

Faculdade de Engenharia da Universidade do Porto



Simulador de casos clínicos do Alert® Paper Free Hospital

João Carlos Martins Eiras

Relatório de Projecto realizado no âmbito do
Mestrado Integrado em Engenharia Informática e de Computação

Orientador: Eng. Gil Gonçalves

Julho de 2008

NOTA: Nos termos do protocolo de projecto e do acordo de confidencialidade celebrado com a ALERT Life Sciences Computing, S.A. ("ALERT"), o presente relatório é confidencial e poderá conter referências a invenções, know-how, desenhos, programas de computador, segredos comerciais, produtos, fórmulas, métodos, planos, especificações, projectos, dados ou obras abrangidos por direitos de propriedade industrial e ou intelectual da ALERT. Este relatório só poderá ser utilizado para efeitos de investigação e de ensino. Qualquer outro tipo de utilização está sujeita a autorização prévia e por escrito da ALERT.

NOTE: In accordance with the terms of the project protocol and the confidentiality agreement signed with Alert Life Sciences Computing, S.A. ("ALERT"), this report is confidential and may contain references to inventions, know-how, drawings, computer software, trade secrets, products, formulas, methods, plans, specifications, projects, data or works protected by ALERT's industrial and/or intellectual property rights. This report may be used solely for research and educational purposes. Any other kind of use requires prior written consent from ALERT.

Simulador de casos clínicos do Alert® Paper Free Hospital

João Carlos Martins Eiras

Relatório de Projecto realizado no âmbito do
Mestrado Integrado em Engenharia Informática e de Computação

Aprovado em provas publicas pelo Júri:

Presidente: nome do presidente do júri (Título):

Arguente: nome do arguente do júri (Título):

Vogal: nome do vogal do júri (Título):

_____ de _____ de 2008

Resumo

Devido às necessidades comerciais, e graças ao crescimento rápido da instituição onde se realizou este projecto, ALERT Life Sciences Computing S.A. (ALERT), decidiu-se criar uma equipa para fazer manutenção dos ambientes de demonstração comercial e formação interna do Alert® Paper Free Hospital. Previamente não havia nenhuma equipa dedicada.

O projecto consistiu-se em duas partes:

- Manutenção de uma instalação do Alert®, usada para demos comerciais e formações internas na empresa;
- Elaboração duma aplicação para fazer gestão e geração de dados numa instalação do Alert® usada para demonstrações comerciais.

O ambiente de demonstrações anteriormente era mantido pela equipa de desenvolvimento, sem haver uma visão unificada das necessidades, problemas e objectivos deste. As intervenções na estrutura tecnológica neste ambiente por vezes deixavam-no instável. Não havia um conjunto de conteúdo clínico disponível previamente acordado entre os diferentes interessados. Para gerar episódios de pacientes, necessários para demonstrar o Alert®, recorria-se a uma aplicação que tinha toda a lógica *hardcoded*, e que ao limpar dados prévios não cobria todos os casos possíveis na base de dados, pois nunca foi devidamente mantida. Estes problemas são parte do que se pretende resolver no contexto deste projecto. Note-se que estes problemas existiam, pois anteriormente não haviam recursos disponíveis anteriormente para lidar com eles.

No contexto deste projecto, iniciou-se uma série de trabalhos, que irão continuar após a conclusão do mesmo. Estes trabalhos terão implicações na organização e manutenção do ambiente de demonstrações. Definiu-se uma organização para as intervenções a realizar sobre este ambiente, como correcções pontuais de problemas, parametrização de instituições de demos e conteúdo clínico, gestão de traduções, definição dos vários intervenientes, *workflows*, etc. Simultaneamente, desenvolveu-se uma nova aplicação que irá permitir ao indivíduo que usa o ambiente de demonstrações, gerir pacientes e episódios, e gerar informação para demonstrações posteriores, pois para o Alert® poder ser demonstrado, este necessita de pacientes nas grelhas com tratamento activo.

Embora o trabalho iniciado requeira mais do que o período de tempo alocado para este projecto, parte do que foi produzido e planeado, foi aceite pelos interessados com bastante expectativa. O trabalho está longe de estar concluído, mas já se produziram algum entregáveis de relevância: especificação de requisitos, o protótipo e a especificação técnica da nova aplicação para fazer gestão e geração de dados no Alert®, e a elaboração de propostas para gestão do ambiente de demonstrações, e gestão de traduções.

Abstract

Due to the commercial needs, and the rapid growth of the institution where this project was conducted, ALERT Life Sciences Computing S.A. (ALERT), it was decided to create a dedicated team to maintain Alert® Paper Free Hospital's commercial demonstration and internal formation environments.

This project consists of two parts:

- Maintenance of an Alert®'s installation used for commercial demonstrations and employee formations inside the company;
- Development of an application to manage and generate data in an Alert® installation used for demonstrations.

The demonstrations environment was maintained by the development team, without the existence of an unified vision of the needs, problems and objectives of it. Interventions in the technological infrastructure of this environment sometimes left it unstable. There was not a set of clinical content available previously accorded between the several parties. To generate episodes, to demonstrate Alert®, there was a small application which had all its business logic hardcoded, and when clearing previous data it didn't cover all the different cases in the data base, because it was never thoroughly developed nor kept. These problems are just a glimpse of what there's to solve in the context of this project. Do note that the problems existed because before there weren't resources available to deal with them.

In the context of this project, a series of tasks were started that will go on after its deadline. These tasks will have impact in the organization and maintenance of the demonstration environment. An organization was defined for how interventions are done in this environment, like bug fixing, demo institutions and clinical content parameterization, definition of the several user groups, workflows, etc. Simultaneously, a new application to manage data in Alert® was developed. This application will allow an individual, which uses the demonstrations environment, to manage patients and episodes, and generate information for further demonstrations, because for Alert® PFH to be demonstrated, it needs patients in the grids with active treatment.

Although the work in course requires more than the time frame allocated for this project, part of what was proposed and planned, was well accepted by the stakeholder with many future expectations. The work is far from complete, but some relevant deliverables are already made: requirements specification, prototype, and technical specification for the new application to manage and generate data in Alert®, proposals and documentation for the management of the demo environment and translations.

Agradecimentos

Agradeço à ALERT, em especial, ao Dr. Manuel Jorge Guimarães, e à equipa de RH, em especial a Maria João Oliveira, por permitirem a realização deste projecto.

Agradeço à Faculdade de Engenharia da Universidade do Porto (FEUP), pelas óptimas condições que disponibilizou ao longo do curso, para minha formação, e em especial ao professor Raul Vidal, pela contribuição enquanto director da LEIC/MIEIC, e ao professor Augusto Sousa, director actual do MIEIC.

Agradeço ao orientador da FEUP, o professor Gil Gonçalves, pelo acompanhamento e aconselhamento, e orientação.

Agradeço aos elementos da minha equipa pelo trabalho realizado em conjunto e apoio, Rita Malheiro, Sérgio Cunha e Hugo Campos.

Índice

1.	Introdução.....	1
1.1.	Introdução.....	1
1.2.	Enquadramento.....	1
1.3.	Descrição do projecto	3
1.4.	Motivação e objectivos.....	4
1.5.	Trabalho realizado.....	5
1.6.	Organização e Temas Abordados no Presente Relatório	7
2.	Revisão bibliográfica.....	8
2.1.	Estado da arte.....	8
2.1.1.	Introdução teórica.....	8
2.1.2.	Tecnologias.....	11
2.2.	Revisão tecnológica.....	14
2.2.1.	Ferramentas.....	14
2.2.2.	Ferramentas utilizadas	19
2.3.	Observações.....	23
3.	Ambiente de demonstrações do Alert®	24
3.1.	Contexto.....	24
3.1.1.	Organização do ambiente	24
3.1.2.	Aplicação antiga de reset.....	25
3.1.3.	Observações	27
3.2.	Soluções propostas.....	28
3.2.1.	Nova aplicação de reset.....	28
3.2.2.	Organização do ambiente de demonstração	30
3.2.3.	Problemas de gestão de traduções.....	34
4.	Especificação de requisitos do Alert® Reset.....	35
4.1.	Funcionalidades Gerais	35
4.2.	Características dos utilizadores finais	35
4.2.1.	Equipas comerciais e de demonstrações	35
4.2.2.	Equipas de formação e <i>key-users</i>	36
4.2.3.	Observações	36
4.3.	Actores.....	36
4.4.	Requisitos não-funcionais	37
4.4.1.	Fiabilidade.....	37
4.4.2.	Eficiência.	37
4.5.	Requisitos funcionais.....	38
4.5.1.	Visão geral	38
4.5.2.	Requisitos do módulo “Gestão de dados“.....	40
4.5.3.	Requisitos do módulo “Gestão de utilizadores”.....	43
4.5.4.	Requisitos do módulo “Geração de dados”.....	44
4.6.	Observações.....	47
5.	Desenvolvimento	48
5.1.	Arquitectura lógica, integração com o Alert®	48
5.2.	Arquitectura física	49
5.3.	Camada de interface de utilizador.....	50

5.4.	Camada de servidor aplicacional	55
5.4.1.	Arquitetura Aplicacional.....	55
5.4.2.	Autenticação	55
5.4.3.	Acesso a dados	55
5.5.	Camada de base de dados.....	56
5.6.	Modelo de dados	58
5.6.1.	Visão geral.....	58
5.6.2.	Diagrama do modelo de dados.....	59
5.6.3.	<i>Workflows</i> de geração de dados	60
5.6.4.	Tabelas.....	63
5.7.	Observações	68
6.	Conclusões e perspectivas de trabalho futuro.....	69
7.	Referências.....	71
8.	Bibliografia.....	72
9.	Anexo A – Apresentação da empresa	74
9.1.	A Alert Life Sciences Computing, S.A.	74
9.2.	O Alert ® Paper Free Hospital	77
9.3.	Metodologia de desenvolvimento	79
10.	Anexo B - Glossário	82
11.	Anexo C – proposta para gestão de traduções	83
11.1.	Problemas de gestão de traduções	83
11.2.	Procedimentos a adoptar.....	84
11.3.	Utilizadores finais	84
11.4.	Requisitos gerais	85
11.5.	Questões técnicas	88
11.6.	Observações	89

1. Introdução

1.1. Introdução

Este relatório documenta o projecto de Mestrado Integrado denominado “Simulador de casos clínicos no Alert® Paper Free Hospital”, realizado 2º semestre do ano lectivo de 2007/08, desde Fevereiro até Julho, ao longo prazo de 20 semanas, no âmbito do curso Mestrado Integrado em Engenharia informática de Computação.

O projecto foi realizado na empresa ALERT Life Sciences Computing S.A, líder no mercado português no segmento do software clínico. A empresa dedica-se à concepção de sistemas de informação para informatizar completamente o processo clínico dos pacientes, inicialmente em ambiente hospitalar, e agora em ambiente *online*. O seu principal produto é o Alert® Paper Free Hospital, também denominado Alert® PFH.

O título deste projecto foi sugerido pelo presidente da ALERT Life Sciences Computing S.A, Dr. Jorge Guimarães, durante as primeiras discussões sobre o trabalho que era necessário realizar. O título descreve bastante bem o ambiente de trabalho e o que se pretende produzir.

1.2. Enquadramento

O Alert® actualmente está instalado em dezenas de hospitais portugueses e centenas de centros de saúde, e prevêem-se para 2008 vários contratos noutros países para além de Portugal. Conta com dezenas de milhares de utilizadores, provavelmente centenas de milhar no futuro próximo. Como se destina ao sector da saúde, pode-se dizer que o Alert® é um software crítico, pois o desempenho dum serviço depende bastante do sistema de informação que usa, e os serviços de saúde lidam com a vida humana. O Alert® PFH abrange já um conjunto vastíssimo de casos de uso, e permite informatizar completamente o processo clínico nas instituições de saúde, sem necessidade de manter registos em papel. Uma das principais capacidades do Alert®, que contribui para o seu sucesso, é o empenho da empresa ALERT em torná-lo interoperável, e a interoperabilidade daí obtida. O Alert® é interoperável com sistemas de informação de laboratórios, sistemas

de admissão e gestão de pacientes, dispositivos auxiliares, etc. A interoperabilidade é conseguida através da camada INTER ALERT¹, que usa normas como HL7¹. A cada versão, o Alert® cresce em tamanho e complexidade.

A ALERT tem duplicado o número de colaboradores anualmente. No início de Junho de 2008, contava com cerca de 500 colaboradores, em várias áreas. Em virtude do crescimento, finalmente houve disponibilidade para criar uma equipa que lidasse com a gestão do ambiente de demonstrações comerciais a tempo inteiro.

Após esta descrição, pode-se ter a percepção da complexidade e vastidão do Alert®

Como qualquer empresa comercial com a sua presença no mercado, a ALERT tem necessidades de vender o seu produto. Uma das táticas usadas para atrair potenciais compradores, é através de demonstrações comerciais. As demonstrações podem tomar diferentes formatos, serem realizadas em diferentes países, em diferentes línguas, ter um guião pré definido, ou serem realizadas *ad libitum*. As demonstrações tipicamente incluem a demonstração directa da utilização do Alert®, para simular casos reais. Por vezes, os espectadores vêem o Alert® pela primeira vez. A equipa de demonstrações trabalha num contexto internacional, multi-cultural.

A empresa ALERT orgulha-se de ter o Alert® traduzido em várias línguas, o que representa uma vantagem competitiva. Actualmente, o Alert® está parcial ou totalmente traduzido em 8 línguas: português, inglês, espanhol, holandês, italiano, francês, eslovaco e português brasileiro.

As demonstrações são realizadas em exposições, em instituições de saúde para clientes, ou remotamente a partir das instalações da empresa. Por exemplo, em 2007, a equipa de demonstrações participou em 29 exposições, sendo realizadas aproximadamente 780 demonstrações só nestas exposições. Realizaram-se 28 demonstrações em instituições portuguesas, 45 em instituições noutros países, 28 demonstrações nos escritórios da ALERT, durante visitas, e 25 demonstrações *online*, através de acesso remoto.² O sucesso do trabalho da equipa de demonstrações tem um impacto claro no sucesso da empresa. Graças à presença da ALERT em exposições internacionais, a ALERT recrutou recursos humanos estratégicos noutros países, conseguiu contratos internacionais, e inaugurou as respectivas subsidiárias, como por exemplo em Singapura e na Holanda.

Durante uma implementação do Alert® num cliente, actualmente são criados dois ambientes: pré-produção e produção. O ambiente de pré-produção é uma instância do Alert® para testar e validação a instalação do Alert®, actualizações de versão, configurações etc. Isto é um passo fulcral para garantir a estabilidade e qualidade da

¹ HL7 (*Health Level 7*) – organização não lucrativa voluntária envolvida no desenvolvimento de normas internacionais usadas em sistemas de informação de saúde. A designação HL7 é habitualmente atribuída às especificações criadas pela organização.

² Esta informação, embora não disponível publicamente, foi apresentada aos colaboradores da ALERT em Janeiro de 2008, no retiro anual da empresa.

aplicação final, no ambiente de produção. O ambiente de produção é a instância do Alert® usada pelos profissionais de saúde para a prossecução das suas tarefas diárias.

Durante a implementação do Alert® num cliente, os utilizadores finais do Alert®, ou seja, os profissionais e auxiliares de saúde, os administrativos e técnicos, recebem formação em Alert®. As formações adquirem contornos muito específicos, e tentam representar fielmente casos reais, adequados à realidade dos utilizadores alvo da formação. A formação é feita no ambiente de pré-produção.

Uma vez que há a necessidade de demonstrações cumulativas do Alert®, os objectivos iniciais deste projecto eram a construção de uma ferramenta que permitisse a limpeza de dados do Alert®, e repor nova informação utilizável para a formação ou demonstração seguinte. Previamente existia uma aplicação que fazia este trabalho, mas que tinha vários problemas, que serão enunciados adiante. A nova ferramenta será chamada de Alert® Reset, pois herda o nome da sua primeira versão.

A ALERT actualmente tem enormíssimas responsabilidades para com os seus clientes, e está a desenvolver uma marca forte. Por isso, como política da empresa, todo o trabalho realizado tem que ter os respectivos procedimentos, *checklists*, e documentação resultante, como forma de preservar conhecimento, aumentar o sentido de responsabilidade, permitir a auditoria, e controlar a qualidade. Este é um facto importantíssimo que irá coabitar com o desenrolar do projecto, e que terá que ser reflectido nos resultados obtidos. Será especialmente importante na resolução de problemas organizacionais que serão expostos mais à frente.

1.3. Descrição do projecto

Pretende-se com este projecto desenvolver uma ferramenta que elimine e crie dados para demonstrações e formações do Alert®, e otimizar o ambiente de demonstrações comerciais. A Alert® Reset será usada pelas equipas de demonstrações e formação para preparar o seu trabalho, de forma autónoma.

O sujeito de trabalho será o ambiente de demonstrações do Alert® PFH. Na sequência da criação de uma equipa de reset, esta ficará com as seguintes responsabilidades:

- Concepção da nova ferramenta para preparar dados para demonstrações comerciais e formações do Alert®;
- Organização do ambiente de demonstrações comerciais e formação interna.

A aplicação de Reset que se quer desenvolver será a segunda versão. A primeira versão, embora tenha servido para a sua tarefa básica, tem problemas que se pretendem resolver. A nova aplicação terá que ter em conta todos os casos de uso dos utilizadores finais, e terá que obedecer a normas de usabilidade e acessibilidade, uma vez que será usada por vários indivíduos na empresa mãe e afiliadas, com diferentes competências.

O ambiente de demonstrações padece de problemas que limita a acção das equipas comerciais. Um dos problemas é a falta de uma aplicação de reset adequada. Os outros

problemas são instabilidade, qualidade de dados, espaço em disco, actualizações de versão problemáticas, falta de traduções ou traduções incorrectas.

Pretende-se resolver um conjunto de problemas actuais e otimizar a forma de trabalhar sobre o ambiente de demonstrações.

Pretende-se concretamente com este projecto:

- Tornar o ambiente de demonstrações estável, fiável, e com capacidade para se adaptar a novos mercados, gerir o ambiente de demos como um cliente a partir de uma cópia estável;
- Colaborar com várias equipas para definir o conteúdo suportado pela aplicação, definir os procedimentos associados aos casos de uso para este ambiente, para alinhar às políticas da empresa, em que todo trabalho deve ter um procedimento associado;
- Incluir na base de dados conteúdos adequados para as diferentes demonstrações, desde conteúdo clínico e aplicacional, a informação operacional, como episódios e histórico dos pacientes;
- Elaborar de uma proposta para gestão de traduções;
- Construir de uma nova aplicação de reset, com o objectivo de ser mais fiável, mais eficiente e mais flexível. Tem que garantir a independência do utilizador.

1.4. Motivação e objectivos

Com este projecto espera-se desenvolver um trabalho multidisciplinar, interagir com diferentes equipas de diferentes áreas de conhecimento, tomar contacto com situações mais reais, obter autonomia nas decisões e acções, e poder contribuir para uma área de bastante relevância de qualquer empresa comercial.

A nova aplicação de reset terá como objectivo gerir dados numa instalação do Alert® PFH, duma forma que não é possível para um cliente (instituição) ou utilizador final da aplicação. Os principais objectivos desta são:

- Garantir a independência total das equipas comerciais, demos e formação, ao suportar os requisitos que serão enunciados neste documento;
- Desenvolver uma aplicação flexível e extensível para gradualmente e facilmente incluir novos casos de uso e funcionalidades;
- Garantir privilégios ilimitados para manipular a informação presente no Alert®;
- Desenvolver uma aplicação de fácil utilização e eficiente, pois terá um conjunto de utilizadores alargado, de diferentes países e culturas.

1.5. Trabalho realizado

O plano de acção adoptado foi elaborado durante este projecto, e à semelhança dos objectivos consistiu em:

- Organização do ambiente de demonstrações e formação interna, análise dos casos de uso, descrição de *workflows* e procedimentos, resolução de problemas na camada tecnológica, e gestão do processo de traduções;
- Conceptualização duma aplicação que permita gerir informação, e gerar dados numa instalação do Alert® PFH adjacente, cujos requisitos foram avaliados através de reuniões com os *stakeholders*, análise da situação actual, e previsão de situações futuras.

Relativamente à organização do ambiente, reestruturou-se a interacção entre as equipas intervenientes no ambiente de demos, quais as tarefas e responsabilidades, como interagem, quais os *workflows* para os diferentes casos de uso, etc. Sem este trabalho, o Alert® Reset traria pouco benefício. Este trabalho foi realizado na sequência dos esforços que têm sido criados na empresa para aumentar o rigor do trabalho efectuado em todas as áreas, e esta área, ambiente de demonstrações, também necessitou desse escrutínio.

A nova versão do Alert® Reset foi um produto de *brainstorming*. Após duas reuniões com elementos da equipa de demonstrações e formação, criou-se uma lista de funcionalidades que a aplicação deveria suportar. Esta lista de funcionalidades era muito específica e cobria principalmente as necessidades da equipa de demonstrações. Esses requisitos foram harmonizadas para incluir mais casos de uso e permitir uma maior flexibilidade de utilização da aplicação final.

Mais tarde, após uma discussão com membros da equipa de formação e acompanhamento presencial dos clientes, concluiu-se que os casos de uso desta equipa, embora bastante mais rigorosos, pois neste caso o Alert® é usado directamente por clínicos, coincidiam parcialmente com os pretendidos pela equipa de demonstrações, e que a harmonização feita contribuiu para satisfação dos requisitos de ambas equipas.

Com esta informação foi elaborado um documento de requisitos, que foi devidamente validado pela responsável da equipa de reset. Os requisitos serão descritos à frente neste documento. Após a conclusão do documento de requisitos, iniciou-se a produção do documento de arquitectura técnica. O documento descreve a arquitectura da aplicação, física e lógica, e documenta pormenores de implementação, com especial ênfase para a base de dados. Com a especificação de requisitos e especificação técnica, desenvolveu-se um protótipo do Alert® Reset que demonstra os casos de uso mais importantes.

O seguinte diagrama esboça o planeamento inicial definido para o projecto. O planeamento foi cumprido salvo alguns atrasos típicos.

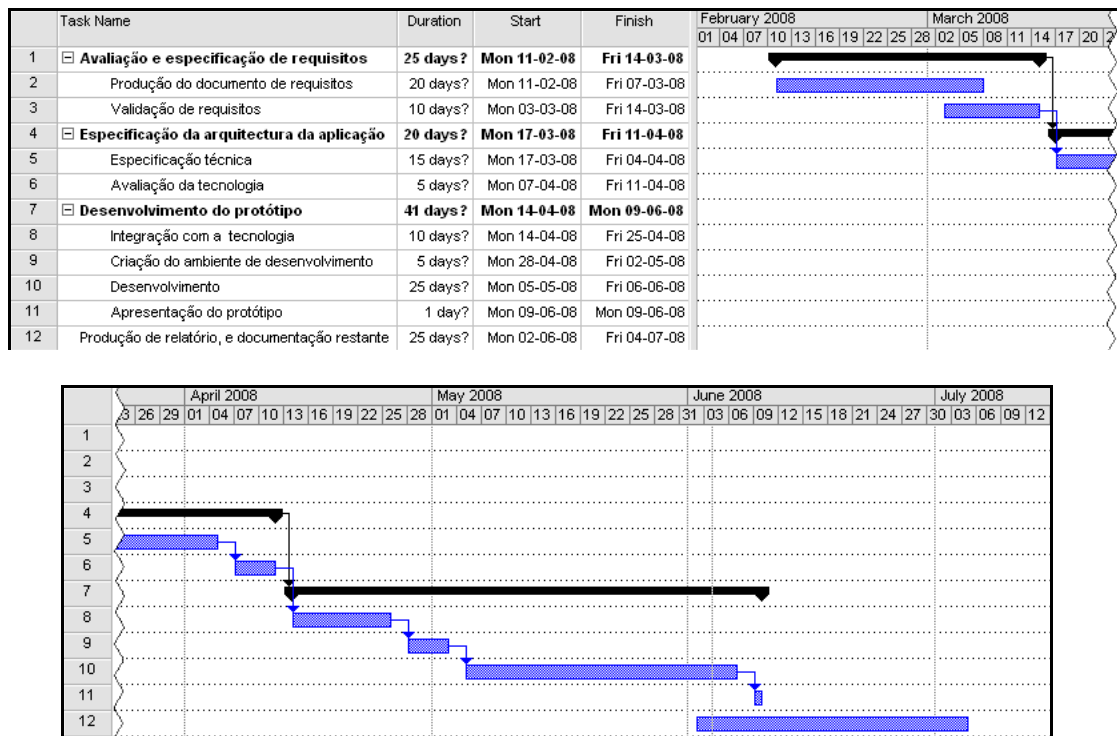


Fig. 1 – Diagrama com planeamento inicial para o desenvolvimento do Alert® Reset

O planeamento era composto pelas seguintes tarefas:

- Produção do documento de requisitos;
- Validação dos requisitos – durante este período de tempo o documento de requisitos foi enviado para elementos chave na empresa para poderem opinar sobre o documento e fornecerem o seu *feedback*. O documento foi também entregue à representante da empresa neste projecto. Em consequência, o documento sofreu ligeiras alterações;
- Especificação técnica – durante este período de tempo, produziu-se o documento de especificação técnica;
- Avaliação da tecnologia – durante este período de tempo avaliou-se de que forma as tecnologias escolhidas iriam condicionar o desenvolvimento, e que dependências teriam no documento de especificação técnica;
- Integração com a tecnologia – durante este período pretendia-se estudar as tecnologias menos familiares para se proceder ao desenvolvimento posterior;
- Criação do ambiente de desenvolvimento – *setup* do controlo de versões, preparação de dependências externas (como bibliotecas de terceiros), criar *framework* básica de suporte à infra-estrutura do protótipo.
- Desenvolvimento do protótipo;
- Apresentação do protótipo, perante os indivíduos da empresa interessados neste projecto, para reunir *feedback* e algumas correcções;

- Produção do relatório de projecto e dos outros entregáveis.

