será a de carregar um modelo, e não a de modificar um modelo. A operação de carregar e construir um modelo segundo esta abordagem é, sem dúvida, a abordagem mais lenta. É necessário aceder a várias tabelas relacionadas, e a muitos campos, apenas para obter a definição do modelo. Na operação de modificação não será comum vários utilizadores tentarem modificar o modelo ao mesmo tempo, uma vez que é um processo que caberá a um conjunto muito específico e restrito de pessoas.

Definição em Formato Binário

A seguinte imagem ilustra o modelo conceptual da base de dados segundo esta abordagem:

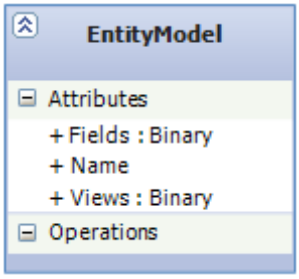

Ilustração 3 – Modelo DynamicEntity em Formato Binário

Esta abordagem persiste o modelo em base de dados em formato binário. Assumindo a existência de uma tabela de Modelos, cada registo representará um modelo e um campo da base de dados deverá conter a definição dos campos e outro campo a definição das vistas, ambos em formato binário.

O desempenho desta abordagem é claramente superior à anterior, e uma vez que o formato é binário (serialização dos dados em memória), o seu carregamento completo implica obter um único registo de base de dados, não relacionado com outras tabelas, e a construção do modelo implica uma reconstrução binária da memória muito rápida. É também o formato mais compacto.

Como principal desvantagem, ao guardar o modelo em formato binário impossibilita a edição da definição por parte de um humano, sem a ajuda de uma ferramenta externa, uma vez que a definição não é legível por um humano. Isto implica construir e disponibilizar uma ferramenta para a edição da definição de modelos, ou a solução não é minimamente viável.

Definição em Formato Texto/XML

A seguinte imagem ilustra o modelo conceptual da base de dados segundo esta abordagem:

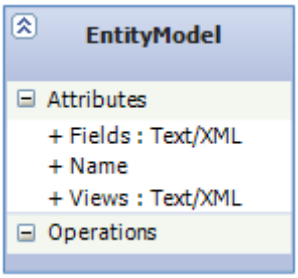

Ilustração 4 – Modelo DynamicEntity em Formato Texto/XML

Tanto o Microsoft Sharepoint como a Microsoft Dynamics CRM como a plataforma Windows Workflow Foundation no que diz respeito a definições de fluxo de processos, persistem os dados em XML. Esta metodologia é muito semelhante à anterior em que guarda a definição num campo de uma tabela de base de dados de modelos, mas agora no formato Texto/XML.

Em termos de desempenho, continua a ser possível obter todo o modelo através de um único registo da base de dados. O conteúdo do campo tem depois de ser tratado por um *parser* de XML, que tratará de validar o XML e construir a definição de campos e de vistas. A primeira abordagem continua a ser mais lenta uma vez que também iria necessitar de uma construção das definições entidade a entidade, acrescentado o facto de ser necessário relacionar várias tabelas para obter o modelo completo. Tem no entanto um desempenho inferior à segunda abordagem, a binária, uma vez que a abordagem binária realiza uma reconstrução dos dados em memória.

Ao persistir o modelo neste formato de texto (XML), é possível que qualquer utilizador experiente e dotado de um manual realize alterações às definições dos campos e das vistas sem ser necessária uma ferramenta externa. As alterações realizadas por um utilizador poderão ser validades por um XSD (*XML Schema Definition*) ou um DTD (*Document Type Definition*) garantindo que a definição de modelos é válida.

Esta metodologia não inibe que se construam ferramentas visualmente ricas de edição do modelo, simplesmente não obriga a que isso aconteça, ao contrário das abordagens anteriores.

Persistência dos Registos Dinâmicos

É também importante analisar as várias formas de persistir os dados dinâmicos na base de dados. De forma um pouco semelhante com o que se passa nos modelos, foram identificadas as seguintes alternativas:

1. **Campo XML com os atributos e respectivos valores**
   Os atributos das entidades e respectivos valores são armazenados em XML na base de dados. Por exemplo, cada paciente tem vários registos de exames, cada um deles com um campo em XML com os atributos e valores do exame;

2. **Utilização do Modelo EAV (Entity-Attribute-Value)** [19]
   Este modelo é bastante comum para a representação de dados clínicos. A definição dos atributos é realizada numa tabela relacionada com uma tabela de entidades (por exemplo, a entidade exame possui os atributos tipo de exame, data do exame e diagnóstico), e existe uma segunda tabela para os valores, associada à tabela de atributos, que especifica os valores de cada atributo, na forma *Key-Value*.

3. **Modificação da Base de Dados (Alter/Create Table)**
   Esta metodologia é utilizada, por exemplo, pela Microsoft Dynamics CRM. Consiste na criação de tabelas, em *runtime*, que representam as entidades, e de colunas da base de dados, que representam os atributos. Logicamente os valores dos atributos são persistidos na respectiva coluna da tabela, como seria esperado num modelo de dados estático, com a diferença do modelo de dados poder ser alterado em *runtime*.

4. **Base de Dados Pré-Construída**
   Esta é a metodologia utilizada, por exemplo, pelo Microsoft Sharepoint. Consiste numa tabela genérica com um número de colunas de determinados tipos pré-estabelecido, e apenas é necessário fazer o mapeamento entre determinada coluna e determinado atributo. Por exemplo, uma tabela contém 80 campos de texto, 20 campos de números inteiros, e assim sucessivamente, e é feito um mapeamento que especifica quais as colunas que determinadas entidades utilizam (os atributos). Esta metodologia evita a alteração da estrutura da base de dados em *runtime*.

Foi realizada uma comparação criteriosa das várias metodologias. Seguem-se os critérios aplicados e sobre os quais será realizada uma reflexão acerca do comportamento que a metodologia aplica. Os critérios foram inspirados em diversas publicações encontradas [19, 20], tal como as conclusões comparativas:

- **Número de Atributos Suportados (Atributos)**
  Qual o limite (se algum) do número de atributos suportados pela metodologia. Aproveita-se para referir também se é possível extrair apenas alguns atributos da base de dados, o que é importante porque alguns atributos podem ser binários e pode não ser desejado carregar a entidade toda, apenas alguns atributos.

- **Obtenção de um Registo (*Browsing*)**
  Qual a complexidade envolvida na obtenção de um registo completo da base de dados, e qual a velocidade comparativamente com as outras alternativas.

- **Obtenção de Muitos Registos (*Bulk Extraction*)**
  Qual a complexidade envolvida na obtenção de uma vasta quantidade de registos da base de dados, e qual a velocidade comparativamente com as outras alternativas.

- **Pedidos de Dados Relacionados (*Ad-hoc Queries*)**
  Qual a complexidade necessária para realizar pedidos que relacionem vários atributos e de várias entidades e qual a velocidade envolvida nesse processo.

Segue-se uma tabela comparativa das vantagens e desvantagens que as metodologias apresentam, de uma forma teórica.

	Atributos	Browsing	Bulk Extraction	Ad-hoc Queries
XML	**Ilimitados** Só é **possível** obter todos os atributos.	Lento (CPU) e complexo (*parsing*).	Extremamente lento (CPU) e complexo	CPU *killer*, ou seja, é tão lento e complexo que não devem ser realizadas.
EAV	**Ilimitados** É **possível** obter apenas um atributo.	Lento (DB) e complexo, visto que o nome do atributo não é guardado junto do valor do atributo.	Muito lento (DB), mas menos que o XML. É também tão complexo como o *browsing*.	Um pouco lento (DB) e muito complexo. É necessário atravessar uma série de tabelas só para compreender quais os registos desejados para ad-hoc.
Modificação da DB	**1.024 ou 30.000** para tabelas *wide.* É **possível** obter apenas um atributo.	Rápido, simples e intuitivo.	Rápido, simples e intuitivo.	Rápido, simples e intuitivo.
DB pré-construída	**1.024 ou 30.000** para tabelas *wide*, mas com tipos pré-estabelecidos. É **possível** obter apenas um atributo.	Rápido, simples mas pouco intuitivo uma vez que os nomes das colunas não correspondem ao nome dos atributos.	Rápido, simples mas também pouco intuitivo.	Rápido, simples mas também pouco intuitivo.

A tabela apresenta claramente uma enorme vantagem na utilização de técnicas de modificação da base de dados em *runtime*. A única desvantagem desta abordagem

face às outras é a complexidade da sua implementação – é a mais complexa de implementar. É portanto a abordagem seleccionada para a implementação.

Comparação das Validações XSD e DTD

O formato XML é altamente configurável, universal, humanamente legível e é um formato extensível. É possível representar qualquer tipo de dados em XML, mesmo dados binários, e todas as linguagens de programação de alto nível possuem *parsers* orientados à leitura deste formato. [21]

Uma vez que uma das grandes vantagens deste formato é ser humanamente legível, é importante garantir que as alterações realizadas por um humano são válidas. Para o efeito, é possível validar toda a estrutura de um documento XML através de XSD ou DTD.

O DTD permite validar o formato de um XML tendo em consideração os seus elementos, atributos e a hierarquia de elementos. [22] Não permite, por exemplo, controlar o tipo de dados interno dos atributos, valores por omissão dos atributos e valores possíveis, listas de elementos com combinações específicas de ordem ou repetição ou relações entre atributos.

O XSD tem muito mais potencialidades, permite futuras extensões, a própria validação é escrita em XML, suporta tipos de dados e suporta *namespaces*. Por todos estes motivos e outros, esta norma é considerada a sucessora do DTD. [23]

Conclusões

A abordagem de persistência de modelos que mais vantagens trás e que mais se enquadra na implementação funcional do SADAC é a persistência em XML, utilizada também pelo Microsoft Sharepoint e Microsoft Dynamics CRM. Para garantir que o utilizador não coloca o XML num estado ilegível, deverá ser produzido um XSD que valide as alterações.

2.3.3. Análise de Sistemas com Registos Pessoais de Saúde

O SADAC não é um sistema que tem como principal objectivo o registo de dados de saúde de forma electrónica, nem pretende centralizar toda a informação de saúde do utente. O SADAC pretende aproximar os utentes de determinada unidade hospitalar, ou cadeia de unidades hospitalares, acompanhando o desenvolvimento do utente. No entanto, porque o sistema irá reunir informações pessoais de saúde do utente, introduzidas por ele e por profissionais de saúde, é importante perceber as actuais estratégias mais importantes para o realizar.

O anexo "*Análise de Sistemas com Registos Pessoais de Saúde*" identificou as principais funcionalidades das duas principais plataformas comerciais que permitem registos pessoais de saúde, o Google Health e o Microsoft HealthVault [24]. O projecto Google Health, tal como referido no documento, foi declarado como encerrado [25], o que permitiu uma análise das principais questões levantadas por profissionais de saúde, técnicos da área e utilizadores, acerca do que terá levado o Google Health a não cumprir os seus objectivos. [26]

O documento em anexo identifica as principais funcionalidades e ajuda a construir um vector de sucesso importante para qualquer sistema de gestão de PHR. É importante ter em consideração as críticas e funcionalidades identificadas ao longo da implementação do SADAC para evitar erros que poderiam ser facilmente reduzidos. Apesar do SADAC não pretender ser um sistema de PHR, é um sistema colaborativo que apresenta funcionalidades muito semelhantes.

3. Objectivos da Investigação e Método de Abordagem

Os objectivos deste estágio são a concepção, detalhe e desenvolvimento da primeira versão do Sistema de Acompanhamento do Desenvolvimento de Adolescentes e Crianças (SADAC).

O SADAC é um sistema que tem como finalidade, por um lado, a melhoria da qualidade de vida dos utentes, na medida em que lhes proporcionará um acompanhamento mais próximo e personalizado mesmo fora da unidade hospitalar, e por outro lado, uma vez que os utentes se tornam mais satisfeitos, melhorar a fidelização dos utentes com a unidade hospitalar.

O sistema proporcionará um acompanhamento do utente através de uma plataforma *online*, o Portal do Utente, que permitirá realizar pedidos de marcação de consultas *online*, a troca de mensagens entre o utente e os profissionais ou auxiliares de saúde, o registo contínuo de medições de parâmetros de saúde e a gestão de alertas direccionados ao utente com sugestão de marcação de consultas de forma cómoda e atempada.

Os clientes finais do sistema são os departamentos de marketing das unidades hospitalares, em especial, do sector privado. Os utilizadores deste sistema serão os profissionais e auxiliares de saúde das unidades hospitalares, que poderão configurar e gerir o sistema, e os utentes, que poderão usufruir dos serviços e gerir a informação disponibilizada através do Portal do Utente.

A primeira subsecção explica os objectivos de implementação do projecto SADAC, ou seja, os módulos que serão efectivamente implementados e que metodologias foram adoptadas para a implementação. A subsecção seguinte consiste nos processos de gestão de qualidade adoptados, em particular, a gestão da documentação, gestão de tarefas e cópias de segurança. A seguinte subsecção detalha o processo de desenvolvimento de *software* adoptado. A última subsecção especifica os riscos mais importantes identificados ao longo do desenvolvimento do projecto.

3.1. Objectivos

A primeira versão do SADAC consiste essencialmente num **Portal do Utente**, um módulo de **Raciocínio Baseado em Regras** e um módulo de **Entidades Dinâmicas**, como o seguinte diagrama de componentes apresenta.

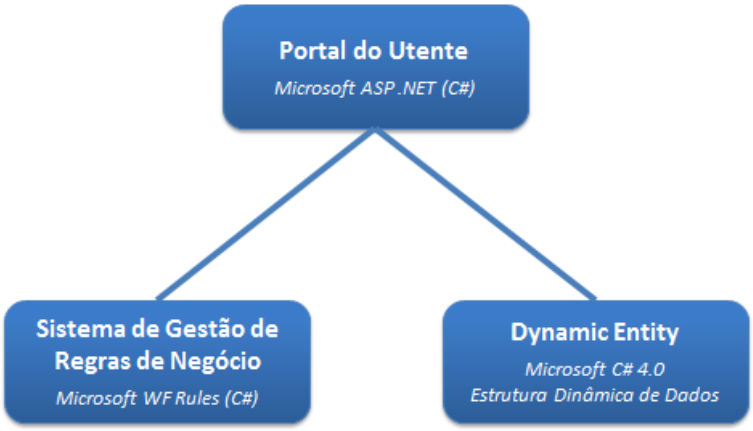

Ilustração 5 – Principais Componentes do SADAC

Devido a **restrições do cliente**, todos os componentes serão desenvolvidos utilizando tecnologias da Microsoft, mais especificamente, **Microsoft C# 4.0**. O sistema de gestão de regras de negócio escolhido, tendo em conta esta restrição, foi o **WF Rules**, utilizado para desenvolver o módulo de Raciocínio Baseado em Regras.

O **Portal do Utente** disponibiliza a informação de saúde aos utentes, alertas, permite a conversação com profissionais de saúde, gestão de pedidos de marcação de consultas e o registo de dados de saúde, como por exemplo, a pressão arterial, o peso, a altura, o índice de glicémia e a pulsação. É importante ter em mente normas de acessibilidade universal [27, 28] no desenho do portal, uma vez que se espera ter um público-alvo muito abrangente e com possíveis capacidades reduzidas.

Os alertas na primeira versão do SADAC serão provenientes do módulo de **Raciocínio Baseado em Regras** (RBR). Este módulo tem como objectivo permitir a codificação de Planos Nacionais de Saúde, em especial o Plano Nacional de Vacinação, e outros protocolos que a unidade hospitalar deseje introduzir. Para além do módulo permitir a codificação de novas regras e gestão das existentes após a aplicação estar instalada (*deployed*), deve conseguir interpretar estas regras e aplica-las aos utentes existentes, gerando alertas que serão visualizados pelos utentes através do Portal do Utente. Os utentes poderão aceitar a sugestão do alerta para a marcação de uma consulta ou ignorar o alerta.

Se o produto for instalado em conjunto com um sistema de gestão de dados clínicos que permita a configuração de um *BackOffice*, como é o caso do produto que o cliente desenvolveu sobre Microsoft Dynamics CRM, o *BackOffice* do portal será realizado por esse produto. Caso contrário, o portal servirá também como *BackOffice* para a configuração do sistema de dados clínicos e do módulo de raciocínio baseado em regras por profissionais e auxiliares de saúde.

O módulo de **Entidades Dinâmicas** do SADAC proporciona a flexibilidade de ajustar a estrutura de dados clínicos de suporte, mesmo após a instalação final da aplicação. A extensão desta estrutura de dados permite incorporar novas entidades e alterar entidades existentes permitindo ao módulo de RBR aceder a estes novos dados para realizar determinadas decisões, sem ser necessária a recompilação do sistema. Por dados clínicos entende-se, por exemplo, exames, tratamentos, substâncias administradas e patologias do utente.

Caso a estrutura de dados do sistema clínico seja sobre Microsoft SQL Server, poderá ser possível configurar o módulo de Entidades Dinâmicas para aceder directamente aos dados clínicos, sem qualquer módulo adicional. Caso não seja possível um acesso directo, terão de ser construídos conversores que importem os dados do sistema clínico para o sistema SADAC. De qualquer forma, o módulo de entidades dinâmicas permite a reconfiguração do sistema sem necessitar de recompilação.

Por exemplo, o sistema pode ser instalado numa unidade hospitalar que tenha o registo dos medicamentos que os utentes recebem, e o módulo de RBR pode ter isso em consideração para perceber se deve ou não despoletar determinado alerta. Mais tarde a versão do sistema clínico recebe um *upgrade* e permite agora a gestão de substâncias ao pormenor. É possível configurar o módulo de entidades dinâmicas para incorporar estas alterações estruturais sem ser necessário voltar a lançar uma nova versão personalizada.

A seguinte ilustração identifica os principais módulos da especificação funcional do SADAC.

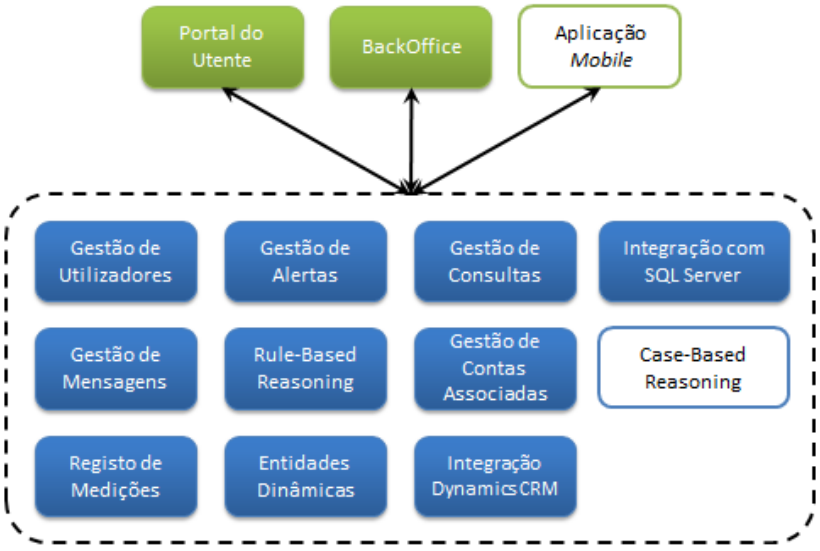

Ilustração 6 – Especificação Funcional do SADAC

Os módulos **preenchidos a cheio** fazem parte da especificação funcional da primeira versão do SADAC e foram completamente concebidos durante o estágio, os módulos **preenchidos a branco** especificam as futuras funcionalidades do SADAC, os módulos a **verde** especificam funcionalidades que interagem directamente com o utilizador (*front-end*), os módulos a **azul** especificam funcionalidades com negócio implícito e sem interface com o utilizador (*back-end*).

É importante referir que esta primeira versão funciona de forma integrada com a Microsoft Dynamics CRM, ou através do módulo de Entidades Dinâmicas em conjunto com um sistema de gestão de dados clínicos. Neste último caso, poderá ser necessário construir um módulo de integração entre o sistema de gestão de dados clínicos específico e o módulo de entidades dinâmicas, no entanto, o SADAC não necessita de ser recompilado.

O módulo de **Raciocínio Baseado em Casos** (*Case-Based Reasoning* – CBR) irá integrar directamente com o módulo interno de gestão de alertas, como já acontece com o módulo de Raciocínio Baseado em Regras (RBR).

A **aplicação *mobile*** é uma possível funcionalidade para o qual o sistema se encontra preparado através da divisão do sistema por camadas. A aplicação *mobile* vai integrar com a camada de negócio de forma muito semelhante ao que já acontece com o Portal do Utente.

3.2. Metodologia de Desenvolvimento de Software

Planeou-se para o primeiro semestre concluir as primeiras fases do ciclo de vida do projecto, a fase de concepção e de elaboração, segundo a RUP. O segundo semestre deveria focar-se na fase de construção e, se possível, de transição, ou seja, a instalação do sistema, também segundo a RUP.

Apesar da metodologia base utilizada ser a RUP, vai ser utilizada uma *framework* de desenvolvimento de *software* ágil – o Scrum [29]. É comum denominar o Scrum como *framework* e não como metodologia uma vez que o Scrum se trata de um conjunto de pequenas metodologias e recomendações deixando em aberto alguns pontos que devem ser complementados. Esta *framework* foi acordada entre a equipa de desenvolvimento e o cliente do projecto que acordou que seria vantajosa a sua utilização, visto permitir trabalhar com uma pequena equipa de desenvolvimento de *software* (até 7 pessoas) ao mesmo tempo que permite ao cliente controlar a ordem das funcionalidades a serem implementadas e ter um *feedback* mais rápido. [30]

O desenvolvimento ágil de *software* (do inglês *Agile Software Development*), ou método ágil, é um conjunto de metodologias de desenvolvimento de *software* que enfatizam a comunicações em tempo real, preferencialmente face-a-face, a documentos escritos, e o trabalho no *software* como uma medida primária de progresso, uma vez que se foca na produtividade da equipa de desenvolvimento. Exemplos de metodologias ágeis são o Scrum, Crystal Clear e a prática de Extreme Programming (XP). [30]

A *framework* de desenvolvimento de *software* Scrum distingue três fases de desenvolvimento no ciclo de vida do projecto [31]:

- ***Pre-Game***

  Esta fase foca-se em dois pontos essenciais: planeamento e arquitectura. O planeamento pretende dar uma visão ao projecto, elaborar a lista de requisitos (*Product Backlog*), identificar a equipa de desenvolvimento e as ferramentas necessárias, elaborar um documento de controlo de riscos e resolver os custos financeiros. A arquitectura ou desenho de alto-nível pretende responder a desafios tecnológicos com as respectivas soluções a adoptar e a aprendizagem das ferramentas e tecnologias necessárias. Corresponde essencialmente às fases Concepção e Elaboração da RUP, de forma mais simplificada.

- ***Game***

  Por vezes chamada como *Development Phase* ou *Sprint Phase*, esta fase dura normalmente uma a quatro semanas, e consiste no ciclo principal de desenvolvimento de *software*. Aqui a equipa de desenvolvimento analisa (olha para o *Product Backlog*), planeia (adiciona algumas tarefas ao *Sprint Backlog*) e implementa as funcionalidades, revê o progresso através do *burndown chart* (gráfico do progresso actual face ao progresso planeado) e avalia desta forma a necessidade de alterar as estratégias de implementação ou as funcionalidades a implementar no *Sprint* actual. Esta fase pode consistir em mais de um *Sprint*, uma vez que pode-se considerar que apenas existe uma *Release* (lançamento oficial de uma nova versão) passadas algumas *Sprints*, ou um *Sprint* pode corresponder exactamente a uma nova *Release*. Corresponde essencialmente à fase Construção da RUP, de forma mais simplificada.

- ***Post-Game***

  Por vezes chamada de *Wrapper*, trata-se da fase da preparação das funcionalidades implementadas no *Sprint* para serem integradas no produto final (a ser oficialmente lançado), a integração propriamente dita, o teste e actualização da documentação. Corresponde essencialmente à fase Transição da RUP, de forma mais simplificada.