O trabalho na organização do ambiente e aplicação Alert® Reset foi intercalado e distribuído durante o tempo alocado ao projecto.

1.6. Organização e Temas Abordados no Presente Relatório

Este documento está estruturado da seguinte forma:

- No capítulo 1 temos a presente introdução, que deverá fornecer ao leitor uma boa base de compreensão sobre o projecto, e que lhe permitirá escolher quais os assuntos que querará saber em mais detalhe;
- No capítulo 2 será efectuado uma análise do actual estado da arte. O problema a analisar será relativo ao Alert®Reset, mais especificamente, à funcionalidade de *workflows*. Como este projecto inicialmente era constituído principalmente pela concepção da aplicação de reset, havia o problema de conseguir garantir a independência do utilizador. E daí surgiu a ideia de lhe dar as ferramentas necessárias para este criar a informação que deseje, com uma mecanismo genérico de *workflows*;
- No capítulo 3 há uma análise detalhada do problema: problemas gerais com a organização do ambiente de demonstrações, problemas com a antiga aplicação de reset, e traduções. No mesmo capítulo documenta-se a solução proposta para a resolução dos problemas e reorganização do ambiente de demonstrações. Não é uma solução técnica, mas um conjunto de políticas e procedimentos a adoptar. E por fim, é delineada a nova aplicação de reset, pois a análise aprofundada desta será feita nos capítulos seguintes;
- No capítulo 4 há uma análise de requisitos do novo Alert® Reset, resultantes de reuniões prévias com alguns dos *stakeholders* desta aplicação, que forneceram um valioso *input*;
- No capítulo 5 descreve-se a especificação técnica do Alert® Reset, como base necessária para o desenvolvimento desta aplicação. Esta especificação inclui a definição das arquitecturas lógica e física, e as bases e regras para o desenvolvimento das três camadas lógicas que irão compor a aplicação;
- No capítulo 6 contem conclusões tiradas deste projecto, e algumas previsões sobre o trabalho que se avizinha;
- Este relatório termina com a bibliografia no capítulo 7, e anexos de seguida.

2. Revisão bibliográfica

Este capítulo sobre estado da arte aborda uma das questões mais complexas e centrais à aplicação de reset., que é o suporte modular, editável e configurável para *workflows*, que serão usados para gerar dados. Esta questão será também aplicável à organização do ambiente de demonstrações: como descrever a interacção entre equipas.

O Alert® PFH é um software vasto. Contêm bastantes funcionalidades que dependem da passagem de informação entre os diferentes actores, de forma organizada. Esta passagem de informação entre actores implica *workflows*.

Actualmente o Alert® têm todos os *workflows* suportados na sua lógica de negócio, de forma pouco customizável, e com configurações limitadas, pois estes foram desenvolvidos e adaptados para os casos de uso dos clientes actuais. Mas, foi proposta a elaboração de um motor de *workflows*, e a respectiva modularização da lógica de negócio para conseguir adaptar o Alert® a qualquer cliente, a qualquer caso de uso, sem novos esforços de desenvolvimento.

2.1. Estado da arte

2.1.1. Introdução teórica

Muito do software actualmente usado em empresas tem um objectivo claro: suportar processos de negócio. Alguns destes processos são automáticos, enquanto outros, provavelmente a maioria, necessitam da intervenção humana. Estes processos podem ser modelados em passos discretos para compor um *workflow*, sobre os quais podem ser construídas aplicações que os suportem.

Um *workflow* é a descrição de uma sequência de operações atómicas, que são executadas ordenadamente para obter um dado resultado. Um *workflow* pode descrever acções humanas ou automáticas, pode ser uma representação fiel dum sistema, ou uma representação abstraída, virtual. É um modelo que pode ser documentado e aprendido. Surgiu inicialmente para a modelação de processos, e entretanto este conceito alastrou-se a diversas áreas de actividade.

Em software, um programa de *workflows* permite ao utilizador desenvolver um modelo de processamento de dados complexo, de forma acessível e simplificada, abstraindo pormenores de estrutura tecnológica e programação. Isto permite uma maior flexibilidade do utilizador final, pois quer este seja técnico ou não, consegue facilmente

criar os seus *workflows*, e de uma forma ou outra influenciar o comportamento de um sistema.

Um *workflow* é constituído por:

- Dados de entrada;
- Tarefas, ou unidades lógicas de processamento, que transformam dados de entrada, em dados de saída, para as tarefas subsequentes;
- Dados de saída, que é o resultado pretendido do *workflow*.

A modelação de *workflows*, consciente ou inconscientemente, é sujeita a um conjunto de padrões que definem a relação entre as diferentes tarefas e o fluxo entre estas [WFP]. Os padrões de *workflow* básicos para controlo de fluxo são:

- *Sequence* (sequência) – passos são processadas sequencialmente;
- *Parallel split* (divisão paralela) – o *workflow* bifurca-se em duas linhas de execução que executam paralelamente;
- *Synchronization* (sincronização) – duas linhas de processamento paralelas juntam-se numa só;
- *Exclusive Choice* (escolha exclusiva) – o *workflow* bifurca-se em dois, mas o processamento segue apenas um caminho baseado em variáveis de condição, ou seja, este é um ponto de decisão;
- *Simple Merge* (junção simples) – linhas de processamento exclusivas, previamente divididas num ponto de decisão, são juntas de novo.

Os seguintes diagramas esboçam estes padrões de *workflows*.

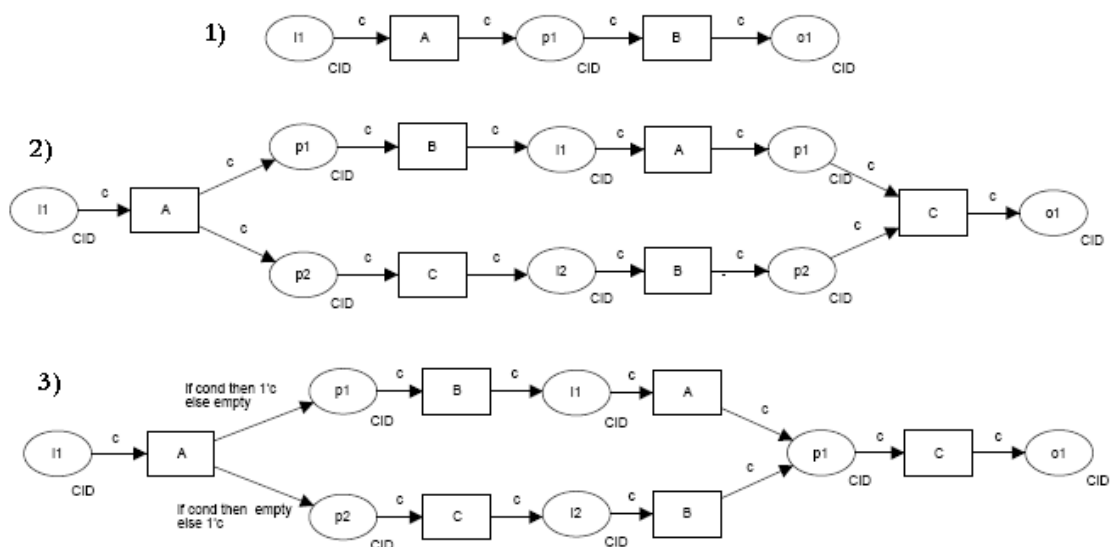


Fig. 2 – Padrões de *workflows*: 1) sequência; 2) divisão paralela e sincronização; 3) escolha exclusiva e junção simples

Existem vários padrões de *workflows*, para além dos básicos indicados, que se dividem em vários tipos:

- *Multiple Instance Patterns* (padrões de múltiplas instâncias) – padrões que implementa a execução paralela de processos semelhantes. Estes podem ocorrer em três situações: quando um *workflow* bifurca a sua execução, quando um *workflow* é despoletado varias vezes após a recepção de sinais; quando dois *workflows* com implementação semelhante são iniciados;
- *State-based Patterns* (padrões orientados ao estado) – padrões que prevêm a contenção de informação significativa sob um dos processos que compõe o *workflow*, que possa afectar a execução deste. O padrão *Exclusive Choice* serve de exemplo;
- *Cancellation and Force Completion Patterns* (padrões para cancelamento e forçar conclusão) – padrões que prevêm a interrupção da execução de um *workflow*, ou o termino forçado;
- *Iteration Patterns* (padrões de iterações) – padrões que modelam ciclos;
- *Termination Patterns* (padrões de termino) – padrões que especificam possíveis circunstâncias sob as quais se pode considerar um *workflow* como terminado;
- *Trigger Patterns* (padrões de gatilho) – padrões que prevêm a recepção de sinais que possam despoletar a execução de novos *workflows*.

Os *workflows* podem ser descritos, em suporte legível, usando linguagens de anotação, ou de representação gráfica. Existem várias linguagens, com diferentes domínios de aplicabilidade, entre as quais:

- WS-BPEL (*Web Services Business Process Execution Language*) – descreve processos de negócio baseada em exclusivamente em *web services*, e é uma norma da OASIS (*Organization for the Advancement of Structured Information Standards*) aprovada em 2007 [WSBPEL]. WS-BPEL é uma linguagem baseada em XML que define vários conceitos para modelar processos de negócio. Define um conjunto básico de estruturas de controlo de fluxo, como condições e ciclos, e elementos que representa o envio e recepção de mensagens de *web services*. Baseia-se em WSDL³ [WSDL] para descrever as interfaces que os *web services* exportam. Depende de outras normas como *XML Schemas*, e *XPath*;
- Wf-XML – protocolo desenvolvido pela *Workflow Management Coalition*⁴, como extensão ao protocolo ASAP (*Asynchronous Service Access Protocol*), desenvolvido pela OASIS, que permite monitorizar a execução de um processo. O Wf-XML

³ WSDL (*Web Services Description Language*) – linguagem para descrever interfaces de *web services*, desenvolvida pela W3C (*World Wide Web Consortium*). É baseada em XML e encontra-se actualmente na versão 2.0, de 2007

⁴ WfMC (*Workflow Management Coalition*) – consorcio dedicado ao desenvolvimento de normas para a interoperabilidade de sistemas de gestão de *workflow*

extende o ASAP para permitir a monitorização e fornecer feedback sobre a execução e estrutura de *web services*. O Wf-XML permite a uma aplicação BPM (*Business process management*) invocar processos noutras aplicações BPM;

- BPMN (*Business Process Modeling Notation*) – descreve exclusivamente processos de negócio recorrendo a notações gráficas, e foi desenvolvido pela *Business Process Management Initiative*;
- XPDL (*XML Process Definition Language*) – é um formato de ficheiro, baseado em XML, criado para armazenar e transportar definições de processos de negócio entre aplicações. Foi certificado como norma internacional pela *Workflow Management Coalition*;
- UML (*Unified Modeling Language*) – linguagem de anotação gráfica usada em *software* para criar modelos abstractos do funcionamento de sistemas. O UML entretanto foi adaptado para descrever outros sistemas e processos não relacionados com *software*;
- EPC (*Event-driven process chain*) – linguagem de anotação de processos de negócio, inicialmente suportada pelo SAP R/3, e agora suportada por várias aplicações no mercado;
- YAWL (*Yet another workflow language*) – linguagem criada por investigadores da Universidade de Tecnologia de *Eindhoven* e Universidade de Tecnologia de *Queensland*. A YAWL tem como objectivo suportar todos os padrões de *workflows*. Actualmente existe uma única implementação da linguagem, num projecto de código livre, com licença LGPL⁵[LGPL];
- jPDL – linguagem desenvolvida para a *framework* JBoss jPMN da Red Hat Inc.

Estas linguagens são suportadas por empresas com bastante reconhecimento na área de gestão de processos como IBM, SAP, Oracle, Red Hat, etc.

2.1.2. Tecnologias

A modelação de processos de negócios, actualmente é implementada por um conjunto vasto de bibliotecas reutilizáveis, que servem como base para o desenvolvimento para aplicações que necessitem de suporte para *workflows*. Estas implementam, com limitações, vários dos padrões de *workflows* existentes, e abrangem diferentes áreas da modelação de processos, como optimização de serviços ou modelação de sistemas.

⁵ LGPL – GNU Lesser General Public License

2.1.2.1. Windows Workflow Foundation.

A *Window Workflow Foundation* (WWF) é desenvolvida pela Microsoft, e é distribuída com a .NET Framework, a partir da versão 3.5 [WWF]. O público-alvo são os programadores. A WWF fornece uma *framework* base sobre a qual podem ser construídas aplicações que façam uso de *workflows*. A WWF foi criada para normalizar a camada tecnológica de *workflows* existente para o Windows, nomeadamente entre os principais produtos da Microsoft, para facilitar integração de ferramentas de suporte aos processos de negócio neste. A *framework* suporta interacção entre aplicações, entra pessoas e aplicações e entre pessoas e pessoas.

A WWF tem três principais objectivos:

- Fornecer uma base tecnológica comum para o Windows. A própria Microsoft tem várias aplicações que fazem uso de *workflows*, mas as implementações diferem, pelo que não faz sentido ter diferentes tecnologias para *workflows*. Ao implementar o suporte para *workflows* na .NET Framework a Microsoft fornece uma base tecnológica para incluir suporte para *workflows* em qualquer software, tanto da própria Microsoft como de terceiros. Futuras versões de vários produtos da Microsoft irão implementar *workflows* com base na WWF, como o MS Office.
- Conceber uma *framework* genérica e completa, que seja reutilizável por aplicações com diferentes casos de uso e domínios, para simplificar o seu desenvolvimento e permitir interoperabilidade imediata entre estas. A WWF fornece funcionalidades de edição de *workflows*, gráfica ou não, execução de *workflows*, directa ou despoletada por eventos. Para implementar esta estrutura genérica, a WWF define um conceito de actividades, que correspondem a acções atómicas isoladas. Sobre estas tarefas depois podem-se implementar controlos como IF/ELSE, WHILE, etc. Pode-se desenvolver novas actividades, ou usufruir das que existem por defeito na WWF.
- Unificar sistemas automáticos e humanos. Os processos automáticos, em que software comunica com software são bastante previsíveis e seguem um conjunto de regras rígidas. Estes podem ser especificados uma vez e depois copiados e reutilizados com poucas ou nenhuma modificação. No entanto, os processos humanos necessitam de ser muito mais flexíveis, pois as pessoas podem mudar de ideias, podem introduzir novos inputs, sugestões, cancelar ou falhar a sua tarefa, pelo que os sistemas humanos são muito mais dinâmicos. Enquanto os sistemas automáticos tendem a ser mais sequenciais, e trocam dados em formatos estruturados e conhecidos, os sistemas humanos são mais baseados em eventos, e troca-se informação em discurso falado, emails, ou documentos. Para conseguir combinar estes dois universos distintos a WWF suporta *workflows* sequenciais, ou baseados em eventos, e a possibilidade de adicionar ou alterar passos de um *workflow* em execução.

As actividades que compõem os *workflows*, e os próprios *workflows*, podem ser programados em qualquer linguagem .NET (C#, VB.NET, C++ .NET), ou em XM.

Infelizmente, não suporta normas nem linguagens de especificação de *workflows* já existentes como WS-BPEL, por defeito, embora o suporte possa ser adicionado graças a *plugins*. Seguem-se exemplos de código:

```
using System.Workflow.ComponentModel;
public class ExampleActivity : Activity{ }

using System.Workflow.Activities;
public class ExampleWorkflow : SequentialWorkflow{ }

<?Mapping XmlNamespace="Activities"
  ClrNamespace="System.Workflow.Activities"
  Assembly="System.Workflow.Activities" ?>
<SequentialWorkflow x:Class="ExampleWorkflow"
  xmlns="Activities" xmlns:x="Definition">
  ...
</SequentialWorkflow>
```

2.1.2.2. JBoss jBPM

A jBoss jBPM é uma *framework* de código livre, desenvolvida em Java, pela jBoss, actualmente uma divisão da Red Hat, Inc [JBPM]. A jBoss jBPM suporta a gestão e construção de *workflows* aplicativos, *workflows web*, automação, processos de negócio, e é escalável. Interopera com vários produtos de *middleware* empresarial baseados em Java EE, e com tecnologias de típicas de sistemas com arquitectura orientada a serviços (*Web services*, *Java messaging*, *J2EE connectors*, *JDBC*, etc). Pode ser corrido como uma aplicação isolada, ou integrada noutras através da *framework* que exporta. A definição dos processos que compõem os *workflows* é totalmente editável. A anotação de processos negócio é feita recorrendo ao BPEL e recorrendo a jPDL, que é uma linguagem desenvolvida para o jPMN, cuja implementação é em código livre. A *framework* é disponibilizada através de subscrições que incluem suporte, certificações, actualizações, e manutenção.

2.1.2.3. Apache ODE (*Orchestration Director Engine*)

O Apache ODE (*Orchestration Director Engine*) é um motor de *workflows* desenvolvido pela *Apache Foundation*, que interpreta e executa processos de negócio escritos em WS-BPEL [AODE]. Este estabelece comunicação entre serviços, efectuando trocas de mensagens, fazendo tratamento de dados e de erro, conforme estabelecido na definição BPEL do processo. Os processos podem durar uma quantidade arbitrária de tempo a executar.

O Apache ODE suporta as seguintes funcionalidades:

- Suporte simultâneo para WS-BPEL 2.0, e BPEL4WS1.1 (linguagem de anotação baseada em WSDL 1.1). Permite compilar, validar e analisar BPEL antes da execução;

- Suporta duas camadas de conectividade distintas, uma baseada em Axis2⁶ [AXIS] e outra que implementa a norma JBI⁷ [JBI], usando ServiceMix⁸;
- API de alto nível que permite integra-lo com qualquer camada de comunicabilidade;
- Capacidade para efectuar instalações ou actualizações em altura de execução (*hot-deployment*);
- Interfaces de utilizador para gerir processos, instâncias aplicacionais e transporte de dados.

2.1.2.4. ActiveBPEL Community Edition Engine

O *ActiveBPEL Community Edition Engine* é um motor de *workflows* de código livre, desenvolvido pela *ActiveBPEL Community*, que implementa o suporte para a linguagem WS-BPEL, incluindo extensões para suportar actividades humanas [ABPEL]. As actividades humanas são suportadas através da especificação BPEL4People⁹ e a especificação WS-HT⁸ [BPEL4PPL]. Tem dois objectivos: ser uma ferramenta pedagógica sobre BPEL, e fornecer um sistema único que integra actividades automáticas e humanas.

2.2. Revisão tecnológica

2.2.1. Ferramentas

2.2.1.1. Workflow Designer para Microsoft Visual Studio

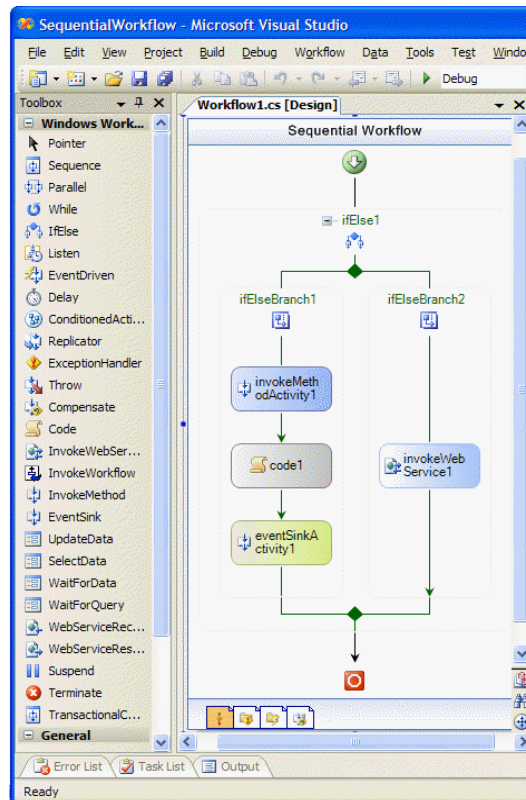
O *plugin Workflow Designer* para o *Microsoft Visual Studio* permite desenvolver sobre a *Windows Workflow Foundation*. A edição de *workflows* pode ser feita visualmente, por código, ou uma mistura dos dois. A ferramenta de edição gráfica contém uma barra lateral com as actividades, e uma área central, onde se constrói o *workflow*. Ao adicionar actividades, pode-se definir propriedades e eventos que pode receber.

⁶ Axis2 – biblioteca desenvolvida pela Apache Foundation, usada como suporte nuclear para Web services em várias aplicações

⁷ JBI (*Java Business Integration*) – especificação desenvolvida pela Java Community Process, com o objectivo de de implementar uma arquitectura orientada a serviços.

⁸ *ServiceMix – framework* para implementação de sistemas distribuídos baseados em serviços, construído de raiz para suportar JBI.

⁹ BPEL4People e WS-HT – extensões ao WS-BPEL para suportar processos humanos, proposta pela IBM e SAP em 2005, e publicada em 2007

Fig. 3 – Ferramenta de edição de *workflows* do Visual Studio

2.2.1.2. jBoss jBPM Visual Process Designer

A *framework* jBoss jBPM inclui ferramentas de desenvolvimento. Uma destas é um *plugin* para o IDE Eclipse. Com este *plugin* podem-se criar projectos do tipo jBPM. A área de trabalho do projecto disponibiliza várias ferramentas para trabalhar com a Framework, tanto em código, como visualmente. Uma dos módulos do *plugin* é o *Visual Process Designer*, que permite a construção de *workflows* através de *drag and drop* com geração automática de código.

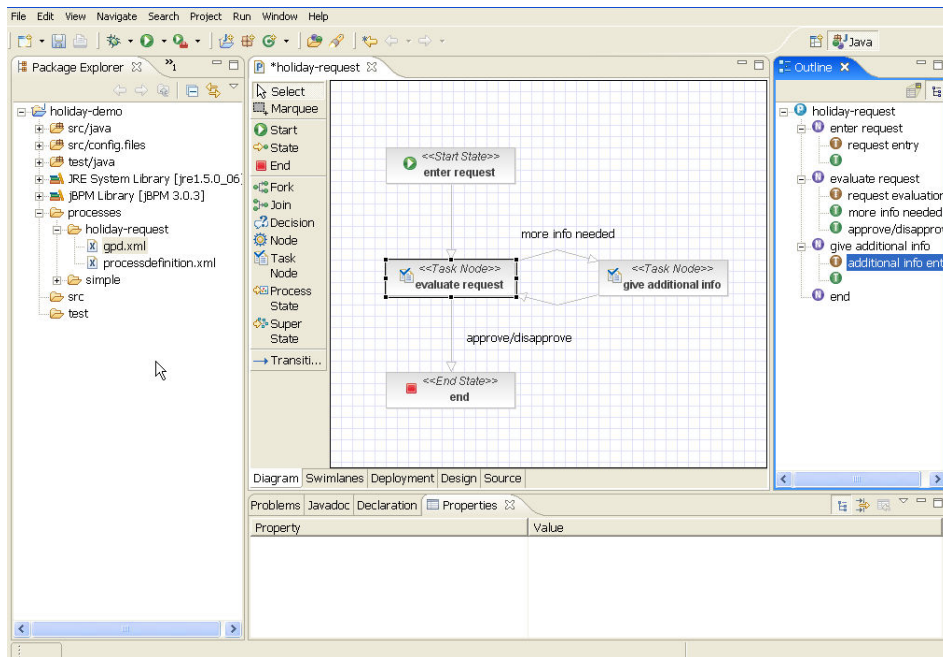


Fig. 4 – jBoss jBPM Visual Process Designer

2.2.1.3. IBM WebSphere Integration Developer

O *WebSphere Integration Developer (WID)* é um IDE (*Integrated Development Environment*) multiplataforma, baseado no IDE Eclipse [WID]. O *WID* suporta a modelação de sistemas e os respectivos processos de negócio, com uma arquitectura baseada em serviços (*SOA – Service Oriented Architecture*).

O *WID* é um componente da suite de produtos *WebSphere* da IBM, orientados à distribuição de serviços em rede. Esta suite inclui, para além do editor, um servidor que gere os processos de negócio, serviços de conectividade, plugins para obter acesso a base de dados relacionais a partir de linguagem de anotação *de workflows*, integração com sistemas ERP (*Enterprise Resource Planning*), CRM (*Customer Relationship Manager*) e RH (Recursos Humanos).

As principais funcionalidades são:

- Integração e reutilização de recursos empresariais no IDE, representados como componentes que podem ser integrados para representar um *workflow*. Esses recursos não se limitam a plataformas tecnológicas. Podem representar recursos humanos, equipas, património, departamentos, etc.
- Permite a construção de *workflows* complexos, usando *drag-and-drop*, sem requerer conhecimentos técnicos, como programação, ou experiencia com a ferramenta.
- Suporta desenvolvimento, testes, depuração e implementação das soluções desenvolvidas.

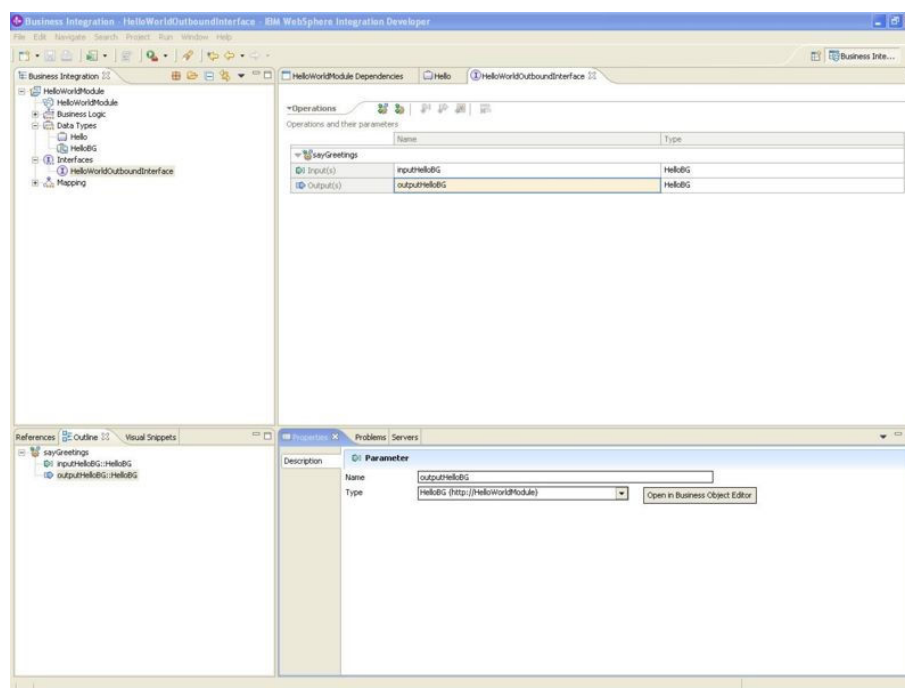


Fig. 5 – IDE WebSphere Integration Developer

2.2.1.4. Taverna WorkBench

O *Taverna WorkBench* é um programa de software livre, para desenhar e executar *workflows* [TAVN]. Foi criado pelo projecto *myGrid*¹⁰[MYGRID] e patrocinado pela OMII-UK. O *Taverna* permite ao utilizador integrar várias ferramentas neste, incluindo suporte para *web services*. O *Taverna* fornece um ambiente gráfico para a elaboração de *workflows* científicos. Descritos na linguagem *Scufl*.¹¹ O motor *workflows* do *Taverna* é disponibilizado separadamente para poder ser integrado noutros projectos.

O *Taverna* foi desenvolvido para permitir a cientistas com poucos conhecimentos técnicos, para desenvolver modelos complexos de análise de dados sobre base de dados e outros recursos electrónicos.

¹⁰ myGrid – framework para computação distribuída, desenvolvida para projecto científicos

¹¹ Scufl (*Simple Conceptual Unified Flow Language*) – linguagem desenvolvida num âmbito do projecto Taverna para a anotação de *workflows*.

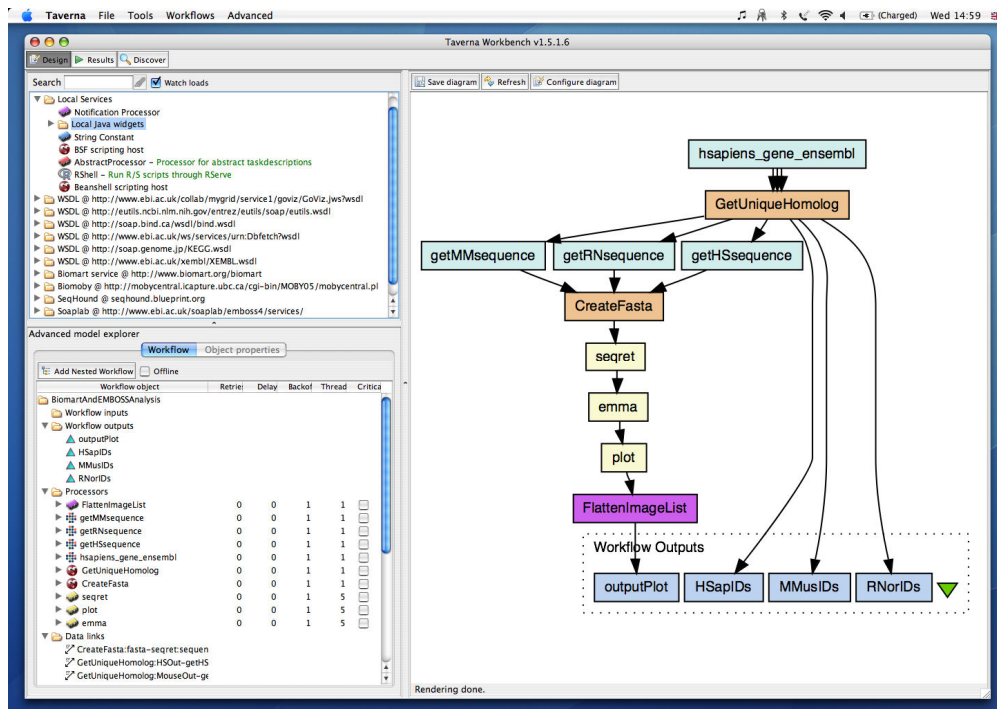


Fig. 6 – Ambiente do *Taverna Workbench* a correr um *workflow* de exemplo.

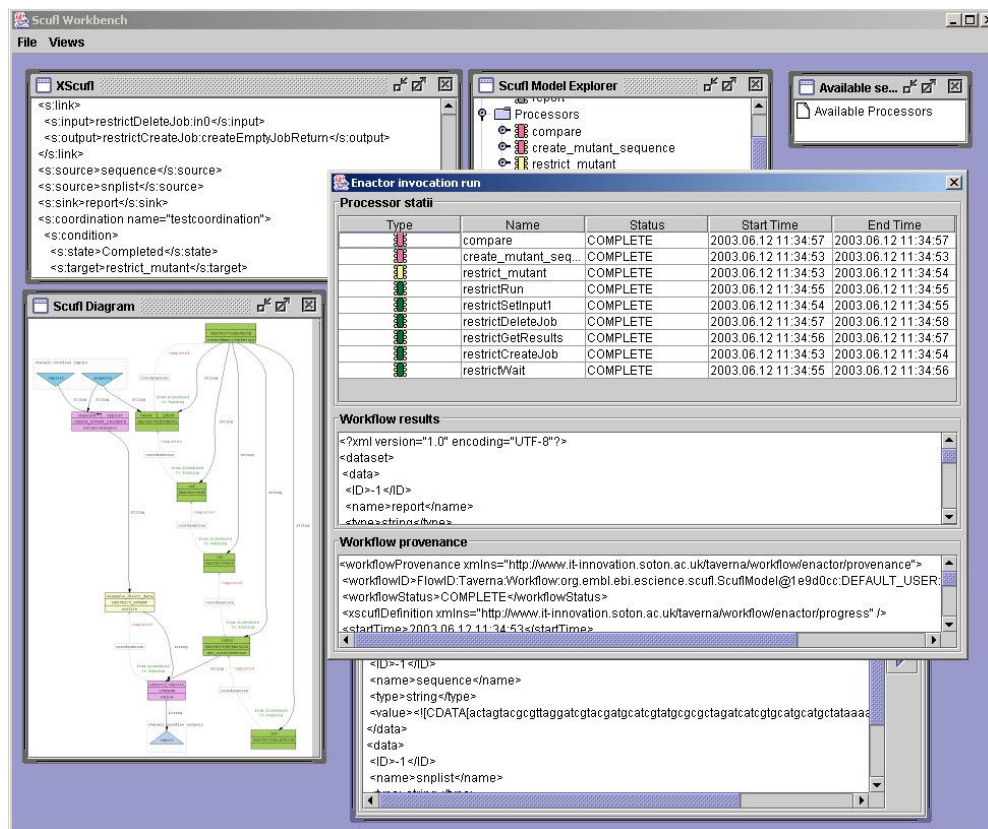
2.2.1.5. Freefluo

Freefluo é uma ferramenta para a construção de *workflows* baseados em *web services* [FFL]. Suporta WSDL para descrição de *web services*, WSFL¹² [XSCUFL], e XScuf¹³. Este é constituído por um núcleo que não implementa nenhuma linguagem de especificação de *workflows* específica, ou arquitectura de execução, de forma a obter maior flexibilidade.

O *Freefluo* é incluído na *framework myGrid* usada para construir sistemas de computação distribuída, principalmente usada para fins científicos.

¹² WSFL (*Web Services Flow Language*) – linguagem desenvolvida pela IBM para descrever a composição de *web services*, entretanto substituída por WS-BPEL.

¹³ XScuf (*XML Simple Conceptual Unified Flow Language*) – derivação de Scuf, linguagem desenvolvida num âmbito do projecto Taverna para a anotação de *workflows*

Fig. 7 – Interface gráfica do *Freeflow*

2.2.2. Ferramentas utilizadas

As ferramentas usadas para a prossecução diária das tarefas deste projecto foram:

2.2.2.1. PL/SQL Developer

O *PL/SQL Developer* é uma ferramenta desenvolvida pela empresa Holandesa “*All Round Automations*” [PLSQLD]. É um IDE dedicado para desenvolver unidade de código para uma base de dados Oracle. Este IDE suporta várias funcionalidades que melhoram a produtividade, como as seguintes:

- Editor de PL/SQL com coloração de sintaxe, *auto-complete* com assistente de código, *refactoring*, *code folding*, *beautifiser* (indentação automática de código), entre outros;
- Depurador de PL/SQL;
- Janelas para analisar o resultado de interrogações de SQL, linha de comando (*SQLPlus*), diagramas do modelo de dados;
- Criação e manutenção de projectos;

- Ferramentas para analisar o desempenho de código e interrogações SQL;
- Ferramentas para criação e manutenção dos vários tipos de objectos que compõem o modelo de dados, como tabelas, restrições, *views*, *triggers*, *packages*, funções, procedimentos e outros;
- Macros para introduzir automaticamente código frequente e macros de teclado;
- Explorador de objectos: permite visualizar todos os objectos presentes na BD à qual o utilizador tem acesso;
- Arquitectura extensível, que permite a instalação de *plugins*. Foi utilizado um *plugin* de controlo de versões para permitir integração com o SVN.

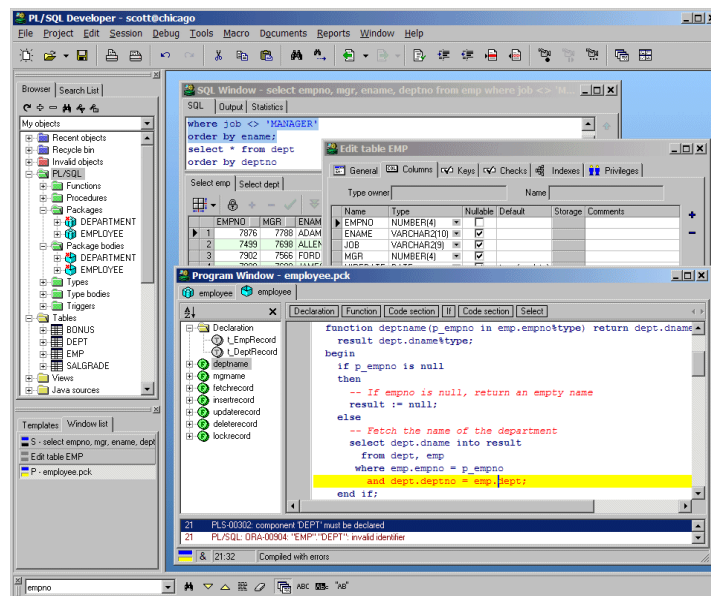


Fig. 8 – PL/SQL Developer

2.2.2.2. Tortoise SVN

O *Tortoise SVN* é a aplicação cliente usada para fazer o controlo de versões do código produzido durante o desenvolvimento da aplicação [TSVN]. Este integra com o explorador de ficheiros do *Windows*, fornecendo pistas visuais sobre o estado dos ficheiros, como ícones, e opções para gerir a árvore de ficheiros, no menu de contexto. O SVN (*Subversioning System*) é um sistema de controlo de versões que surgiu em 2000, cuja adopção pelo mercado é cada vez maior, sendo utilizado por empresas e grandes projectos *open source*, como *Apache Software Foundation*, *KDE*, *GCC*, etc. O SVN tem as seguintes funcionalidades:

- Os *commits* (acto de enviar modificações para o servidor) são acções controladas, cuja interrupção ou sucesso garantem sempre a consistência e não-corrupção do repositório;

- Renomear, copiar, mover e apagar ficheiros mantém a história de versões completa;
- Árvores de directorias são versionadas, e os meta-dados associados à directoria (informação presente apenas no sistema de ficheiros) são também versionados;
- Ligações simbólicas para ficheiros ou directorias são versionadas;
- Suporte para versionamento de ficheiro binários;
- Criar ramos, e atribuir etiquetas (*tags*) às árvores de directorias, são operações que são executadas sempre em tempo constante, e não dependem do tamanho da árvore a ser versionada;
- Todas as trocas de informação cliente-servidor e vice-versa usam *diffs*, o que torna o custo das operações proporcional ao tamanho das modificações, e não dos ficheiros;
- Possibilidade de fazer registos de actividades para formatos de saída estruturados como XML;
- Licenças *open source*, com suporte para várias línguas internacionais;
- Possibilidade de bloquear acesso a determinados ficheiros;
- Autenticação baseada no local da árvore a ser controlada;
- Bibliotecas em Python, Perl, e Java.

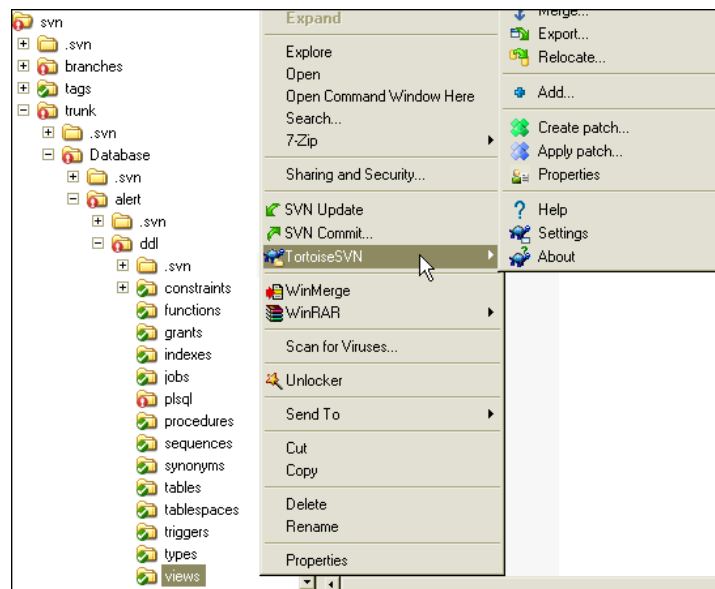


Fig. 9 – Integração do *Tortoise SVN* com o explorador de ficheiros do sistema operativo Microsoft Windows

2.2.2.3. Adobe Flash

Adobe Flash, ou simplesmente *Flash*, é um programa que usa uma tecnologia de gráficos vectoriais utilizados para criar animações interactivas, desenvolvidas e comercializado pela *Adobe Systems* (que recentemente adquiriu a *Macromedia*) [FLASH].

Os arquivos executáveis gerados pelo Flash, chamados de SWF (*Shockwave Flash File*), podem ser visualizados numa página de navegação de Internet, ou utilizando o Flash Player. Em versões recentes (a partir da 5), a *Macromedia* expandiu a utilização do Flash para além de simples animações, tornando-o uma ferramenta de desenvolvimento de aplicações complexas. Isso graças aos avanços na linguagem *ActionScript*, que é a linguagem de programação utilizada em aplicações SWF e que se encontra na terceira versão.

O *ActionScript* é uma linguagem de programação baseada em *ECMAScript*, revisão 3. As semelhanças são óbvias, sendo muito fácil a conversão entre elas e a consequente adaptação de uma linguagem para a outra. Em *ActionScript* existem classes, bibliotecas e componentes (por exemplo *Movieclips*) que podem ser associadas às mesmas. As classes são sempre escritas num ficheiro externo de texto, tendo estes ficheiros uma extensão ".as". As classes são extensões ao *ActionScript* original e existem já classes provenientes da própria linguagem, tais como as classes relativas aos componentes básicos do Flash. As classes permitem que o programa se torne mais simples e permitem a modularidade do mesmo, com consequente reaproveitamento de código, características essenciais em qualquer linguagem orientada a objectos.

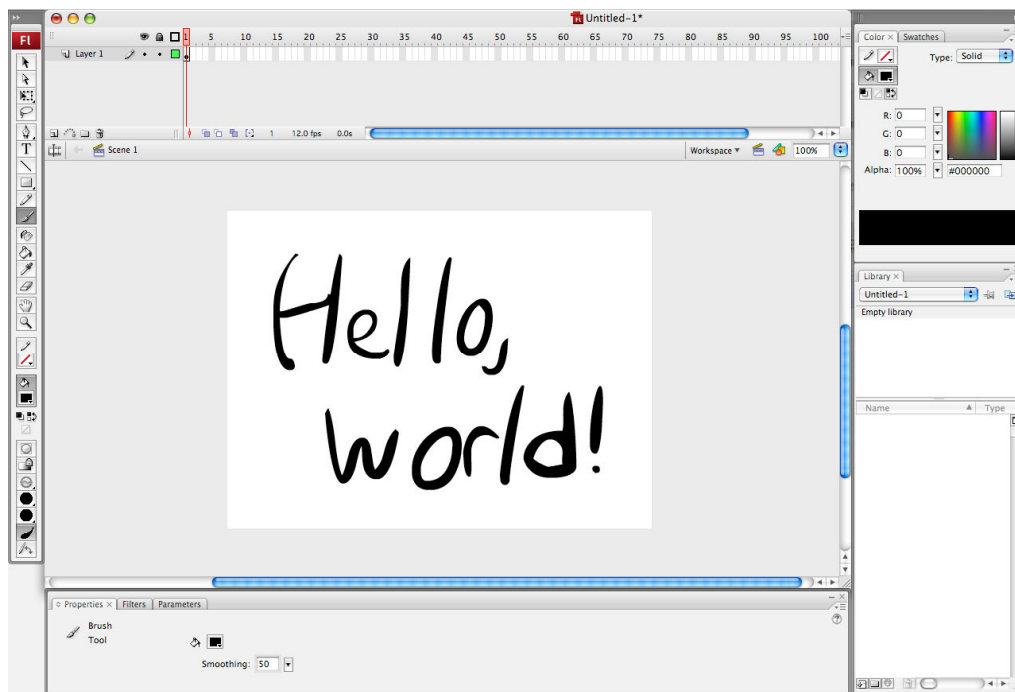


Fig. 10 – Imagem do IDE Adobe Flash

2.3. Observações

Após o estudo dos conceitos, tecnologias e ferramentas aqui apresentados, podem-se fazer comparações com os resultados pretendidos deste projecto. Os conceitos teóricos apresentados permitem ter uma boa base de como se deve conceber uma forma de armazenar e relacionar os componentes que constituem o *workflow*.

A ideia clara de padrões de *workflows* fornece contexto para a preparação do modelo de dados e da organização da lógica de negócio. Embora hajam linguagens de anotação de processos populares, no contexto deste projecto, a aplicação que se irá construir irá usufruir de integração com o Alert® PFH, e irá partilhar a mesma base tecnológica, pois é uma das restrições impostas à partida. No entanto, as linguagens de anotação, em conjunção com os padrões de *workflows*, fornecem valiosas pistas de como se podem estruturar os elementos para atingir o fim pretendido.

A avaliação de *frameworks* e motores de *workflows* permite fazer um resumo de quais as normas mais bem aceites no mercado, e de que forma estas são implementadas. De forma geral, a organização de sistemas com arquitectura orientada a serviços é bastante popular, pois os conceitos intrínsecos à lógica de negócio de uma aplicação específica são bastante difíceis de generalizar numa especificação que possa ser assumida como uma norma internacional. Mas, quer se analise um sistema maior ou menor, existem sempre alguns conceitos que prevalecem: a flexibilidade e elaboração de estruturas de dados flexíveis, com um motor central independente dos processos cuja execução delega. Esta abordagem é seguida na implementação do motor de execução dos *workflows* do Alert® Reset, pois os casos de uso do Alert® PFH são vastíssimos, pelo que não é possível especializar a aplicação apenas para alguns casos.

Existem várias ferramentas de edição visual de *workflows*, que tem vários elementos em comum, como uma área central para construção do *workflow* através de *drag-and-drop*, com um ecrã de detalhe para editar propriedades específicas de uma tarefa em particular, e geração de código automática. O Alert® Reset irá fornecer uma interface semelhante, mas com restrições, pois existem determinadas tarefas que são dependentes da execução de tarefas anteriores. Estas restrições serão elaboradas no capítulo de avaliação de requisitos e especificação técnica. Não será desenvolvida uma funcionalidade em específico para gerar código, mas ter-se-á que desenvolver uma estrutura de dados que possa ser gerada na aplicação cliente, enviada para o servidor, armazenada e executada. Como a composição de um *workflow* pode ter um número arbitrário de processos, com um número arbitrário de relações entre estes, a transmissão de dados será feita recorrendo a um formato estruturada como XML, que será interpretado e gravado numa base de dados relacional.

3. Ambiente de demonstrações do Alert®

3.1. Contexto

3.1.1. Organização do ambiente

As máquinas de demonstrações são usadas para demonstrações do Alert® em feiras, exposições, clientes, apresentações a convidados e visitas da empresa, comunicação social, e para dar formação aos recém-empregados da ALERT. Anualmente, o Alert® é demonstrado a milhares de pessoas

O ambiente de demos consiste numa instância do Alert®, da última versão estável, que é monitorizada e costumizada para satisfazer as necessidades das equipas de demonstrações. Quando há necessidade, é feita uma cópia do disco rígido do servidor, que tem este ambiente, para uma imagem que corre numa máquina virtual nas máquinas dos indivíduos das equipas comerciais. Estas cópias são usadas para todo o tipo de apresentações.

As equipas de demonstrações têm sido fortemente condicionadas pela instabilidade e qualidade dos dados da instalação do Alert usada para este fim.

O seguinte diagrama ilustra a organização prévia deste ambiente (os rectângulos representam equipas, as setas representam a passagem de entregáveis):

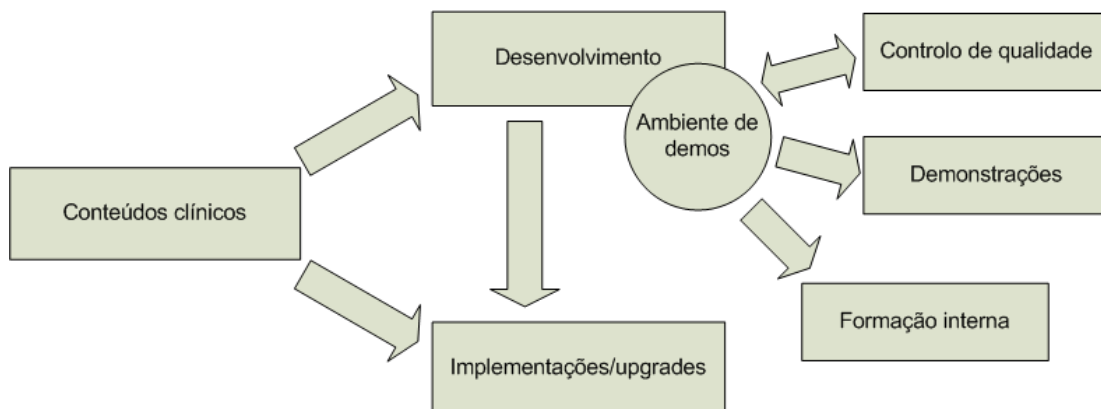


Fig. 11 – Diagrama com a organização antiga do ambiente de demonstrações

Antes, o ambiente de demonstrações estava ao cargo da equipa de desenvolvimento.

Havia uma sub-equipa do grupo de controlo de qualidade que testava e monitorizava o Alert® neste ambiente, embora não pudesse fazer intervenções na base de dados. Esta equipa reportava problemas no Alert®, através do sistema de *bug tracking* usado internamente, para a equipa de desenvolvimento resolver. Quando os elementos da equipa de desenvolvimento tinham disponibilidade, podiam resolver os problemas, parcial ou totalmente. Como a máquina era alvo de intervenções de várias pessoas da equipa de desenvolvimento, por vezes estas intervenções simultâneas podiam fazer com que código fosse perdido na máquina, que surgissem novos erros, ou que o ambiente ficasse instável.

A equipa de demonstrações, por vezes, faz demonstrações em que têm que seguir um guião, que representa um caso real. Nos guiões, prevê-se a utilização de determinado conteúdo clínico. Por isso, esse conteúdo tem que ser disponibilizado. Isso implica a introdução de novo conteúdo, ou a parametrização de conteúdo já existente para ficar associado a novas especialidades clínicas. A definição deste conteúdo, e associação com outro conteúdo e especialidades clínicas, é definida pela equipa de conteúdos clínicos, que faz um trabalho semelhante para o conteúdo clínico presente nos clientes.

Os upgrades do Alert® nesse ambiente eram feitos pela equipa de DBA (administração de base de dados), agora denominada equipa de arquitectura tecnológica. Essa equipa faz parte da equipa de desenvolvimento.

Periodicamente, é feita uma cópia do ambiente para imagens para correr como máquina virtual, ficando essa tarefa à responsabilidade da equipa de redes e *hardware*.

Desta organização os procedimentos de definição do conteúdo clínico, upgrades e cópia funcionavam correctamente, pois eram atribuídos a equipas especializadas nestas tarefas. No entanto, a equipa de desenvolvimento tem outras responsabilidades para com o desenvolvimento do Alert®. A pouca disponibilidade desta equipa, tamanho enorme, e falta de visão global dos objectivos deste ambiente, resultaram numa solução pouco óptima para a manutenção deste.