A ilustração seguinte apresenta o ciclo completo do desenvolvimento em Scrum, com as três fases acima detalhadas:

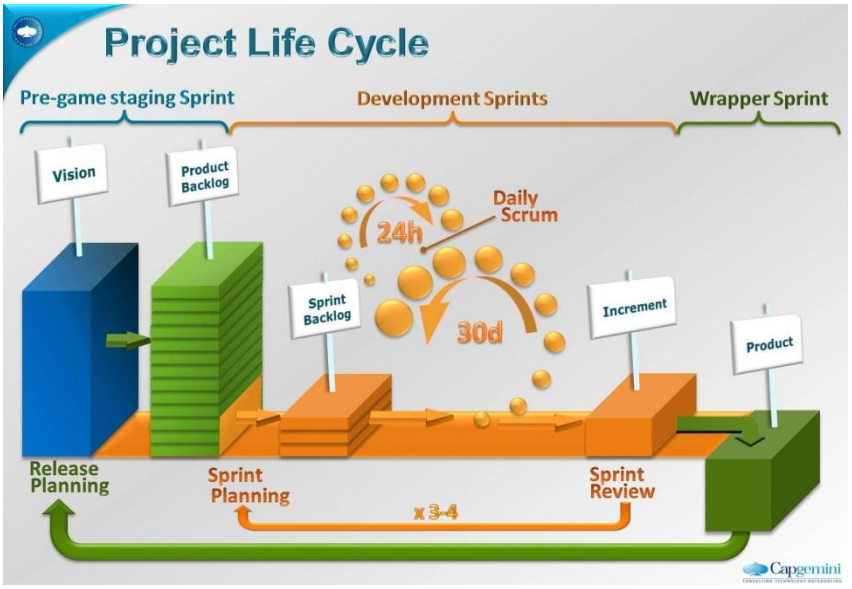

Ilustração 7 – Ciclo de Desenvolvimento Scrum[3]

[3] Imagem retirada de http://mscop.be.capgemini.com/tag/scrum/

Segue-se uma imagem com o processo principal do desenvolvimento utilizando a *framework* Scrum (fase *game*) mais detalhado:

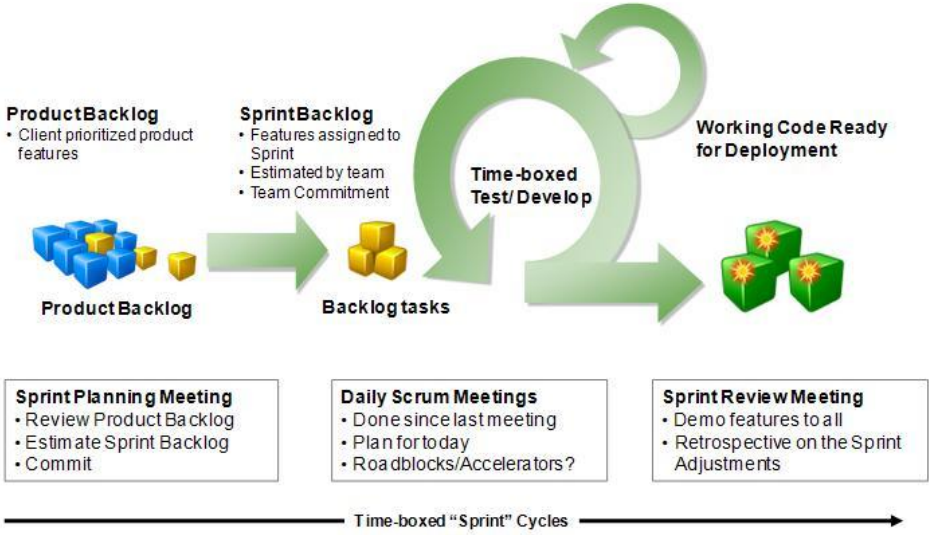

Ilustração 8 – Ciclo Principal de Desenvolvimento em Scrum[4]

O desenvolvimento em Scrum assenta sobre três papéis principais:

- ***Product Owner*** – Dr. Eduardo Oliveira (BOND)

  Este papel corresponde normalmente ao cliente final, caso não seja possível, deve ser uma pessoa que consiga defender o produto numa perspectiva de valor comercial, tendo em conta as expectativas do público-alvo.

  Determina que funcionalidades devem ser implementadas na próxima *Sprint*, uma vez que o seu papel é fundamentalmente orientar o desenvolvimento das funcionalidades para que o produto final fique o mais próximo possível do que os clientes finais desejam, baseando-se por exemplo, em estudos de mercado. Deve saber, por exemplo, que funcionalidades são essenciais, quais são as mais importantes, e quais são mero detalhe, logo menos importantes.

- ***Scrum Master*** – Eng. Alcides Marques (IPN)

  Assegura que o processo de desenvolvimento está a ser correctamente implementado, motiva, protege e reduz o *stress* da equipa e desbloqueia os problemas encontrados pela equipa para que possa trabalhar o mais suavemente possível. Foca-se na melhoria da equipa e da qualidade do produto.

[4] Imagem retirada de http://agilescrum.biz/six-sigma-and-agile-software-development/

- ***Scrum Team*** – Fernando Rocha (Estagiário)

  Responsável por construir as funcionalidades acordadas para cada *Sprint* e demonstrar o que desenvolveram. O *Product Owner* terá a demonstração das funcionalidades em conta para definir o que deve ser desenvolvido a seguir.

O primeiro semestre do estágio correspondeu à fase de *Pre-Game* do Scrum. No segundo semestre pretendeu-se realizar o conjunto de acções necessárias para se concluir a primeira *Release* do produto, o seu teste e a sua integração, concluindo assim uma iteração completa do ciclo Scrum.

Esta metodologia de desenvolvimento impõe reuniões muito importantes a ocorrer durante o desenvolvimento: as *Sprint Planning Meetings*, as *Daily Scrum Meetings* e as *Sprint Review Meetings*. [32, 33]

Sprint Planning Meeting

A *sprint planning meeting* ocorre no primeiro dia da sprint, dia que é totalmente ocupado com esta reunião, mesmo que não demore o dia todo. A reunião é dividida em duas partes, uma primeira parte com o *product owner*, *scrum master* e a *scrum team*, e uma segunda parte com o *scrum master* e a *scrum team*.

Na primeira parte o *product owner* descreve as funcionalidades que entende serem mais importantes a desenvolver na *sprint*. Estas funcionalidades são traduzidas em *user stories*, e ficam registadas no *product backlog* do projecto. Idealmente o *product backlog* deve conter todas as *user stories* discutidas até ao momento. Os restantes participantes colocam dúvidas, sugestões e tentam compreender como as funcionalidades devem ser implementadas, ou seja, dividem a *user story* em tarefas. No final da primeira parte deve existir uma lista de tarefas, ordenada por prioridade, passíveis de serem desenvolvidas na *sprint*.

Na segunda parte é realizada uma estimativa para a duração das tarefas, normalmente através de técnicas de *planning poker* [34]. No *planning poker*, o *scrum master* lidera a reunião e apresenta as tarefas. Cada elemento da *scrum team* mostra, ao mesmo tempo, uma carta com um valor de horas associado, caso os valores sejam muito dispersos, discute-se o porquê dessa dispersão e acorda-se um tempo estimado, caso sejam muito semelhantes, o tempo estimado é evidente. Independentemente da técnica, o objectivo é a estimativa de horas ou quantidade de trabalho associado a cada tarefa. É através desta estimativa e durante esta reunião que a equipa decide o quanto se compromete a realizar na *sprint*, e os objectivos essenciais a cumprir na *sprint* para ser considerada um sucesso.

Daily Scrum Meeting

Na *daily scrum meeting*, ou *daily standup* como por vezes denominada, o *scrum master* tem como objectivo compreender o estado de implementação da *sprint* e as principais dificuldades que a equipa está a encontrar. Para isso, dispõe de apenas dez minutos, e de preferência em pé. Esta reunião não pretende de todo resolver problemas, apenas dar os problemas a conhecer ao *scrum master* e ao resto da *scrum team*, para uma eventual partilha de conhecimentos. Os problemas podem ser discutidos posteriormente, se necessário, e os vários elementos podem se ajudar, incluindo o *scrum master*. Esta reunião é importante para compreender se o objectivo da *sprint* vai ser cumprido, ou se está em risco, avaliando o resultante *burndown chart* até ao momento.

Sprint Review and Retrospective Meeting

Esta reunião envolve os mesmos participantes que a *sprint planning meeting* e tem como objectivo apresentar a toda a equipa, em especial ao *product owner*, o que foi efectivamente implementado na sprint. Também esta reunião se divide em duas partes, a primeira com todos os envolvidos e a segunda com o *scrum master* e a *scrum team*. Também não devem ser planeadas tarefas para este dia.

Na primeira parte o *product owner* percorre as tarefas realizadas procurando compreender o que foi realizado e qual o estado geral da implementação das *user stories*. Caso se acredite que a *user story* está concluída, deve ser realizada uma pequena demonstração da funcionalidade que comprove os respectivos critérios de aceitação. Caso os critérios sejam cumpridos, a *user story* é marcada como completa, caso contrário, é necessário compreender e registar o que mais falta realizar.

Na segunda parte o *scrum master* analisa o que correu bem, o que correu mal e o que pode ser melhorado na próxima *sprint*. Normalmente o *product owner* não está presente nesta reunião, uma vez que é mais técnica. É importante a equipa se focar também em potenciais problemas de desempenho e outras métricas qualitativas. O *scrum master* deve tentar compreender os impedimentos comuns e tentar proporcionar o melhor ambiente de desenvolvimento possível para melhorar a produtividade da equipa.

Todas estas reuniões tomaram lugar durante o desenvolvimento do projecto SADAC, conforme recomendado.

3.3. Gestão de Qualidade

Durante o estágio adoptei algumas normas de desenvolvimento de *software* para organizar o meu trabalho e assegurar um bom desempenho e qualidade. Esta secção descreve as ferramentas, processos e metodologias utilizadas para esse efeito.

3.3.1. Documentação

Para facilitar a organização, todos os documentos, em especial os documentos a serem oficialmente entregues, respeitam a seguinte regra para construção do seu nome:

<Nome do Documento> v<Versão><Iniciais do Autor>.<Extensão>

Por exemplo:

Documento de Requisitos v0.2FR.docx

A versão inferior do documento avança sempre que após ser modificado, seja enviado (publicado) a alguém. A versão superior avança sempre que o documento é considerado estável, ou seja, as pessoas que o reviram o considerem de acordo com o estado actual do projecto. Estar de acordo com o estado actual não impede novas alterações, uma vez que é sempre possível tentar melhorar o documento ou actualizá-lo para assimilar novas informações.

3.3.2. Plataforma de Gestão de Tarefas

Todas as tarefas executadas durante o estágio, e a respectiva duração, foram sendo registadas numa plataforma comum ao LIS, no Microsoft SharePoint. Este registo permite saber quanto tempo foi ocupado nas tarefas, quanto estava previsto, e quais serão as próximas tarefas a fazer. Este registo minucioso permite ao orientador ter um melhor controlo sobre as tarefas a serem realizadas e ao mesmo tempo permite-me a mim também ter informação mais detalhada sobre como o *Gantt* planeado está a ser cumprido.

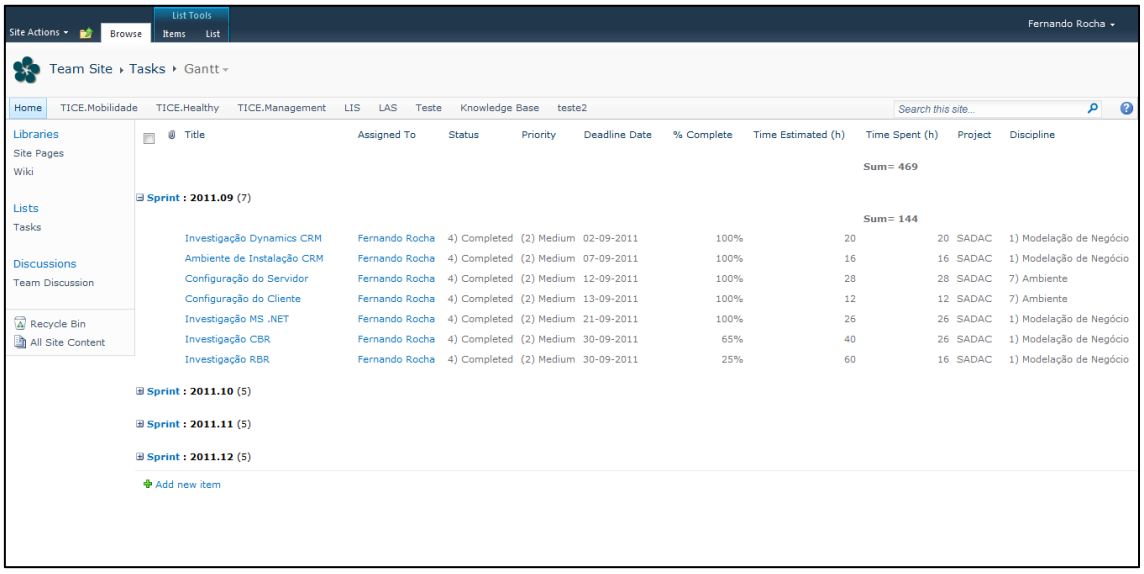

Ilustração 9 – Página do SharePoint com Tarefas

A imagem apresenta as tarefas agrupadas por mês. Cada tarefa apresenta o título, pessoa responsável, estado, prioridade, deadline, percentagem de trabalho realizado, o tempo estimado, o tempo despendido, o projecto e a disciplina RUP (*IBM Rational Unified Process*) [35] correspondente. É ainda possível abrir a tarefa e ver mais informações, como por exemplo, a sua descrição detalhada.

Este registo de tarefas permite também uma fácil construção do *burndown chart*, utilizado na metodologia de desenvolvimento de *software* Scrum, acima descrita.

3.3.3. *Backup* da Documentação

Utilizei a versão gratuita da aplicação Dropbox[5] para assegurar o *backup de* toda a documentação. Esta aplicação efectua uma cópia de segurança dos dados seleccionados para um local remoto na Internet, de forma automática e em segundo-plano. Para reduzir a quantidade de informação sem *backup*, deve-se garantir que a *Dropbox* tem tempo para sincronizar completamente essa informação com o servidor, ou no mínimo, uma vez por semana. Esta estratégia permite também o controlo de versões dos ficheiros, apesar de não ser esse o principal objectivo, proporcionando uma forma fácil de recuperar versões anteriores de ficheiros ou ficheiros eliminados.

3.3.4. *Backup* do Código-Fonte

Para assegurar o *backup* de todo o código-fonte (*source code*) e recursos associados será utilizado um repositório Git[6] (repositório distribuído de controlo de

[5] Disponível gratuitamente em https://www.dropbox.com/
[6] Disponível gratuitamente em http://git-scm.com/

versões). Este repositório permite o *backup* do código fonte ao mesmo tempo que permite o desenvolvimento colaborativo de aplicações e gestão de versões de ficheiros. Este sistema de controlo de versões é distribuído, isso permite que o código fonte não se encontre apenas no repositório, mas também em todos os computadores que o tenham subscrito, tornando o processo mais resistente a riscos de perda. Para que o repositório e todos os computadores associados tenham sempre a versão mais actualizada, deve-se enviar as alterações, no mínimo, uma vez por semana.

3.3.5. Reuniões Periódicas

As reuniões são muito importantes no processo de desenvolvimento de *software* – permitem a discussão de problemas, soluções, ideias e perspectivas. A *framework* de desenvolvimento utilizada, o Scrum, premia reuniões face-a-face em prol de uma comunicação mais aberta, humana e activa. Ao contrário de pedidos formalmente elaborados em função de, por exemplo, alterações da interface com o utilizador ou novos requisitos, documentos comuns noutras metodologias de desenvolvimento, a *framework* Scrum retira a maioria da burocracia documental e estimula à comunicação face-a-face.

No segundo semestre, fase *game*, foi seguida a recomendação de reuniões segundo a *framework* Scrum, as *sprint planning meetings*, as *daily scrum meetings* e as *sprint review and retrospective meetings*. Algumas reuniões *planning* e *review and retrospective* foram realizadas por intermédio da aplicação Skype[7], uma vez que o cliente se encontra em Lisboa. As *sprint planning meetings* ocorreram no início de cada *sprint*, as *daily scrum meetings* ocorreram entre as 10 e as 11 da manhã e as *sprint review and retrospective meetings* no último dia das *sprints*. O primeiro e o último dia de cada *sprint* ficou sempre reservado para estas reuniões.

A técnica de estimativa do tempo das tarefas nas *sprint planning meetings* foi através da discussão do que a tarefa envolve e tendo como medida mínima um dia (8 horas) e como medida máxima 10 dias (80 horas).

3.4. Gestão de Riscos

A gestão de riscos é o processo que identifica, avalia e prioriza os riscos (definido no ISO 31000 como *o efeito da incerteza nos objectivos*, quer seja positivo ou negativo), seguido de uma estratégia coordenada para aplicar recursos para reduzir,

[7] Aplicação disponível em http://skype.com/

monitorizar e controlar a probabilidade ou impacto desses eventos, tendo também em conta o custo económico que isso envolverá. [36]

Os riscos podem ocorrer em qualquer fase do projecto, podem ter causas naturais (ou mesmo desastres naturais), acidentes, ataques deliberados ou indeliberados, ou simples eventos com causas incertas ou imprevisíveis. A *framework* Scrum, apesar de simplificada face a outras, por exemplo o desenvolvimento em Cascata [37], não coloca de parte a gestão de riscos.

Para ter noção dos riscos que se correm e tentar evitar os mesmos, é importante ter sobre monitorização os principais riscos, aqueles cujo impacto e probabilidade de ocorrerem são maiores. Essa lista deve ser revista periodicamente, colocando ou removendo riscos dada a actualização da sua probabilidade de ocorrer e impacto. De facto, devido à natureza tão geral dos riscos, não é recomendado fazer o levantamento e registo de todos os riscos que podem ocorrer, apenas aqueles considerados mais importantes, uma vez que tudo pode ser levado ao extremo do incerto, ainda mais numa era tão dependente da tecnologia. Nada nos garante que o nosso computador permaneça funcional durante as próximas horas, nem sequer conseguimos estimar com precisão o tempo que efectivamente vai estar funcional, por muito que o tentássemos [38], é necessário por isso ser pragmático e focar apenas os riscos mais prováveis e com maior impacto.

Os seguintes subcapítulos descrevem as estratégias de mitigação para os principais riscos identificados.

3.4.1. Imprecisão das Funcionalidades

Em projectos de *software* é bastante comum existir uma imprecisão na definição das funcionalidades devido à falha na comunicação entre engenheiros de *software* e engenheiros de negócio (ou cliente). Esta imprecisão pode levar a implementar funcionalidades de forma errada, ou no limite, funcionalidades despropositadas. Este é um risco que ocorre ao longo de todo o desenvolvimento do projecto.

Estratégia de Mitigação

Para evitar que este problema ocorra é necessário garantir que a visão do produto do cliente coincide com a visão do produto da equipa de desenvolvimento. Para esse efeito, a equipa de desenvolvimento participa nas reuniões com o cliente. É o cliente que determina também as próximas tarefas a serem implementadas. O

cliente deve também validar a visão de implementação das tarefas mais complexas através de protótipos de baixa-fidelidade (*mockups*).

Ocorrência do Risco

Análise se os requisitos foram implementados consoante a descrição e consoante os protótipos de baixa-fidelidade, e diálogo com o cliente face-a-face para encontrar uma solução viável.

3.4.2. Inexistência de Dados Reais

Este risco consiste na impossibilidade de obter dados de exemplo reais de determinada unidade hospitalar. Esta necessidade é fundamental para a implementação do Raciocínio Baseado em Casos (CBR), uma vez que é necessário revelar uma estrutura de dados subjacente e casos reais para permitir a correcta avaliação se o algoritmo funciona como desejado. Dados fabricados não fariam qualquer sentido e os resultados seriam inconclusivos.

Estratégia de Mitigação

Deve-se tentar obter dados tanto através de parcerias com unidades hospitalares como também de outras formas. Enquanto não for possível obter dados reais de exemplo é sempre possível desenvolver o resto da aplicação à excepção do módulo de CBR.

Ocorrência do Risco

No caso de não ser possível obter dados de exemplo antes de instalar a aplicação num cliente, talvez seja possível nessa altura extrair alguns dados de forma anónima para a implementação do módulo de CBR, ou pelo menos testar o seu sucesso directamente na unidade hospitalar. Se mesmo assim for impossível, a funcionalidade de CBR terá que aguardar pela existência de dados reais.

3.4.3. Inexistência de Cliente Final (Unidade Hospitalar)

Este risco foi identificado na primeira reunião com o cliente do IPN, e foi mais tarde confirmado o insucesso de angariação de uma unidade hospitalar parceira. Este risco agrava o risco de não existirem dados de exemplo, uma vez que se fosse possível ter uma unidade hospitalar parceira, provavelmente iria fornecer a estrutura de dados e dados necessários para o projecto prosseguir, caso contrário, dificulta esta obtenção. Também implica directamente com as funcionalidades que devem ser implementadas primeiro, uma vez que sem cliente final as funcionalidades são

escolhidas pela utilidade que se estima ser relevante e não pela utilidade que o cliente final lhe atribui.

Por fim, a introdução da Microsoft Dynamics CRM como plataforma de gestão de dados clínicos juntamente com o SADAC torna-se um risco se não for encontrado um cliente final aberto a estes sistemas. A plataforma Microsoft Dynamics CRM implica custos adicionais elevados e pode não ser desejada pelo cliente final uma vez que poderá já ter um sistema de gestão de dados clínicos e não queira abdicar dele.

Estratégia de Mitigação

É necessário um esforço para angariar um cliente final, tentando envolve-lo no processo de implementação e escolha de funcionalidades e tentando poupá-lo de processos burocráticos ou dispendiosos, destacando as funcionalidades planeadas que o cliente mais desejar.

Ocorrência do Risco

Enquanto não for possível angariar um cliente final, as funcionalidades devem ser implementadas pela relevância que o cliente determinar. Não sendo possível garantir que o SADAC terá sucesso juntamente com a Microsoft Dynamics CRM, o SADAC terá que ter em consideração poder ser instalado de forma individual (sem a Dynamics CRM) e directamente integrado com uma plataforma de gestão de dados clínicos a definir pelo cliente final, aquando da sua instalação.

3.4.4. Escolha do WF Rules

A escolha desta tecnologia como sistema de gestão de regras de negócio pode ser um risco na medida em que não é um produto comercial e não suporta as optimizações RETE que outros produtos como o Drools suportam. Para além dos problemas de performance, o WF Rules é um sistema bastante fechado em si, as suas funcionalidades estão bem documentadas mas se for desejado esticar um pouco as suas capacidades, poderá ser necessário utilizar técnicas avançadas de reflexão para implementar as funcionalidades. No limite, pode não ser possível utilizar este produto devido a alguma restrição de implementação.

Estratégia de Mitigação

Deve ser realizado um estudo a esta ferramenta para apurar a sua capacidade de dar resposta às necessidades funcionais da definição e execução de regras do SADAC. Deverá ser implementado um protótipo funcional que demonstre a capacidade de definição e execução de regras sobre um contexto dinâmico de dados.

Ocorrência do Risco

Caso não seja possível realizar alguma operação através do WF Rules, deverá ser investigada uma forma de resolver o problema, mesmo que por reflexão. Se não for possível resolver o problema, deve ser encontrado um novo produto para substituir o WF Rules.

4. Processo de Implementação

Este capítulo destaca os **principais desafios** encontrados durante o processo de implementação do SADAC e os detalhes mais importantes. Os módulos que apresentaram os maiores desafios de desenho e implementação foram: o módulo de **Raciocínio Baseado em Regras** (RBR), o módulo de **Entidades Dinâmicas** e o Portal do Utente, nomeadamente a **Arquitectura de *Plugins*** e o ***Routing***.

4.1. Raciocínio Baseado em Regras

O módulo de Raciocínio Baseado em Regras foi implementado tendo por base o sistema de gestão de regras de negócio WF Rules. O WF Rules é um sistema gratuito e nativo da plataforma Microsoft .NET 3.0 (e superiores), integrado na plataforma Windows Workflow Foundation. Como é um sistema gratuito apresenta algumas limitações a nível de desempenho e de funcionalidades.

Como o uso do WF Rules foi considerado uma escolha tecnológica arriscada, a estratégia de mitigação foi realizar um estudo das suas capacidades e implementação de uma aplicação (protótipo) que comprovasse a capacidade funcional do sistema.

Em termos de desempenho, vários artigos afirmam que dificilmente se justifica a utilização de um motor de regras comercial se não for necessário utilizar regras com lógica de negócio complexa [39, 40], o que não é o caso dos Planos Nacionais de Saúde. De facto, não é sequer necessário qualquer encadeamento de regras (*forward/backward chaining*) ou reavaliação das condições, uma vez que não ocorrem alterações do conhecimento acerca do utente – uma execução sequencial é ideal.

A comparação de desempenho entre o sistema baseado em RETE da Microsoft, o Microsoft BizTalk Business Rules Engine (MS BRE), e o WF Rules demonstra claramente que o WF Rules apresenta um desempenho linear aceitável, e no caso de ser possível aproveitar a condição “ELSE” das regras e não usar encadeamento, o desempenho pode ser até ligeiramente superior ao MS BRE. [40]

Após este estudo, restou ainda compreender se o WF Rules apresentava todas as funcionalidades internas em termos de API, necessárias para o SADAC. Para o efeito, foi construído um protótipo funcional com um contexto de execução muito semelhante ao do SADAC, e com algumas regras de negócio do Plano Nacional de Vacinação. Segue-se a imagem da aplicação desenvolvida de raiz:

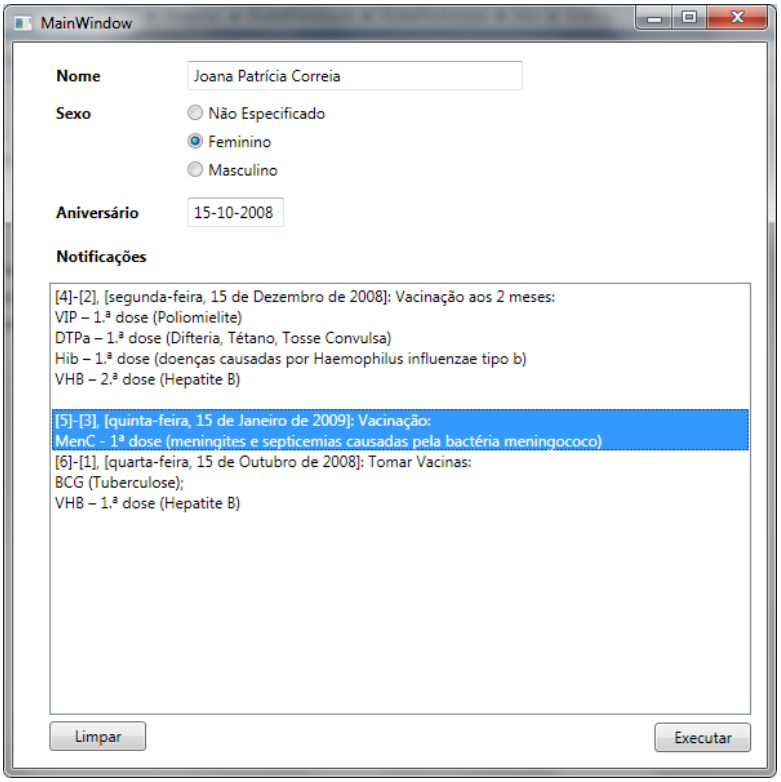

Ilustração 10 – Protótipo de Execução do WF Rules

As regras foram introduzidas utilizando um exemplo da Microsoft alterado para permitir a estrutura de dados do SADAC. Segue-se a aplicação utilizada para a introdução das regras no protótipo:

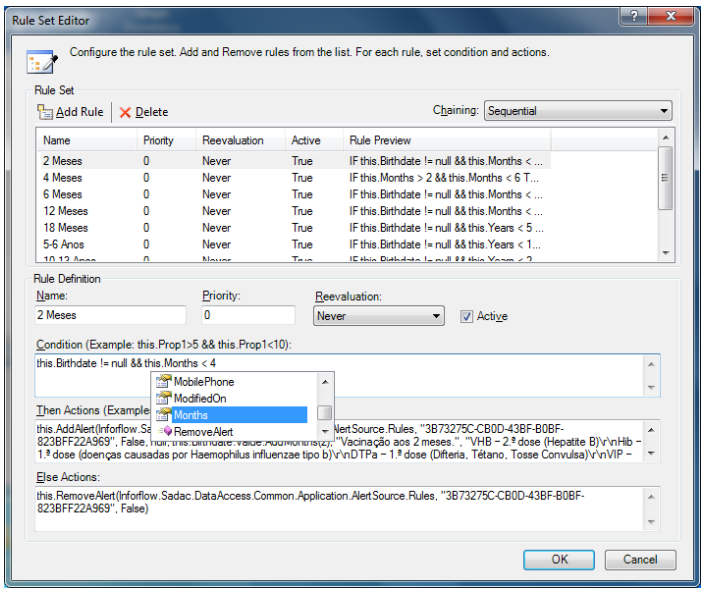

Ilustração 11 – RuleSet Toolkit Modificado

A elaboração do protótipo foi um sucesso, mas como foi utilizada parte de uma aplicação da Microsoft para introduzir as regras, foi necessário perceber como

construir uma aplicação de raiz para a introdução de regras no formato “IF-THEN-ELSE”.

A ferramenta produzida pela Microsoft tirava partido de classes, métodos, estruturas e interfaces *internal*, ou seja, acessíveis apenas dentro do mesmo *assembly*. Após algumas pesquisas na Internet foi possível confirmar que a Microsoft não tinha uma API de negócio para a compilação das regras, isto é, para a tradução das regras do formato de texto “IF-THEN-ELSE” para o formato binário (modelo de dados), sem a ajuda da janela de edição de regras anteriormente apresentada e incorporada dentro da plataforma .NET. [41, 42]

Esta preocupante limitação levou à necessidade de descompilação do *assembly* que possuía a janela de edição de regras (*System.Workflow.Activities.Rules.Design.RuleSetDialog*), através de uma versão de demonstração da ferramenta **JetBrains ReSharper 6.1**. A análise do código da janela de edição de regras revelou duas classes *internal* denominadas **Parser** e **DesignerHelpers**. Como as classes são *internal*, não seria possível saber da sua existência senão através da descompilação do *assembly*, e não é possível aceder às mesmas senão através de processos avançados de reflexão de código.

Foram implementadas duas classes que através do *design pattern* ***adapter*** (muitas vezes chamado por ***wrapper***), acedem às respectivas classes *internal* através de reflexão de código e expõem alguns dos métodos necessários. Segue-se um resumo do *adapter* do Parser:

```
internal static class Parser
{
    #region Private Constants
    private const string TypeName = "System.Workflow.Activities.Rules.Parser";
    private static readonly Type ParserType;
    private static readonly ConstructorInfo ParserConstructorInfo;
    #endregion

    #region Static Constructor
    static Parser()
    {
        Assembly ruleAssembly = typeof(Rule).Assembly;
        ParserType = ruleAssembly.GetType(TypeName);
        if (ParserType != null)
        {
            ParserConstructorInfo = ParserType.GetConstructor(BindingFlags.NonPublic | BindingFlags.Instance, null, new[] { typeof(R
        }
    }
    #endregion

    #region Public Methods
    public static RuleExpressionCondition ParseCondition(string expression, RuleValidation ruleValidation)...
    public static List<RuleAction> ParseStatementList(string expression, RuleValidation ruleValidation)...
    public static List<RuleAction> ParseSingleStatement(string expression, RuleValidation ruleValidation)...
    #endregion

    #region Private Methods
    private static object ExecuteMethod(string name, object[] ctorParameters, object[] methodParameters)...
    #endregion
}
```

Ilustração 12 – Resumo do Adapter da Classe Parser

Estes dois *adapters* permitiram compilar as regras do formato de texto "IF-THEN-ELSE" para o formato binário. O formato binário permite a execução e manipulação de regras e permite, através de um método de serialização, a persistência das regras em formato XML na Base de Dados.

É importante ter em consideração que o acesso a classes e métodos não públicos não é recomendado porque pode quebrar o negócio inerente ao sistema e as interfaces podem ser alteradas numa actualização, sem aviso prévio, inutilizando os *adapters*. Felizmente a plataforma Microsoft .NET suporta todos os ambientes de execução anteriores, isto é, a versão 4.0 da Microsoft .NET suporta nativamente os ambientes de execução 2.0, 3.0, 3.5 e 4.0. A aplicação foi compilada para a versão 4.0, o que significa que mesmo que sejam lançadas e instaladas versões superiores, ela irá correr sempre no contexto do Microsoft .NET 4.0 para o qual foi compilada.

A versão da plataforma Microsoft .NET pode ser alterada para o SADAC sem ser necessário actualizar a versão da plataforma do módulo de RBR. No entanto, caso seja desejado recompilar o módulo de RBR para uma nova versão da plataforma, o *adapter* poderá ter que ser modificado. O *adapter* utiliza apenas cinco métodos *internal*, tornando difícil a ocorrência de alterações que impliquem esses métodos, tendo ainda em consideração que os métodos *internal* são públicos a nível do *assembly*, ou seja, ao contrário dos métodos *private*, é assumido um compromisso de API em termos de pacote (*package*).

4.2. Entidades Dinâmicas

O módulo de Entidades Dinâmicas foi construído de raiz e com o objectivo de substituir a Microsoft Dynamics CRM no que diz respeito a dados dinâmicos.

Este módulo permite a configuração ao pormenor da sua estrutura de dados, permitindo criar novos registos através de formulários personalizados e de listas com vistas configuráveis, em *runtime*. Como referido anteriormente, é um componente muito semelhante com os componentes utilizados pelo Microsoft SharePoint e pela Microsoft Dynamics CRM. Foi realizado um estudo para perceber como estes produtos guardavam a definição das entidades e dos campos, e de que forma a estrutura da base de dados se adaptava às alterações.

O módulo é composto por dois componentes, o componente ***DynamicEntity.Core*** e o componente ***DynamicEntity.Editor***. O *Core*, tal como o nome indica, compreende toda a lógica de negócio e acesso a dados. O *Editor*

consiste num componente visual para Microsoft ASP .NET de edição de entidades dinâmicas e toda a lógica que isso envolve.

O seguinte modelo de dados do componente *Core* do módulo de Entidades Dinâmicas apresenta a estrutura das principais funcionalidades.

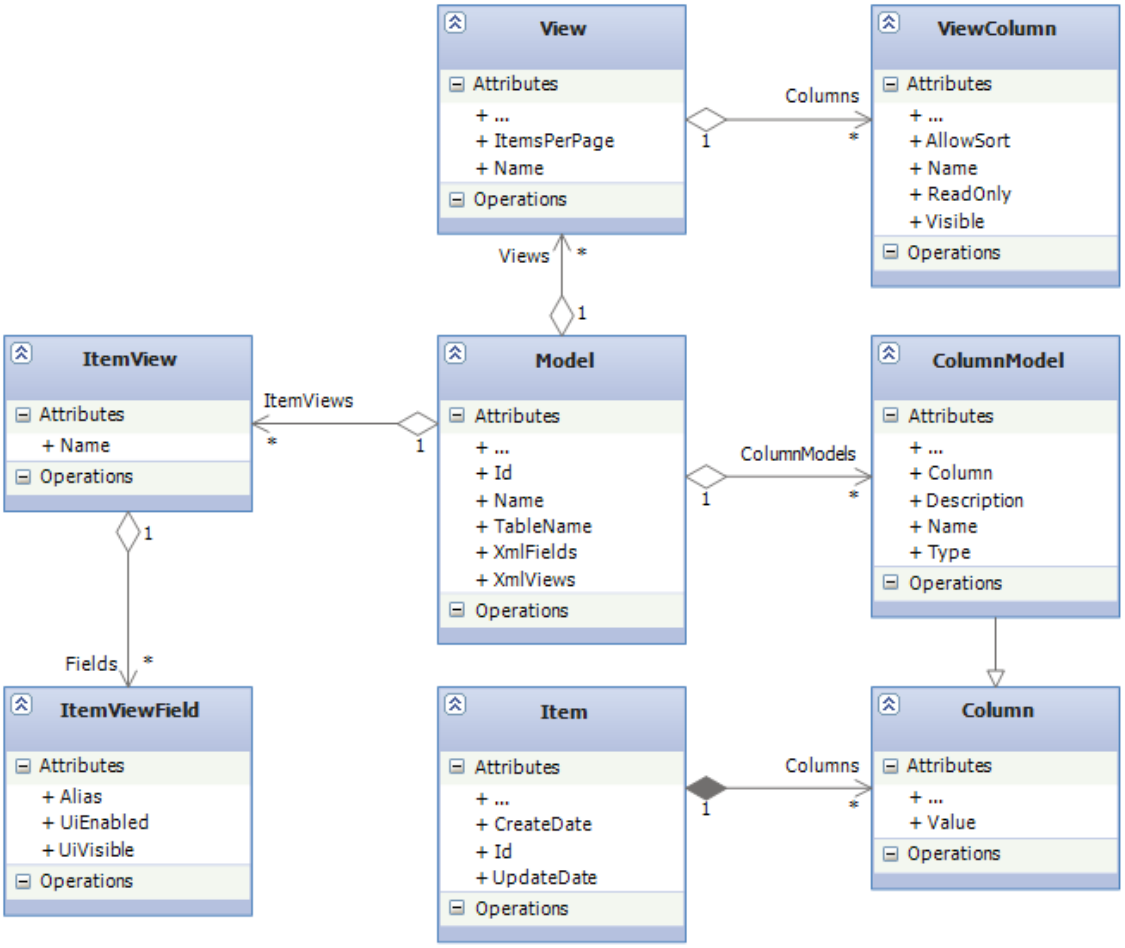

Ilustração 13 – Modelo de Dados da Dynamic Entity (*Core*)

A classe “Model” concentra a maioria das funcionalidades – consiste numa lista de vistas (*Views*) utilizadas para configurar as listas de entidades apresentadas ao utilizador, uma lista de modelos de colunas (*ColumnModels*) utilizada para descrever as propriedades que a entidade possui e a configuração do formulário, e uma lista de vistas do registo (*ItemViews*) utilizada para ocultar ou desactivar determinados campos ao utilizador nos formulários.

A classe “Item” corresponde a um registo carregado da base de dados. É possível obter o seu “Id” e as suas colunas com os respectivos valores já associados.

Para exemplificar, começa-se por pedir para carregar determinado modelo, por exemplo, com o nome “Substâncias”. Este modelo permite listar todas as entidades

associadas através da propriedade *Views*. Podemos imaginar existirem duas vistas disponíveis:

- Uma que lista as substâncias apresentando apenas o código da substância e o nome;
- Outra mais completa, que apresenta o código da substância, o nome e a descrição.

Na altura de listar as substâncias, pode ser pré-seleccionada a vista desejada ou pode ser dada a oportunidade ao utilizador de alterar a vista. A seguinte imagem ilustra a primeira vista anteriormente especificada:

Ilustração 14 – Exemplo de Lista de Substâncias

No caso dos formulários passa-se algo muito semelhante. Obtém-se o modelo "Substâncias" e pede-se para o componente "DynamicEntity.Editor" apresentar o formulário de criação para o respectivo modelo. Na página ASP o componente desenha dinamicamente os campos definidos para a criação da entidade. Da mesma forma que se pede para apresentar o formulário de criação, pode-se pedir para apresentar o formulário de edição ou de visualização, sendo para isso necessário também definir o "Id" do registo a editar/visualizar. A seguinte imagem apresenta um exemplo de um formulário de edição de uma substância:

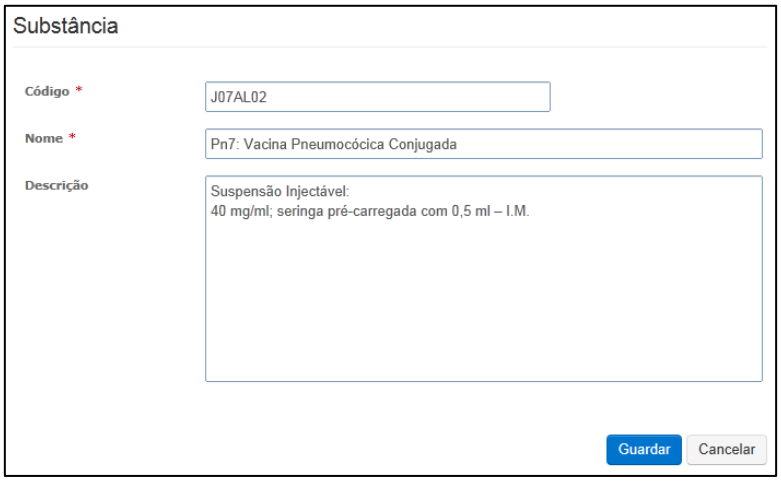

Ilustração 15 – Formulário de Edição de uma Substância

Como nem sempre é desejado que todos os perfis de utilizadores tenham acesso a todos os campos da entidade, as vistas sobre os campos permitem configurar que campos devem ser ocultados ou desactivados quando determinada vista é seleccionada. Não é obrigatório serem configuradas vistas sobre os campos.

Para além dos comuns campos de texto, o módulo de entidades dinâmicas é capaz de construir formulários com a capacidade de introduzir imagens/documentos, anexos, datas, datas e horas, caixas de selecção múltipla, caixas de selecção única, *DropDownLists* com valores pré-definidos, *DropDownLists* com valores referentes a outras entidades (por exemplo, um *DropDownList* com os utilizadores do sistema) e caixas de texto com suporte de formatação avançada. Podem posteriormente ser implementados mais componentes de edição se for necessário.

A edição da definição das entidades, das vistas e das vistas dos formulários é realizada por intermédio de XML. Segue-se um exemplo do XML que define uma entidade denominada “Substância”.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<DynamicEntity TableName="Substance" Versioned="false"
               AllowAttachments="false" Shared="true">
  <Fields>
    <Field Alias="Code" Name="Código" Type="String" Column="Code"
           Mandatory="true" Max="20" uiType="TextBox" uiWidth="60%"/>
    <Field Alias="Name" Name="Nome" Type="String" Column="Name"
           Mandatory="true" Max="100" uiType="TextBox" uiWidth="98%"/>
    <Field Alias="Description" Name="Descrição" Type="String"
           Column="Description" Mandatory="false" uiType="TextBox"
           MultiLine="true" Max="500" uiRows="12" uiWidth="98%"/>
  </Fields>
</DynamicEntity>
```

Uma vez que a criação e alteração das definições serão realizadas por um utilizador, torna-se importante validar o XML para garantir a sua integridade. A

validação do XML dá-se por intermédio de dois ficheiros XSD (um para cada tipo de XML), que valida toda a integridade do XML, tanto em termos de estrutura de elementos, como de propriedades e valores das propriedades. Os ficheiros XSD podem ser utilizador por outras aplicações para facilitar a edição e validar o XML.

Uma vez que o módulo foi produzido de forma independente da aplicação SADAC, foi implementada uma aplicação para ajudar no desenvolvimento do módulo, que permitisse listar todos os modelos existentes, as entradas das tabelas, introduzir e editar modelos existentes e introduzir ou editar registos existente. Esta ferramenta foi denominada por ***DynamicEntity Development Tool*** (*DynamicEntity.Developer*). Segue-se um *print-screen* da aplicação ligada à base de dados do SADAC em modo de edição da entidade "Substância".

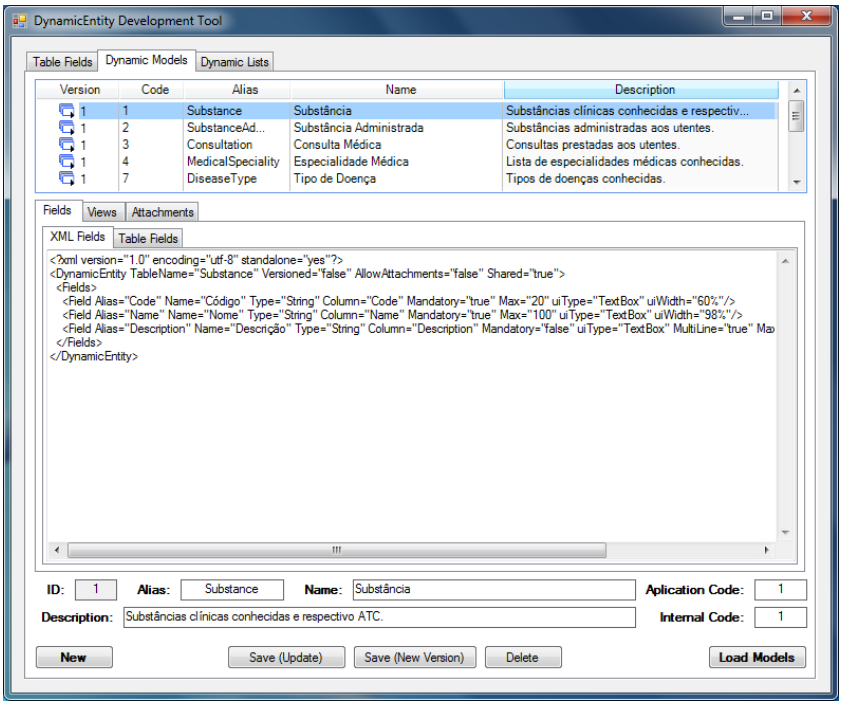

Ilustração 16 – DynamicEntity Development Tool

Esta ferramenta *standalone* consegue configurar as entidades dinâmicas muito mais rapidamente que uma ferramenta *web*, por isso foi utilizada com grande frequência no desenvolvimento da Dynamic Entity e para *debugging* de problemas com as entidades dinâmicas do SADAC.

A ferramenta permite também detectar se a estrutura da base de dados se encontra consistente com os modelos de dados dinâmicos. Em termos de estrutura da base de dados, são utilizadas as seguintes tabelas base:

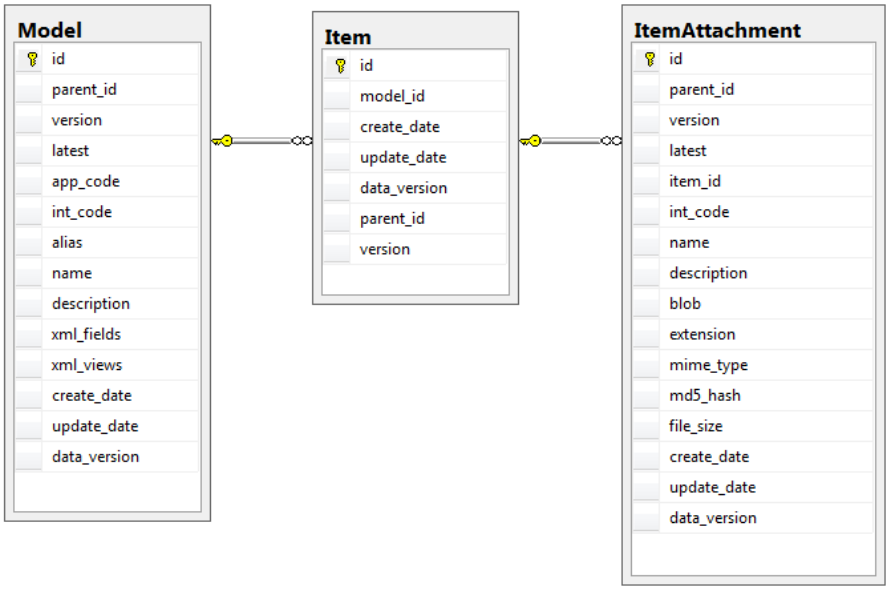

Ilustração 17 – Modelo Físico de Base de Dados da Dynamic Entity

A tabela “Model” contém todas as definições dos modelos no formato XML, nos campos “xml_fields” e “xml_views”. A tabela “Item” é a tabela predefinida para conter os registos da concretização do modelo, isto é, as entidades definidas no modelo. A tabela “ItemAttachment” contém os ficheiros anexados aos registos da tabela “Item”.

O modelo físico de base de dados acima descrito não está completo. Os registos podem ser guardados em tabelas com nomes especificados pelo modelo, sem ser na tabela “Item”. O modelo especifica também se deve ou não ser criada uma tabela de anexos para os registos ou se os registos não vão suportar anexos. Esta estratégia é a mesma que a utilizada pelo Microsoft SharePoint e Microsoft Dynamics CRM. De forma semelhante a estes produtos, a Dynamic Entity constrói o SQL necessário para criar ou alterar as tabelas existentes na Base de Dados de acordo com as definições dos modelos.

O processo de actualização da estrutura da base de dados é automático e acontece ao persistir um modelo. O processo utiliza a *design pattern Transfer Data Object* para construir um objecto que contempla todas as alterações necessárias realizar à Base de Dados. Foi utilizada a *design pattern Row Data Gateway*, permitindo um encapsulamento dos registos da base dados em objectos que se gerem a si próprios na base de dados.

Consistência de Dados

A Dynamic Entity utiliza transacções para realizar operações mais complexas (*design pattern Transaction Script*) e utiliza ainda a coluna “data_version” para

concretizar a *design pattern Optimistic Offline Lock*. Como torna possível chamar os métodos de manipulação da base de dados (*Save* / *Delete*) com uma transacção aberta e definida, concretiza a *design pattern Pessimistic Offline Lock*, cobrindo todas as possibilidades de falha de consistência de dados. As operações com a base de dados são atómicas (*Insert*, *Update*, *Delete*), sendo apenas necessárias transacções para operações mais complexas da parte do programador, por exemplo, inserir vários registos numa tabela, ou caso algum falhe, não inserir nenhum.

Segurança

O módulo de entidades dinâmicas está construído sobre Microsoft SQL Server, permitindo a utilizar todos os procedimentos de segurança inerentes a esse produto. O Microsoft SQL Server permite a encriptação de determinadas colunas, de determinadas tabelas ou de toda a base de dados. Permite também realizar cópias de segurança dos dados de forma periódica para o mesmo servidor ou para outros.

O protocolo de comunicação com a Base de Dados é o SQL, protoloco que o vasto leque de aplicações utiliza para comunicar com Bases de Dados. É um protocolo seguro e que implementa as devidas protecções contra ataques de *eavesdropping*, *man-in-the-middle*, repudiação de dados, *denial of service* (DoS), session hijacking, entre outros. O protocolo foi desenhado para ser seguro.

Desempenho

O módulo de entidades dinâmicas converte todas as operações em puro SQL, e permite que os dados sejam paginados ao nível da base de dados, sem necessidade de serem carregados para o servidor aplicacional. O módulo permite também a utilização de todos os mecanismos de escalabilidade do servidor de SQL, por exemplo, a utilização de NLB (*Network Load Balancing*) nos servidores de base de dados. Por esse motivo, prevê-se um desempenho muito semelhante à utilização de outro ORM (*Object-Relational Mapping*) sobre SQL Server, por exemplo, a Entity Framework. No entanto, deverão ser realizados testes de desempenho neste sentido.

Foram feitos testes exaustivos de desempenho com o objectivo de identificar *bottlenecks* de desempenho e optimizar ao máximo os *bottlenecks*. Para o efeito foram utilizadas as excelente ferramentas de ***Profiling*** do Microsoft Visual Studio. Esta ferramenta identifica automaticamente os *hotspots* (ou *bottlenecks*) da aplicação e permite comparar o desempenho de várias abordagens para o mesmo problema. Permite ainda monitorizar não só o desempenho em termos de tempo de processador, mas também em termos de operações I/O, instrumentação (número de chamadas) e concorrência.

Foi realizado um teste de desempenho (tempo de processador) ao carregar 1000 entidades dinâmicas, 10 campos cada entidade (4 de texto, 2 inteiros, 2 decimais, 2 datas), da Base de Dados. Esta operação estava a demorar cerca de **2.203,4 ± 10,31** milissegundos (cerca de 2 segundos), algo que independentemente de saber se era ou não exagerado (uma vez que se estava a carregar 1000 entidades), era importante compreender se existia algum *bottleneck* na operação, e onde se localizava. Ao correr a ferramenta de *profiling* foram detectados dois *bottlenecks* dentro de um método, conforme a próxima imagem apresenta.

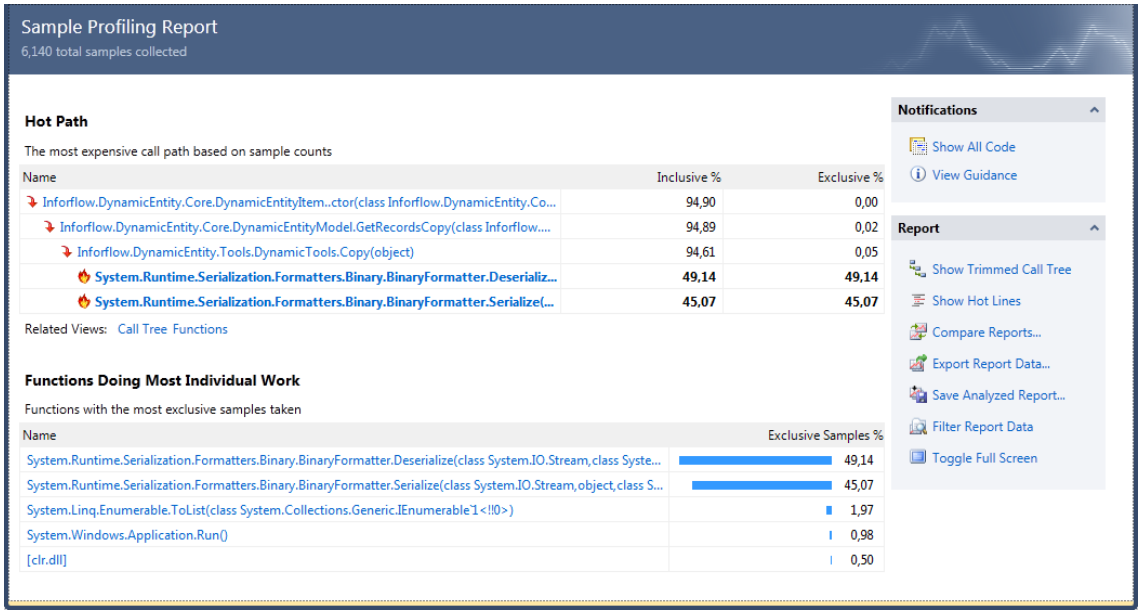

Ilustração 18 – Profiling de Carregamento de 1000 Entidades Dinâmicas

Extraordinariamente o *profiler* detectou automaticamente um *bottleneck* no método “Copy” provocado por dois métodos de serialização que o método utilizava. Este *bottleneck* ocupava **94,61%** de todo o tempo de execução a carregar as entidades da base de dados. Após uma investigação de formas alternativas de realizar a operação sem utilizar técnicas de serialização, foi encontrada uma solução alternativa utilizando a metodologia de ***Shallow Copy*** (cópia superficial), que conseguia o mesmo efeito sem serialização. Foram alteradas apenas **seis linhas de código** (a que chamava o método “Copy” e o método “ShallowCopy”) e voltou-se a realizar o *profiling*.

Surpreendentemente o *profiler* não detectou qualquer *bottleneck* de tempo de processador. Em termos de tempo de execução passou para **15,5 ± 1,84** milissegundos, uma extraordinária melhoria de desempenho (*SpeedUp* [43]) de **14.215,48%**. Se não fosse realizado este processo de *profiling* este *bottleneck* iria muito provavelmente passar despercebido uma vez que não parece muito significativo

demorar 2,2 segundos a carregar 1000 entidades dinâmicas da base de dados (sem paginação), mas o impacto de performance em ambiente real seria com certeza notório e seria afectado pela concorrência de pedidos.

Foram realizadas outras optimizações menos significativas e por essa razão não mereceram destaque nesta secção. De uma forma geral o desempenho do módulo foi levado a sério em todo o processo de implementação e a plataforma Microsoft .NET 4.0 foi explorada (ferramentas e estruturas de dados) com o objectivo de optimizar o desempenho.

4.3. Arquitectura de *Plugins* do Portal do Utente

O Portal do Utente, por si só, não apresenta muitos desafios a nível de apresentação, nem a nível de negócio. No entanto, a nível de arquitectura foi necessário implementar uma arquitectura com as três camadas bem definidas e com a capacidade da camada de acesso a dados ser concretizada por *plugins*. A arquitectura de *plugins* foi desenhada e realizada de raiz e com o objectivo de permitir integrar com sistemas externos, por exemplo, com a Microsoft Dynamics CRM, através de *plugins* específicos para essa finalidade. Da mesma forma que é possível desenvolver *plugins* específicos para a Dynamics CRM é também possível desenvolver *plugins* para Microsoft SQL Server para persistir os dados com estrutura estática e utilizar o módulo de entidades dinâmicas para configurar e persistir os dados com estrutura dinâmica.

O Microsoft C# 4.0 permite através da reflexão de código e por intermédio de Interfaces comuns a comunicação com *plugins*. Foi necessário ter em consideração que o contexto de execução é ASP .NET e não WinForms, sendo mais complexo lidar com *plugins*. Foi também necessário considerar que é possível que os *plugins* tenham dependências externas necessárias para a sua execução, como é o caso dos *plugins* de acesso a dados da Microsoft Dynamics CRM.

A arquitectura de *plugins* foi implementada através da implementação de um atributo “*DataProviderPluginAttribute*” que permite especificar o nome e descrição do *plugin* ao marcar uma classe derivada da classe abstracta do *DataProvider* desejado. O exemplo seguinte demonstra a marcação da classe *SqlDataProvider* como *plugin* de acesso a dados para o motor de regras.