3.1.2. Aplicação antiga de reset

Para preparar informação para uma demonstração no Alert®, havia uma aplicação chamada de Reset, pois apagava dados antigos, e repunha dados novos. A aplicação tinha o seguinte aspecto:

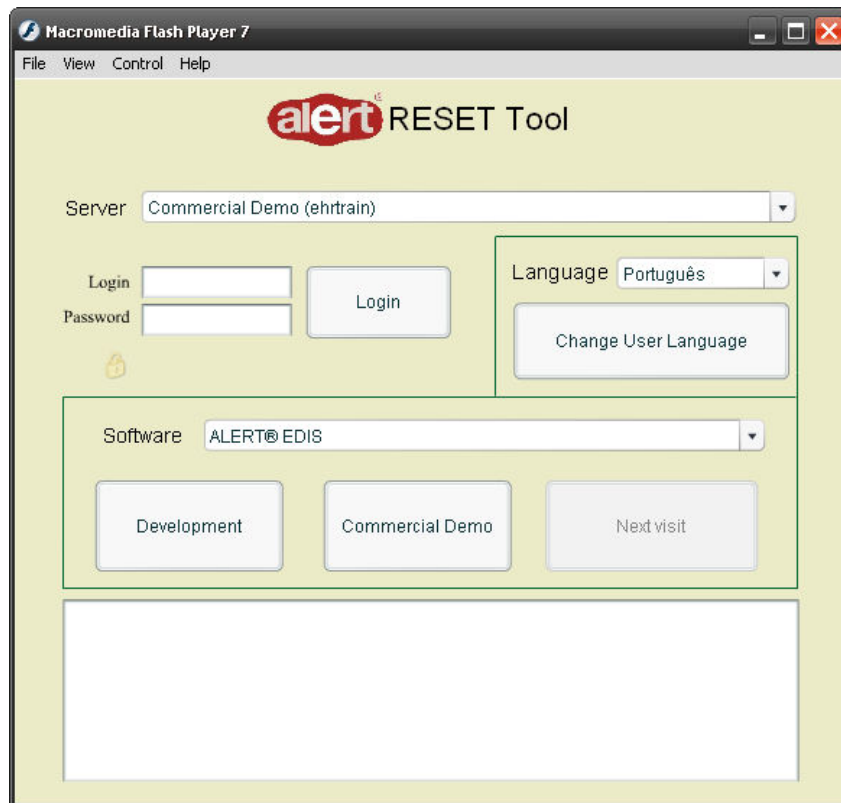


Fig. 12 – Antiga aplicação de reset

A antiga aplicação de reset tinha vários problemas.

Em primeiro lugar, esta permitia escolher um dos vários ambientes disponíveis, como urgência, internamento, ou consulta externa, e com um clique de um botão, a aplicação apagava todos os episódios existentes nesses ambientes, e criava novos. Como o Alert tem uma organização multi-instituição, multi-ambiente, a aplicação já dava a escolha ao utilizador do ambiente a limpar. Mas as instituições estavam programadas no reset, *hardcoded*, de forma inflexível.

A limpeza de episódios anteriormente, apagava dados das tabelas que compõem o modelo de dados, mas com o crescimento da base de dados, este algoritmo facilmente falhava devido às restrições de integridade na base de dados (chaves estrangeiras). Entretanto, antes de início deste projecto, o reset foi alterado para passar registos de uma instituição A para uma instituição B e passar de um paciente X para um paciente Y. Esta instituição e paciente depois não eram visíveis na aplicação. Esta situação acumulou durante bastante tempo bastantes dados na base de dados que nunca mais seriam usados, o que prejudica o desempenho da máquina, e aumenta o tamanho necessário em disco para conter a base de dados.

Simultaneamente, como a passagem de dados para outra instituição e paciente não valida restrições de integridade, não se tem a garantia que toda a informação é passada. Nas várias tabelas que guardam a informação registada num episódio, existem várias referências ao paciente e instituição, redundância para melhorar o desempenho. Por vezes, era

registada informação em tabelas cujos dados não estavam a ser correctamente migrados, e por isso gerava-se “lixo” em episódios novos. Após um reset, pacientes continuavam com tratamento activo ou tinham histórico.

Em segundo lugar, o reset disponibilizava a opção de fazer uma mudança de língua da aplicação. É possível, a partir do Alert®, cada utilizador individualmente alterar a língua em que vê o Alert® na sua área pessoal. A mudança de língua no reset altera a língua de todos as contas de utilizador, e muda configurações para se adequarem melhor ao mercado onde falam essa língua. Por exemplo, a mudança de língua para o inglês disponibiliza conteúdo clínico americano, muda o fuso horário da instituição para um fuso horário americano, muda as unidades de sinais vitais para o sistema americano, muda o tipo de triagem usado na urgência para a triagem americana, etc.

Em terceiro lugar, a máquina comercial actualmente é usada como repositório de traduções para todas as línguas, pois é sempre neste ambiente que surgem as primeiras necessidades de tradução, e é sempre neste ambiente que se fazem validações e correcções de traduções, pois esta máquina é usada e revista nas várias línguas para preparar as demonstrações, e é usada por pessoas que falam as outras línguas nativamente.

Em quarto lugar, a instância do Alert® usada para demonstrações não foi uma instalação feita de raiz para este propósito, mas foi uma cópia integral do ambiente de desenvolvimento em 2006, pelo que há uma quantidade de dados redundantes significativa, e de fraca qualidade.

Em quinto lugar, as equipas de formação, tanto internas na empresa, como em clientes, e as equipas de demonstrações têm um conjunto de requisitos, que precisam de ver satisfeitos, mas que a aplicação de reset não suporta. Esses requisitos prevêem a gestão de alguma informação central do Alert®, e a possibilidade de pré-programar um conjunto de dados que estarão disponíveis no Alert®, repetidamente, para várias demonstrações e formações.

3.1.3. Observações

Em súmula, esta situação tem resultado nos seguintes problemas:

- Instabilidade e descontrolo, pois fazem-se demasiadas intervenções no ambiente por diferentes pessoas;
- Conteúdo clínico e parametrizações inadequadas aos mercados-alvo, o que reduz o realismo das demonstrações, e torna a execução dos guiões mais difícil;
- Problemas constantes de espaço em disco e qualidade de dados devidos a resets sucessivos;
- Devido às intervenções na base de dados, problemas de espaço em disco e qualidade dos dados, os upgrades geralmente correm mal, ocorrem bastantes erros, o que obriga a que hajam testes exaustivos e mais intervenções para corrigir os problemas resultantes;

- As traduções por vezes estão incorrectas ou dessincronizadas com outros textos noutras línguas, ou estão em falta, pois o trabalho de tradução e validação é feito a partir deste ambiente, que já devia ter traduções validadas, em conjunto com o conteúdo clínico;
- Limitações do que pode ser demonstrado no Alert®, pois a aplicação de reset não gera informação para demonstrar certas funcionalidades, ou para realizar um certo guião, e como consequência, as demonstrações e formações não seguem por vezes os guiões planeados.

3.2. Soluções propostas

3.2.1. Nova aplicação de reset

A proposta deste projecto continha a seguinte lista de objectivos:

- Remover duma instalação do Alert® dados desnecessários de testes e demonstrações anteriores;
- Mudança de língua do Alert®, e outras configurações que variam por localização;
- Criação e gestão de casos clínicos específicos, baseados em vários critérios, como especialidade clínica;
- Configuração do *workflow* pretendido para as demonstrações, que inclui prescrição de tratamentos, exames, análise, medicação, etc;
- Mudança de hora e fuso horário de uma instalação.

Esta lista de objectivos, embora limitada para se adequar aos prazos do projecto, já toca num dos pontos fulcrais do problema: criação e gestão de casos clínicos, ou seja, episódios com tratamento, mudança de língua e configurações. A funcionalidade de mudança de língua por si já era fonte de problemas, pois permitia ao utilizador com um toque de um botão que reconfigurasse completamente a base de dados.

Após uma reunião inicial com alguns dos *stakeholders* da aplicação de reset, ou seja, membros da equipa de demonstrações e formação, obteve-se a seguinte lista de requisitos:

- Manipulação de datas de episódios pelo utilizador com acesso à funcionalidade reset;
- Registo de informação em episódios anteriores;
- Escolher tipos de consulta para os episódios da grelha principal (aplicações de ambulatório);
- Configurações como tipo de triagem e outros;
- Criação de pacientes com números de episódios e processo clínico, com nome para várias línguas, com fotografia e preferências a nível de departamento e instituição;

- Registo de tratamento nos episódios da grelha principal.

Esta lista já representa parcialmente as necessidades reais das equipas de demonstrações, embora não represente correctamente os casos de uso da equipa de formação. As equipas de formação têm outras necessidades mais complexas. Por exemplo, se um formador quiser dar uma formação para 10 obstetras, este vai querer que o Alert® disponha de 11 mulheres com gravidezes activas, 1 para cada obstetra e uma para o formador, e assim fazem registos, cada um no seu paciente. Se o formador quiser dar formação a 10 pediatras, este quer 11 crianças. Estes casos de uso representam necessidades reais.

Será que esta informação, lista de requisitos e *user stories*, representa realmente as necessidades do utilizador? De que forma se podem integrar e normalizar estes requisitos? Note-se que uma das regras para facilitar o desenvolvimento no novo reset e o trabalho dos demonstradores e formadores era não duplicar funcionalidades que o Alert® já suporta.

Segundo o perito em usabilidade Jakob Nielsen: “*The most common way to get usability wrong is to listen to what users say rather than actually watching what they do*”¹⁴[JKN], ou seja “A forma mais comum de obter má usabilidade é ouvir o que os utilizadores dizem, em vez de observar o que eles fazem”. Embora esta frase seja um bocado agressiva, pode-se tirar dela algumas ilações para o problema actual.

A manipulação de datas em episódios anteriores e o registo de informação em episódios anteriores tem o mesmo objectivo: produzir histórico nos pacientes. Então por que não simplificar a tarefa, ao permitir o utilizador fazer registos directamente no Alert® e preservar esse episódio? Este é o processo usado actualmente, mas requer acesso à base de dados, na qual se coloca um ID de episódio (ID da tabela da base de dados) numa tabela que guarda IDs de episódios que não podem ser apagados durante um reset. Esse episódio então é fechado em vez de ser apagado.

A solução consistirá numa funcionalidade, que permita ao utilizador escolher os episódios que bem deseje, alterar as suas datas de início, e bloqueá-lo de forma semelhante como já era feito directamente na base de dados. Agora o utilizador irá dispor desta funcionalidade num interface amigável e independente. Este será obrigado a registar toda a informação que desejar no episódio antes deste ser fechado pelo próximo reset, ou pelo utilizador. E assim não se necessita de fornecer uma nova funcionalidade que permita registar informação nestes, pois os casos de uso seriam tão vastos como o próprio Alert®.

Os requisitos para escolher tipos de consulta e permitir alterar configurações já são funcionalidades que estão a ser desenvolvidas para o Alert®, disponíveis respectivamente no *backoffice* da agenda e no *backoffice* aplicacional. Ou seja, o utilizador irá usufruir das ferramentas que o Alert® dispõe para introduzir esta informação.

¹⁴ Jakob Nielsen, “Top-10 Application-Design Mistakes”

Como o Alert® PFH actualmente não dispõe de um software para fazer a gestão de pacientes, pois esta funcionalidade sempre esteve delegada a sistemas externos já existentes nos hospitais, a aplicação irá permitir criar pacientes, associar informação pessoal e fotos.

O requisito mais complexo é o registo de tratamento em episódios. Este coincide com o que as equipas de formação desejam, pois estas precisam de uma forma de obter episódios no Alert® já com informação registada. A possibilidade de criar agendamentos de consultas de especialidade também será satisfeita por este requisito.

Inicialmente, a resolução proposta para este requisito consistia em criar *snapshots* da base de dados, mas tal processo era tendencialmente pouco eficiente, requeria licenças especiais de Oracle, pois essa funcionalidade só está disponível nas edições *enterprise*, não permitia fazer reset selectivo apenas de alguns ambientes, e após recuperar a *snapshot*, o tratamento estava com os tempos quando foi criado, ou seja, entretanto ganhou bastante atraso.

Este requisito será satisfeito com uma nova funcionalidade: a possibilidade do utilizador construir uma árvore de que representa um *workflow* ramificado. Esse *workflow* ao ser executado irá gerar informação no Alert®. Cada nó da árvore representa uma tarefa executável na aplicação, como prescrever um medicamento, dar uma alta, ou criar um episódio. Cada tarefa pode ser executada dentro de um contexto: prescrever medicação tem que ser executado no contexto de um episódio, fazer colheitas tem que ser executado no contexto de uma análise laboratorial.

Esta nova funcionalidade, embora não prevista nos requisitos iniciais, irá permitir ao utilizador especificar exactamente que informação quer ver na aplicação, e irá permitir ao utilizador criar conjuntos de *workflows* para executar nas alturas adequadas.

3.2.2. Organização do ambiente de demonstração

3.2.2.1. Casos de uso

Para a resolução dos problemas do ambiente de demonstrações enunciados, há que reorganizar e repensá-lo. Tendo em conta os diferentes casos de uso, a nova organização, embora muito semelhante à anterior, poderá ser esboçada da seguinte forma (os rectângulos representam equipas, as setas representam a passagem de entregáveis):

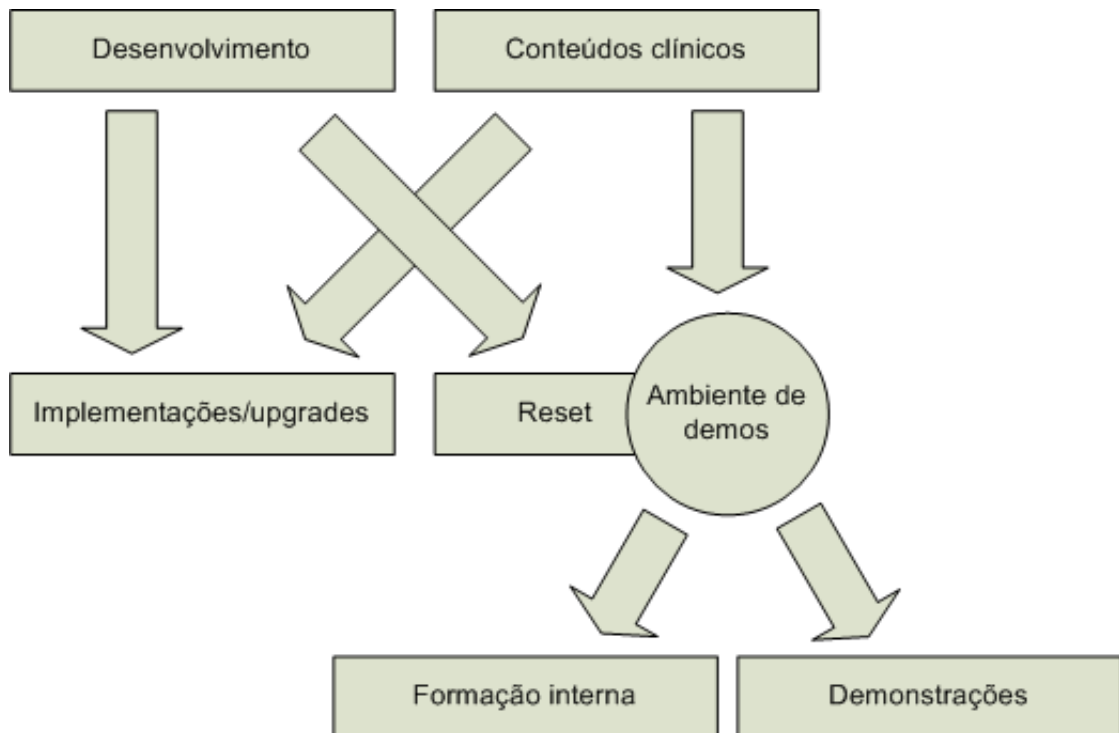


Fig 13 – Nova organização do ambiente de demonstrações

A equipa de desenvolvimento deixará de fazer intervenções no ambiente de demonstrações, excepto para situações pontuais e urgentes em que algum *bug* tem que ser corrigido. Desenvolvimentos específicos para demonstrações serão feitos por esta equipa. A equipa de arquitectura tecnológica manterá a instância da base de dados e fará os upgrades de versão.

A equipa de conteúdos clínicos irá trabalhar em conjunto com a equipa de demonstrações e reset para definir claramente o conteúdo clínico para os vários mercados alvo das demonstrações. Estas equipas identificarão quais os mercados alvo, quais as necessidades de demonstração, que ambiente serão disponibilizados, qual o histórico dos pacientes, qual o conjunto de MCDT's (meios complementares de diagnóstico e terapêutica), medicação, diagnósticos, e outros conteúdos usados pelos mercados alvo. Com a clara definição destes universos irá caber à equipa de Reset, que será responsável por manter a base de dados do Alert® deste ambiente, introduzir o conteúdo, devidamente associados às instituições dos mercados pretendidos. Todo o conteúdo clínico terá que partir obrigatoriamente da equipa de conteúdos clínicos, pois este terá que partir dos universos de conteúdo suportados e terá que ter os IDs da base de dados sincronizados com os outros ambientes. Por defeito, o conteúdo utilizado para um mercado em específico será o *MyAlert fusion*, que é a denominação atribuída ao universo de conteúdo clínico por mercado.

A equipa de reset irá manter a base de dados do Alert®, ou seja, parametrizar conteúdo disponível na aplicação, como clínico, institucional e pacientes; testar a aplicação, incluindo testar as funcionalidades do Alert®, parametrizações, e depurar problemas; fazer

as intervenções necessárias para a resolução de eventuais *bugs*, caso seja pertinente a resolução e não se possa esperar por um *fix* de versão; e fazer a manutenção da aplicação de Reset. Caso existam desenvolvimentos específicos para demonstrações, a equipa de reset irá receber um conjunto de código e documentação e irá aplicar as modificações necessárias no Alert®.

A equipa de demos será o cliente final deste ambiente, e irá receber cópias deste ambiente para correr localmente nas máquinas usadas para as demonstrações comerciais. Esta equipa, em conjunto com a equipa de reset irá elaborar as necessidades de conteúdo necessário para a aplicação. A equipa de demos irá também a continuar a desempenhar as tarefas habituais de preparação das apresentações, que obviamente resultam em testes da aplicação, directos ou indirectos. Caberá a esta equipa e à equipa de conteúdos definir claramente os diferentes cenários de utilização do Alert®, tendo em conta rigorosidade clínica, para se construírem guiões de demonstrações que serão usados obrigatoriamente.

O seguinte diagrama ilustra os casos de usos enunciados:

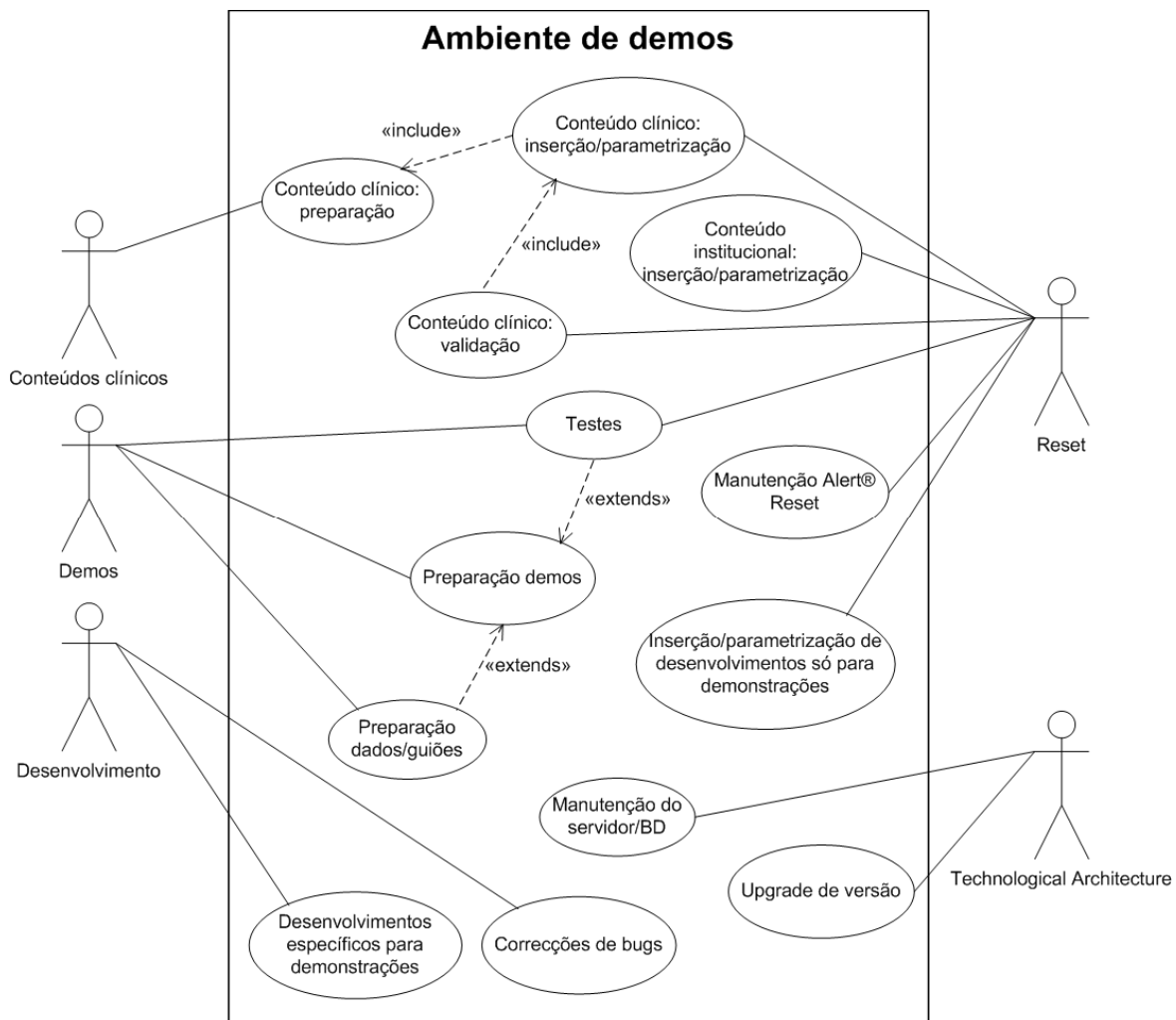


Fig. 14 – Casos de uso do ambiente de demonstrações

Haverá um conjunto de instituições por mercado, cada uma com as configurações adequadas, e contas de utilizador dedicadas. No ambiente deverão existir instituições para hospital, centro de saúde, clínica e P1, por mercado. Haverá conjuntos de pacientes para cada mercado. Cada paciente terá um conjunto de episódios e histórico dentro das instituições do mercado alvo. Esta informação seguirá critérios de rigorosidade para se assemelhar o melhor possível a casos clínicos reais.

A organização proposta implica limitar os acessos à base de dados do ambiente de demonstrações. Então, como é que se garante a manutenção e estabilidade da base de dados?

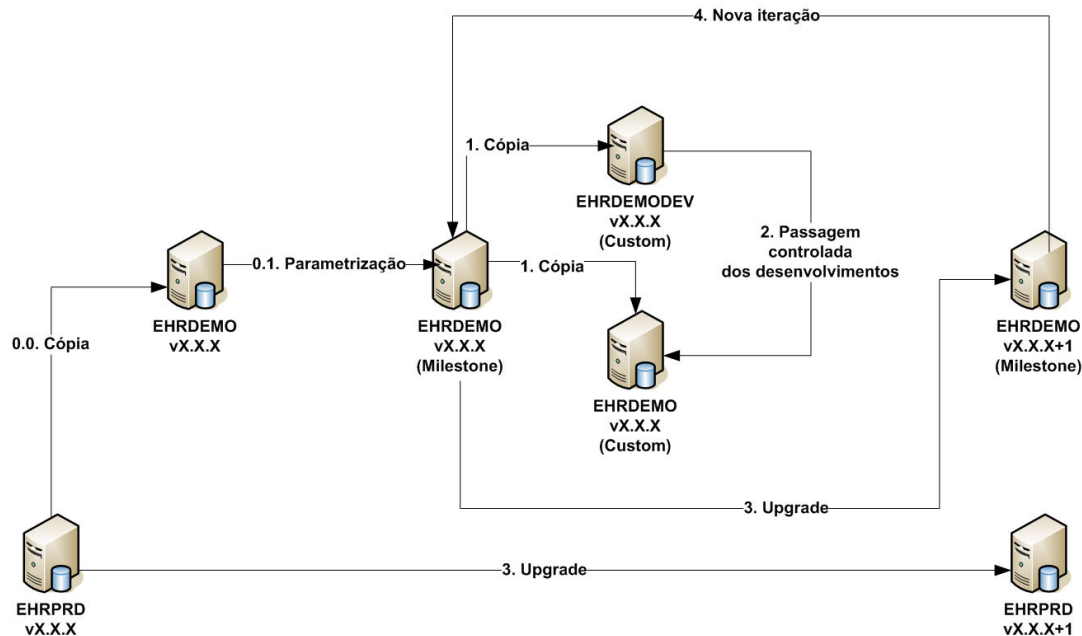
3.2.2.2. Versionamento do ambiente de demonstrações

Será feita uma cópia da base de dados de produção, usada com base de novas instalações, que servirá de base para demonstrações. A partir desse momento o ambiente de demonstrações será tratado como um cliente, e só a equipa de reset e arquitectura técnica terão acesso à base de dados deste ambiente.

A gestão de funcionalidades específicas para demos, irá incluir vários passos intermédios para garantir a estabilidade da máquina final, e uma melhor organização do trabalho efectuado.

Sucintamente, o ambiente de demos chamar-se-á EHRDEMO. Após a cópia inicial do ambiente de produção (EHRPRD), ou upgrade, e devidas parametrizações para essa versão, obtêm a EHRDEMO *Milestone*, uma nova instância. Desta máquina faz-se uma nova cópia, sobre a qual se podem colocar desenvolvimentos específicos para demos, caso hajam, que será testada. Esta nova cópia chamar-se-á EHRDEMO *Custom*. A partir deste ambiente fazem-se as cópias para as máquinas comerciais. Novos upgrades incidem sempre sobre a EHRDEMO *Milestone*, por têm uma base de dados intacta, ou seja, informação que só haja na EHRDEMO *Custom* será perdida. É uma desvantagem do processo, mas garante-se que os upgrade correm sempre numa base de dados estável, cujo conteúdo e código é claramente conhecido, e assim minimizam-se os problemas.

O seguinte diagrama exemplifica o *workflow* da gestão destes ambientes.



1. As parametrizações que sejam válidas para a versão seguinte devem ser feitas tanto nos ambientes Custom como no ambiente Milestone, para que persistam após nova cópia.
2. Esta cópia faz com que as alterações nos ambientes Custom sejam perdidas.
3. Caso haja desenvolvimentos relevantes nos ambientes Custom que tenham de se manter, devem ser introduzidos no produto, reflectindo-se nos scripts de upgrade posteriores.