```
[RulesDataProviderPlugin("Microsoft Sql Server", "Microsoft Sql Server Data Access PlugIn for Rules Engine.")]
public class SqlDataProvider : RulesEngineDataProvider
{
    Private Variables

    Public Properties

    Public Overridden Properties

    Constructors

    Public Overridden Methods

    Override IDisposable
}
```

Ilustração 19 – Exemplo de Plugin de Acesso a Dados

Uma DLL que contenha uma classe como esta e seja colocado no directório "*Plugins*" do Portal de Utente será automaticamente reconhecida como *plugin*. A configuração de qual *plugin* de acesso a dados utilizar acontece no ficheiro *web.config*, como apresentado a seguir:

```
<sadac>
  <plugins>
    <rulesEngineDataAccess>
      <rulesProviders defaultName="SqlDataProvider">
        <add name="SqlDataProvider"
             type="Bond.RulesEngine.DataAccess.Sql.SqlDataProvider"
             connectionStringName="SadacDataSql" />
        <add name="XrmDataProvider"
             type="Bond.RulesEngine.DataAccess.Xrm.XrmDataProvider"
             connectionStringName="XrmData" />
      </rulesProviders>
    </rulesEngineDataAccess>
(…)
    <applicationDataAccess>
(…)
    </applicationDataAccess>
  </plugins>
</sadac>
```

A configuração permite especificar os vários *plugins* disponíveis e qual a *ConnectionString* (texto que contém os parâmetros de ligação) que o *plugin* deve usar. O acesso a dados dar-se-á sempre através do *plugin* seleccionado pelo parâmetro "defaultName", a não ser que seja especificado um *plugin* específico por código.

Para a pesquisa e carregamento de *plugins* para dentro da aplicação foram necessárias técnicas que envolvem reflexão de código, classes abstractas e atributos personalizados.

4.4. *Routing* do Portal do Utente

O Portal do Utente foi construído utilizando as avançadas capacidades de *Routing* do lado do servidor. [44] O *routing* permite optimizar a indexação da aplicação

nos motores de procura (SEO – *Search-Engine Optimization*), uma vez que o URL é um dos principais campos a considerar na indexação e permite construir URLs mais pequenos e que tenham mais significado para o utilizador, permitindo até ao utilizador utilizar o URL como se fosse mais uma forma de interacção *expert* com a aplicação. [45] O *routing* é, portanto, uma mais-valia tanto em termos de usabilidade como de indexação e organização de conteúdos em motores de pesquisa.

A utilização de *routing* permite converter endereços do género:

```
www.sadac.pt/ShowProductsByCategory.aspx?CategoryID=1
```

Em endereços virtuais mais intuitivos:

```
www.sadac.pt/Categories/Beverages
```

A possibilidade de utilizar *routing* em aplicações ASP .NET sem MVC (*Model-View-Controller*) é relativamente recente, e não é uma tarefa de todo trivial. Esta capacidade apenas foi introduzida na versão Microsoft .NET 3.5 SP1, sendo até então apenas possível utilizar *routing* em projectos MVC. [46]

Uma aplicação ASP .NET comum utiliza um *SiteMap* construído através do *provider* nativo *XmlSiteMapProvider* que permite organizar as várias páginas existentes e atribuir-lhes um título, descrição e associar permissões através das *roles* do utilizador autenticado (papeis). É então possível ligar o *SiteMap* a um componente *Menu*, e o *Menu* torna-se capaz de perceber automaticamente em que página se está e em que parte da hierarquia de páginas a página se encontra (*breadcrumbs*), permitindo ainda esconder as entradas a que o utilizador não possui acesso.

A utilização de *routing* acontece desligada de todo este processo natural. O *SiteMap* deixa de ser capaz de mapear páginas, uma vez que o *routing* mapeia páginas físicas em endereços virtuais e o *SiteMap* apenas sabe lidar com páginas físicas. Deixa de ser portanto possível configurar através do *SiteMap* as permissões de acesso e de ligar o *SiteMap* ao *Menu* da aplicação. Outro problema é que a configuração do mapeamento de endereços virtuais para páginas físicas (*routing*) se encontra num ficheiro à parte do *SiteMap*, sendo necessário duplicar esta configuração.

Para ultrapassar estes obstáculos foi necessário derivar um *provider* novo que conseguisse interpretar os novos endereços virtuais com *routing* do ficheiro de configuração do *SiteMap* e mapeasse esses endereços em páginas físicas e configurasse de forma central o *SiteMap* e o *routing*.

A forma utilizada para o *provider* conseguir enganar o *SiteMap* fazendo-o crer que o URL utilizado era de uma página física e não um endereço virtual foi através de reflexão de código. Isso permitiu ao *SiteMap* utilizar o endereço físico da página para os efeitos de verificação de permissões ao mesmo tempo que utiliza o endereço virtual para redireccionar o utilizar e perceber em que ponto da aplicação o utilizador se encontra.

Esta técnica não é completamente original [47, 48], mas apenas foram encontradas duas tentativas incompletas de o alcançar e pouco documentadas. As técnicas descritas pelos autores encontrados não conservavam o contexto de segurança do *SiteMap* nem permitiam centralizar a configuração completa do *routing* através do ficheiro de configuração do *SiteMap*.

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns=http://schemas.microsoft.com/AspNet/SiteMap-File-1.0
    enableLocalization="true">
  <siteMapNode title="SADAC Home Page" id="home" url="home"
      physicalFile="~/Pages/FrontOffice/Home.aspx">
    <siteMapNode title="Mensagens"
        physicalFile="~/Pages/FrontOffice/Messages/Recent.aspx">
      <siteMapNode title="Recentes" id="messages-recent" url="messages/recent"
          physicalFile="~/Pages/FrontOffice/Messages/Recent.aspx" />
      <siteMapNode title="Histórico" id="messages-log" url="messages/log"
          physicalFile="~/Pages/FrontOffice/Messages/History.aspx" />
      <siteMapNode title="Nova" id="messages-new" url="messages/new" visible="false"
          physicalFile="~/Pages/FrontOffice/Messages/Edit.aspx" />
      <siteMapNode title="Visualizar" id="messages-edit" url="messages/view/{id}"
          physicalFile="~/Pages/FrontOffice/Messages/Edit.aspx" visible="false"
          constraints="{'id', '(\d{1,9})|([\da-fA-F]{8}-[\da-fA-F]{4}-[\da-fA-F]{4}-
[\da-fA-F]{4}-[\da-fA-F]{12})'}" />
    </siteMapNode>
  </siteMapNode>
</siteMap>
```

O excerto acima descrito da configuração do *SiteMap* exemplifica como foi possível configurar o *routing* e organizar a estrutura de páginas do SADAC num ficheiro conjunto.

5. Levantamento e Análise de Requisitos

O levantamento de requisitos foi elaborado segundo um conjunto de Técnicas Facilitadas de Especificação de Aplicações (*Facilitated Application Specification Techniques* – FAST) [49]. Estas técnicas visam contrair o afastamento entre a equipa de desenvolvimento e o cliente, reduzindo as ambiguidades na informação, o volume de documentação, evitar sessões formais de pergunta/resposta e melhorar o ambiente de trabalho entre as partes envolvidas. Segundo a RUP, o objectivo não é fechar todos os requisitos, apenas os necessários para formular uma opinião, uma vez que o processo de desenvolvimento é iterativo [50].

Os requisitos foram classificados segundo a especificação de Sommerville descrita no seu livro "Software Engineering" [51]. Segundo este autor, os Requisitos Funcionais descrevem o que o sistema deve disponibilizar/assegurar e como se deve comportar em determinadas condições. Os Requisitos Não Funcionais são restrições aos serviços oferecidos pelo sistema.

A priorização dos requisitos foi elaborada segundo a metodologia MoSCoW [52]. Esta metodologia utiliza quatro termos que classificam a prioridade do requisito: MUST (M), SHOULD (S), COULD (C) e WON'T (W).

A especificação de cada requisito está elaborada com recurso a *User Stories* [53]. Esta estratégia é muito comum em metodologias de desenvolvimento ágil uma vez que incita à comunicação face-a-face entre o cliente e o gestor ou programador implicando a discussão de ideias [54].

Esta secção apresenta apenas os principais requisitos e *mockups* do SADAC. O anexo "*Levantamento e Análise de Requisitos*" detalha a fundamentação e contém a lista completa de requisitos identificados e de protótipos de baixa-fidelidade (*mockups*) elaborados.

5.1. Requisitos

Segue-se um requisito funcional do Portal do Utente:

Código	WEB-RF-02	**Prioridade**	MUST
Título	Efectuar Pedidos de Marcação de Consultas		
Enquanto	Utente		
Quero	Efectuar um novo pedido de marcação de consulta		
Para que	Não tenha que me dirigir à unidade hospitalar para marcar a consulta; Não tenha que estar dependente de um serviço com um horário específico; Evite aguardar a disponibilidade de um serviço telefónico; Evite gastar dinheiro a ligar para um assistente		
Critério de Aceitação			
Dado que	O utente encontra-se autenticado		
Quando	Acede ao menu “Consultas”; Prime o botão para realizar um novo pedido de marcação; Preenche correctamente o formulário e submete-o;		
Então	O pedido de marcação é inserido no sistema com sucesso		

Os requisitos são essencialmente compostos por um código, uma prioridade, um título, uma descrição e um critério de aceitação. O critério de aceitação deve validar se o requisito foi ou não correctamente implementado.

Segue-se um requisito não funcional, de produto, do módulo de Entidades Dinâmicas:

Código	DYN-NFP-02	**Prioridade**	MUST
Título	Desempenho dos Formulários		
Enquanto	Cliente do Produto		
Quero	Que o tempo de construção dos formulários dinâmicos seja aceitável quando comparado com a construção estática		
Para que	Seja viável a sua utilização		
Critério de Aceitação			
Dado que	Um formulário com 10 propriedades diversas		
Quando	For pedida a sua construção		
Então	O tempo de construção completa deve ser inferior a 3 segundos [55]		

Mesmo para requisitos não funcionais, o critério de aceitação continua a conseguir validar se o requisito foi ou não cumprido.

5.2. Protótipos de Baixa-Fidelidade

A elaboração do protótipo de baixa-fidelidade (*mockups*) teve em consideração várias recomendações de *design* para Web Sites, numa perspectiva de melhorar significativamente a experiência do utilizador com a aplicação. Segue-se o resumo das principais recomendações consideradas. O anexo "*Levantamento e Análise de Requisitos*" contém a descrição completa.

- Tipo de menu escolhido. [56, p. 226]
- Uso de botões embebidos em conteúdo dinâmico. [56, p. 234]
- Ordenação por omissão nas listagens. [56, p. 245]
- Dinâmica dos menus. [56, p. 246]
- Tipos de letra utilizados. [56, p. 246]
- Páginas com objectivos simples e específicos. [56, pp. 407-416]
- Campos com etiquetas e ícones de botões com *tooltips*. [56, p. 479]
- Utilização de linguagem acessível e familiar. [56, p. 253]
- Possibilidade de reordenar, procurar, ou filtrar grandes quantidades de dados. [56, p. 233]
- Formatações dos campos. [56, p. 256]
- Justificação do texto à esquerda. [56, pp. 445-446]

Os seguintes *mockups* descrevem o desenho idealizado para a primeira página e para a página dos pedidos de marcação de consultas do Portal do Utente:

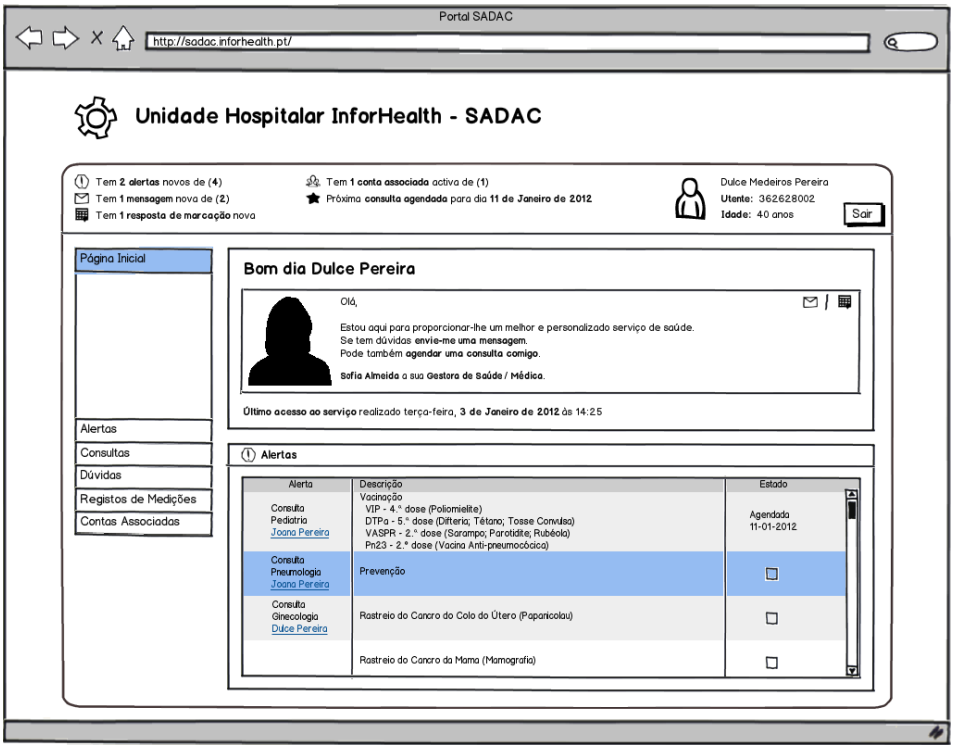

Ilustração 20 – Mockup da Página Principal do Portal do Utente

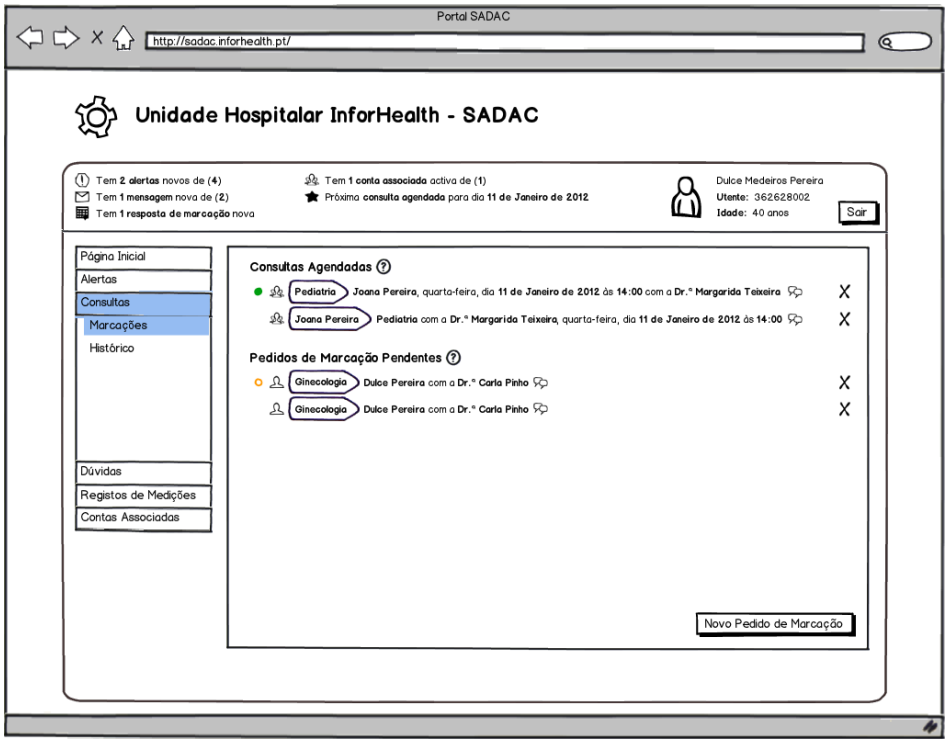

Ilustração 21 – Mockup dos Pedidos de Marcação de Consultas

6. Arquitectura

Este capítulo descreve a arquitectura do sistema a desenvolver segundo os dois primeiros níveis de abstracção – interacção entre os sistemas externos e interacção interna entre os módulos. Apresenta também o modelo de dados do módulo de Raciocínio Baseado em Regras (RBR) e do Portal do Utente.

A primeira versão da aplicação consiste no Portal do Utente, um módulo de RBR e um módulo de Entidades Dinâmicas. O portal disponibilizará uma Interface com o Utilizador para permitir ao utente aceder à sua informação e aos alertas através de um explorador de Internet. Permitirá também aos profissionais e auxiliares de saúde configurar o módulo de RBR e o módulo de entidades dinâmicas.

O cliente dispõe de uma aplicação de gestão de dados clínicos, criada sobre a plataforma Microsoft Dynamics CRM, que contém informação clínica acerca dos utentes, nomeadamente as consultas, informação pessoal, tratamentos, diagnósticos e exames.

Existem portanto duas possibilidades de *deployment*, o SADAC pode ser instalado de forma isolada, com o seu próprio *BackOffice* e gestão dinâmica da estrutura de dados clínicos, ou partilhada com um sistema de gestão de dados clínicos, por exemplo, a Microsoft Dynamics CRM.

6.1. Visão de Nível 0 – Deployment

O objectivo desta visão é focar no papel do SADAC no envolvimento com outros agentes externos no ambiente de instalação.

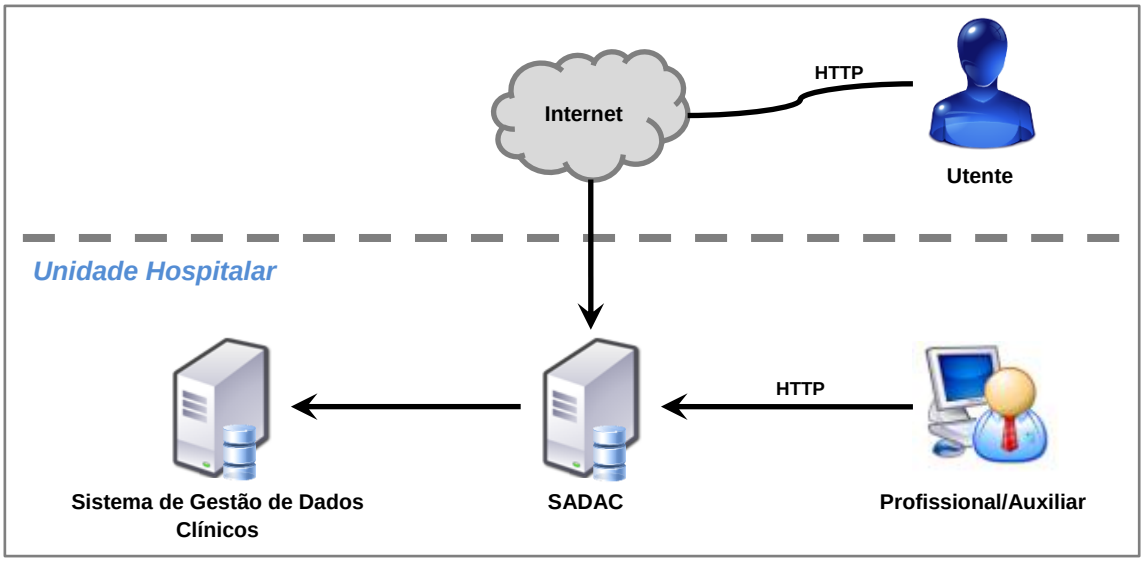

Ilustração 22 – Visão Nível 0 – Deployment

O utente poderá aceder ao Portal do Utente do SADAC através da Internet e os Profissionais de Saúde de dentro da Unidade Hospitalar. O Portal do Utente poderá comunicar com um sistema de gestão de dados clínicos com o objectivo de obter os dados clínicos necessários.

É importante compreender que, apesar de existirem duas formas de instalação da aplicação SADAC, o SADAC não está preparado para a gestão de dados clínicos, necessitando de uma plataforma de gestão de dados clínicos para obter, de alguma forma, esses dados. Como referido anteriormente, os dados podem ser partilhados pelas duas plataformas, até porque o acesso do SADAC a esses dados será **só-de-leitura**, ou pode ser através de sincronização.

O próximo esquema demonstra a estrutura interna da arquitectura de *plugins* da camada de acesso a dados do SADAC. Existem três tipos de acesso a dados, um para o RBR, outro para o Portal do Utente e outro para os Dados Clínicos. É possível aceder a cada um dos tipos de dados de diferentes formas. Na necessidade da aplicação comunicar com outro sistema de gestão de dados clínicos ou outro tipo de DBMS sem ser o Microsoft SQL Server, apenas será necessário implementar o *plugin* para acesso aos dados clínicos desse sistema.