Fig 15 – *workflow* do versionamento do ambiente de demonstrações

3.2.3. Problemas de gestão de traduções

O ambiente de demonstrações, assim como todo o Alert®, por vezes carece de traduções, ou tem traduções dessincronizadas, ou incorrecções, ou de fraca qualidade. Este problema não é só específico do ambiente de demonstrações, embora seja o primeiro local onde estes são visíveis, pois o ambiente de demonstrações é geralmente o primeiro ambiente onde são necessárias traduções.

Foi elaborada uma proposta para a resolução de muitos dos problemas existentes, mas como não se enquadra nos objectivos deste projecto, está disponível em anexo.

4. Especificação de requisitos do Alert® Reset

4.1. Funcionalidades Gerais

Os requisitos da aplicação de reset resumem-se ao seguinte:

- Limpeza de dados de demonstrações anteriores, mais especificamente, remoção dos episódios previamente utilizados noutras demonstrações, e todo o *workflow* associado.
 - O utilizador deve escolher qual os ambientes que quer limpar;
 - O utilizador deve escolher episódios que não devem ser apagados, para manter histórico
- Alteração de data de episódios e de toda a informação registada nos mesmos
- Simular sistemas externos administrativos, para criação e gestão de pacientes, e criação e gestão de episódios.
- Mecanismo para gerar dados na aplicação. Por exemplo, uma formação de enfermeiros requer pacientes com episódios activos e tratamento prescrito mas não executado.
- Autenticação do utilizador, opcional.

4.2. Características dos utilizadores finais

4.2.1. Equipas comerciais e de demonstrações

As equipas comerciais são constituídas por indivíduos pertencentes à ALERT, subsidiárias, ou a distribuidoras de software com as quais a ALERT tem acordos de distribuição. Estas equipas realizam demonstrações da aplicação em clientes, em reuniões, em feiras de tecnologia, etc.

Estas equipas apresentam o Alert® em diferentes mercados, com diferentes necessidades e conteúdos. Por vezes as demonstrações não precisam de simular o caso real com exactidão, podendo-se registar qualquer tipo de informação na aplicação, pois o que é

importante é demonstrar as funcionalidades, *workflow*, *design*, etc. Noutras situações pode-se que seguir um guião com conteúdo muito específico.

Durante demonstrações, geralmente só há uma pessoa a utilizar o Alert®.

4.2.2. Equipas de formação e *key-users*

As equipas de formação são constituídas por indivíduos, que dão formação a futuros utilizadores do Alert®. Estas realizam as suas actividades nos clientes, e as formações requerem um ambiente adequado ao contexto da formação. Por exemplo, numa formação de obstetras, necessita-se de mulheres grávidas nas grelhas de pacientes, uma para cada formando e uma para o formador. Numa formação de enfermagem, deve-se ter vários pacientes nas grelhas com tratamento requisitado, para os enfermeiros o executarem.

Como as equipas de formação realizam o seu trabalho sempre no contexto da mesma instituição, é necessário ter a instalação do Alert® devidamente configurada e adaptada à realidade dessa.

Para além dos formadores, existem os *key-users*, que são utilizadores finais da aplicação, cuja formação e conhecimento de Alert® são bastante desenvolvidos. Estes *key-users* estarão presentes em mercados onde a ALERT tem menos presença, ou realizou acções prévias, e estes também realizarão acções de formação com novos utilizadores.

Durante as formações, os formandos e os formadores usam o Alert® simultaneamente.

4.2.3. Observações

Embora tenham metodologias de trabalho bastante diferentes, as equipas usufruem dos mesmos requisitos. O nível de detalhe com que querem usar a aplicação depende contexto do seu trabalho.

Denote-se que a aplicação será usada por inúmeras pessoas de diferentes capacidades técnicas e de diferentes culturas, pelo que a aplicação deverá ter bastante em conta conceitos como acessibilidade, localização, e simplicidade.

4.3. Actores

Existem dois actores:

- **Utilizador** – representa o utilizador final da aplicação que irá usufruir de todos os requisitos da aplicação.
- **Administrador** – o administrador será responsável pela criação e edição de utilizadores da aplicação. O administrador também é um utilizador e herda todos os requisitos deste.

O utilizador irá usufruir de todos os requisitos da aplicação, à excepção da gestão de utilizadores, pois esse requisito é reservado ao administrador.

A identificação do utilizador da aplicação como “utilizador” ou “administrador” implica uma prévia autenticação.

4.4. Requisitos não-funcionais

4.4.1. Fiabilidade

O principal requisito não funcional é a fiabilidade.

A aplicação será o ponto central de gestão de toda a informação na instalação do Alert® adjacente. Os utilizadores finais irão usar a aplicação para preparar todos os cenários de utilização da aplicação, para fazer limpeza de informação desnecessária, e preservação de informação útil. O principal objectivo da aplicação, é permitir fazer a gestão destes dados, pelo que não podem ser introduzidos erros na aplicação, pois tal poderia comprometer seriamente a demonstração da aplicação.

4.4.2. Eficiência.

Existem casos de uso que internamente implicam alterar grandes conjuntos de informação, como alterar a data de um episódio, e todos os registos feitos neste, ou remover episódios activos de um dado ambiente. Em várias situações, há a necessidade de realizar estas acções antes ou durante demonstrações, como em feiras. Por isso, estes processos terão que decorrer de forma eficiente.

4.5. Requisitos funcionais

4.5.1. Visão geral

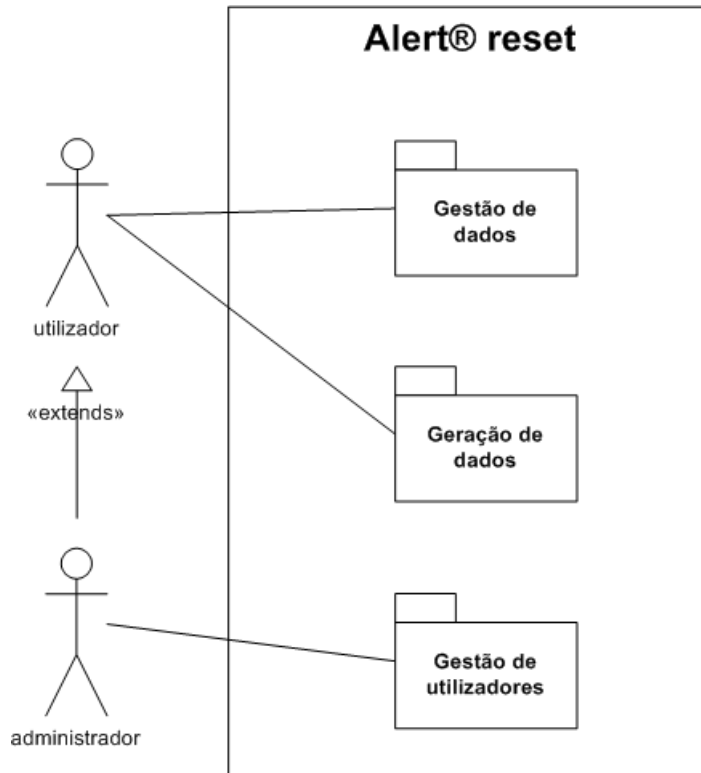


Fig. 16 – Módulos da aplicação de reset

4.5.1.1. Gestão de dados

Existem dois conjuntos de dados que a aplicação deverá permitir gerir: pacientes e episódios. Embora estes dois conjuntos de informação sejam geridos por ferramentas distintas, estas serão bastante semelhantes em funcionalidade, e integradas no mesmo módulo.

A aplicação irá dispor de uma interface que permite navegar entre as diferentes instituições e ambientes dentro destas (urgência, internamento, consulta, etc.). Dentro de cada um destes ambientes, o utilizador pode consultar detalhes sobre cada episódio, alterar as datas de início dos episódios, bloquear episódios para não serem apagados. Desta forma consegue-se preservar histórico.

A aplicação irá também fornecer listagens de todos os pacientes, e episódios associados, activos e inactivos, agrupados por mercado. Adjacente a esta listagem a aplicação deverá permitir a criação de pacientes.

4.5.1.2. Geração de dados

Existe a necessidade de gerar episódios na aplicação, de gerar informação e tratamento destes episódios, pois, a introdução desta informação no Alert® manualmente pelo demonstrador ou formador poderia não ser oportuna, interferir com uma demonstração a decorrer, ou repetitiva. A informação que se irá gerar obedecerá a guiões predefinidos, e deverá representar com exactidão casos clínicos reais.

Por exemplo: um *workflow* pode criar um episódio de urgência de um indivíduo adulto, triá-lo com um fluxograma “Catástrofe primária”, mover o paciente para um serviço de ortopedia, atribuir um diagnóstico relacionado com problemas ortopédicos, e a respectiva medicação. Depois durante uma formação de enfermeiros, estes partiam deste ponto para efectuar o tratamento do paciente.

Por isso, a aplicação irá suportar uma funcionalidade que permite ao utilizador especificar um *workflow* de geração informação. Este *workflow* será composto por várias tarefas que serão executadas sequencialmente. Tarefas podem depender de outras, podem definir uma hierarquia. Por exemplo, o utilizador pode criar um *workflow* que faça uma limpeza de dados num certo ambiente, crie 6 episódios, prescreva medicação num episódio, requisite um transporte noutra. Este *workflow* então é gravado na base de dados, para poder ser activado, e reutilizado.

O utilizador, com o toque de um botão, depois poderá executar qualquer *workflow*.

4.5.1.3. Gestão de utilizadores

A aplicação estará disponível em ambiente controlado, e será usada por um conjunto de utilizadores predeterminados. Embora a segurança não seja um requisito da aplicação, a autenticação apenas previne o uso indevido da aplicação, e permite registar o que cada utilizador fez nesta.

Existem duas situações em que esta aplicação estará disponível:

- Numa máquina de acesso geral não restrito, em que a autenticação garante que apenas indivíduos legítimos têm acesso a esta;
- Numa máquina de demonstrações, tipicamente um computador portátil, que é utilizada por poucas pessoas, ou só por uma, na qual a autenticação é desnecessária pois só é possível usá-la se se tiver acesso físico;

Por isso a autenticação estará activa opcionalmente.

4.5.2. Requisitos do módulo “Gestão de dados”

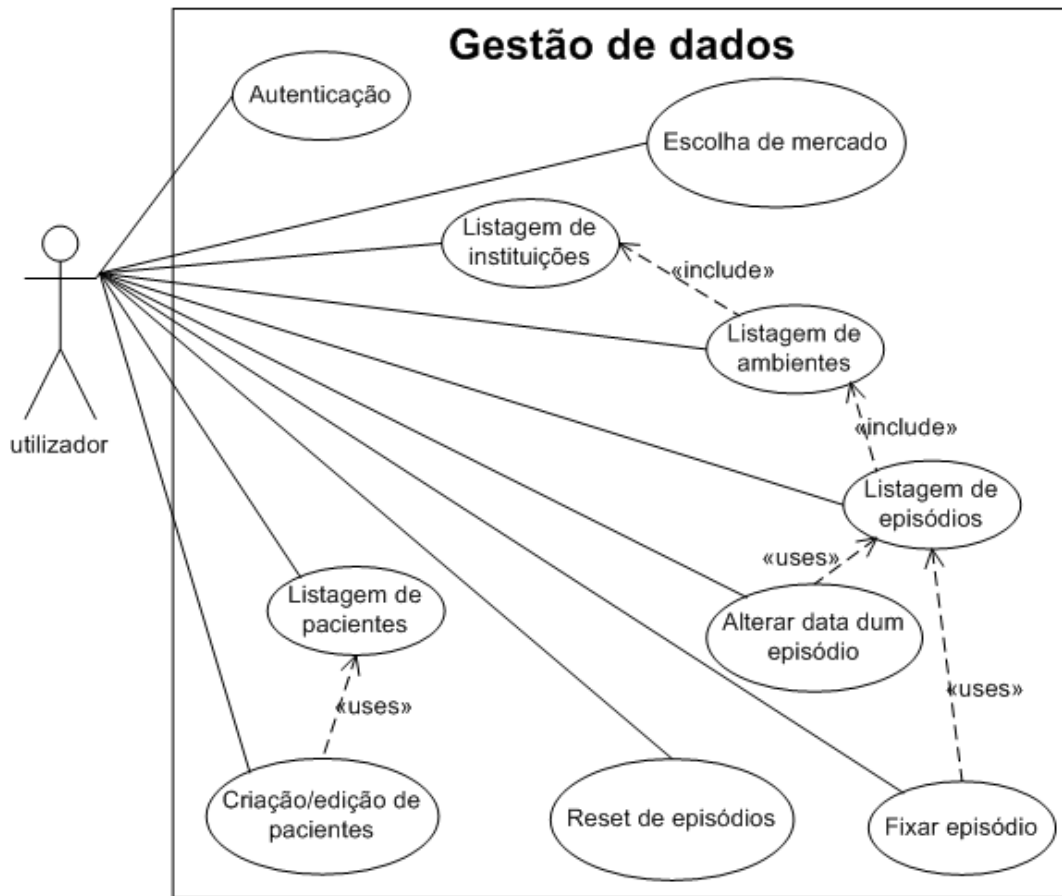


Fig. 17 – Casos de uso do módulo “Gestão de dados”

4.5.2.1. Autenticação

A autenticação valida a identidade do utilizador perante a aplicação. A autenticação é o único requisito acessível ao utilizador não autenticado.

A autenticação é feita com nome de utilizador e palavra-passe. O utilizador deve obter acesso à aplicação, caso se autentique com informação adequada.

4.5.2.2. Escolha de mercado

O mercado identifica em que contexto se está a trabalhar: mercado português, americano, espanhol, etc. A escolha deste irá filtrar dados na aplicação para o utilizador se poder concentrar apenas num conjunto de informação relevante.

Os mercados disponíveis para escolha estarão parametrizados na base de dados, juntamente com as instituições.

A aplicação deverá filtrar informação relevante apenas ao mercado escolhido, e deverá mostrar toda a informação disponível caso o utilizador escolha visualizar informação de todos os mercados.

4.5.2.3. Listagem de instituições

A aplicação deverá fornecer uma listagem de todas as instituições presentes na aplicação, que permita aceder a dados dentro destas.

4.5.2.4. Listagem de ambientes

A aplicação deverá fornecer uma listagem de todos os ambientes existentes numa instituição pré-seleccionada. Entende-se por ambientes a urgência o internamento, a consulta, etc.

4.5.2.5. Listagem de episódios

A aplicação deverá fornecer uma listagem de todos os episódios existentes num dado ambiente de uma instituição, organizada por pacientes e ordenável.

4.5.2.6. Alterar data de um episódio

A aplicação deverá permitir especificar uma data de início para um episódio já existente, e todos os registos efectuados neste serão alterados para manter o intervalo de tempo constante, relativo ao início do episódio. Este caso de uso permite ao utilizador criar vários episódios para o mesmo paciente no Alert®, e espaçar os episódios no tempo, para simular uma cronologia.

O utilizador deverá especificar um episódio e uma data de início, e assim e toda a informação registada neste deverá ser deslocada linearmente no tempo, mantendo os intervalos entre o início do episódio e os dados registados constantes.

4.5.2.7. Reset de episódios

Como para realizar demonstrações e formação se deve ter sempre um conjunto de episódio limpos na aplicação, sem informação prévia, a aplicação deverá implementar uma funcionalidade que apaga episódios existentes de utilizações anteriores. O utilizador poderá escolher que ambientes e/ou instituições quer limpar. Como o reset poderá ser uma operação demorada, este deverá ser assíncrono.

O utilizador deverá indicar que ambientes e/ou instituições pretende limpar, e após o reset, os ambientes escolhidos não poderão ter nenhum episódio activo ou inactivo. A única forma de preservar um episódio é fixá-lo.

4.5.2.8. Fixação de episódios

Como o reset de episódios apaga um conjunto predeterminado de episódios, o utilizador pode escolher episódios para preservar. Estes episódios depois serão disponibilizados na aplicação como histórico do paciente.

O utilizador escolherá um conjunto de episódios, e durante o reset, os episódios fixados nunca podem ser apagados.

4.5.2.9. Listagem de pacientes

A aplicação deverá disponibilizar uma listagem de todos os pacientes disponíveis por mercado. A filtragem de pacientes por mercado permitirá distinguir pacientes com nomes na língua nativa desse mercado, com histórico nas instituições desse mercado.

Para cada paciente, a aplicação deverá disponibilizar a lista de todos os episódios que este paciente tem na aplicação, activos e inactivos.

4.5.2.10. Criação/edição de pacientes

O registo de pacientes nas instituições de saúde, e admissão destes, tipicamente é realizado em sistemas externos ao Alert®. Como o Alert® actualmente não suporta esta funcionalidade, a aplicação irá permitir ao utilizador consultar listagens de pacientes e criar novos, ou editar a informação de pacientes já existentes. A aplicação deverá suportar o *upload* de fotografias.

Após a criação dum paciente, este deverá estar disponível para criação de episódios, e deverá ser visível no Alert®. A criação de episódios será abrangida pelo módulo de geração de dados.

4.5.3. Requisitos do módulo “Gestão de utilizadores”

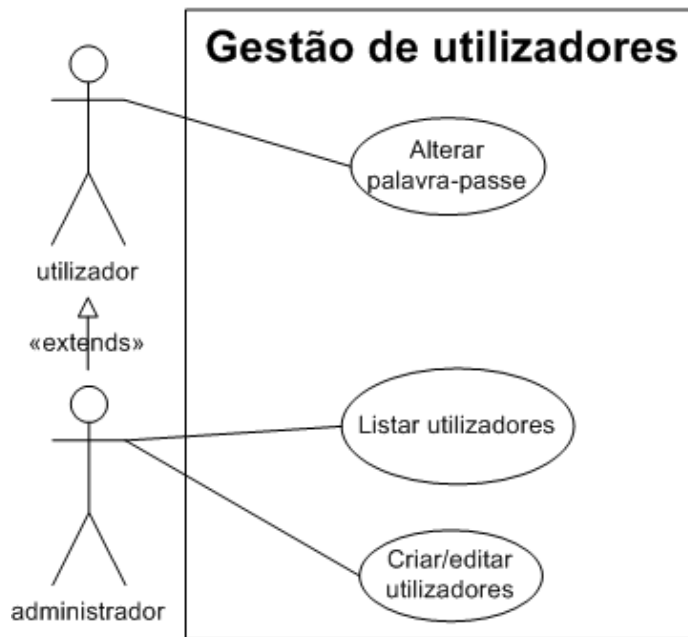


Fig. 18 – Casos de uso do módulo “Gestão de utilizadores”

4.5.3.1. Listar utilizadores

O administrador deverá ter acesso a uma listagem de todos os utilizadores da aplicação, com informação como nome de utilizador e data de último acesso.

4.5.3.2. Criar e editar utilizadores

O administrador da aplicação terá a possibilidade de criar novos utilizadores, e/ou redefinir informação de outros utilizadores, como palavra-passe, ou bloquear contas.

Os utilizadores introduzidos no sistema pelo administrador deverão ter a possibilidade de se autenticarem e usarem a aplicação, caso a conta destes não esteja bloqueada.

4.5.3.3. Alterar palavra-passe

Cada utilizador pode alterar a sua palavra-passe após se autenticar.

A alteração implica a introdução da palavra-passe original, e da nova repetida para confirmação. Caso o utilizador se esqueça da sua palavra passe, então o administrador pode alterar a palavra-passe ao utilizador.

4.5.4. Requisitos do módulo “Geração de dados”

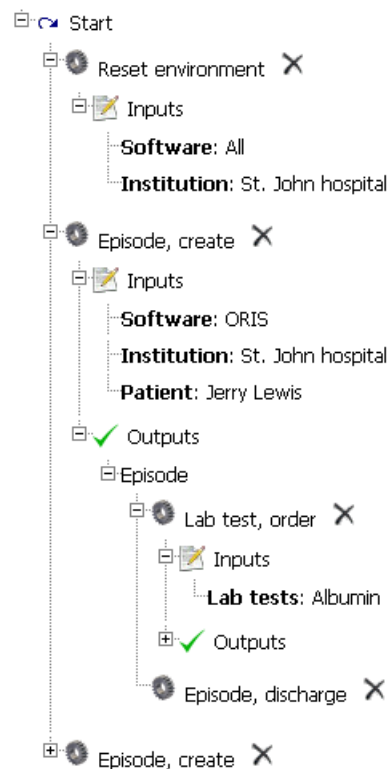


Fig. 19 – Casos de uso do módulo “Geração de dados”

Um *workflow* de geração de dados representa uma sequência de passos que a aplicação deve executar sequencialmente para gerar uma quantidade de dados. Um *workflow* é constituído por tarefas que tem parâmetros de entrada e de saída. As tarefas podem ser associadas em árvore hierarquicamente. Uma tarefa pode só ser executada quando as tarefas anteriores estiverem concluídas. Uma tarefa que tenha outras subjacentes deverá ter como output a informação necessária para o input das tarefas filhas.

As tarefas devem ser associadas de forma lógica para simular a utilização do Alert®, pois estas serão executadas dessa forma.

Exemplo:

Fig. 20 – *Workflow* de geração de dados

4.5.4.1. Listar *workflows* de geração de dados

A aplicação deverá fornecer ao utilizador uma listagem de todos os *workflows* existentes, com informação como nome, mercado, criador, data de criação, data de última modificação, data de última utilização e número de utilizações.

4.5.4.2. Criar/editar *workflows*

O requisito mais complexo e mais importante da aplicação, será a possibilidade do utilizador criar os seus próprios *workflows* de geração de dados. O utilizador terá a possibilidade de definir que tarefas podem ser executadas, com que ordem, hierarquia e parâmetros de entrada. No entanto, estas tarefas relacionam-se com algumas restrições.

Algumas tarefas requerem contexto para executar, e por isso só podem ser filhas de outras determinadas tarefas, para poderem ser executadas. Caso a principal tarefa de um *workflow* dependa de contexto, este não pode ser executado directamente, mas pode ser executado dentro de outro *workflow*, desde que este segundo lhe forneça o contexto necessário. Uma tarefa nunca pode alterar o contexto para uma tarefa irmã (no mesmo nível da árvore), apenas o pode alterar para tarefas filhas.

Por exemplo: a tarefa requisitar uma análise laboratorial, requer que esta seja executada no contexto de um episódio. Por isso, pode-se executar a tarefa que cria episódios, e sobre um episódio resultante (output da tarefa), executar a tarefa que requisita

a análise. Ou, pode-se criar um *workflow* cuja principal tarefa é a requisição de uma análise de laboratório, mas como esta depende do contexto de um episódio, este *workflow* não poderá ser executado individualmente. Este apenas poderá ser executado dentro de outros *workflows*, como se tratasse de outra tarefa. Desta forma pode-se agrupar e reutilizar *workflows*.

A aplicação deverá impedir a geração de *workflows* inválidos, ou seja, cuja passagem de informação sobre contextos seja incompatível entre as tarefas mãe e filha.

A funcionalidade será genérica e irá conter um conjunto variável de tarefas. O utilizador não pode criar novas tarefas. Estas serão desenvolvidas e integradas na aplicação, pois criar tarefas implica interagir com a lógica de negócio do Alert®.

Após um *workflow* estar criado, este deverá estar disponível para execução na grelha de *workflow*.

4.5.4.3. Executar *workflows*

O utilizador poderá escolher um *workflow* da grelha respectiva, e executá-lo. Como um *workflow* poderá demorar uma quantidade significativa de tempo a executar, esta execução deverá ser assíncrona.

Caso as tarefas executadas alterem dados passíveis de ser visualizados no Alert®, as modificações deverão ser igualmente visíveis.

Dado o assincronismo deste requisito, a interacção é descrita pelo seguinte diagrama.

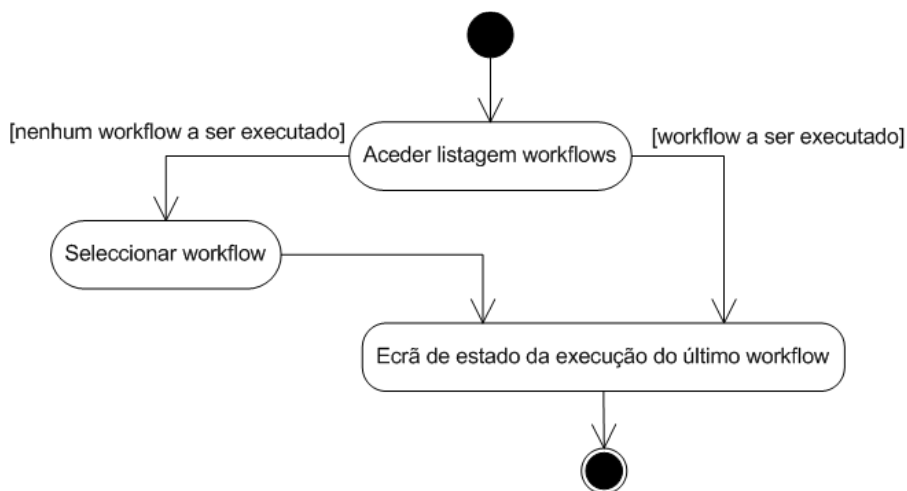


Fig. 21 – Interacção com a funcionalidade execução de *workflows*

4.5.4.4. Executar reset de episódios

Este caso de uso é um caso particular do requisito “Executar *workflows*”. O reset de episódios implica apagar todos os episódios activos de um ambiente (instituição e software), para posteriormente ser gerada de novo informação nestes ambientes.

Todos os episódios activos dos ambientes escolhidos deverão ser removidos da aplicação. Logo isso implica que, se um utilizador entrar no Alert®, este não deverá ver nenhum episódio activo.

4.5.4.5. Criar episódios

Este caso de uso é um caso particular do requisito “Executar *workflows*”, e é descrito à parte pois é a tarefa mais importante que a funcionalidade deverá suportar, juntamente com o reset de episódios. O utilizador escolherá uma instituição, ambiente e um paciente para o qual será gerado um novo episódio.

Após a execução desta tarefa, os episódios criados deverão estar visíveis no Alert® nos ambientes e instituições em que foram criados.