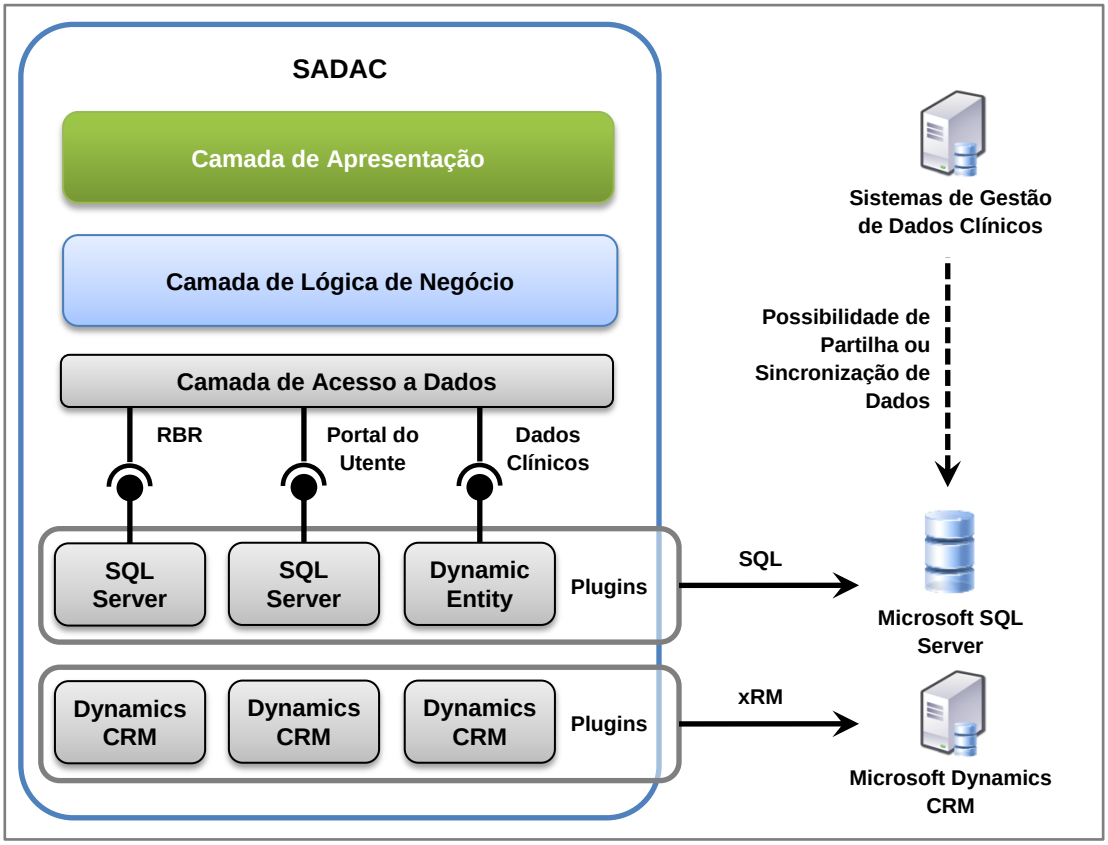

Ilustração 23 – Plugins de Acesso a Dados

O módulo Dynamic Entity pode também ser utilizado para partilhar a Base de Dados do sistema de gestão de dados clínicos, através da *design pattern Shared Database*, uma vez que o módulo permite configurar ao detalhe a estrutura de dados. Permite também que sejam utilizadas técnicas de sincronização de dados entre a estrutura de dados configurada pelo módulo Dynamic Entity e o sistema de gestão de dados clínicos, concretizando a *design pattern Maintain Data Copies*.

Continua a ser também possível, como anteriormente referido, criar um *plugin* de acesso a dados clínicos para comunicar directamente com outros sistemas de gestão de dados clínicos.

6.2. Visão de Nível 1 – Módulos

O seguinte diagrama apresenta os módulos internos da aplicação divididos pelas várias camadas: apresentação, lógica de negócio e acesso a dados.

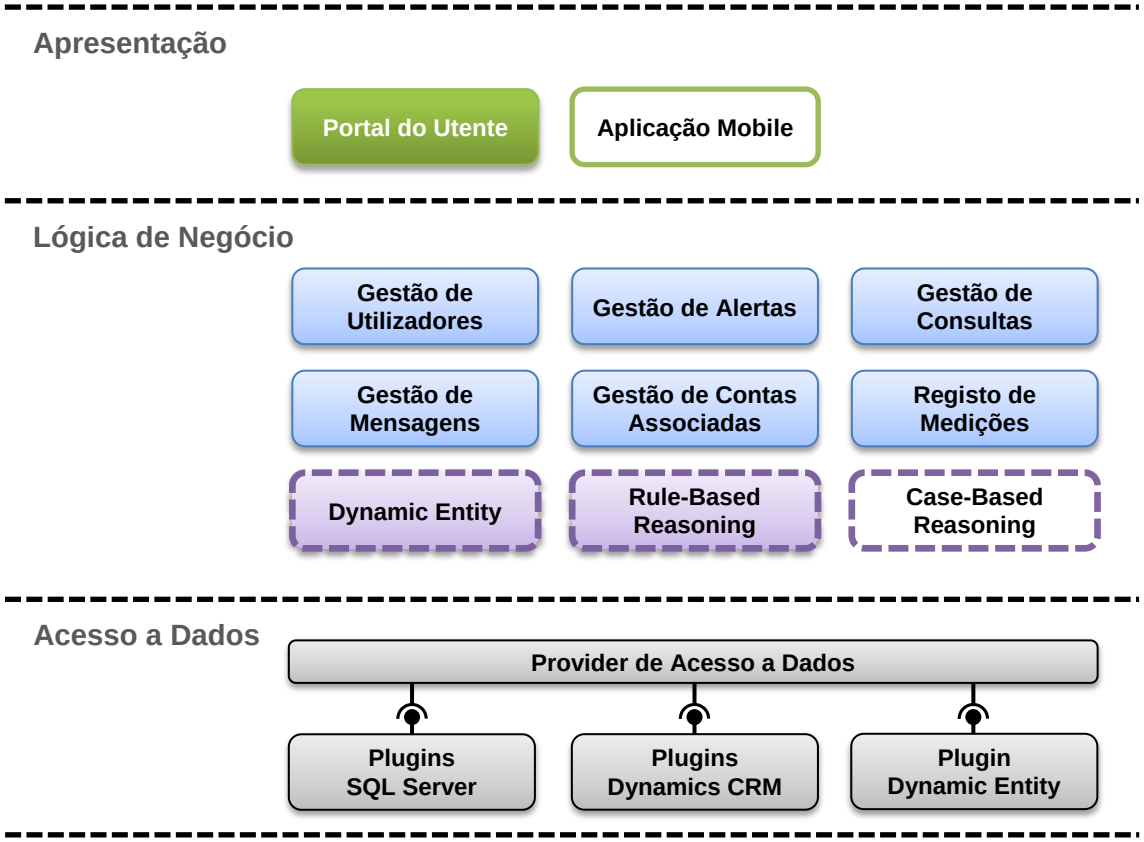

Ilustração 24 – Visão Nível 1 – Módulos

Os módulos que correspondem à primeira versão do SADAC estão marcados a **cheio**, os módulos que serão posteriormente implementados estão marcados a **branco** (vazios). A primeira versão não incluí uma aplicação *mobile* nem envolve Raciocínio Baseado em Casos (*Case-Based Reasoning* – CBR).

A camada de lógica de negócio do SADAC apresenta seis **módulos internos**, que correspondem a macro funcionalidades, e três **módulos externos** (Dynamic Entity, RBR e CBR), marcados a tracejado. O módulo Dynamic Entity integra apenas com o *Plugin* Dynamic Entity. Os outros dois módulos externos são utilizados por vários módulos de negócio e de acesso a dados do SADAC.

Os *plugins* utilizam a *design pattern Table Data Gateway* para obter os registos da base de dados, e cada registo utiliza a *design pattern Row Data Gateway* encapsulando as operações que envolvem a manipulação dos dados. A concretização destas *design patterns* é forçada por uma classe abstracta que abstrai a implementação dos métodos necessários, esta *design pattern* de comunicação denomina-se *Abstract Interface*. A comunicação entre a camada de acesso a dados e os *plugins* é realizada através da *design pattern Dependency Inversion* – ambas as camadas referenciam uma camada comum, evitando possíveis referências circulares. [57, 5] O *provider* de acesso a dados permite gerir o carregamento de *plugins* procurando pela respectiva implementação da interface nos *plugins*.

O carregamento de *plugins* é realizado por reflexão de código e utiliza a *design pattern Common Design Type* [5] para listar e carregar o *plugin* desejado. Também foi utilizada a *design pattern Data Transfer Object* uma vez que o objecto que corresponde a um registo da base de dados é passado da camada de acesso a dados, até à camada de apresentação para visualização e construção de listagens. [57]

6.3. Modelo Conceptual de Dados

A seguinte imagem apresenta o modelo conceptual de dados do módulo de Raciocínio Baseado em Regras.

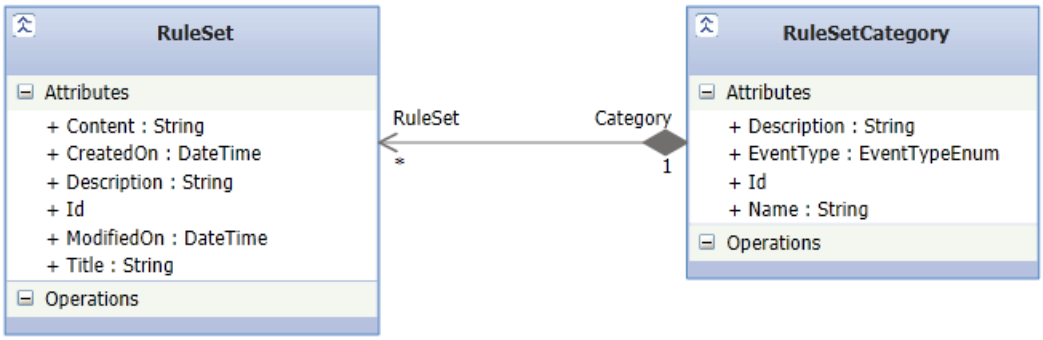

Ilustração 25 – Modelo Conceptual de Dados do RBR

O modelo conceptual do RBR foca-se na necessidade de guardar os grupos de regras (*RuleSet*) e as categorias dos grupos de regras (*RuleSetCategory*). As regras estão descritas em XML dentro da propriedade “Content” da entidade “RuleSet”.

Segue-se o modelo conceptual de dados do Portal do Utente.

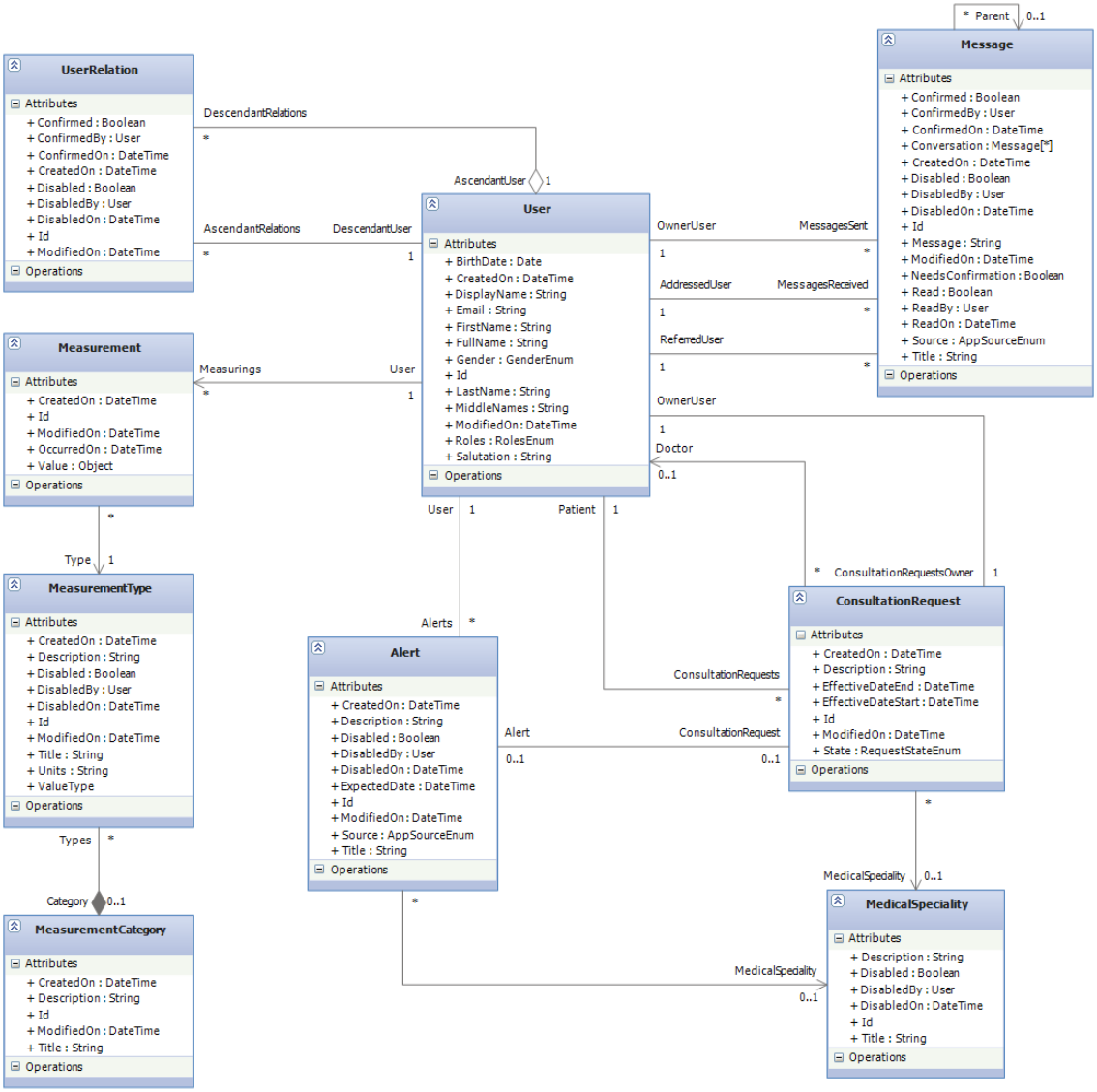

Ilustração 26 – Modelo Conceptual de Dados do Portal do Utente

É possível observar que todas as entidades dependem sempre, de alguma forma, do utilizador. O utilizador pode ser um utente, um auxiliar, um perito da unidade hospitalar, um profissional de saúde da unidade hospitalar ou um administrador.

O utente (*User*) pode consultar as suas contas associadas (*UserRelation*), as suas mensagens enviadas (*Message*), os pedidos de marcação de consultas (*ConsultationRequest*), os seus alertas (*Alert*) e as suas medições de parâmetros de saúde (*Measurement*). Estas são as principais entidades.

Os outros papéis (auxiliar, profissional de saúde e administrador) terão acesso ao *BackOffice* (se não for realizado por outra aplicação externa como a Microsoft Dynamics CRM), e terão, portanto, outro tipo de acesso mais elevado no Portal.

6.4. Segurança

O SADAC é um sistema enquadrado no sector da saúde. Contém informação pessoal, sensível e confidencial dos utentes e vai comunicar com outros sistemas de gestão de dados clínicos. A segurança torna-se portanto crucial neste contexto.

A Microsoft desenvolveu uma recomendação com as seis principais categorias de ameaças a considerar quando se desenvolve uma aplicação informática, denominada por STRIDE [58, 59], acrónimo de *Spoofing*, *Tampering*, *Repudiation*, *Information Disclosure*, *Denial of Service*, e *Elevation of Privilege*. De seguida é apresentada a metodologia utilizada para mitigar cada uma destas ameaças.

Ameaça	Estratégias de Segurança
Spoofing O atacante consegue adquirir as credenciais da vítima.	Os dados de autenticação dos utilizadores são guardados sob a forma de uma *hash*, nomeadamente a SHA-1. É ainda possível encriptar a base de dados.
Tampering Modificação e/ou destruição dos dados de uma vítima através da sua intercepção.	Utilização de protocolos de comunicação seguros, nomeadamente SSL e TLS, sobre HTTP, *WebServices* e SQL.
Repudiation Permitir a um atacante negar ter realizado uma operação, evadindo-se dessa responsabilidade e possíveis consequências.	Utilização de políticas de auditoria em todas as operações de modificação dos dados da base de dados. Todas as operações do portal do utente verificam se o utilizador está autorizado a realizar as operações antes de as realizar.
Information Disclosure Indevida exposição de informação confidencial, à qual o utilizador não deveria ter acesso.	Utilização das estratégias referidas nas categorias de *Spoofing*, *Tampering* e *Elevation of Privilege* para assegurar que o utilizador não tem acesso a dados confidenciais. A aplicação verifica se o utilizador tem permissões para aceder a todas as informações pedidas, e está preparado para negar o acesso caso não possua.

Ameaça	Estratégias de Segurança
Denial of Service O atacante provoca a rejeição do acesso a serviços válidos e legítimos por parte de outros utilizadores, por exemplo, colocando um servidor *web* temporariamente fora de serviço.	O sistema está construído para poder ser dividido por diferentes máquinas e a utilização de técnicas de NLB (*Network Load Balancing*) escalando eventuais cargas excessivas de acessos. O servidor *web* sob o qual o sistema foi desenvolvido e o sistema operativo Windows já implementam técnicas eficazes para ignorar ataques de *Denial of Service*.
Elevation of Privilege Um utilizador desprivilegiado consegue um acesso privilegiado ao sistema, adquirindo privilégios para comprometer ou destruir o sistema por completo.	A aplicação verifica a identidade do utilizador e respectivas permissões antes de qualquer operação, e utiliza protocolos de comunicação seguros, evitando possíveis ataques a sessões abertas por outros utilizadores. O IIS 7 corre num contexto não privilegiado e em conjunto com o Windows Server 2008/2012 implementam técnicas eficazes de protecção contra possível elevação de privilégios.

A análise das estratégias de segurança segundo o conjunto de ameaças mais comuns permite garantir que a aplicação não é desenvolvida sobre um contexto inseguro e inutilizável. A reflexão sobre estas estratégias não garante um sistema perfeito, mas garante que a aplicação contém as devidas propriedades de segurança esperadas pelos seus utilizadores.

Uma vez que a aplicação principal a ser desenvolvida é *web*, foram também considerados os dez riscos de segurança mais críticos em aplicações *web* segundo a recomendação *Open Web Application Security Project* (OWASP) [60]. A OWASP é uma comunidade aberta dedicada em assegurar às organizações o desenvolvimento, compra e realização de manutenção de aplicações confiáveis. Para cumprir esse objectivo, disponibiliza ferramentas e informação relacionada com segurança.

Seguem-se os dez riscos de segurança mais críticos identificados pela OWASP e respectivas estratégias de segurança adoptadas.

Risco	Estratégias de Segurança
A1. SQL Injection Injecção de código SQL através de uma página legítima.	Nunca é realizada a concatenação de *strings* dadas pelo utilizador para construir o SQL. Todos os parâmetros de SQL são explicitamente identificados. Não existe portanto risco de SQL Injection.

Risco	Estratégias de Segurança
A2. Cross-Site Scripting (XSS) Esta exploração de comportamento ocorre quando o servidor envia dados para o *browser* sem realizar as devidas operações de *escaping*.	A plataforma ASP .NET faz *escaping* automático de todo o conteúdo dos controlos. De facto, se não for desejado realizar *escaping*, é necessário referir explicitamente no controlo. O ASP .NET também protege contra a introdução de código HTML em controlos, evitando, duplamente o problema de XSS.
A3. Autenticação e Gestão de Sessões Permitir o acesso a dados de autenticação, *hijack* de sessões, ou outro tipo de exploração para assumir a identidade de outra pessoa.	A aplicação corre sobre o IIS, que suporta os últimos avanços em segurança (TLS 1.2) permitindo ocultar toda a transmissão de dados. A plataforma ASP .NET tem integrada uma gestão de sessão bastante segura. A palavra-passe de autenticação não é armazenada em lugar algum, apenas a *hash*.
A4. Referência Directa a um Objecto de forma Insegura Exposição de um acesso a um ficheiro ou a uma interface que não verifica as permissões de utilizador.	O único acesso à camada de negócio do SADAC é através do portal do utente. O portal do utente verifica sempre se o utilizador tem as credenciais necessárias para aceder à informação pedida, antes de realizar o pedido. Caso não tenha, é apresentada uma página de acesso negado, e nenhuma operação é realizada.
A5 Cross-Site Request Forgery (CSRF) Um *website* força o *browser* (por exemplo, através de uma IFrame invisível) a visitar outro *website*, onde a vítima se encontra autenticada, realizando um pedido aparentemente legítimo.	O ASP .NET suporta automaticamente um *token* de autenticação que é partilhado por POST e apenas válido para a sessão em curso. Existem também *hidden fields* que ajudam na identificação da legitimidade do pedido. Esta é a recomendação apresentada pela OWASP.
A6. Má Configuração de Segurança É necessário garantir que existe uma boa configuração de segurança e evitar configurações por omissão.	O IIS requer várias configurações de segurança próprias, por exemplo, a criação de uma *MachineKey* única por cada aplicação *web*. Para além disso, o *web.config* permite uma fácil e rápida gestão de configurações. O *software* utilizado no SADAC é, de momento, o mais recente e pode ser actualizado no futuro.
A7. Armazenamento Criptográfico Inseguro Deve-se persistir os dados sensíveis de forma devidamente encriptada.	Toda a camada de acesso a dados suporta a encriptação de dados através do SQL Server. A palavra-passe é armazenada como *hash* para não correr o risco de vir a ser roubada.

Risco	Estratégias de Segurança
A8. Falhas na Restrição de Acesso a URLs Não deve ser possível aceder a URLs cujo utilizador não deva ter acesso.	A restrição de endereços é realizada por um módulo específico do ASP .NET que assegura que apenas determinados tipos de utilizadores, ou utilizadores específicos, tenham acesso a determinados URLs. Esta verificação toma lugar ainda antes de a página ser chamada no servidor.
A9. Protecção Insuficiente da Camada de Transporte Não usar encriptação ou uso de algoritmos fracos, certificados expirados ou má utilização de certificados.	A camada de transporte é gerida pelo IIS que suporta os mais recentes avanços na encriptação de dados da camada de transporte (TLS 1.2).
A10. Redireccionamentos ou Forwards não Validados A utilização de redireccionamentos ou *forwards* não validados pode levar o utilizador a visitar *websites* de *phishing* e *malware* sem se aperceber.	O SADAC não utiliza *forwards*. Os *redirects* utilizados são explicitamente declarados na aplicação, não envolvendo parâmetros específicos acessíveis ao utilizador, e todos os parâmetros e acções são validados com as permissões do utilizador autenticado. O uso de *routing* permitiu também a criação de endereços virtuais que protegem os endereços reais das páginas do servidor.

Após analisar ambas as recomendações de segurança (Microsoft e OWASP), deve-se referir que todos os módulos de acesso a dados (incluindo o de Entidades Dinâmicas) suportam a utilização de encriptação da base de dados e balanceamento de pedidos (NLB). Em especial, o módulo de entidades dinâmicas realiza operações atómicas (inserção, edição, eliminação), colocando do lado do servidor a gestão de concorrência de pedidos e de segurança dos dados.

7. Testes

Esta secção descreve os testes realizados ao sistema SADAC com o objectivo de testar as funcionalidades implementadas e a sua qualidade de implementação, segundo os requisitos.

Foram realizados quatro tipos de testes: testes de integração, unitários, funcionais (de aceitação) e não funcionais (de aceitação).

Muitos testes foram automatizados através do ambiente de desenvolvimento integrado (IDE) do Microsoft Visual Studio. O Visual Studio apresenta excepcionais capacidades de teste aos projectos em desenvolvimento, permite a criação de testes unitários e testes à camada de apresentação (*web*). [61]

A ferramenta de testes unitários permite executar scripts antes de iniciar e após terminar os testes e estabelecer um contexto de execução antes de executar cada grupo de testes unitários e após essa execução. Os testes são executados num projecto à parte e sobre as classes de acesso a dados e de lógica de negócio. É possível configurar listas ordenadas de testes, se for desejado que os testes sejam executados por determinada ordem. A execução dos testes regista os testes que passaram, o respectivo tempo de execução de cada teste, e os testes que falharam juntamente com a descrição do erro e ferramentas de *debugging* do teste.

A ferramenta de testes à camada de apresentação *web* permite a criação de testes extremamente personalizados, passo a passo, ou a gravação de uma sessão através de um *plugin* integrado no Internet Explorer que grava todas as operações realizadas para posteriormente as automatizar. A gravação consegue obter automaticamente muitos dos parâmetros desejados para a validação, podendo nem sequer ser necessário ajustar os parâmetros de validação. No caso de ser desejada uma regra específica de validação, é possível configurar para cada pedido a inserção ou validação dos seguintes parâmetros: valores dos campos do formulário, valores dos controlos, texto específico no HTML através de *parsers*, URL de um *WebService*/*WebSite*, parâmetros passados por GET/POST, parâmetros do cabeçalho HTTP/HTTPS, *cookies* e até mesmo utilizar expressões regulares para capturar determinado texto no meio de um pedido (HTML, JSON ou outro formato).

7.1. Cenário de Testes

Todos os testes foram realizados num cenário de teste semelhante ao de produção. Segue-se a descrição do cenário.

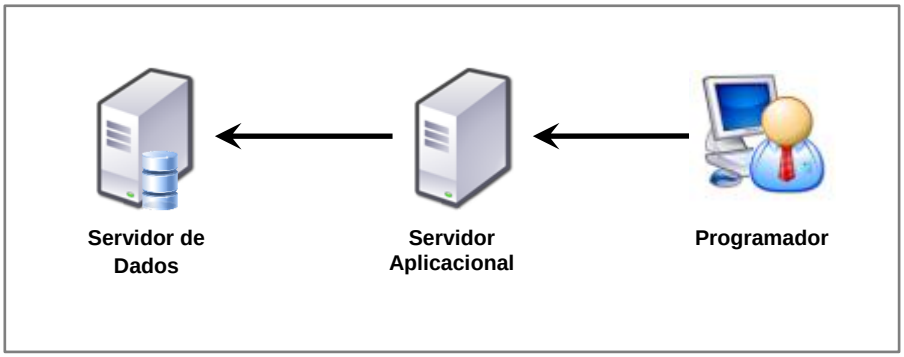

Ilustração 27 – Cenário de Testes

Os servidores são máquinas virtuais com as seguintes características:

- Processador Intel Core i7-2630QM 2.0GHz;
- 8Gb de Memória RAM;
- Sistema Operativo Windows Server 2008 de 64 bits;
- Disco rígido de 1Tb.