4.5.4.6. Gerar informação em episódios

Este caso de uso abrange um conjunto de casos particulares do requisito “Executar *workflows*”, e complementa a criação de episódios, e o reset. É necessário em formações por vezes criar episódios com informação já disponível. A criação de informação pode resultar na requisição de tratamento, registos de texto livre ou templates, prescrição de medicação, etc.

O utilizador deverá colocar todas as tarefas deste tipo sob a criação de episódios, para o *workflow* “saber” qual o episódio onde tem que ser gerada informação. Depois o utilizador deverá executar um *workflow* que inclui estas tarefas.

Caso as tarefas executadas alterem ou criem dados passíveis de ser visualizados no Alert®, as modificações deverão ser igualmente visíveis.

4.6. Observações

Após as reuniões com vários elementos da empresa, ligadas a demonstrações comerciais, interessadas neste trabalho, obteve-se uma lista de requisitos bastante específica, que cobria casos de uso limitados.

Para conseguir abranger mais casos de uso possíveis, e preparar a aplicação para o futuro, os requisitos aqui especificados são bastante genéricos e deixam em aberto bastantes possibilidades sobre o resultado final, pois pretende-se que a avaliação mais detalhada dos requisitos seja feita iterativamente durante o desenvolvimento, em conjunto com o feedback contínuo de alguns dos utilizadores finais da aplicação.

A lista de requisitos proposta satisfaz completamente os casos de uso inicialmente propostos para esta aplicação.

5. Desenvolvimento

5.1. Arquitectura lógica, integração com o Alert®

A suite de produtos Alert® abrange diferentes tecnologias e estruturas. A aplicação Alert® reset incide apenas sobre a parte clínica, ou seja, o Alert® PFH.

O Alert® PFH usufrui de uma arquitectura lógica de 3 camadas: base de dados Oracle, servidor aplicacional Tomcat (que age como proxy entre a interface cliente e a base de dados), e aplicação cliente feita em Adobe Flash.

O Alert® reset irá necessitar de integração da base de dados com o Alert® PFH, por isso a camada de dados será partilhada.

Actualmente, a aplicação cliente em Flash do Alert®, requer a utilização de um ficheiro executável, e a disponibilização de um ficheiro de configuração local. O reset será acedido a partir do *web browser* para simplificar o acesso à aplicação.

Para o cliente poder aceder à aplicação, esta precisa de ser disponibilizada por HTTP, ou seja, a base de dados será englobada por um servidor aplicacional que disponibilize a parte do cliente da aplicação, e permita a comunicação nos dois sentidos.

Para reaproveitar o *know-how* e infra-estruturas existentes, o servidor será também uma instância de Tomcat. Pelo mesmo motivo a interface será feita em Flash.

Resumidamente, o Alert® Reset irá usufruir também de uma organização em 3 camadas, base de dados Oracle, servidor Tomcat, e interface em Flash, acessível através de *web browser*.

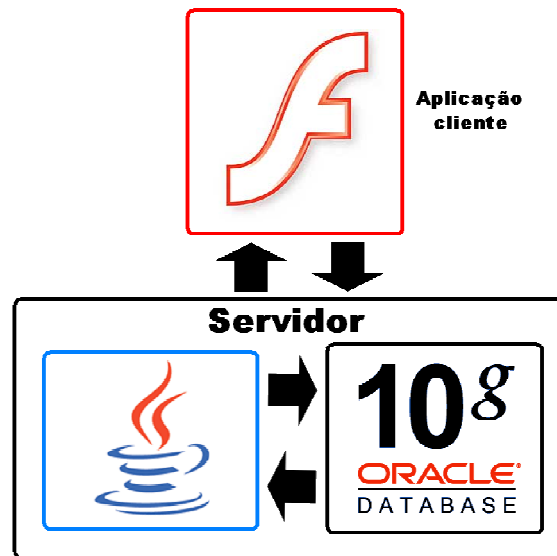


Fig. 22 – Arquitetura lógica do Alert®

5.2. Arquitetura física

Tal como o Alert® PFH, o Alert® Reset terá uma disposição física em que um utilizador num posto de trabalho acede à aplicação através de uma aplicação cliente. Estes podem estar em máquinas diferentes, ou na mesma máquina. O servidor pode ser disponibilizado através de uma máquina virtual que corre numa máquina comercial, e se têm acesso a essa máquina virtual localmente. Esta última configuração é típica nas máquinas de demonstrações.

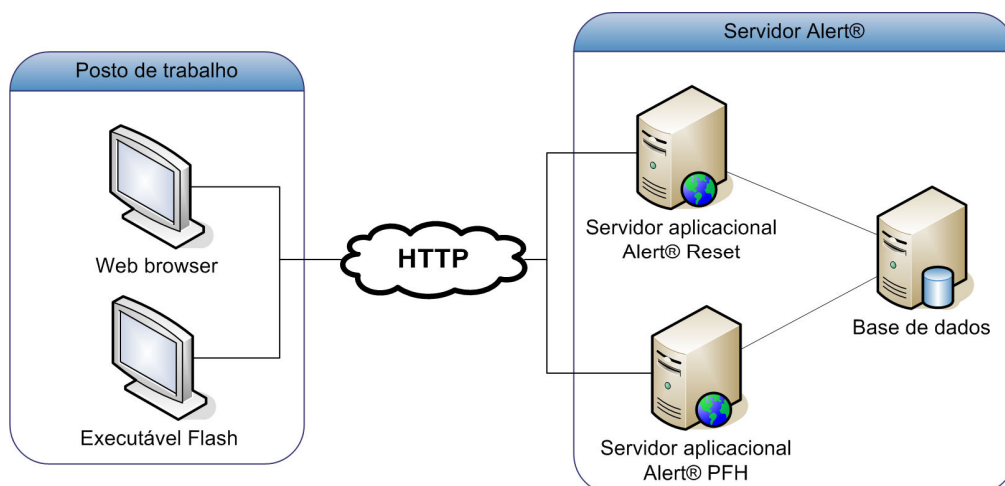


Fig. 23 – Arquitetura física

5.3. Camada de interface de utilizador

Para permitir facilmente a utilização de vários tipos de alfabetos (latino, asiáticos, etc) a interface de utilizador será desenvolvida toda em Flash 9 ou superior e Actionscript 3, e ir-se-á usar *device fonts*, ou seja, os tipos de letra não serão embebidos nos ficheiros resultantes, mas serão obtidos na altura da execução, a partir do sistema operativo.

A interface será composta por vários ecrãs, que representam unidades lógicas de funcionalidade. A forma como estes ecrãs se relacionam será programa nesta camada, e não na camada de base de dados, como no Alert®, pois a aplicação de reset não tem a necessidade de se criar vários perfis de utilizador de forma genérica e flexível. No entanto, a base de dados terá que validar sempre as permissões do utilizador para realizar uma determinada acção.

A aplicação irá dispor duma interface com navegação estilo *wizard*, em que o utilizador confirma passos sequenciais, podendo voltar atrás. Esta abordagem permite ao utilizador ter uma melhor percepção da sequência de passos que a aplicação está a realizar, facilita a aprendizagem, limita a informação disponível a cada passo, e permite a passagem de informação de ecrã para ecrã.

Os ecrãs da aplicação estarão organizados se seguinte forma:

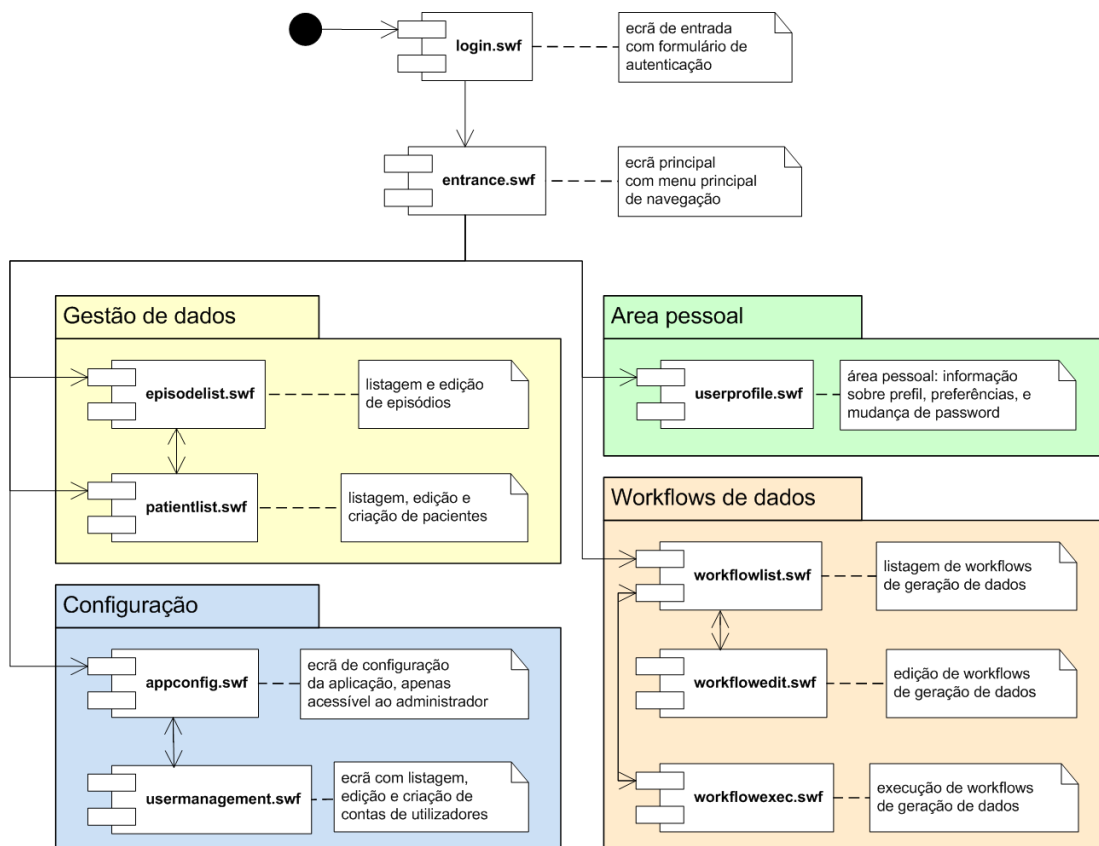


Fig. 24 – Diagrama com ecrãs que compõem a interface de utilizador.

O seguinte diagrama demonstra o fluxo de navegação na aplicação:

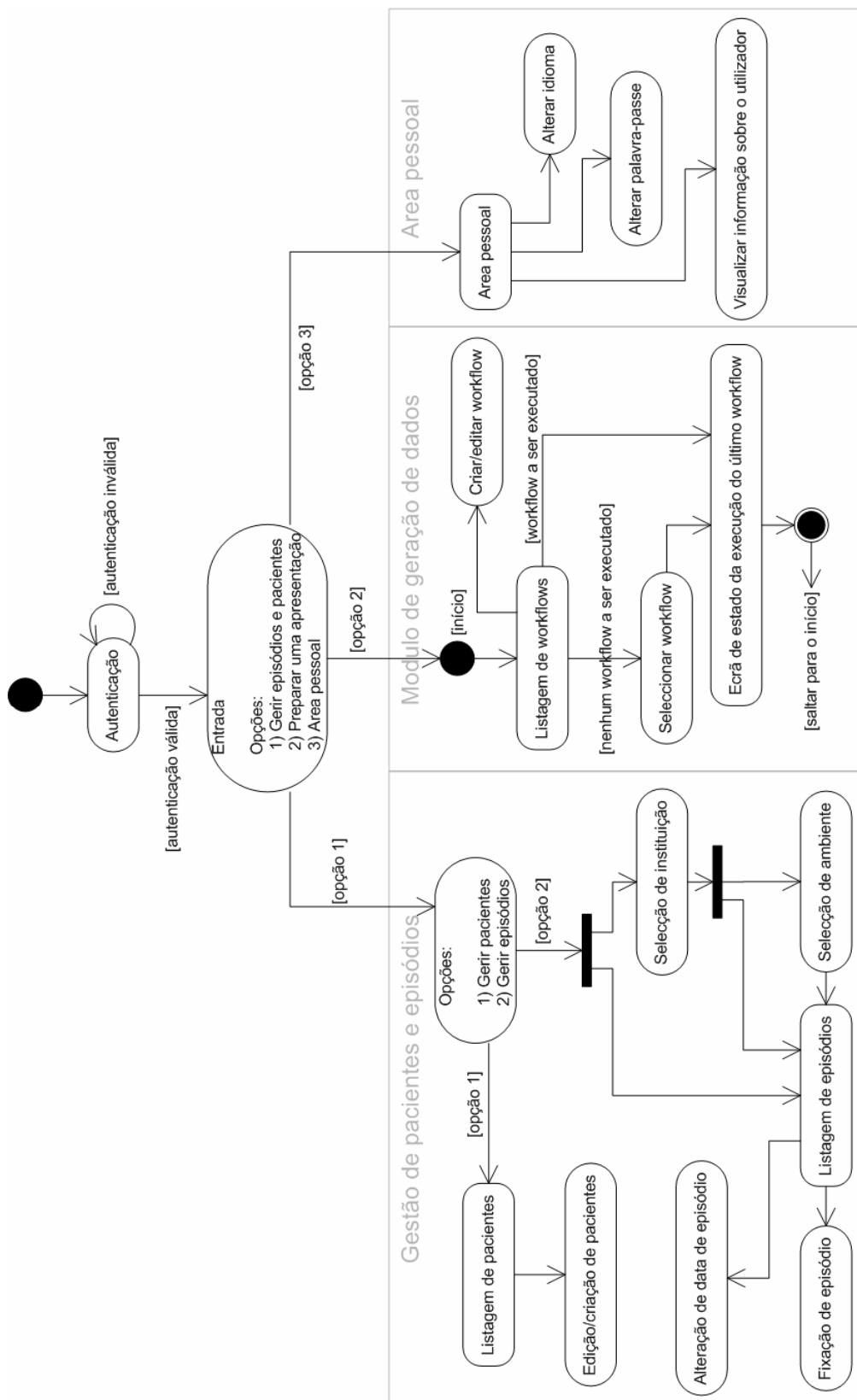


Fig. 25 - Diagrama que exemplifica a navegação na aplicação

O utilizador ao abrir a aplicação vê o ecrã de login (login.swf) que lhe pedirá as credenciais de utilizador para se poder autenticar. Após a autenticação com sucesso, aparece o ecrã de entrada no ambiente de trabalho (entrance.swf) que tem um conjunto de opções que o utilizador pode escolher, sobre a próxima acção a realizar. Este pode escolher 4 opções, que são:

- Preparar uma apresentação;
- Gerir episódios;
- Gerir pacientes;
- Ir para a área pessoal.

A opção “Gerir pacientes” apresentará ao utilizador um novo ecrã com uma grelha dos pacientes disponíveis na aplicação, e com um formulário que permita a edição e criação de pacientes, com fotografia. O utilizador poderá editar todos os atributos pessoais do paciente e a fotografia. A grelha de pacientes deverá filtrar por mercado. O facto de um paciente pertencer a um mercado será determinado pelo utilizador, que associará o paciente ao mercado preferido.

Patient t	Description	Gender	Date of birth	Number of episodes	Market
David Jackson	Orthopedic problems	Male	23-12-1981	2	USA
Jerry Lewis	Diabetes	Male	13-05-1972	3	USA

Episode type	Institution	Start date	Status	Locked
ORIS	HDC	02-04-2008 01:56	Closed	Yes
INPATIENT	HDC	24-04-2008 03:39	Closed	No
CARE	HPCC	21-02-2008 05:15	Closed	No

Mary King	Cardic history	Female	02-02-1956	2	USA
Michael Bloom	Skin cancer	Male	14-07-1978	2	USA

Patient information

Name: Market:

Description: Birth date:

Gender:

Fig. 26 – Grelha de pacientes do protótipo do Alert® Reset

A opção “Gerir episódios” apresentará ao utilizador um ecrã (episodelist.swf) com uma vista dividida em 3 secções:

- Uma árvore que agrega instituições e os ambientes disponíveis nesta
- Uma grelha que mostra os episódios disponíveis na instituição e ambiente em causa, que por defeito mostra todos.

- Um formulário que permite editar informação do episódio. Actualmente prevê-se que o episódio possa ser bloqueado, ou desbloqueado, e que a data de início possa ser alterada, para poder recolocar o episódio no tempo.

Este ecrã terá o seguinte *layout*:

Patient	Episode type	Institution	Start date	Status	Locked
David Jackson	OUTPATIENT	HDC	26-03-2008 02:15	Closed	Yes
David Jackson	EDIS	HSJ	02-04-2008 22:25	Active	No
Jerry Lewis	ORIS	HDC	02-04-2008 01:56	Closed	Yes
Jerry Lewis	INPATIENT	HDC	24-04-2008 03:39	Closed	No
Jerry Lewis	CARE	HPCC	21-02-2008 05:15	Closed	No
Mary King	EDIS	HSJ	06-03-2008 15:51	Closed	Yes
Mary King	CARE	HPCC	16-02-2008 21:24	Active	Yes
Michael Bloom	OUTPATIENT	HDC	09-03-2008 04:42	Closed	Yes
Michael Bloom	INPATIENT	HSJ	15-04-2008 13:32	Active	No

Episode information

Patient: Status:

Institution: Locked: ☐ Yes ☒ No

Type: Start date:

Fig. 27 – Grelha de episódios do protótipo do Alert® Reset

A opção “Preparar uma apresentação” irá abrir um ecrã (*workflowlist.swf*) que lista todos os *workflows* de geração de dados, com várias opções de filtragem e ordenação, como data de criação, criador, nome, descrição, número de execuções, data da última execução.

Após a selecção de um *workflow* e o clique no botão de execução, é apresentado outro ecrã (*workflowexec.swf*) que apresenta o estado actual da execução do *workflow*. A informação será lida através de *polling*, pois o protocolo AMF (usado pelo *Flash Remoting* para transporte dos pedidos RPC através da rede) não suporta eventos com origem no servidor, e a respectiva API do lado do cliente.

O utilizador poderá seleccionar um *workflow* para edição, e então abre-se o ecrã (*workflowedit.swf*) de edição/criação de *workflows* de geração de dados. Este ecrã terá um *layout* partido em 3 secções:

- Uma barra lateral, estilo barra de ferramentas, com todas as tarefas possíveis de execução, com um campo de pesquisa para ser mais fácil encontrar o que se pretende. As tarefas estarão dispostas verticalmente, estilo tabela com uma coluna, em que cada célula contém o nome da tarefa, e uma descrição, possivelmente só visível quando a tarefa tiver foco, para tornar a vista mais leve.
- Uma árvore central que irá representar o *workflow*. Esta árvore irá agregar tarefas hierárquica e ordenadamente. O utilizador poderá ordenar as tarefas entre elas através de *drag-and-drop*. A árvore será construída ao arrastar uma tarefa da barra lateral para cima da árvore. Ao largar a tarefa sobre a árvore, esta será colocada no fim nível da árvore e que é largada.

- Um formulário na secção inferior do ecrã que irá mostrar informação relativa à tarefa actualmente seleccionada. Este formulário será o ponto de introdução de inputs para a tarefa. Por exemplo, a tarefa criar episódio requer um paciente, uma instituição e um tipo de episódio. Por isso este formulário iria disponibilizar uma lista de pacientes para o utilizador escolher, uma lista de instituições e ambientes dentro destas.

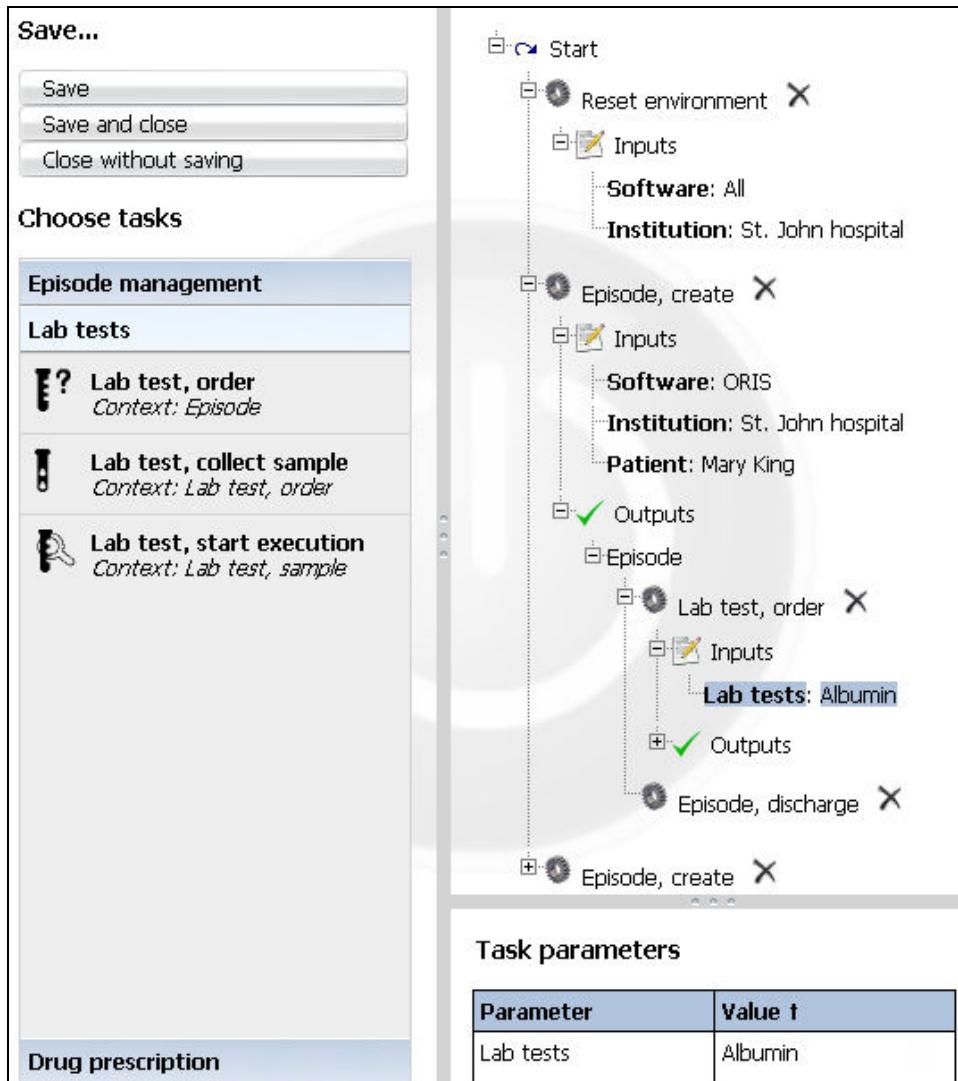


Fig. 27 – Editor de *workflows* do protótipo do Alert® Reset

Finalmente, o utilizador poderá abrir um ecrã (userprofile.swf) com informação sobre a sua conta de utilizador, como registo de entradas na aplicação, preferências de utilizador, incluindo língua da interface, mercado preferencial, e um formulário para alteração de palavra-passe.

5.4. Camada de servidor aplicacional

5.4.1. Arquitectura Aplicacional

A camada de Java na aplicação será responsável por invocar métodos na base de dados, e produzir o output adequado. A lógica de negócio relativa a controlo de sessões estará implementada nesta camada. Toda a outra lógica de negócio estará implementada na base de dados.

5.4.2. Autenticação

Uma vez que a aplicação terá autenticação, embora opcional, isto necessita que haja controlo de sessões de utilização. Após o utilizador se autenticar, é atribuída à sessão um ID de utilizador que será usado para todos os registos de dados efectuados na aplicação. Sempre que seja efectuado um registo na base de dados, deverá ser sempre gravado o ID do utilizador que o criou. Sempre que houver um pedido ao servidor, este irá verificar se a sessão actual está devidamente autenticada, e só perante este cenário, é que irá permitir a execução do pedido. Caso contrário retorna um erro de acesso negado. Esta validação é feita da camada de Java, e o código que a faz é gerado automaticamente pelo JPublisher. Esta validação é transversal a todos *packages* de PL/SQL que sejam acessíveis através de RPC. Por isso, para o utilizador se poder autenticar, haverá um único *package* que não requer autenticação, que será o *package* que realiza essa autenticação.

Caso a autenticação esteja desactivada, será atribuído um ID de utilizador genérico.

5.4.3. Acesso a dados

Como indicado anteriormente, para cada *package* na base de dados, será gerada uma classe com o JPublisher, que implementará métodos que reflectem as funções implementadas no mesmo *package* na base de dados.

O flash invocará funções da base de dados através de RPC, que por sua vez faz pedidos AMF (*Action Message Format*) ao servidor aplicacional. O servidor recebe o pedido e invoca a função pedida. Essa função será representada no servidor através de serviços escritos em Java.

Por exemplo, o seguinte *package* seria representado por:

```
CREATE OR REPLACE PACKAGE pk_my_package IS
  FUNCTION get_data
  (
    i_param_1 IN NUMBER,
    o_param1  OUT VARCHAR
  ) RETURN BOOLEAN;
END pk_my_package;
```

```

public class get_dataOut{
    String o_param1_out;
    Boolean __return;
}

public interface pk_my_packageInterface{
    public get_dataOut get_data(Double i_param_1);
}

public interface pk_my_packageBase implements
pk_my_packageInterface{
    public pk_my_packageBase() { /*...*/ }
    public get_dataOut get_data(Double i_param_1) { /*...*/ }
}

public class pk_my_packageService extends pk_my_packageBase
implements pk_my_packageInterface{
    public get_dataOut get_data(Double i_param_1) { /*...*/ }
}

```

- **pk_my_packageInterface** contém apenas os cabeçalhos
- **pk_my_packageBase** contém lógica de acesso à base de dados
- **pk_my_packageService** Faz validação de input, tratamento de erro, e internamente invoca as funções do **pk_my_packageBase**

5.5. Camada de base de dados

5.5.1.1. Schemas

A camada de dados da aplicação Alert® Reset estará na *schema* ALERTRESET. Esta terá acesso limitado apenas a algumas tabelas (leitura apenas) e *packages* (execução de funções) da *schema* ALERT, onde está localizado o Alert® PFH.

5.5.1.2. Regras para chaves primárias

Todas as tabelas tem que ter uma chave primária com o formato “id_<nome da tabela>”. Este ID pode ser numérico ou alfanumérico. No caso de conteúdo aplicacional, este ID deverá ser alfanumérico, cujo valor será definido manualmente. O valor poderá reflectir o conteúdo que a linha da tabela descreve. No caso de conteúdo gerado pela aplicação, este ID deverá ser numérico e gerado sequencialmente pela aplicação.

5.5.1.3. Regras para datas

Todas as datas em todas as tabelas terão que ser forçosamente do tipo **TIMESTAMP WITH LOCAL TIME ZONE** para preservar fuso horário, e interagir bem

com a mudança de hora do Verão. As colunas deverão ter nome no formato “DT_<descrição>_TSTZ”.

Todas as tabelas terão sempre duas colunas: DT_CREATED_TSTZ e DT_UPDATED_TSTZ que indicam sempre, respectivamente, a data e hora de criação da linha na tabela, e da última actualização. Estas colunas deverão ser preenchidas com um *trigger*.

Excepcionalmente, as colunas podem ser do tipo DATE em casos devidamente ponderados.

5.5.1.4. Regras para textos da interface de utilizador

De forma a suportar diferentes codificações de caracteres, as colunas que armazenam texto escrito pelo utilizador, e as colunas que armazenam textos da interface gráfica, deverão ser do tipo NVARCHAR2.

5.5.1.5. Organização da lógica de negócio

Todo o código da lógica de negócio deverá estar agrupado em funções dentro de *packages* na base de dados.

O *packages* seguem a nomenclatura “PK_<descrição>” em que “descrição” será uma expressão que represente o conteúdo do *package*. O uso de funções individuais é desencorajado, a não ser que sejam funções bastante genéricas para uso no resto do PL/SQL, pois estas não serão acessíveis pela camada de Java.

5.5.1.6. Tratamento de erro

De forma a localizar situações de erro na aplicação, todas as funções deverão ter um bloco *EXCEPTION* que captura todos os erros. O tipo de retorno deverá ser *BOOLEAN*, em que *TRUE* representa sucesso, e *FALSE* erro. Este bloco deverá fazer *ROLLBACK* caso a função tenha modificado tabelas. Deverão ser gravados, na funcionalidade de *logging*, os erros ocorridos. A função em causa, retorna *FALSE* caso ocorra um erro não previsto. Quando se invocam outras funções, deve-se sempre detectar se essa função retorna também *FALSE*. Caso positivo, a função actual termina a execução, propaga a mensagem de erro para fora e retorna *FALSE*. A mensagem de erro deverá ser sempre o último parâmetro da função e deverá ser do tipo NVARCHAR2.

Exemplo:

```

CREATE OR REPLACE PACKAGE BODY pk_my_package IS
  FUNCTION my_func
  (
    i_param IN NUMBER,
    o_error OUT NOCOPY NVARCHAR2
  ) IS
  BEGIN
    IF NOT pk_other_package.other_func(i_param, o_error)
    THEN
      RETURN FALSE;
    END IF;
    RETURN TRUE;
  EXCEPTION
    WHEN OTHERS THEN
      log_error('PK_MY_PACKAGE.MY_FUNC', SQLERRM);
      o_error := get_message('DB_GENERIC_ERROR');
      RETURN FALSE;
  END;
END;

```

5.6. Modelo de dados

5.6.1. Visão geral

O modelo de dados divide-se em 5 áreas:

- Gestão de contas utilizadores, preferências e permissões;
- *Logging* para *debugging*, erros, e registo de sessões e pedidos http;
- Textos e ícones para a interface de utilizador;
- Gestão de mercados, episódios e pacientes do Alert® PFH;
- *Workflows* de geração de dados;
- Configuração da própria aplicação.

5.6.2. Diagrama do modelo de dados

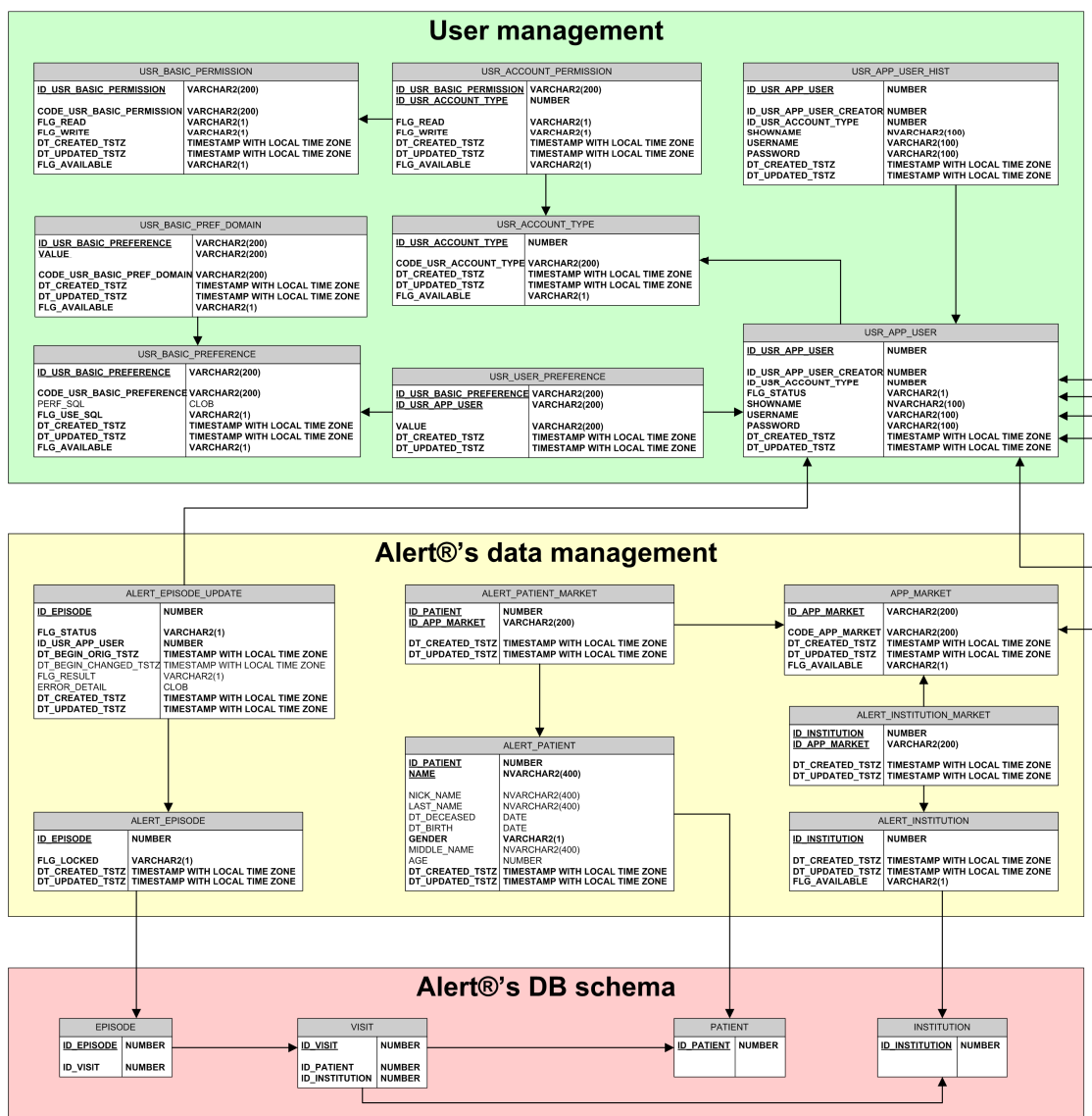


Fig. 28 – Diagrama do modelo de dados (parte 1)

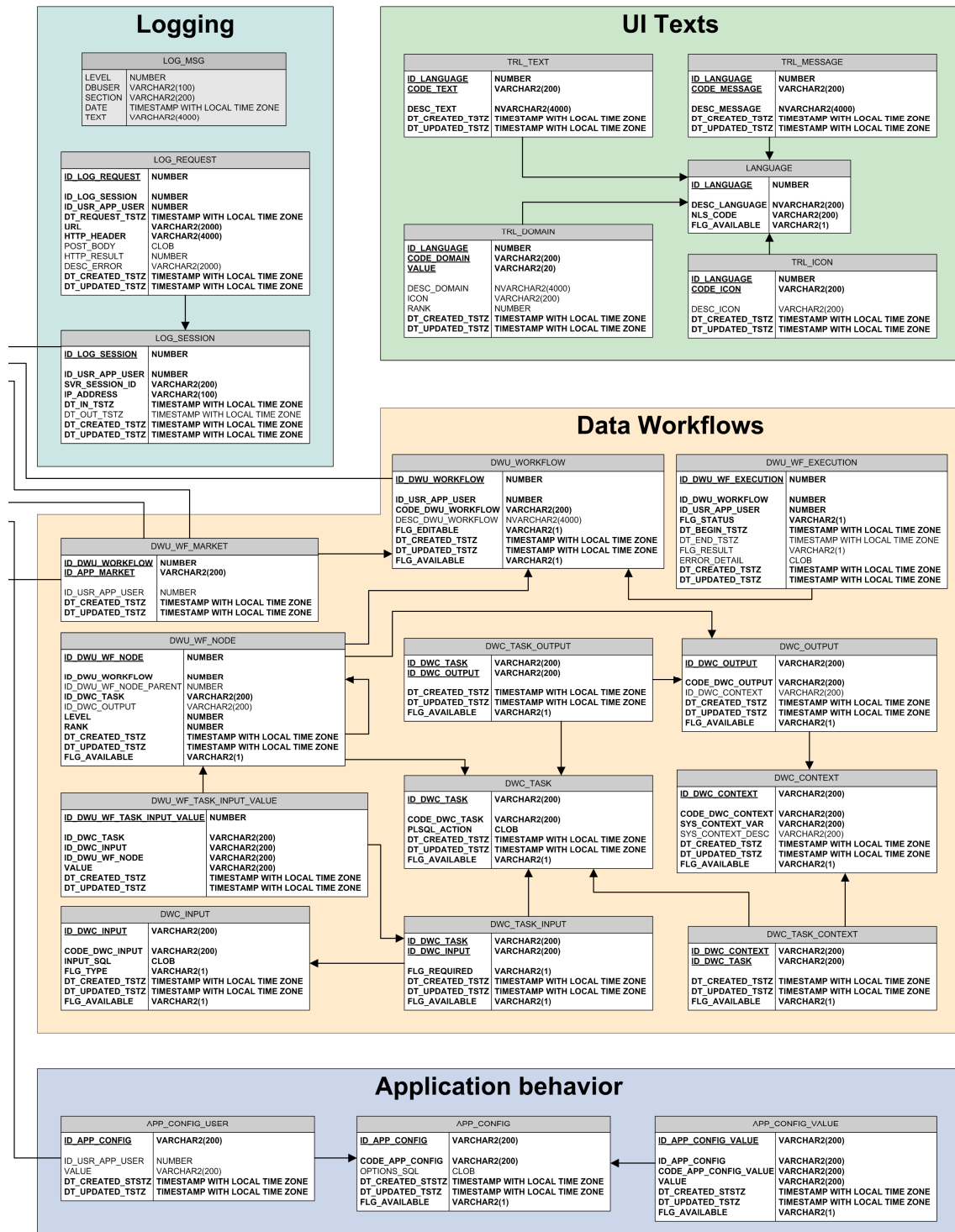


Fig. 29 – Diagrama do modelo de dados (parte 2)

5.6.3. Workflows de geração de dados

Dada a sua complexidade, o modelo de dados da funcionalidade de geração de dados é aqui descrito.

Um *workflow* de geração de dados é composto por várias tarefas, dispostas hierarquicamente.

Uma tarefa tem parâmetros de entrada, indicados pelo utilizador, parâmetros de saída predefinidos, e um contexto sob o qual deve executar.

Um parâmetro de saída pode definir um novo contexto. Por exemplo, a tarefa “criar episódio” gera um episódio de output, que define o contexto de um episódio. Sob esse contexto pode-se associar outra tarefa. Ou seja, obtêm-se duas tarefas, em que uma é filha, pois foi associada a um dos outputs da tarefa mãe.

Estes conceitos são mapeados nas tabelas DWC_TASK (define a tarefa), DWC_CONTEXT (define contexto), DWC_INPUT (define *input* de tarefa) e DWC_OUTPUT (define *output* de tarefa). Ainda existem as respectivas tabelas de ligação, pois as relações entre tarefas, contextos, inputs e outputs são de muitos para muitos.

Através da interface de utilizador, este pode associar tarefas hierarquicamente. A organização de um *workflow* assemelha-se a uma árvore com vários nós por nível. O utilizador instância uma tarefa, que representa um nó da árvore (DWU_WF_NODE). Para essa tarefa, especifica os diferentes valores dos parâmetros de entrada (DWU_WF_TASK_INPUT_VALUE). Já se sabe à partida que a tarefa vai ter parâmetros de saída (DWC_OUTPUT), e que cada parâmetro de saída pode definir um contexto (DWC_CONTEXT), logo, sob este parâmetro de saída pode-se adicionar outra tarefa (DWU_WF_NODE), que depende do contexto local.

Um *workflow* é representado na tabela DWU_WORKFLOW.

A execução de *workflows* não pode ser paralela, pois a lógica de negócio do Alert® actualmente faz modificação na base de dados, e depois faz COMMIT das modificações. Isto provoca dois problemas:

- Durante a execução de um *workflow*, antes deste terminar, as modificações podem ser visíveis por outras transacções na base de dados, o que inclui a possibilidade de ter dois *workflows* a executar sequencialmente e que a informação que geram seja visível pelo outro antes de terminarem. Isto gera um problema de concorrência, pois pode-se ter um *workflow* que gera episódios a executar, e outro que os elimina logo de seguida, por isso o primeiro *workflow* irá falhar, o gerar dados inconsistentes;
- Caso haja uma tarefa que falhe, durante a execução de um *workflow*, como já existem COMMITs intermédios, não é possível fazer um ROLLBACK a todas as alterações produzidas. Por isso ter-se-á que fazer scripts de limpeza de dados.

Estes dois problemas têm duas soluções possíveis:

- Alterar todo o PL/SQL da base de dados para fazer COMMIT apenas condicionalmente;
- Limitar a funcionalidade de execução de *workflows* ao permitir a execução de apenas um *workflow* a cada instante.

A primeira solução é exequível, mas implica exportar todo o código da base de dados para ficheiros de texto, correr scripts com expressões regular para substituir todas as instruções COMMIT por uma execução condicional, e recompilar todo o código. No entanto esta solução é muito possível de erro, e implica modificações profundas.

A segunda situação é mais prática, pois a execução de *workflows* é controlada pela tabela DWU_WF_EXECUTION. Assim, a cada instante só poderá haver um único *workflow* a ser executado. Os *workflows* tem representação em árvore, dada a hierarquia de contextos, mas a sua execução terá que ser sequencial, pois execuções paralelas podem afectar-se mutuamente, pelos mesmos motivos. Como os *workflow* são construídos pelo utilizador, em que este especifica parâmetros de entrada, os outputs destas tarefas serão conhecidos, pelo que a especificação do Alert® Reset não prevê pontos de decisão durante a execução de *workflows*.

As seguintes imagens esboçam o aspecto inicial do *workflow*, organizado em árvore, e o respectivo fluxo de execução:

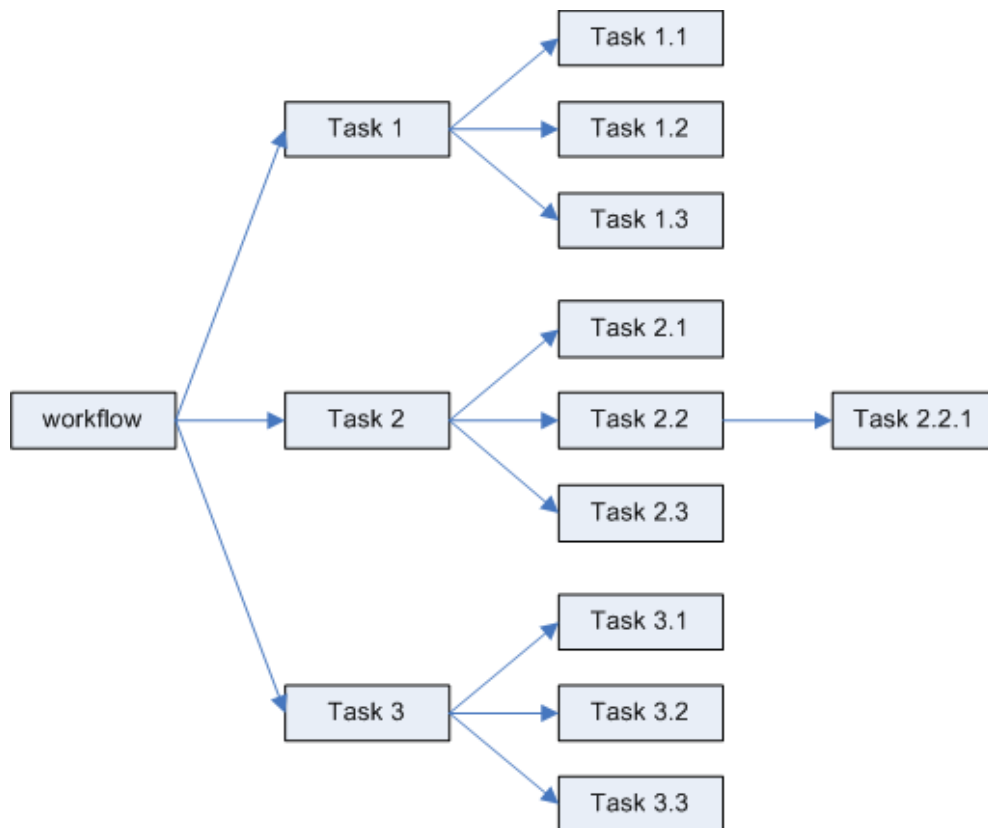
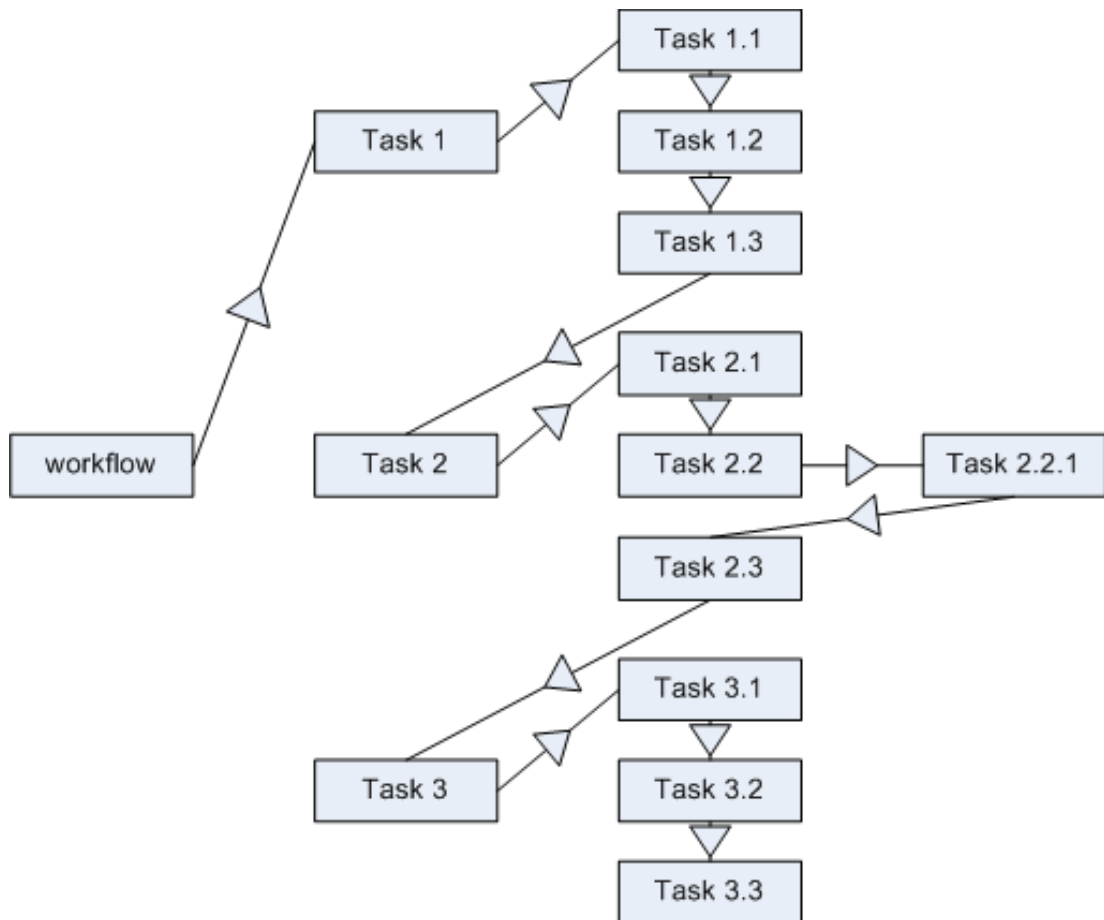


Fig. 30 – Representação hierárquica de um *workflow* do Alert® Reset

Fig. 31 – Fluxo de execução de um *workflow* do Alert® Reset

Quando houver ocorrer um erro durante a execução do *workflow*, este é abortado, e serão corridos scripts de limpeza dos dados criados previamente.

5.6.4. Tabelas

A nomenclatura das tabelas segue a seguinte regra:

- Tabelas do módulo de gestão de utilizadores são prefixadas com USR.
- Tabelas do módulo de *logging* são prefixadas com LOG.
- Tabelas com textos da interface de utilizador são prefixadas com TRL, com a excepção da tabela LANGUAGE.
- Tabelas que mapeiam conteúdo da *schema* ALERT da base de dados, para fazer gestão destes, são prefixadas com ALERT.
- Tabelas que fazem parte da funcionalidade de *workflows* de geração de dados são prefixadas por DWC e DWU. DWC é o prefixo atribuído às tabelas com dados configurados manualmente para serem disponibilizados na

funcionalidade para criação dos *workflows*. DWU é o prefixo atribuído às tabelas cuja informação é introduzida pelo utilizador na aplicação.

- Tabelas que influenciam o comportamento do Alert® Reset, por ex., configurações, são prefixadas por APP.

5.6.4.1. Módulo de gestão de utilizadores

USR_ACCOUNT_TYPE

Esta tabela define um tipo de conta de utilizador, à qual podem ser atribuídas permissões de utilização da aplicação.

USR_BASIC_PERMISSION

Esta tabela define uma permissão que pode ser atribuída a um grupo de utilizadores.

USR_ACCOUNT_PERMISSION

Esta tabela associa uma permissão a um tipo de conta de utilizador.

USR_APP_USER

Esta tabela guarda o registo de uma conta de utilizador.

USR_APP_USER_HIST

Esta tabela guarda um histórico (backup) da tabela USR_APP_USER. Sempre que algum dado é alterado na USR_APP_USER, é efectuado um backup na USR_APP_USER_HIST.

USR_BASIC_PREFERENCE

Esta tabela define uma preferência básica, que o utilizador poderá escolher, para customizar a aplicação.

USR_BASIC_PREF_DOMAIN

Esta tabela guarda um conjunto de valores possíveis para uma dada preferência, caso não haja outra tabela que possa conter valores possíveis. Por exemplo: a escolha de mercado ou língua incide sobre tabelas existentes, enquanto que a escolha se a sessão deve expirar automaticamente não.

USR_USER_PREFERENCE

Esta tabela guarda valor escolhido pelo utilizador para uma preferência. Cabe à lógica de negócio validar estes valores para modelar o seu comportamento.

5.6.4.2. Módulo de *logging*

LOG_MSG

Esta *view* referencia directamente a tabela ALERTLOG.TLOG usada pelo Alert® para efectuar *logging*. O Alert® Reset irá usar a mesma funcionalidade.

LOG_REQUEST

Esta tabela regista todos os pedidos http feitos ao servidor.

LOG_SESSION

Esta tabela regista todas as sessões de utilização do servidor.

5.6.4.3. Módulo com textos da interface de utilizador**LANGUAGE**

Esta tabela guarda os idiomas disponíveis na aplicação.

TRL_DOMAIN

Esta tabela guarda textos associados a diferentes valores que uma dada coluna pode ter na base de dados. Por exemplo, consideremos a tabela DWU_WF_EXECUTION, coluna FLG_STATUS. A coluna CODE_DOMAIN teria o valor “DWU_WF_EXECUTION.FLG_STATUS”, a coluna VALUE teria vários valores possíveis (R,E,F), um por linha da tabela TRL_DOMAIN. O trinómio CODE_DOMAIN,VALUE,ID_LANGUAGE é chave única e define o texto que representa o valor da coluna DWU_WF_EXECUTION.FLG_STATUS.

TRL_ICON

Esta tabela guarda ícones. Como os ícones podem conter texto, por vezes têm-se que os definir baseado numa língua.

TRL_MESSAGE

Esta tabela contém mensagens que não estão associadas a nenhum tipo de conteúdo na base de dados, tipicamente usadas para cabeçalhos de grelhas.

TRL_TEXT

Esta tabela contém textos que estão associados a conteúdos de tabelas. Em várias tabelas, existe conteúdo que é identificado por um ID único, e esse conteúdo por vezes é apresentado na interface de utilizador, logo necessita de tradução. Nas tabelas que contém esse conteúdo, haverá uma ou mais colunas cujo nome tem o prefixo CODE. O valor dessa coluna será replicado na tabela TRL_TEXT e multiplicado pelas línguas existentes. Assim associa-se uma única linha na tabela original, a várias linhas com texto traduzido na TRL_TEXT.