O servidor de dados possui o Microsoft SQL Server 2012. O servidor aplicacional possui o IIS 7.0, o Microsoft Dynamics CRM 2012 e o Microsoft Visual Studio 2008.

7.2. Testes Unitários

Os testes funcionais avaliam o estado de implementação de pedaços de código, normalmente classes. As várias classes de acesso a dados e de negócio foram testadas. A execução dos respectivos métodos públicos foi testada em função do que seria esperado que os métodos realizassem ou retornassem.

A execução de testes unitários foi automatizada através do ambiente de desenvolvimento Microsoft Visual Studio. Os testes unitários automáticos são importantes durante o desenvolvimento para assegurar o correcto funcionamento das classes apesar das modificações realizadas. Alterações em determinadas classes podem interferir com o funcionamento de outras classes, de forma imprevisível, tornando importante correr os testes unitários no final de cada *sprint*.

Foram realizados testes unitários às principais funcionalidades das camadas de acesso a dados e de lógica de negócio, conforme a seguinte imagem apresenta.

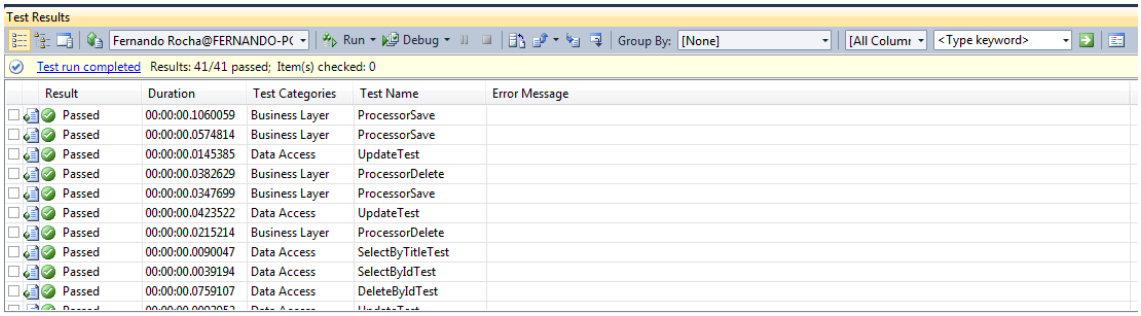

Ilustração 28 – Resultado dos Testes Unitários

Como a imagem apresenta, todos os testes unitários foram executados com sucesso. No total, foram realizados 41 testes unitários **automatizados**, todos com sucesso.

7.3. Testes Funcionais

Os testes funcionais são importantes para demonstrar a capacidade do sistema cumprir determinada funcionalidade. Segundo a *framework* Scrum, todos os requisitos (descritos em *user stories*), apresentam um ou vários critérios de aceitação, critérios esses que servem para comprovar se o requisito foi correctamente implementado. A execução de testes funcionais automatizados pode, neste sentido, ser complexa de implementar, uma vez que a interface com o utilizador é muito dinâmica durante o processo inicial de desenvolvimento.

Todos os requisitos implementados foram devidamente validados com o *product owner* através da verificação **manual** do critério de validação, na *sprint review meeting*. Para o efeito, cada requisito implicou uma pequena demonstração da funcionalidade para ser validado.

Apesar da verificação manual do critério de validação, é importante construir testes automáticos que validem pelo menos as principais funcionalidades da aplicação. A execução automática dos testes permite validar uma das principais componentes da aplicação, a interface com o utilizador. Por muito bem que as camadas inferiores se comportem, se o utilizador não conseguir realizar adequadamente determinada operação, de nada adianta.

Por esse motivo, foram elaborados 6 testes **automatizados** que validam várias funcionalidades a nível da interface com o utilizador. Os testes validam desde a simples acção de autenticação segundo determinado perfil, até à capacidade de introdução, visualização e modificação de entidades. Os processos de elaboração e manutenção de testes complexos despendem de muito tempo, sendo portanto importante encontrar um equilíbrio saudável entre a quantidade de testes e a

implementação de funcionalidades. Muitas funcionalidades deixam até de ser válidas ao longo do tempo com as sucessivas alterações da aplicação, e nesse sentido, o tempo despendido na elaboração do teste automatizado muitas vezes não se justifica. Os testes funcionais tornam-se mais importantes numa fase de maior manutenção da aplicação, e não nesta fase inicial em que existe uma forte implementação de novas funcionalidades.

7.4. Testes de Integração

Uma vez que o SADAC comunica com vários sistemas externos, é importante testar a correcta comunicação entre os diversos sistemas. O SADAC possui também *plugins* cuja integração é realizada em *runtime*, tornando também importante testar se todos os *plugins* desenvolvidos conseguem ser dinamicamente carregados para o contexto de execução da aplicação. A execução de testes de integração foi **automatizada** através do ambiente de desenvolvimento Microsoft Visual Studio.

Foram realizados testes entre os seis *plugins* desenvolvidos e a aplicação, entre o Microsoft SQL Server e os respectivos *plugins* de acesso a dados, e entre a Microsoft Dynamics e os respectivos *plugins* de acesso a dados.

Os testes de integração dos *plugins* consistem na tentativa de carregamento do *plugin* para o contexto de execução seguida de um acesso a dados a partir do *plugin* para garantir que o *plugin* consegue ligar-se ao sistema que contém os dados (Microsoft Dynamics ou Microsoft SQL Server).

Todos os testes foram executados com sucesso até ao momento. Os testes de integração elaborados de forma automatizada cobrem os seguintes

7.5. Testes Não Funcionais

Os testes não funcionais são muito importantes para verificar a viabilidade do sistema num ambiente real. O SADAC conta com dois módulos cuja implementação necessita de validação extra funcional, o módulo de Raciocínio Baseado em Regras (RBR) e o módulo de Entidades Dinâmicas. O anexo “Testes Não Funcionais” apresenta os valores brutos e respectivos desvios padrões de todos os testes não funcionais realizados. Esta secção apresenta apenas os principais.

É importante referir que os requisitos não funcionais foram devidamente validados pelo *product owner*. A maioria dos requisitos não funcionais, por exemplo, “Utilização de Tecnologias Microsoft”, são muito simples de validar e não fazendo sentido elaborar testes automatizados.

Os requisitos não funcionais aqui apresentados são requisitos mais complexos que necessitaram de um estudo mais elaborado para determinar se estão ou não a ser cumpridos. Todas as medições apresentadas foram repetidas **dez vezes**, realizada uma **média** do tempo de execução, e associado um **desvio padrão**.

7.5.1. Módulo de Raciocínio Baseado em Regras

O módulo de raciocínio baseado em regras apresenta o requisito não funcional "**RBR-NFP-01**" referente ao desempenho do módulo, tornando necessário um estudo cuidado para compreender o comportamento do módulo quando se faz variar os atributos relacionados, desde valores claramente baixos a valores possivelmente mais elevados que a realidade.

Os gráficos que se seguem demonstram o desempenho do módulo quando se varia o número de utilizadores existentes na base de dados, sobre o conjunto de regras referentes ao Plano Nacional de Vacinação (PNV) em vigor. As regras introduzidas foram validadas pelo *product owner*.

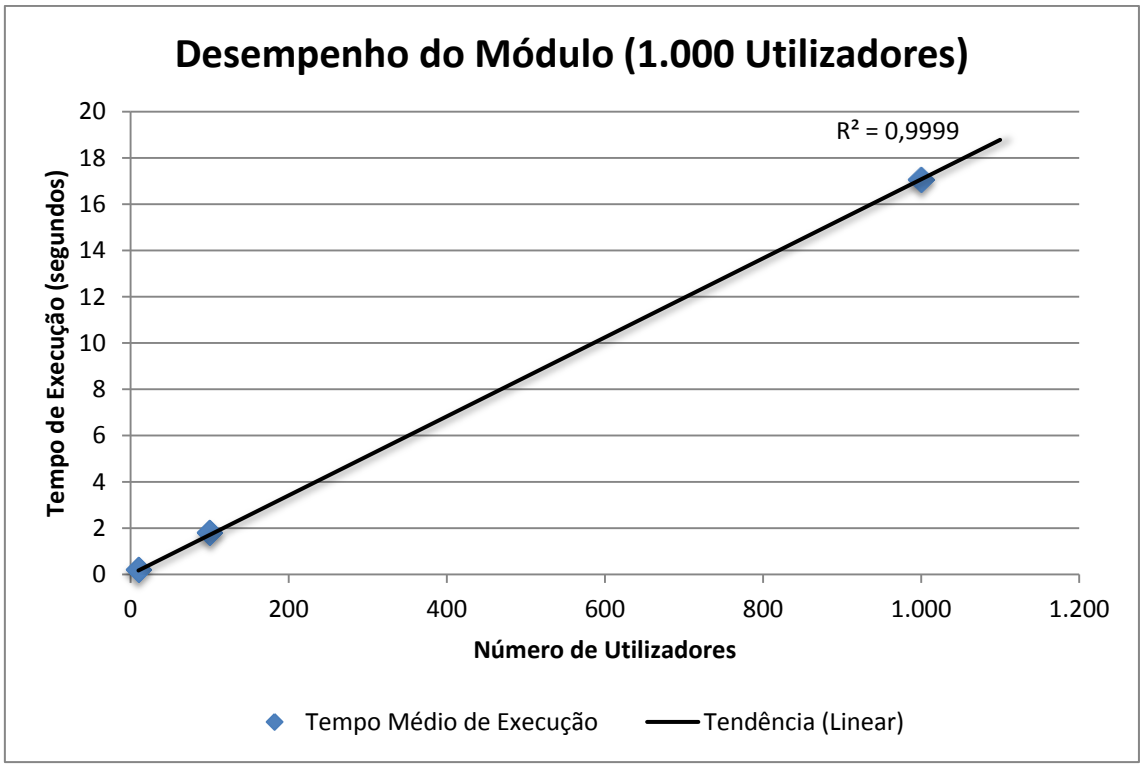

Ilustração 29 – Desempenho do Módulo de RBR até 1.000 Utilizadores

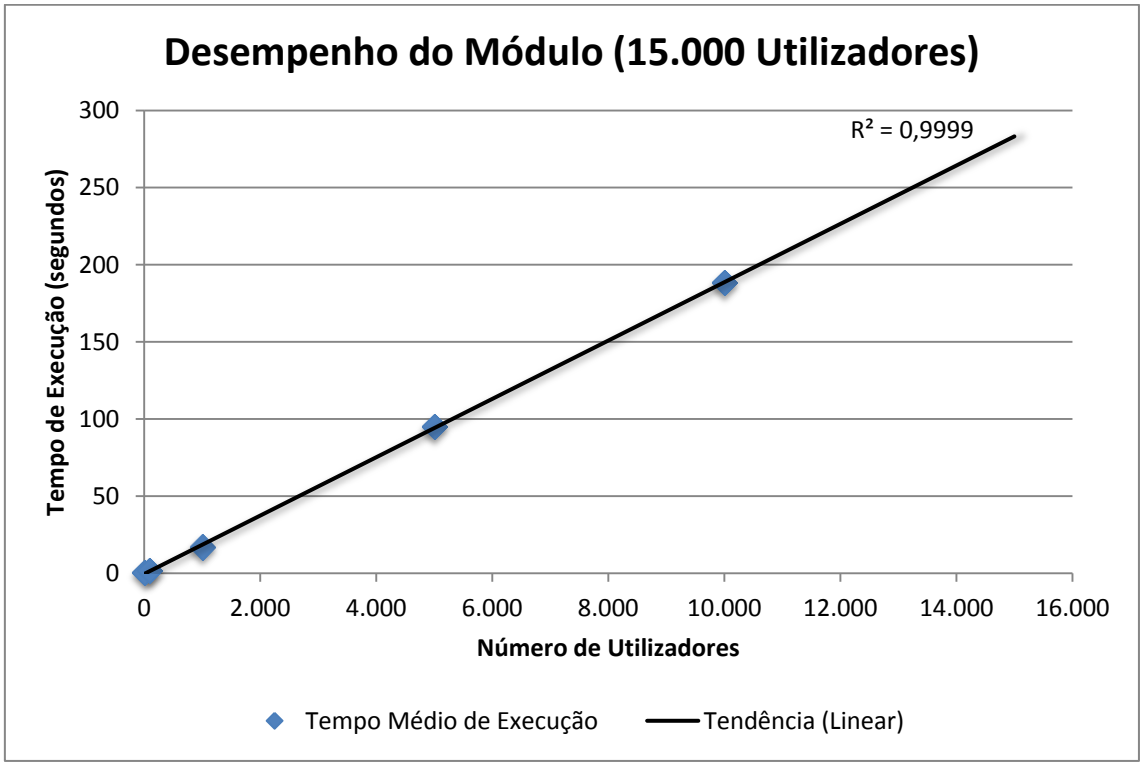

Ilustração 30 – Desempenho do Módulo de RBR até 15.000 Utilizadores

Como é possível observar, o módulo de RBR apresenta um desempenho quase perfeitamente linear (R^2=0,9999) no que diz respeito a um aumento do número de utilizadores.

Os seguintes gráficos apresentam a relação entre o desempenho do módulo e o aumento do número de regras. O número de regras para o Plano Nacional de Vacinação (PNV) ronda as 10 regras (20 contando com as negações), esta será portanto a unidade base. O aumento de regras pode ser realizado de duas formas, aumentando o número do grupo de regras ou aumentando o número de regras por grupo. Em termos de desempenho é significativo testar ambas as possibilidades uma vez que o motor de regras WF Rules corre um grupo de regras de forma atómica, e poderá possuir técnicas que optimizam o desempenho neste sentido.

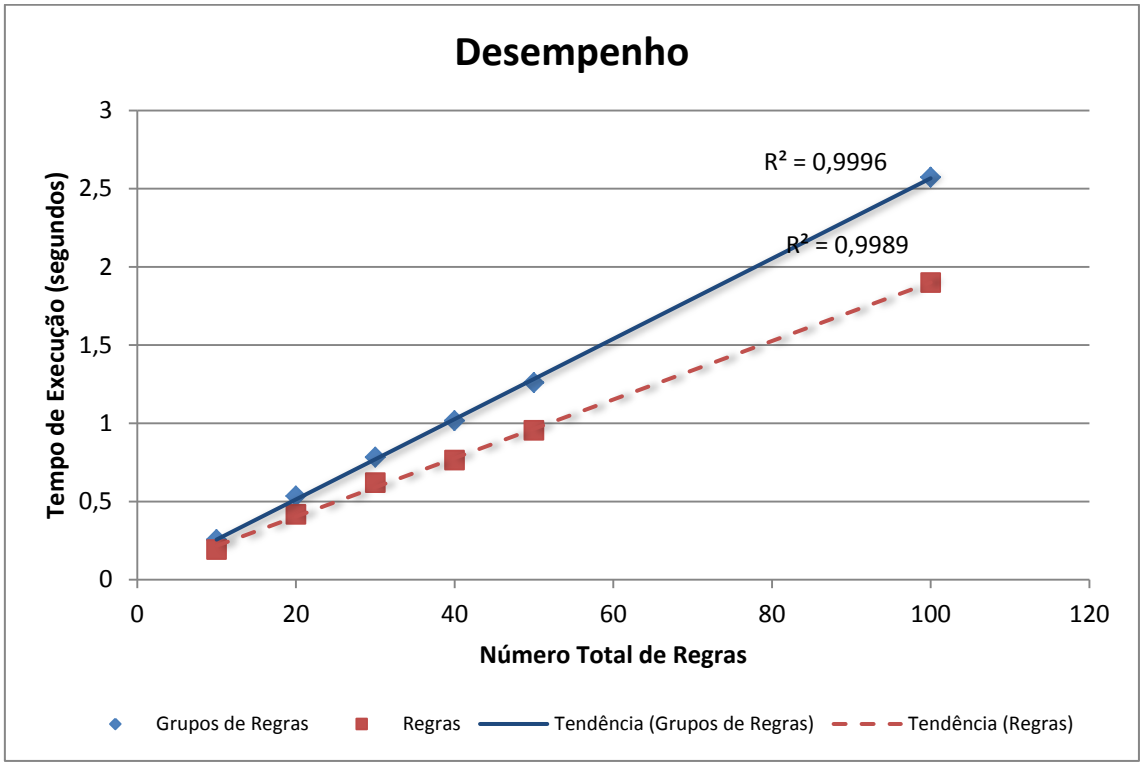

Ilustração 31 - Desempenho do Módulo de RBR

Como é possível observar, o desempenho do módulo com a variação do número de regras é também quase perfeitamente linear ($R^2 \cong 1$). O desempenho do módulo com o aumento do número de grupos de regras é ligeiramente inferior, de uma forma proporcional, ao desempenho aumentando o número de regras dentro do grupo de regras.

O requisito não funcional "RBR-NFP-01" demonstra-se cumprido.

7.5.2. Módulo de Entidades Dinâmicas

O módulo de entidades dinâmicas apresenta dois requisitos não funcionais importantes estudar:

- "**DYN-NFP-01**" referente ao desempenho do módulo Dynamic Entity (DE) em comparação com a Entity Framework (EF);
- "**DYN-NFP-02**" referente ao desempenho dos formulários dinâmicos em comparação com formulários estáticos.

É portanto necessário um estudo cuidado para compreender o comportamento do módulo quando se faz variar os atributos relacionados, desde valores claramente baixos a valores possivelmente mais elevados que a realidade.

As seguintes imagens são gráficos que ilustram a comparação entre o desempenho da Dynamic Entity e da Entity Framework, requisito **DYN-NFP-01**.

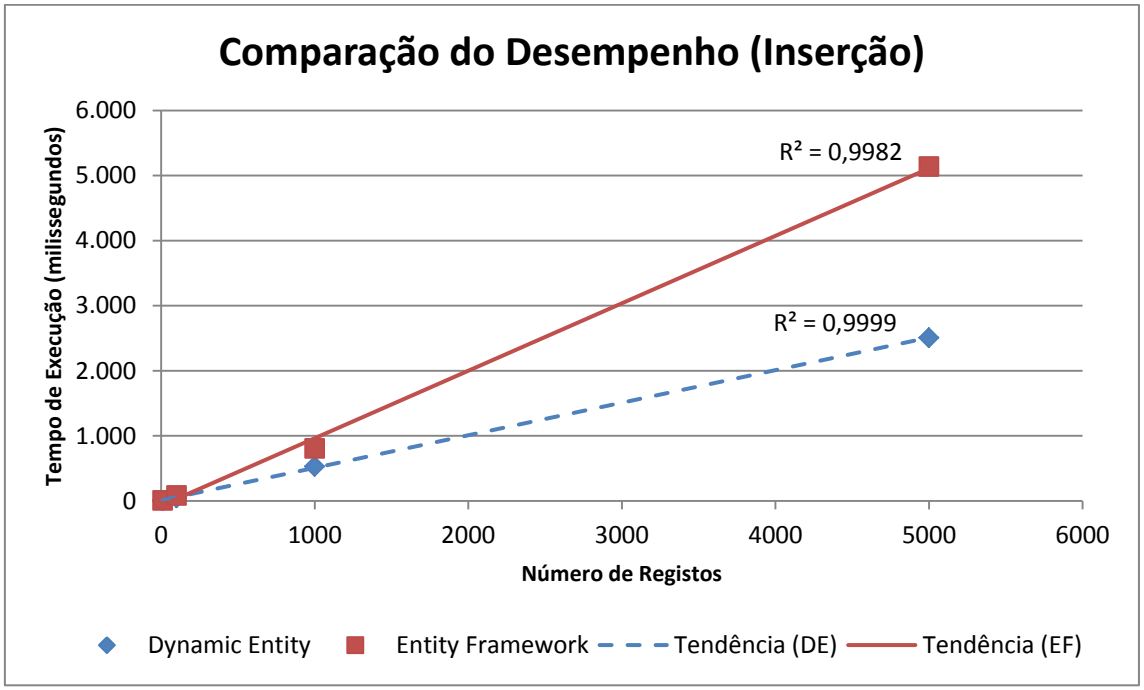

Ilustração 32 – Comparação do Desempenho entre a EF e DE (Insert)

Esta comparação demostra que a Entity Framework possui um desempenho ligeiramente inferior que a Dynamic Entity no que diz respeito a inserções de dados na base de dados. A seguinte imagem apresenta a tendência dessa diferença.

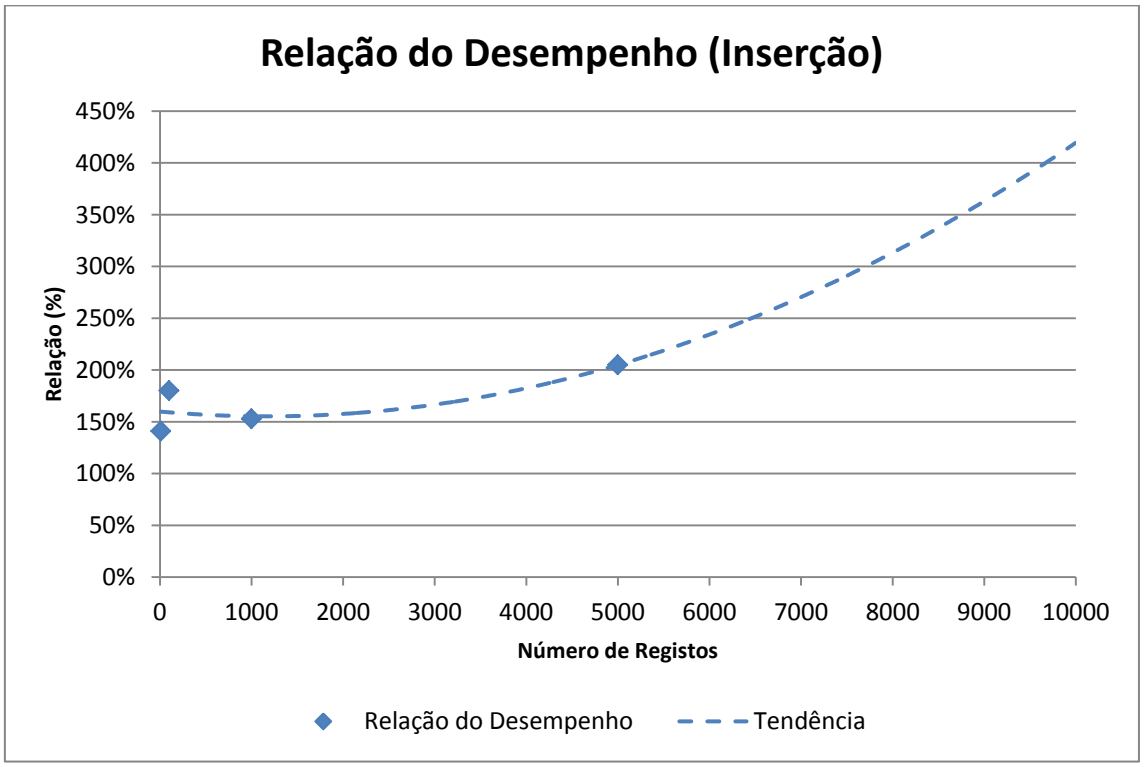

Ilustração 33 – Relação do Desempenho entre a EF e DE (Inserção)

O gráfico anterior demonstra um afastamento da diferença entre o desempenho da Dynamic Entity e da Entity Framework, em que a Dynamic Entity supera cada vez mais a Entity Framework para as operações de inserção de dados.

O seguinte gráfico apresenta a comparação entre o desempenho (tempo de execução) da Dynamic Entity face ao desempenho da Entity Framework para a operação de obtenção de dados.

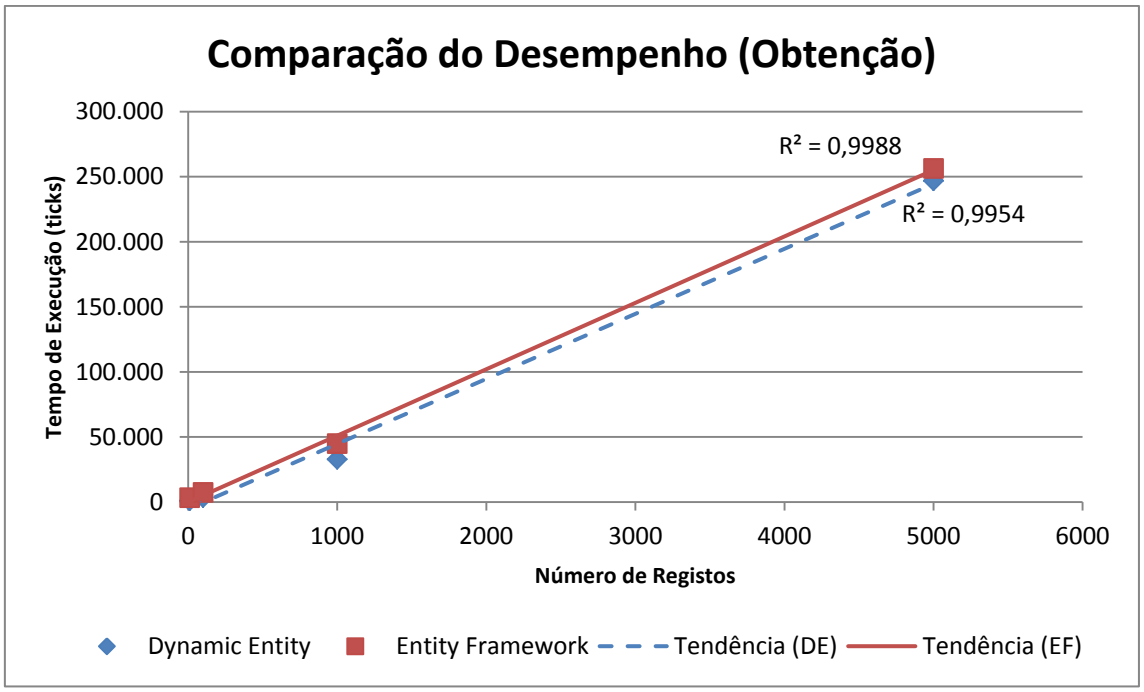

Ilustração 34 – Comparação do Desempenho entre a EF e DE (Obtenção)

O gráfico anterior demonstra uma relação praticamente linear entre o aumento de registos e o tempo necessário para obtenção de dados. A Dynamic Entity demonstra um desempenho ligeiramente superior à Entity Framework, mesmo para grandes quantidades de dados. É importante compreender como esta relação se comporta, ou seja, se o desempenho da Dynamic Entity será sempre superior, se tenderá a tornar-se inferior ou se tende a estacionar num valor.

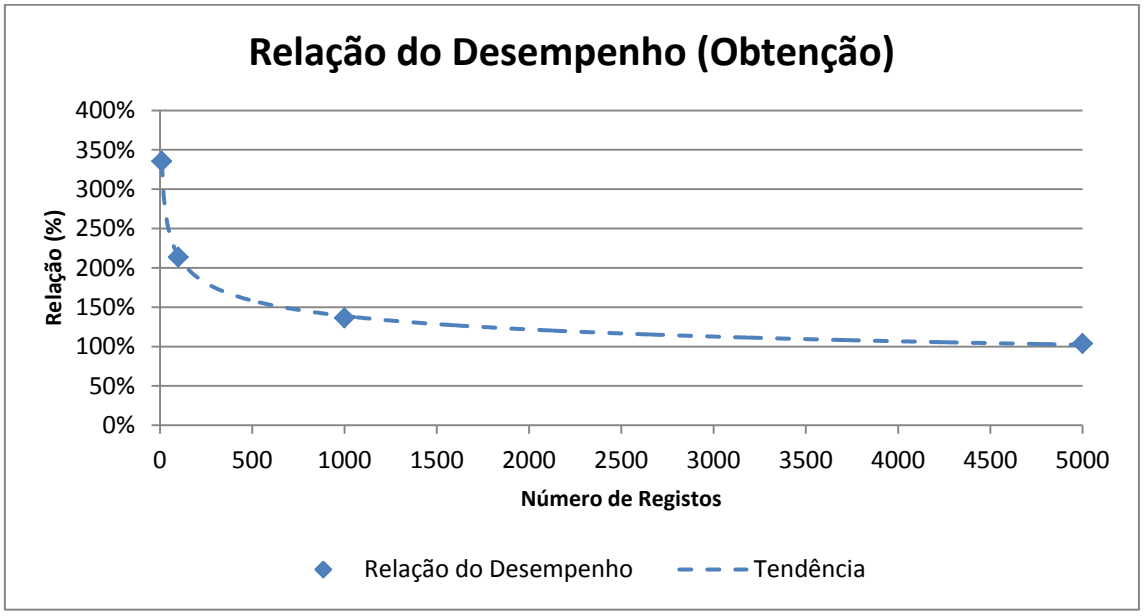

Ilustração 35 – Relação do Desempenho entre a EF e DE (Obtenção)

O gráfico anterior demonstra uma evolução potencial da relação do desempenho, isto é, o desempenho da Dynamic Entity tende a estacionar para valores próximos do desempenho da Entity Framework (100%) para números de registos muito elevados.

Estes dados demonstraram que a Dynamic Entity não só apresenta um tempo de execução menor do que três vezes o da Entity Framework (cumprindo o critério de aceitação estipulado no requisito não funcional DYN-NFP-01), como apresenta um desempenho ainda melhor. Este desempenho superior justifica-se por se tratar de um componente com um nível de abstracção inferior ao da Entity Framework, por exemplo, a Dynamic Entity não implementa um contexto com um gestor de repositório de entidades, a Entity Framework tem portanto um *overhead* de gerir esse repositório.

Segue-se o gráfico que compara os tempos de construção do formulário dinâmico em comparação com um formulário estático, referente ao requisito não funcional DYN-NFP-02. Fez-se variar o número de campos do formulário para conseguir compreender a relação entre os tempos de construção.

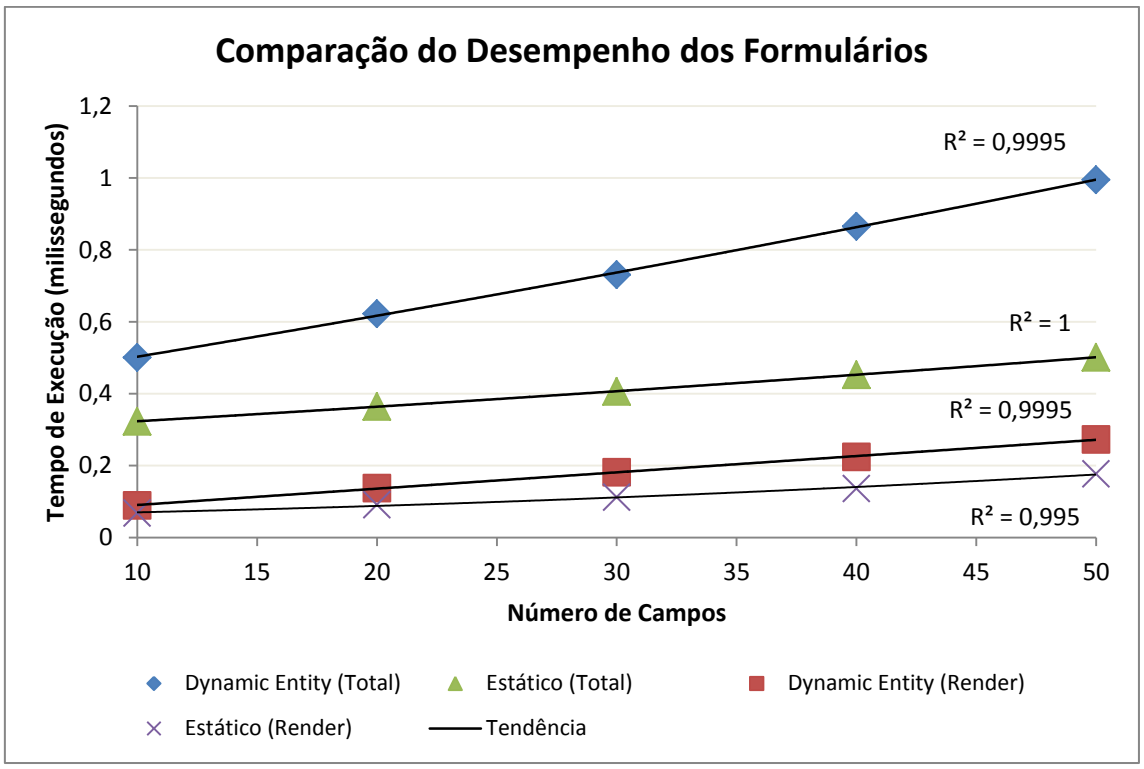

Ilustração 36 - Comparação dos Tempos de Criação do Formulário de Inserção

Através dos dados dos tempos de execução foi ainda representada a relação entre estes tempos. A relação realiza-se dividindo os tempos de execução dos formulários estáticos pelos tempos de execução dos formulários dinâmicos (do módulo de entidades dinâmicas). Portanto, um valor de 100% significa que os formulários dinâmicos apresentam um desempenho (em termos de tempo de execução) semelhante aos estáticos, valores superiores a 100% significa que apresentam um desempenho superior, valores inferiores significa que apresentam um desempenho inferior.

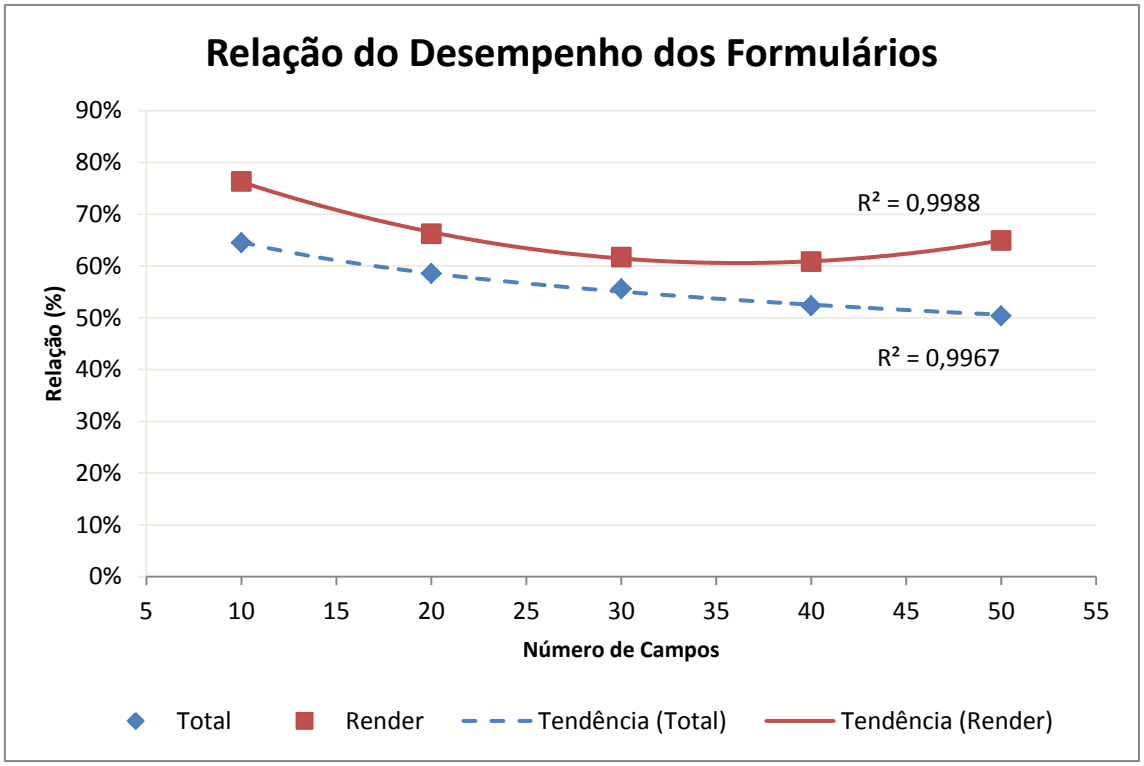

Ilustração 37 – Relação dos Tempos de Execução do Formulário de Inserção

O gráfico demonstra que apesar do desempenho, em termos de tempo de execução, da construção do formulário de inserção do módulo de entidades dinâmicas ser ligeiramente inferior (não ultrapassa o dobro do tempo), a relação entre os tempos de execução está a convergir para um valor fixo, ou até mesmo eventualmente recuar (função polinomial de 2º grau). Deve-se ainda ter em consideração que raramente existirá um formulário de 50 campos, logo o desempenho do formulário dinâmico será muito mais semelhante ao estático.

8. Estado da Implementação

Esta secção resume o estado actual do trabalho realizado. Para simplificar a descrição, são apresentados os requisitos dos três componentes principais, o seu estado de implementação (data em que foi pela primeira vez verificado através do critério de aceitação) e o seu estado de validação por parte dos *stakeholders* do projecto (data de validação).

Uma vez que o Portal do Utente é um componente com uma interface visual de interacção com o utilizador, o *design* actual é também apresentado nesta secção como forma de comparação com o protótipo de baixa-fidelidade anteriormente apresentado.

8.1.1. Portal do Utente

A seguinte tabela descreve o estado de implementação dos requisitos:

Requisito		Prioridade	Implementação	Validação
Requisitos Funcionais				
WEB-RF-01	Visualizar Histórico e Calendarização Actual de Consultas	COULD	-	-
WEB-RF-02	Efectuar Pedidos de Marcação de Consultas	COULD	-	-
WEB-RF-03	Visualizar Pedidos de Marcação de Consultas	COULD	-	-
WEB-RF-04	Visualizar os Alertas Activos	MUST	07-06-2012	18-06-2012
WEB-RF-05	Inserir Registos de Medições Periódicas	COULD	-	-
WEB-RF-06	Visualizar Registos de Medições Periódicas	COULD	-	-
WEB-RF-07	Colocação de Dúvidas	WON'T	-	-
WEB-RF-08	Visualizar Histórico Pessoal de Dúvidas	WON'T	-	-
WEB-RF-09	Adicionar Contas Associadas	COULD	-	-
WEB-RF-10	Gerir Informações das Contas Associadas	COULD	-	-
WEB-RF-11	Gerir Informações dos Utentes	MUST	07-06-2012	18-06-2012
WEB-RF-12	Gerir Pedidos de Marcação de Consultas	SHOULD	15-06-2012	18-06-2012
WEB-RF-13	Gerir Contas Associadas	COULD	15-06-2012	18-06-2012
WEB-RF-14	Confirmar/Rejeitar Alertas Despoletados	WON'T	-	-
WEB-RF-15	Permitir Alertas Sujeitos a Confirmação	WON'T	-	-
WEB-RF-16	Controlo de Edição de Regras	WON'T	-	-
WEB-RF-17	Interface de Gestão do Módulo de Raciocínio Baseado em Regras	MUST	14-05-2012	31-05-2012
WEB-RF-18	Interface de Gestão do Módulo de Entidades Dinâmicas	MUST	05-06-2012	18-06-2012

Requisito		Prioridade	Implementação	Validação
Requisitos Não Funcionais – Produto				
WEB-NFP-01	Protecção dos Dados de Autenticação	MUST	30-03-2012	30-03-2012
WEB-NFP-02	Protecção da Informação Pessoal dos Utentes	SHOULD	22-03-2012	30-03-2012
WEB-NFP-03	Capacidade de Auditoria	WON'T	-	-
Requisitos Não Funcionais – Organizacionais				
WEB-NFO-01	Utilização Exclusiva de Tecnologias Microsoft	MUST	02-02-2012	02-02-2012
WEB-NFO-02	Utilização de Recursos Localizados	SHOULD	29-03-2012	30-03-2012
WEB-NFO-02	Localização de Recursos para a Língua Portuguesa	MUST	29-03-2012	30-03-2012
Requisitos Não Funcionais – Externos				
WEB-NFE-01	Interoperabilidade com o Módulo de Raciocínio Baseado em Regras	MUST	27-03-2012	30-03-2012
WEB-NFE-02	Interoperabilidade com o Módulo de Entidades Dinâmicas	MUST	05-06-2012	18-06-2012
WEB-NFE-03	Plugins de Acesso a Dados para a Plataforma Microsoft Dynamics CRM	MUST	27-03-2012	30-03-2012
WEB-NFE-04	Plugins de Acesso a Dados Aplicacionais para Microsoft SQL Server	MUST	27-03-2012	30-03-2012
WEB-NFE-05	Plugin de Acesso a Dados Clínicos para Entidades Dinâmicas	MUST	05-06-2012	18-06-2012

Tabela 2 – Tabela de Requisitos do Portal do Utente

Foi possível concluir os objectivos de implementação do Portal do Utente, uma vez que todos os requisitos com a prioridade "*MUST*" foram implementados e validados pelos *stakeholders*. A implementação do novo módulo de Entidades Dinâmicas implicou alterações críticas nas prioridades dos requisitos, uma vez que este módulo não estava inicialmente planeado. Mesmo assim apenas não foi possível cumprir alguns requisitos "*COULD*", ou seja, menos prioritários.

8.1.2. Design do Portal do Utente

Esta secção apresenta o *design* actual do Portal do Utente. Dependendo do tipo de utilizador autenticado a interface do Portal do Utente pode alterar ligeiramente para restringir o acesso do utilizador aos locais a que não possui acesso. Por exemplo, o botão "BackOffice" não aparece aos utilizadores que apenas tenham o papel "Utente" no seu perfil.

Seguem-se dois *print-screens* do Portal do Utente no estado actual de implementação. O primeiro é referente ao *FrontOffice* (acesso por parte do utente

comum), o outro é referente ao *BackOffice*, acesso por parte de profissionais de saúde, auxiliares e administradores.

Ilustração 38 – Página Principal do Portal do Utente (FrontOffice)

Ilustração 39 – Página das Entidades Dinâmicas (BackOffice)

Como é possível observar a versão actual da aplicação assemelha-se bastante ao protótipo de baixa-fidelidade elaborado na fase de concepção do projecto. O design

ainda tem bastante trabalho a ser realizado, mas de momento o cliente prioriza as funcionalidades ao design.

8.1.3. Módulo de Raciocínio Baseado em Regras

A seguinte tabela descreve o estado de implementação dos requisitos:

Requisito		Prioridade	Implementação	Validação
Requisitos Funcionais				
RBR-RF-01	Gestão da Base de Dados de Regras	MUST	14-05-2012	31-05-2012
RBR-RF-02	Geração de Alertas	MUST	07-06-2012	18-06-2012
RBR-RF-03	Grupos de Regras	WON'T	14-05-2012	31-05-2012
RBR-RF-04	Permitir Activar/Desactivar Grupos de Regras	WON'T	14-05-2012	31-05-2012
Requisitos Não Funcionais – Produto				
RBR-NFP-01	Desempenho do Módulo	MUST	08-07-2012	09-07-2012
Requisitos Não Funcionais – Organizacionais				
RBR-NFO-01	Utilização Exclusiva de Tecnologias Microsoft	MUST	02-02-2012	02-02-2012
RBR-NFO-02	Utilização de Recursos Localizados	SHOULD	14-05-2012	31-05-2012
RBR-NFO-03	Localização de Recursos para a Língua Portuguesa	MUST	14-05-2012	31-05-2012
Requisitos Não Funcionais – Externos				
RBR-NFE-01	Interoperabilidade com o Portal do Utente	MUST	27-03-2012	30-03-2012

Tabela 3 – Tabela de Requisitos do Módulo de RBR

A implementação do módulo de Raciocínio Baseado em Regras (RBR) foi um **sucesso**. Todos os requisitos com a prioridade "*MUST*" foram implementados, até mesmo os dois requisitos marcados como "*WON'T*", que apenas seriam implementados em futuras versões do SADAC.

Os dois requisitos "*WON'T*" foram implementados porque o sistema de gestão de regras de negócio utilizado, o WF Rules, já suportava essas funcionalidades de base (grupos de regras e activar ou desactivar grupos de regras), tornando pouco significativo o custo de implementação.

8.1.4. Módulo de Entidades Dinâmicas

A seguinte tabela descreve o estado de implementação dos requisitos:

Requisito		Prioridade	Implementação	Validação
Requisitos Funcionais				
DYN-RF-01	Configuração em Runtime	MUST	05-06-2012	18-06-2012
DYN-RF-02	Entidades em Diferentes Tabelas	MUST	05-06-2012	18-06-2012
DYN-RF-03	Formulário Dinâmico	MUST	05-06-2012	18-06-2012
DYN-RF-04	Listagens de Entidades Dinâmicas	MUST	05-06-2012	18-06-2012
Requisitos Não Funcionais – Produto				
DYN-NFP-01	Desempenho do Módulo	MUST	08-07-2012	09-07-2012
DYN-NFP-02	Desempenho dos Formulários	MUST	08-07-2012	09-07-2012
Requisitos Não Funcionais – Organizacionais				
DYN-NFO-01	Utilização Exclusiva de Tecnologias Microsoft	MUST	02-02-2012	02-02-2012
Requisitos Não Funcionais – Externos				
DYN-NFE-01	Interoperabilidade com o Portal do Utente	MUST	05-06-2012	18-06-2012

Tabela 4 – Tabela de Requisitos do Módulo de Entidades Dinâmicas

A implementação do módulo de Raciocínio Baseado em Regras (RBR) foi um sucesso. Todos os requisitos com a prioridade "*MUST*" foram implementados, até mesmo dois dos três requisitos marcados como "*WON'T*", ou seja, não estava prevista a sua implementação nesta versão.

Os dois requisitos "*WON'T*" foram implementados porque o sistema de gestão de regras de negócio utilizado, o WF Rules, já suportava essas funcionalidades de base, não aumentando significativamente o custo da sua implementação.

9. Planeamento

Este capítulo descreve o planeamento do estágio, os desvios que ocorreram e respectivos motivos e estratégias de atenuação. As tarefas a vermelho no *gantt* de controlo significam que não ocorreram como planeado, ou seja, foram realizadas com uma duração diferente da planeada.

9.1. Primeiro Semestre

A calendarização inicial previa a conclusão das fases de concepção e elaboração da primeira versão do projecto, no primeiro semestre de estágio. A primeira fase consiste em analisar, com as partes interessadas, a viabilidade do projecto, fazendo uma aprendizagem e análise das tecnologias, o estudo do estado da arte e alguns requisitos principais do sistema para a primeira versão. De seguida, a formulação da visão do projecto, a refinação dos detalhes dos requisitos e a elaboração da arquitectura e desenho do sistema.

O seguinte *Gantt* descreve o planeamento inicial das tarefas do primeiro semestre de estágio.

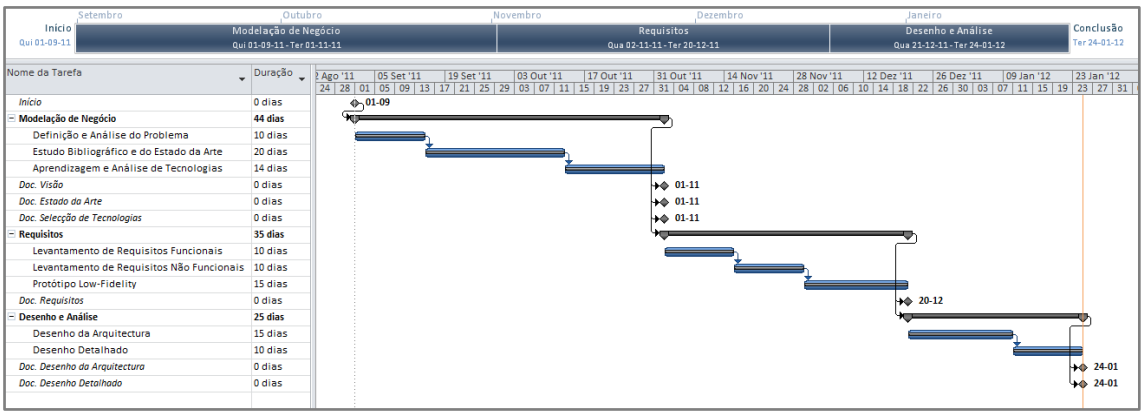

Ilustração 40 – Gantt Planeado do 1º Semestre

O *Gantt* planeado previa uma duração longa para as tarefas de “estudo do estado da arte” e “aprendizagem e análise de tecnologias”. Foi necessário adquirir conhecimentos acerca da plataforma Microsoft .NET, do Raciocínio Baseado em Regras e de Sistemas de Gestão de Regras de Negócio. Existiu portanto, uma forte necessidade de aprendizagem de conceitos complexos completamente novos.

Em termos de datas concretas, a fase de Modelação de Negócio foi prevista terminar a 1 de Novembro de 2011, a fase de Requisitos a 20 de Dezembro de 2011 e a fase de Desenho e Análise a 24 de Janeiro de 2012.

O seguinte *Gantt* descreve de que forma a execução das tarefas ocorreu e se diferenciou com o que foi inicialmente planeado.

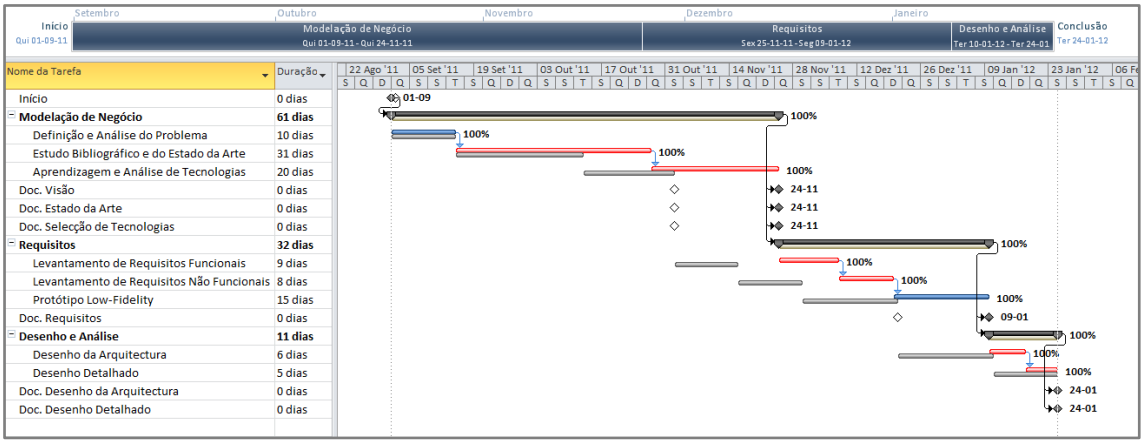

Ilustração 41 – Gantt de Controlo do 1º Semestre

Ocorreu uma clara derrapagem (23 dias) no estudo do estado da arte e na aprendizagem de novas tecnologias (primeira fase). Esta derrapagem deveu-se essencialmente a três factores:

- Ambiguidade dos objectivos colocados pelo cliente;
- Subestimação da complexidade do problema e das tecnologias;
- Atraso de duas semanas na reunião com o cliente.

Este tipo de dificuldades é previsível em projectos com um cliente, principalmente se não for um cliente local, uma vez que existe sempre a necessidade de negociação de datas de reunião e problemas em transmitir correctamente a ideia do projecto.

Na segunda reunião, no dia 4 de Novembro de 2011, foi-nos esclarecido que os principais objectivos do projecto não eram exactamente os que inicialmente se tinha compreendido. Foi, nessa altura, descrito um cenário com um dispositivo móvel a aceder a um *Web Site* (ou aplicação móvel) com a capacidade de agendar consultas, visualizar as actuais consultas, esclarecer dúvidas, registar medições periódicas e apresentar recomendações dadas pelo sistema (obtidas pelo sistema baseado em regras).

A visão inicial do projecto dava mais ênfase ao Raciocínio Baseado em Casos e à integração com os dados da Microsoft Dynamics CRM. Em termos de negócio, a alteração nos objectivos principais modificou a visão do projecto, acrescentou-lhe mais complexidade e impossibilitou a sua execução completa dentro da duração deste estágio. Em termos de estágio, apesar do conhecimento adquirido no estudo do

Raciocínio Baseado em Casos e da viabilidade da integração com a Microsoft Dynamics CRM ser útil, deixou de ser o principal foco, atrasando o meu Estudo do Estado da Arte. A integração do produto com a plataforma Dynamics CRM será tão subtil que o utilizador do produto poderá não se aperceber que as duas estão integradas. O produto será instalado como uma aplicação completamente à parte.

Para a redução do atraso foi necessário aumentar a carga de trabalho e tentar antever os próximos problemas e obter uma estratégia de resolução de forma atempada.

A tarefa de requisitos durou pouco menos que o esperado – seria esperado demorar 35 dias e demorou cerca de 32 até a primeira versão estar terminada. Todos os requisitos e *mockups* foram aprovados até dia 1 de Fevereiro, antes da primeira reunião de planeamento da *sprint*.

Apesar do estudo do estado da arte e a análise de tecnologias ter demorado bastante mais do que o previsto, com esforço, foi possível cumprir todos os objectivos do primeiro semestre, tornando esta fase de desenvolvimento um sucesso.

9.2. Segundo Semestre

O segundo semestre consistiu na conclusão das duas outras fases, a de construção e a de transição, do processo de desenvolvimento de *software*. O seguinte *Gantt* apresenta o planeamento das principais tarefas:

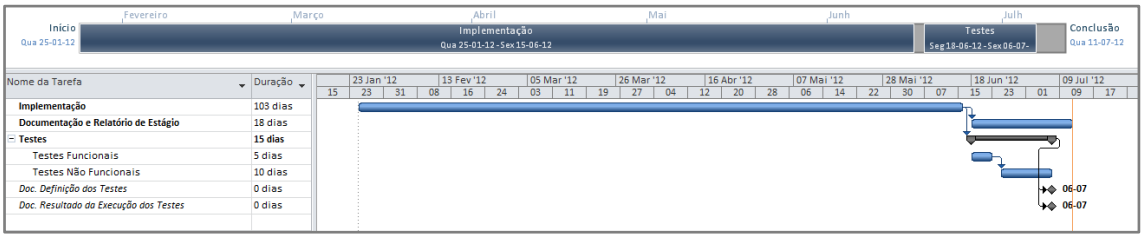

Ilustração 42 – Gantt Planeado do 2º Semestre

O planeamento previa 103 dias para a implementação segundo a *framework* Scrum, ou seja, cerca de quatro *Sprints*, admitindo que cada *Sprint* terá uma duração média de um mês.

Previa-se a duração de 18 dias para escrita do relatório de estágio, e ao mesmo tempo, 15 dias para realizar os testes funcionais e não funcionais.

O seguinte *Gantt* descreve de que forma a execução das tarefas ocorreu e se diferenciou com o que foi inicialmente planeado para o segundo semestre.

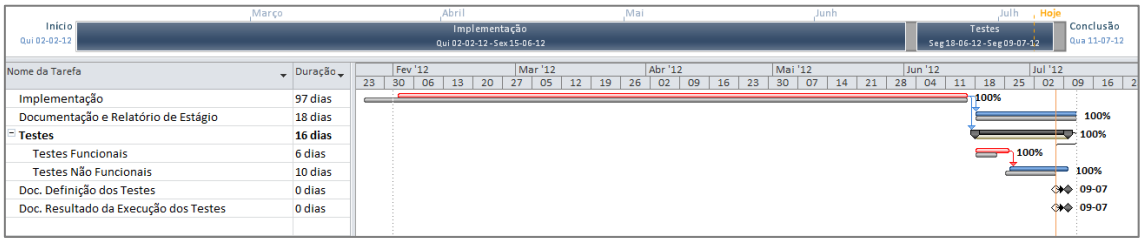

Ilustração 43 – Gantt de Controlo do 2º Semestre

A tarefa de implementação foi iniciada cerca de 6 dias depois do previsto, uma vez que se decidiu aproveitar para ter algumas reuniões e ajustar alguns detalhes da documentação sincronizando o início da fase de implementação com o dia 2 de Fevereiro. O início da primeira *sprint* tomou lugar no dia 2 de Fevereiro e a última sprint durou apenas duas semanas, conforme planeado, terminando no dia 15 de Junho.

De uma forma geral, ocorreu apenas um pequeno atraso na realização dos testes, demorando mais um dia do que o esperado, derrapando até muito próximo da entrega final do estágio.

As tarefas realizadas em cada sprint e respectivos *burndown charts* da sua execução encontram-se no anexo "*Sprints de Implementação*".

10. Conclusões

O objectivo principal desta dissertação foi a análise e desenvolvimento da primeira versão do Sistema de Acompanhamento do Desenvolvimento de Adolescentes e Crianças (SADAC).

A primeira versão do SADAC consiste em três grandes módulos, o Portal do Utente, o módulo de Raciocínio Baseado Em Regras (RBR) e o módulo de Entidades Dinâmicas – todo o sistema desenvolvido sobre a plataforma Microsoft .NET 4.0.

As principais dificuldades prenderam-se com a dimensão e complexidade da aplicação. Devido a esta elevada dimensão e complexidade, o estudo do estado da arte e a aprendizagem de tecnologias demorou mais do que o esperado. Contudo, permitiu adquirir conhecimentos de metodologias e tecnologias direccionadas para os problemas identificados conduzindo a uma solução sólida, fundamentada, funcional e com um óptimo desempenho.

A arquitectura do sistema é modular e explicitamente dividida por camadas, permitindo uma fácil integração vertical ou horizontal com outros sistemas que se possam desenvolver. Permite também a alteração de um componente sem comprometer os restantes componentes, e uma vez que a arquitectura se encontra bem delineada, a alteração de código torna-se intuitiva. Houve também uma forte preocupação na criação de pontos de extensibilidade, por exemplo, o uso de Interfaces e de *Generics* [62], permitindo uma rápida, sólida e homogenia implementação de novas funcionalidades.

A arquitectura de *plugins* permite o *deploy* de novos *plugins* mesmo após a aplicação ter sido instalada e sem a aplicação ter tido conhecimento prévio (durante a compilação) da existência desses *pugins* específicos. O código base de classes abstractas e interfaces permite uma rápida e intuitiva criação de novos *plugins*.

Este estágio atravessou várias dificuldades de negócio, por exemplo, quando não foi possível angariar um cliente desejado, resultando na introdução de um novo requisito a meio de uma *sprint*, alteração muito significativa uma vez que deu origem ao módulo de Entidades Dinâmicas; e de indefinição de objectivos concretos, resultando num estudo e análise do estado da arte iniciais do módulo de Raciocínio Baseado em Casos, sendo mais tarde decidido que seria implementado fora do âmbito deste estágio.

Foram também ultrapassadas grandes dificuldades tecnológicas, por exemplo, o risco do WF Rules, a base do módulo de Raciocínio Baseado em Regras, não

permitir a implementação das funcionalidades, esteve muito próximo de ocorrer, e o módulo de Entidades Dinâmicas trouxe preocupações acrescidas a nível estrutural e de desempenho, sendo necessária uma análise mais exaustiva e o uso do *profiler do Microsoft Visual Studio*.

Durante a análise prévia realizada ao WF Rules, necessária exactamente para prevenir a ocorrência de problemas durante a implementação, o produto foi testado a nível de desempenho, da capacidade de codificação de regras e a nível funcional. Foi então descoberto que o WF Rules não permite a codificação (e compilação) de regras de outra forma senão através da janela de edição de regras embutida na plataforma Microsoft .NET, o que inviabilizaria a sua utilização em ambiente *Web*. Felizmente uma análise mais exaustiva e avançada através da descompilação de código permitiu perceber que seria possível resolver o problema através da chamada de quatro métodos *internal*.

A arquitectura do módulo de Entidades Dinâmicas é simultaneamente surpreendente, simples e funcional. Qualquer pessoa com conhecimentos de XML e com o manual técnico do módulo consegue configurar novas entidades, formulários, listas, e sem se aperceber, toda a estrutura da base de dados necessária para o efeito, tudo em *runtime* e validado ao detalhe através de XSD. No entanto, onde este módulo sem dúvida que mais surpreendeu foi ao nível do desempenho, ultrapassando a conceituada Entity Framework, plataforma ORM que ajuda no desenvolvimento de aplicações mas não permite a alteração da estrutura da base de dados em *runtime*. Este módulo, como qualquer outro, pode ser facilmente reutilizado noutras aplicações, mesmo que não sejam da área da saúde.

A realização dos testes foi bastante enriquecedora – foi utilizada a automação de testes do Microsoft Visual Studio. A criação dos testes unitários é bastante intuitiva e poderosa e a criação dos testes de validação *web* possibilita uma gravação detalhada do teste a reproduzir e, se desejado, uma configuração do teste para adicionar mais regras específicas de validação.

Esperava-se que a *framework* de desenvolvimento Scrum fosse stressante e difícil de aplicar, no entanto, apesar de um pouco stressante, provocou um sentido de responsabilidade pelo cumprimento dos objectivos delineados ao mesmo tempo que se demonstrou flexível e recompensadora a nível de mérito. A participação da equipa de desenvolvimento (*scrum master* e *scrum team*) nas reuniões de planeamento da *sprint* tornam o processo de desenvolvimento mais humano, compreensivo, flexível e menos burocrático. A *sprint* de Maio foi alterada a meio do desenvolvimento, mas

todas as partes tiveram que concordar e assumir as possíveis consequências, de uma forma muito compreensiva. O *burndown chart* revelou-se uma excelente ferramenta para toda a equipa de desenvolvimento e de gestão para o *scrum master*. Foi uma experiência gratificante e enriquecedora participar nesta *framework* de desenvolvimento ágil.

Apesar das diversas dificuldades, foi possível cumprir as *sprints* planeadas com um esforço adicional. A *framework* de desenvolvimento Scrum permitiu um controlo mais fino e realístico sobre as tarefas em desenvolvimento através dos *burndown charts* ao mesmo tempo que permitiu ao cliente ter uma maior flexibilidade das funcionalidades a implementar. Este controlo e sobriedade do processo de desenvolvimento permitiu que o desenvolvimento do SADAC apesar de um pouco atribulado tenha sido um sucesso até ao momento.

11. Bibliografia

[1] A. Romeira, “Brincar Cientificamente,” *PME NEWS,* vol. XI, 2009.

[2] Direcção Geral da Saúde, “Saúde 24,” 2007. [Online]. Available: http://www.saude24.pt/. [Acedido em 17 Janeiro 2012].

[3] G. Wiseman, “Real-World Rule Engines,” *InfoQ,* 2006.

[4] W3Counter, “W3 Counter - Global Web Stats,” 30 Novembro 2011. [Online]. Available: http://www.w3counter.com/globalstats.php?year=2011&month=11. [Acedido em 23 Dezembro 2011].

[5] Microsoft Corporation, “Chapter 5: Layered Application Guidelines,” 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/ee658109.aspx. [Acedido em 20 Fevereiro 2012].

[6] Microsoft Corporation, “Chapter 3: Architectural Patterns and Styles,” 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/ee658117.aspx. [Acedido em 20 Fevereiro 2012].

[7] K. Marquardt, “Patterns for Plug-Ins,” 1999.

[8] Microsoft Corporation, “System Connections,” 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/ff647963. [Acedido em 20 Fevereiro 2012].

[9] Microsoft Corporation, “Presentation Integration,” 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/ff649847.aspx. [Acedido em 20 Fevereiro 2012].

[10] Microsoft Corporation, Microsoft Dynamics CRM 2011 for Independent Software Venders, 2010.

[11] J. J. Townsend, Microsoft Dynamics CRM and SharePoint Platforms, Washington, DC: Information Strategies, 2008, p. 21.

[12] S. Parsons, M. Kubat e M. Dohnal, A Robust Logic for Rule-Based Reasoning Under Uncertainty, The Hague, 1992.

[13] Wikipedia, “Semantic Reasoner,” 23 Dezembro 2011. [Online]. Available: http://en.wikipedia.org/wiki/Rule_engines. [Acedido em 28 Dezembro 2011].

[14] Wikipedia, “Business Rules Engine,” 25 Novembro 2011. [Online]. Available: http://en.wikipedia.org/wiki/Business_rules_engine. [Acedido em 5 Janeiro 2012].

[15] P. Lin, “Forward vs Backward Chaining,” 2006. [Online]. Available: http://legacy.drools.codehaus.org/Forward+Vs+Backward+chaining. [Acedido em 3 Janeiro 2012].

[16] J. Freeman-Hargis, “Methods of Rule-Based Systems,” 23 Novembro 2002. [Online]. Available: http://ai-depot.com/Tutorial/RuleBased-Methods.html. [Acedido em 2 Janeiro 2012].

[17] Wikipedia, “Business Rule Management System,” 3 Janeiro 2012. [Online]. Available: http://en.wikipedia.org/wiki/Business_rule_management_system. [Acedido em 3 Janeiro 2012].

[18] J. Owen, “Business Rules Management Systems,” InfoWorld, 25 Junho 2004. [Online]. Available: http://www.infoworld.com/t/business/business-rules-management-systems-548. [Acedido em 3 Janeiro 2012].

[19] Wikipedia, “Entity–attribute–value model,” [Online]. Available: http://en.wikipedia.org/wiki/Entity%E2%80%93attribute%E2%80%93value_model. [Acedido em 10 Fevereiro 2012].

[20] A. Bertrand, “What is so bad about EAV, anyway?,” 19 Novembro 2009. [Online]. Available: http://sqlblog.com/blogs/aaron_bertrand/archive/2009/11/19/what-is-so-bad-about-eav-anyway.aspx. [Acedido em 10 Fevereiro 2012].

[21] Wikipedia, “XML,” 19 Junho 2012. [Online]. Available: http://pt.wikipedia.org/wiki/XML. [Acedido em 20 Junho 2012].

[22] W3Schools, “DTD - XML Building Blocks,” [Online]. Available: http://www.w3schools.com/dtd/dtd_building.asp. [Acedido em 20 Junho 2012].

[23] W3Schools, “Introduction to XML Schema,” [Online]. Available: http://www.w3schools.com/schema/schema_intro.asp. [Acedido em 20 Junho 2012].

[24] A. Sunyaev, D. Chornyi, C. Mauro e H. Krcmar, “Evaluation Framework for Personal Health Records: Microsoft HealthVault vs. Google Health,” *Proceedings of the 43rd Hawaii International Conference on System Sciences,* 2010.

[25] B. Dolan, “Official: Google Health shuts down because it couldn’t scale adoption,” 24 Junho 2011. [Online]. Available: http://mobihealthnews.com/11453/official-google-health-shuts-down-because-it-couldnt-scale/. [Acedido em 21 Junho 2012].

[26] B. Dolan, “10 Reasons why Google Health failed,” 27 Junho 2011. [Online]. Available: http://mobihealthnews.com/11480/10-reasons-why-google-health-failed/. [Acedido em 18 Junho 2012].

[27] Black Widow Web Design Limited, “Black Widow Web Design,” 2012. [Online]. Available: http://blackwidows.co.uk/resources/access/universal-accessibility.php. [Acedido em 18 Junho 2012].

[28] Wikipedia, “Universal design,” 2 Maio 2012. [Online]. Available: http://en.wikipedia.org/wiki/Universal_design. [Acedido em 18 Junho 2012].

[29] Scrum.org, “What is Scrum?,” 2011. [Online]. Available: http://www.scrum.org/. [Acedido em 10 Janeiro 2012].

[30] Wikipedia, “Scrum (development),” 10 Janeiro 2012. [Online]. Available: http://en.wikipedia.org/wiki/Scrum_(development). [Acedido em 11 Janeiro 2012].

[31] ScrumMethodology.org, “SCRUM Phases,” 2009. [Online]. Available: http://www.scrummethodology.org/scrum-phases.html. [Acedido em 10 Janeiro 2012].

[32] V. Teles, “Scrum,” 26 Agosto 2009. [Online]. Available: http://improveit.com.br/scrum. [Acedido em 9 Janeiro 2012].

[33] Scrum Methodology, “Scrum Meetings,” [Online]. Available: http://scrummethodology.com/scrum-meetings/. [Acedido em 10 Janeiro 2012].

[34] M. Cohn, “Planning Poker in Detail,” 18 Fevereiro 2012. [Online]. Available: http://www.planningpoker.com/detail.html. [Acedido em 10 Janeiro 2012].

[35] IBM, “IBM Rational Unified Process (RUP),” 2007. [Online]. Available: http://www-01.ibm.com/software/awdtools/rup/. [Acedido em 11 Janeiro 2012].

[36] International Organization for Standardization, “Risk Management - Principles and Guidelines on Implementation,” *ISO/DIS 31000,* 2009.

[37] Wikipedia, “Modelo em Cascata,” 4 Dezembro 2011. [Online]. Available: http://pt.wikipedia.org/wiki/Modelo_em_cascata. [Acedido em 11 Janeiro 2012].

[38] W. Heisenberg, “Ueber den anschaulichen Inhalt der quantentheoretischen Kinematik and Mechanik,” *Zeitschrift für Physik,* n.º 43, pp. 172-198, 1927.

[39] C. Young, “BizTalk Server or WF for rules and tracking?,” 13 Julho 2007. [Online]. Available: http://geekswithblogs.net/cyoung/archive/2007/07/19/114061.aspx. [Acedido em 13 Fevereiro 2012].

[40] C. Young, “WF Rules and MS BRE - Comparing Performance,” 12 Agosto 2007. [Online]. Available: http://geekswithblogs.net/cyoung/archive/2007/08/12/114597.aspx. [Acedido em 14 Fevereiro 2012].

[41] B. Crawford, “Hacking into the Windows Workflow Rules Engine,” 14 Março 2009. [Online]. Available: http://beaucrawford.net/post/Hacking-into-the-Windows-Workflow-Rules-Engine.aspx. [Acedido em 19 Fevereiro 2012].

[42] R. Suharta, “Dynamic Business Rules Helper Using Windows Workflow Rules Engine,” 18 Outubro 2011. [Online]. Available: http://rsuharta.wordpress.com/2011/10/18/dynamic-business-rules-helper-using-windows-workflow-rules-engine/. [Acedido em 19 Fevereiro 2012].

[43] Wikipedia, “Speedup,” 2012. [Online]. Available: http://en.wikipedia.org/wiki/Speedup.

[Acedido em 28 Junho 2012].

[44] Microsoft Corporation, "ASP.NET Routing," [Online]. Available: http://msdn.microsoft.com/en-us/library/cc668201.aspx. [Acedido em 10 Abril 2012].

[45] C. Martinez, "ASP .Net Tutorial: SEO Friendly Pages with Metadata and URL Routing," 11 Maio 2011. [Online]. Available: http://www.aspnet-tutorials.com/asp-net-tutorial-seo-friendly-pages-with-metadata-and-url-routing. [Acedido em 21 Março 2012].

[46] S. Mitchell, "Using ASP.NET Routing Without ASP.NET MVC," 13 Maio 2009. [Online]. Available: http://www.4guysfromrolla.com/articles/051309-1.aspx. [Acedido em 23 Março 2012].

[47] Zyphrax, "SiteMap and Url routing," [Online]. Available: http://stackoverflow.com/questions/2165568/sitemap-and-url-routing. [Acedido em 24 Maio 2012].

[48] G. Lower, "SiteMap and Routing: So Happy Together," 20 Dezembro 2010. [Online]. Available: http://forums.asp.net/t/1634943.aspx/1. [Acedido em 24 Maio 2012].

[49] K. Aggarwal e Y. Singh, Software Engineering (2nd Edition), New Age International, 2006.

[50] Wikipedia, "IBM Rational Unified Process," 23 Setembro 2011. [Online]. Available: http://pt.wikipedia.org/wiki/IBM_Rational_Unified_Process. [Acedido em 23 Janeiro 2012].

[51] I. Sommerville, Software Engineering (7th Edition), United Kingdom: Addison-Wesley, 2004.

[52] Wikipedia, "MoSCoW Method," 17 Janeiro 2012. [Online]. Available: http://en.wikipedia.org/wiki/MoSCoW_Method. [Acedido em 23 Janeiro 2012].

[53] S. W. Ambler, "Introduction to User Stories," 2009. [Online]. Available: http://www.agilemodeling.com/artifacts/userStory.htm. [Acedido em 23 Janeiro 2012].

[54] Wikipedia, "User Story," 31 Dezembro 2011. [Online]. Available: http://en.wikipedia.org/wiki/User_story. [Acedido em 23 Janeiro 2012].

[55] P. Onori, "Design for Speed," 21 Fevereiro 2012. [Online]. Available: http://somerandomdude.com/2012/02/21/design-for-speed/. [Acedido em 15 Março 2012].

[56] B. Shneiderman, C. Plaisant, M. Cohen e S. Jacobs, Designing the User Interface: Strategies for Effective Human-Computer Interaction, 5ª ed., Addison Wesley, 2009.

[57] Microsoft Corporation, "Chapter 8: Data Layer Guidelines," [Online]. Available: http://msdn.microsoft.com/en-us/library/ee658127.aspx#PatternMap. [Acedido em 16 Abril 2012].

[58] Microsoft Corporation, "Uncover Security Design Flaws Using The STRIDE Approach," [Online]. Available: http://msdn.microsoft.com/en-us/magazine/cc163519.aspx. [Acedido em 20 Fevereiro 2012].

[59] Microsoft Corporation, "The STRIDE Threat Model," [Online]. Available: http://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx. [Acedido em 20 Fevereiro 2012].

[60] The Open Web Application Security Project, "OWASP Top 10 - 2010," 2010. [Online]. Available: https://www.owasp.org/index.php/Main_Page. [Acedido em 10 Fevereiro 2012].

[61] Microsoft Corporation, "How to: Create a Test Project," [Online]. Available: http://msdn.microsoft.com/en-us/library/ms182413(v=vs.90).aspx. [Acedido em 20 Fevereiro 2012].

[62] Microsoft Corporation, "An Introduction to C# Generics," [Online]. Available: http://msdn.microsoft.com/en-us/library/ms379564(v=vs.80).aspx. [Acedido em 8 Julho 2012].

12. Anexos

Segue-se a lista dos documentos anexados a este relatório de estágio.

1. Análise de Integração de Sistemas

O documento resume a investigação das várias metodologias comuns para a integração entre sistemas. Existem essencialmente três formas de integração: Integração a nível da Apresentação; Integração Funcional; Integração de Dados. A integração mais indicada para o SADAC é a integração de dados.

2. Análise de Sistemas com Registos Pessoais de Saúde

O anexo resume o estudo realizado a dois sistemas que lidam com registos pessoais de saúde, o Google Health e o Microsoft HealthVault, com o objectivo de recolher as suas principais funcionalidades, identificar as principais forças e fraquezas.

3. Análise de Sistemas de Gestão de Regras de Negócio

Este documento descreve o processo de análise aos motores de regras de negócio identificados (33). O processo de análise envolve uma criteriosa exclusão, detalhe e comparação de motores de regras.

4. DynamicEntity - Manual Técnico

A DynamicEntity, ou módulo de Entidades Dinâmicas, é um componente que necessita de um forte apoio ao programador e ao utilizador final. Permite criar entidades dinâmicas mesmo após o *deploy* da aplicação, definidas pelo utilizador final. De momento ainda não foi desenvolvido um editor de entidades dinâmicas, por esse motivo o ainda é necessário especificar as entidades através de XML. Este manual ajuda nessa especificação de entidades através de XML e ajuda na utilização do módulo através de código.

5. Enquadramento do Estudo do Estado da Arte

Este documento desenvolve de forma mais exaustiva o enquadramento do estado da arte explorando os vários conceitos técnicos e metodologias envolvidas.

O documento começa por dar um enquadramento do conceito técnico de Raciocínio Baseado em Regras, dentro da área de Inteligência Artificial. Este conceito leva à necessidade de definir o que são Regras de Negócio e respectivas categorias.

O processamento de regras de negócio é essencialmente o objectivo de um Motor de Regras. São discutidas várias metodologias de execução dos motores de regras (*forward-chaining*, *backward-chaining* e híbridas) e optimizações por parte do

processo de inferência (Rete I, Rete II, Rete III, Treat, Rete* e Leaps). Para facilitar a integração dos motores de regras com outros sistemas, são discutidas várias normas que alguns motores de regras implementam.

Por fim são descritas algumas limitações dos sistemas baseados em regras.

6. Levantamento e Análise de Requisitos

O documento "*Levantamento e Análise de Requisitos*" detalha as metodologias utilizadas na recolha, análise e descrição dos requisitos da aplicação SADAC e dos seus módulos constituintes.

7. Sprints de Implementação

O anexo "*Sprints de Implementação*" detalha as tarefas realizadas nas *sprints* durante a fase de implementação do projecto SADAC e respectivos *burndown charts*. Detalha também as derrapagens ocorridas, motivos e procedimentos adoptados para recuperar as derrapagens.

8. Testes Não Funcionais

O documento "*Testes Não Funcionais*" apresenta os dados puros, com respectivos desvios padrões, dos testes não funcionais realizados aos módulos de Raciocínio Baseado em Regras (RBR) e de Entidades Dinâmicas. Apresenta também todos os gráficos resultantes dos dados.