5.6.4.4. Módulo dos *workflows* de geração de dados**DWC_CONTEXT**

Esta tabela define um contexto, sob o qual podem ser executadas tarefas. Um contexto pode ser um episódio, uma análise requisitada, uma colheita, uma triagem, etc, ou seja um conjunto de informação. Esta informação transita da tarefa mais para uma tarefa filha usando a API que o Oracle fornece para contextos: `dbms_session.set_context`.

DWC_INPUT

Esta tabela define um parâmetro de entrada que pode ser associado a tarefas. Os parâmetros podem ser a instituição, o ambiente (urgência, internamento), um paciente, etc.

DWC_OUTPUT

Esta tabela define um output resultante da execução de uma tarefa. Um output pode implicar um novo contexto. Por exemplo, a tarefa “Criar episódio” pode ter como output um episódio, que define um novo contexto, sob o qual pode ser requisitado tratamento.

DWC_TASK

Esta tabela representa uma tarefa que pode ser executada durante a execução de um *workflow* de geração de dados.

DWC_TASK_CONTEXT

Esta tabela faz a associação entre uma tarefa e os contextos em que pode ser realizada. Por exemplo: os sinais vitais podem ser lidos sob o contexto de uma triagem, ou sob o contexto de um episódio. A atribuição de um diagnóstico pode ser efectuada sob o contexto de um episódio, ou aquando a requisição de um MCDT.

DWC_TASK_INPUT

Esta tabela faz a associação entre uma tarefa e os parâmetros de input que esta requer.

DWC_TASK_OUTPUT

Esta tabela faz a associação entre uma tarefa e os outputs que esta produz.

DWU_WF_EXECUTION

Esta tabela representa a execução de um *workflow* de geração de dados.

DWU_WF_MARKET

Esta tabela associa um *workflow* a um mercado. Assim consegue-se filtrar *workflows* que efectuem acção específicas para um dado mercado.

DWU_WF_NODE

Esta tabela guarda a associação entre tarefas num *workflow* de geração de dados, tal como criada através da interface de utilizador. Um *workflow* consiste numa hierarquia de passos que são executados sequencialmente. Essa hierarquia pode ser representada em árvore. Cada nó da árvore associa-se ao seu nó pai, ao *workflow* a que pertence, à tarefa que vai ser executada, e ao output produzido pela tarefa mãe, no caso de necessitar de contexto, e esteja num nível inferior da árvore. Ao executar um *workflow*, as tarefas são executadas sequencialmente. Para uma dada tarefa executar, a tarefa irmã imediatamente anterior deverá ter executado, assim como todas as suas filhas, após esta.

DWU_WF_TASK_INPUT_VALUE

Esta tabela guarda o valor que o utilizador introduziu para um dado input numa dada tarefa, num nó da árvore que compõem um *workflow*.

DWU_WORKFLOW

Esta tabela guarda o registo principal de um *workflow* de geração de dados. Os *workflows* suportados poderão ser criados conforme as necessidades do utilizador, ou poderão ser criados à priori numa máquina central para serem distribuídos para outras máquinas, ou propagados com cópias do Alert®.

5.6.4.5. Módulo de gestão de dados do Alert® PFH**ALERT_EPISODE**

Esta tabela mapeia um episódio da base de dados do Alert® PFH para um episódio cuja informação foi editada no Alert Reset.

ALERT_EPISODE_UPDATE

Esta tabela guarda a informação relativamente à tarefa de alterar a data de um episódio.

ALERT_INSTITUTION

Esta tabela mapeia uma instituição do Alert® PFH para ser visível no Alert® Reset.

ALERT_INSTITUTION_MARKET

Esta tabela mapeia a tabela INSTITUTION da base de dados do Alert® PFH, para usar no Alert® Reset.

ALERT_PATIENT

Esta tabela mapeia a tabela PATIENT da base de dados do Alert® PFH, para usar no Alert® Reset. Contém várias colunas existentes na tabela original, como cópia de segurança.

ALERT_PATIENT_MARKET

Esta tabela associa um paciente a um mercado. Assim é possível distinguir pacientes, por ex., franceses e ingleses, e disponibiliza-los apenas nos contextos adequados.

5.6.4.6. Configurações**APP_CONFIG**

Esta tabela guarda uma definição de uma configuração da aplicação. Estas configurações só deverão estar disponíveis nos perfis de administrador.

APP_CONFIG_USER

Esta tabela guarda o valor de uma configuração introduzido pelo utilizador.

APP_CONFIG_VALUE

Esta tabela guarda os diferentes valores que uma configuração pode ter. Será utilizada em complemento com a coluna OPTIONS_SQL da tabela APP_CONFIG. Quando os valores de configuração não tem representação noutras tabelas, logo não se

pode usar a coluna `OPTIONS_SQL`, esta tabela actua como auxiliar ao guardar os vários valores possíveis.

APP_MARKET

Esta tabela indica os diferentes mercados disponíveis na aplicação (PT, US, FR, ...).

5.7. Observações

A arquitectura aqui descrita fornece a integração necessária com o Alert® PFH. Essa integração é obrigatória na camada de dados. As camadas de Java e UI serão suportadas por um servidor separado do Alert®, embora possa correr na mesma máquina.

A camada de interface de utilizador será desenvolvida de raiz, sem dependências com o código actual do Alert®, pois o Alert® actualmente não é feito com Flash 9.

A camada de Java irá seguir as metodologias actualmente em uso no desenvolvimento desta camada no Alert®.

O Alert® Reset irá correr sobre uma instância dedicada de servidor aplicacional para facilitar a instalação e minimizar o impacto da sua presença no Alert® PFH. Por exemplo, sempre que se actualizar código no servidor do Reset, o servidor do Alert® PFH não é afectado.

A conceptualização da camada de base de dados foi feita de raiz para este projecto, e as únicas dependências que tem com a *schema* do Alert® PFH são indispensáveis. Esta independência facilita o processo de desenvolvimento do Reset, e posteriores upgrades e instalações. O modelo de dados poderá sofrer mutações ou adições no futuro para cobrir casos de uso não previstos.

6. Conclusões e perspectivas de trabalho futuro

Todo o trabalho realizado durante este projecto foi realizado com bastante motivação, e sentido de responsabilidade. Este projecto arrancou com os objectivos iniciais, enquadrados no período de tempo previsto para este projecto, que se centravam sobre a conceptualização de uma aplicação para dar apoio às demonstrações comerciais: avaliação de requisitos, especificação técnica, e prototipagem. Mas, dentro do contexto deste projecto, o ambiente de demonstrações tinha vários problemas, pelo que o trabalho realizado passou a incluir tarefas de planeamento e organização do ambiente de demonstração completo.

Foi possível pegar num problemas que já durava à bastante tempo, que se agravava diariamente, e propor um conjunto de soluções, que estão em curso, e que progressivamente vão mostrando os seus frutos.

Os objectivos de conceptualização da nova aplicação de apoio às demonstrações foram cumpridos na integridade. Está completamente documentada, planeada, tem um protótipo, e já se iniciou o seu desenvolvimento definitivo, embora este se preveja demorado, pois toda a base de código será escrita de raiz e testada. De início haviam bastantes expectativas sobre a primeira conceptualização da aplicação de reset, ou seja, a especificação de requisitos, pois este é um dos problemas mais pertinentes. As expectativas foram satisfeitas, houve boa aceitação dos requisitos, pois resolve o problema de forma simplista e eficaz. A concepção de uma ferramenta para a edição de *workflow* que reutilização a lógica de negócio do Alert®, dentro do contexto da empresa, até pode ser visto como algo inovador.

Por outro lado, a interacção entre as várias equipas e o ambiente de demonstração progressivamente vai adoptando as novas medidas propostas. Por exemplo, actualmente a equipa de desenvolvimento já não precisa de despende tempo em intervenções no ambiente de demonstrações, o que permite que esta se concentre mais nas suas prioridades. Isto representa obviamente uma poupança de custos para a empresa. Da mesma forma agora a equipa de reset dedica-se completamente ao ambiente de demonstrações. Ou seja, actualmente os problemas de espaço em disco e desempenho foram resolvidos definitivamente, embora os problemas de qualidade de dados só sejam resolvidos com o versionamento proposto para o ambiente de demonstrações, e com a nova aplicação de reset. Progressivamente também se vai adicionando novo conteúdo clínico ao Alert® e

traduções, a uma taxa maior do que quando não havia uma equipa dedicada. Mas ainda estão por criar as novas instituições, novas contas de utilizador, novos mercados.

Embora se tenham cumprido os objectivos definidos para este projecto, o trabalho iniciado está longe de concluir. Por isso, advinha-se que o trabalho futuro ainda se centre sobre este problema.

Após as primeiras de iterações de desenvolvimento do Alert® Reset, este poderá evoluir para uma ferramenta operacional mais poderosa, como por exemplo, fornecer uma ferramenta de monitorização, e de criação de *workflows* para o Alert®. A prototipagem de uma aplicação que permita construir um *workflow* hierárquico, como o Reset, irá fornecer um valioso input para a conceptualização de um futuro motor de *workflows* no Alert®. O Reset irá progredir, e isso implica suportar cada vez mais casos de uso, o que obriga a uma maior divisão atómica do Alert® em passos, que depois possam ser agrupados nas sequências que compõe os *workflows*.

Também foi proposto a possibilidade de usar a funcionalidade de *workflows* como ferramenta de testes unitários. Actualmente, os testes do Alert® são principalmente testes de usabilidade da aplicação, do ponto de vista do utilizador, enquanto que a infraestrutura tecnológica subjacente não é alvo de testes intensivos, o que por vezes origina problemas de qualidade de dados, ou inconsistências. Ao suportar a criação e execução de *workflow*, poder-se-á adicionar *scripts* de validação de dados, e assim se reutilizava a aplicação como *suite* de testes unitários.

O problema das traduções, representa outro dos grandes obstáculos, que já esteve nas mãos da equipa de reset, e espera-se que transite para outra equipa dedicada, cujo acompanhamento pelos prévios responsáveis será importante. Os casos de uso para o problema das traduções são sérios, os grupos de utilizadores, vários, os problemas de arquitectura, complexos. Este será um projecto, que após a primeira entrega, terá grande visibilidade dentro da empresa.

Este projecto contribuiu bastante para a formação pessoal e profissional, pois permitiu a interactividade com equipas multidisciplinares, das várias áreas que compõe o Alert®. Na ALERT há bastante empenho por parte de todos os colaboradores, em constituir uma empresa bastante reconhecida pelo seu valor, e garantir que o Alert® PFH, é a melhor aplicação na sua área, e que se consegue transmitir esse valor para terceiros.

7. Referências

- **[WFP]** <http://www.workflowpatterns.com/>, consultado em 02-07-2008
- **[WSDL]** <http://www.w3.org/TR/wsdl20/>, consultado em 24-06-2008
- **[WFMC]** <http://www.wfmc.org/>, consultado em 29-06-2008
- **[LGPL]** <http://www.gnu.org/licenses/lgpl.html>, consultado em 25-06-2008
- **[WWF]** <http://netfx3.com/content/WFHome.aspx>,
<http://msdn.microsoft.com/en-us/library/aa480215.aspx>, consultado em 26-06-2008
- **[JBPM]** Website oficial: <http://www.jboss.org/jbossjbpm/>, consultado em 28-06-2008
- **[AODE]** <http://ode.apache.org/>, consultado em 26-06-2008
- **[ABPEL]** <http://www.activevos.com/community-open-source.php>,
consultado em 28-06-2008
- **[AXIS]** <http://ws.apache.org/axis2/>, consultado em 25-06-2008
- **[JBI]** <http://jcp.org/aboutJava/communityprocess/final/jsr208/index.html>,
consultado em 24-06-2008
- **[BP4PPL]** <http://bpel.xml.org/news/bpel4people-and-ws-humantask-get-reference-implementation> consultado em 27-06-2008
- **[WID]** <http://www-306.ibm.com/software/integration/wid/>, consultado em 28-06-2008
- **[MYGRID]** <http://www.mygrid.org.uk/>, consultado em 27-06-2008
- **[TAVN]** <http://taverna.sf.net/>, consultado em 28-06-2008
- **[XSCUFL]** <http://www.ebi.ac.uk/~tmo/mygrid/XScuflSpecification.html>,
consultado em 26-06-2008
- **[FFL]** <http://freefluo.sourceforge.net/>, consultado em 01-07-2008
- **[WSBP4PPL]** <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>, consultado em 01-07-2008
- **[PLSQLD]** <http://www.allroundautomations.nl/plsqldev.html>
- **[TSVN]** <http://tortoisesvn.tigris.org/>
- **[FLASH]** <http://www.adobe.com/products/flash/>, consultado em 01-07-2008

8. Bibliografia

- BPMN Information Home; 01-07-2008
<http://www.bpmn.org/>
- Object Management Group - UML; 01-07-2008
<http://www.uml.org/>
- SRIVASTAVA, Tulika, STOKOL, Glenn, Oracle Database 10g: PL/SQL fundamentals, edition 2.1, Oracle Univertisy, 2006
- SRIVASTAVA, Tulika, STOKOL, Glenn, Oracle Database 10g: Develop PL/SQL program Units, edition 2.1, Oracle Univertisy, 2006
- GREENBERG, Nancy, RAUT, Aniket, Oracle Database 10g: Advanced PL/SQL, edition 1.0, 2004

Anexos

9. Anexo A – Apresentação da empresa

9.1. A Alert Life Sciences Computing, S.A.

A Alert Life Sciences Computing, S.A, anteriormente chamada MNI – Médicos Na Internet, Saúde Na Internet, S.A, situada no Edifício Lake Towers, Rua Daciano Baptista Marques, n.º 245, Torre D, 4400-617 Vila Nova de Gaia – Portugal, é uma organização com início de actividade em Dezembro 1999.



Fig. 32 – Instalações actuais da ALERT

A ALERT foi criada pelo fundador e actual Presidente da Comissão Executiva Jorge Guimarães, com o objectivo inicial de desenvolver soluções web inovadores na área da saúde (www.saudenainternet.pt, www.listamedica.com).

Em 2001, alargou a sua actividade à construção de websites para outras organizações. Assim, a ALERT conseguiu um portfólio de dezenas de sites para operadores nacionais de saúde, incluindo nos seus clientes sociedades médicas, faculdades, ordens profissionais e laboratórios de indústria farmacêutica. Desta forma, obteve a maior

rede nacional de saúde on-line, constituindo aquilo que ainda hoje se designa por rede de sites MNI entre os quais se destacam o Instituto Português de Reumatismo, Nortemédico e Ordem dos Médicos Dentistas.

A ALERT evoluiu e alargou a sua actividade a outras áreas de actuação.

A exploração do mercado dos serviços de saúde iniciou-se com a informatização da Triagem de Manchester (Alert® Manchester) e, posteriormente, com o desenvolvimento e colocação no mercado de uma solução orientada para a gestão de episódios de urgência hospitalar (Alert® ER).

A partir do ano de 2004, iniciou-se o alargamento da gama de produtos Alert®, com produtos direccionados para a análise estatística de dados clínicos e financeiros (Alert® Data Warehouse), para os centros de saúde (Alert® Care), consulta externa hospitalar (Alert® Outpatient), para a gestão das filas de espera (Alert® Waiting Room), informatização do bloco operatório (Alert® ORIS) e informatização global do internamento (Alert® Inpatient).

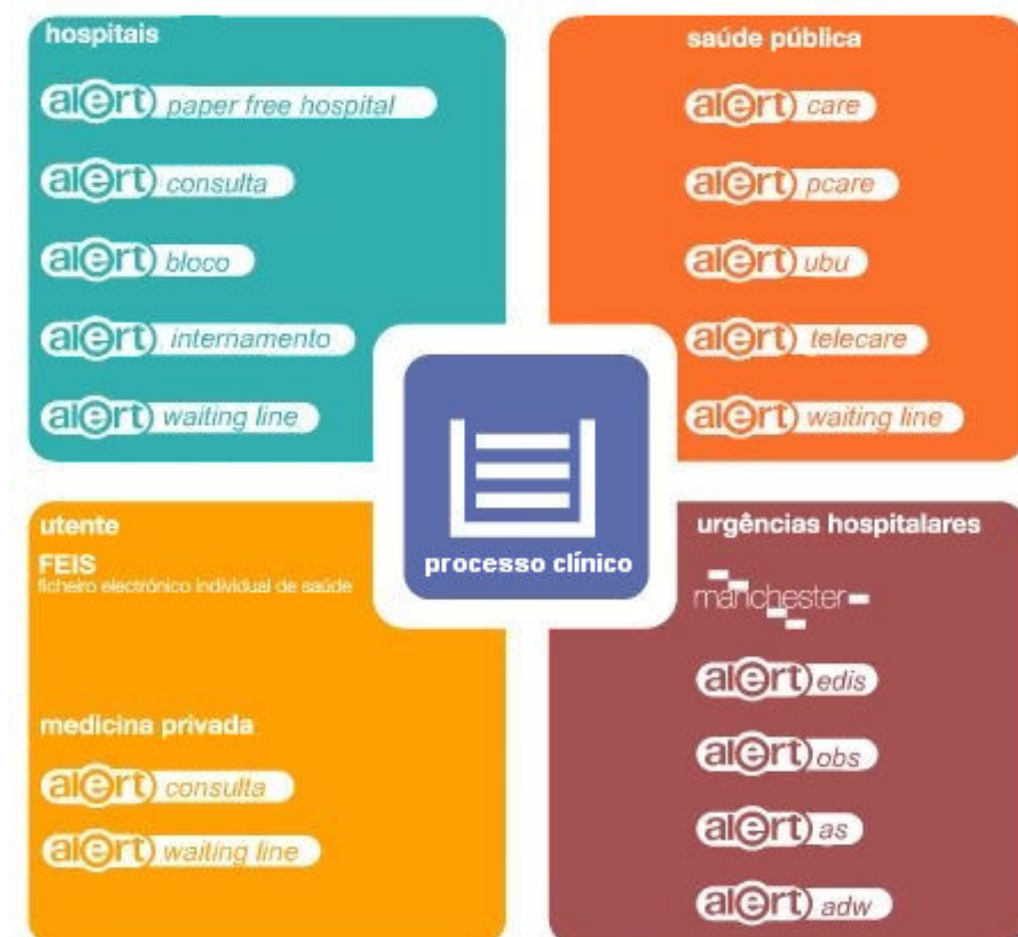


Fig. 33 – Família de produtos Alert®

A aceitação destes produtos por parte de todos os intervenientes na área da saúde foi excelente. Deste modo, o sucesso destas soluções levou a ALERT a lançar o Alert® Paper Free Hospital, projecto para a informatização global de todos os serviços clínicos de um hospital, que consiste numa aglomeração dos produtos anteriormente desenvolvidos, graças à sua interoperabilidade e integrabilidade.

Em virtude do rápido crescimento registado e para responder ao interesse dos investidores, efectuou a transformação para Sociedade Anónima em 2001, facturou acima de 12.5 milhões de euros no ano de 2006 e 23 milhões de euros em 2007.

Actualmente tem como missão apresentar uma solução global para a informatização de serviços hospitalares que possibilite o registo, a interligação, a reutilização e a análise de toda a informação clínica.

Consciente do pioneirismo da sua actividade, a ALERT empreende um esforço muito grande para se tornar uma empresa ainda maior. O nome da empresa e as marcas associadas ao Alert®, encontram-se registadas em vários países, e algumas das funcionalidades mais importantes estão patenteadas. Detém participação parcial ou total sobre distribuidores de software em países como Espanha, Itália, Holanda, e constituiu filiais nos EUA e Singapura, onde novos clientes estão a ser angariados. Actualmente tem presença em 26 países. Como medida para facilitar a internacionalização, e adequação ao segmento de mercado abordado, alterou o nome de MNI – Médicos na Internet para Alert Life Sciences Computing, no início de 2007.

A empresa é *Platinum Corporate Member* da HIMSS (*Healthcare Information and Management Systems Society*) e tem participado nas iniciativas IHE (*Integrating the Healthcare Enterprise*), com a obtenção pelo Alert® de 3 certificações de interoperabilidade.

Em Junho de 2008 a ALERT consistia num grupo de várias empresas:

- Alert Life Sciences Computing, S.A., empresa mãe portuguesa, que se dedica ao desenvolvimento do Alert® PFH, distribuição e implementação do Alert® em Portugal;
- Alert Life Sciences Computing, Inc, filial Americana, cujo principal objectivo é a distribuição e implementação do Alert® nos EUA;
- Innova Auria, S.I – empresa que se dedica à implementação e distribuição do Alert® no mercado espanhol e sul-americano;
- Alert BV – subsidiária sediada na Holanda que se dedica a implementação e distribuição do Alert® neste país;
- Alert BRASIL – subsidiária sediada em Belo Horizonte, estado de Minas Gerais no Brasil, que se dedica a implementação e distribuição do Alert® neste país. Esta subsidiária foi construída em virtude do contracto formado com o governo estatal de Minas Gerais para a informatização de toda a rede de saúde regional do estado;
- Alert ASIA – subsidiária sediada em Singapura que se dedica a implementação e distribuição do Alert® no mercado asiático;

- myPartner HealthCare, Software Solutions, S.A. – empresa portuguesa, adquirida pela, na altura, MNI, em Junho de 2006, que se dedica exclusivamente ao desenvolvimento do Alert® ERP, orientado para a gestão de stocks de material hospitalar, logística, recursos humanos, finanças, facturação, e do Alert® CRM, para gerir a relação com os clientes. Entretanto a ALERT já adquiriu a totalidade da myPartner, logo esta é parte integrante da ALERT, mas é aqui descrita para referência histórica.

Em Junho de 2007 contava com cerca de 500 colaboradores permanentes incluindo clínicos, designers, arquitectos, informáticos, matemáticos, equipas de acompanhamento presencial, etc. Está acreditada como entidade formadora pelo INOFOR (Instituto para a Inovação na Formação).

Encontra-se assim numa posição privilegiada para o fornecimento de soluções inovadoras e flexíveis, sendo detentora de produtos e conhecimentos únicos para a prossecução dos actuais objectivos na prestação de cuidados de saúde.

9.2. O Alert ® Paper Free Hospital

O Alert® Paper Free Hospital (Alert® PFH) é uma solução para informatização integral de hospitais, que possibilita o registo, a interligação, a reutilização e a análise de toda a informação relacionada com a realidade clínica hospitalar, eliminando o papel como suporte da informação clínica. Caracteriza-se por um interface gráfico inovador e intuitivo, feito a pensar nos utilizadores finais.

O conjunto de software e hardware do Alert® PFH está especialmente concebido para:

- Aumentar a satisfação dos utentes e profissionais;
- Aumentar a eficácia dos serviços hospitalares;
- Interligar a informação;
- Auxiliar a gestão;
- Controlar os custos;
- Melhorar o desempenho dos profissionais.

O Alert® PFH, em conjunto com o Alert ® Electronic Health Record, possibilita, em tempo real, o registo e consulta do processo clínico dos utentes e a análise de toda a actividade clínica do hospital.

O Alert® PFH permite aumentar a eficácia dos serviços hospitalares através da realização de registos sobre o paciente e o episódio, do registo rápido e intuitivo da actividade clínica e do workflow intra-hospitalar.

Todos os intervenientes (médicos, enfermeiros, assistentes sociais, pessoal auxiliar, etc.) nos serviços hospitalares têm no Alert® PFH acesso a níveis de informação

diferenciados dependendo das suas competências, necessidades e privilégios através de perfis de utilizador e identificação biométrica.

Com o software Alert® PFH, o registo da informação é efectuado em tempo real, permitindo a partilha de informação, facilitando o acesso ao processo clínico do utente e evitando problemas de legibilidade e de comunicação.

O sistema Alert® PFH caracteriza-se por:

- Conceito de workflow;
- Monitores sensíveis ao tacto (touch-screen);
- Interface gráfico inovador especialmente desenvolvido para esta aplicação;
- Identificação biométrica dos utilizadores;
- Reconhecimento dos utentes por fotografia e código de barras;
- Construção modular;
- Compatibilidade HL7;
- Interoperabilidade IHE (Integrating the Healthcare Enterprise);
- SNOMED CT;
- ICNP (CIPE);
- LOINC;
- ICD9, ICD10;
- Bases de dados de alergias, doenças, fármacos, procedimentos e meios auxiliares de diagnóstico;
- Possibilidade de incluir protocolos com ajuda técnica;
- *Back Office*;
- *Data Warehouse*.

O Alert® PFH também funciona como um elemento integrador de outras soluções de software, incluindo software administrativo.

O software Alert® PFH inclui um sistema de alertas que assegura a capacidade de vigilância sobre o workflow dos pacientes, dá conta da lista de tarefas por realizar ou de eventos que necessitem da intervenção de determinado utilizador, minorando tempos de espera e a possibilidade de erro humano.

O Alert® PFH tem uma forte componente de gestão, permitindo:

- a gestão dos recursos humanos e técnicos disponíveis através de um olhar em tempo real sobre o funcionamento dos serviços;
- a análise e avaliação objectiva do funcionamento de cada unidade, com base em instrumentos complementares de análise e apoio à decisão, quer em tempo real quer a posteriori.

A concepção do Alert® PFH está a cargo de uma equipa multidisciplinar, entre os quais, administradores hospitalares, médicos, enfermeiros, auxiliares, administrativos, engenheiros informáticos e designers de comunicação.



Nome	Processo clínico	Consulta de	Tipo						
09:02h	M / 31 Ricardo Pereira 138517	Cardiologia	1						
09:32h	M / 56 António Manuel Campos 52393	Cardiologia	1						
10:02h	F / 42 Manuela Santos 76824	Obstetrícia	1						
10:32h	F / 51 Alberta Castro 547145	Medicina	1						
11:02h	M / 73 Vitor Soares 5479879	Medicina	1						
11:32h	M / 7 Diogo Castro 228649752	Pediatria Médica	1						
12:02h	F / 23 Mariana Silveira 228532323	Diabetes	1						
12:32h	M / 62 Joaquim Oliveira 346E72BBD	Medicina	1						
13:02h	M / 23m Alexandre Santos 228389324	Otorrinolaringologia	1						

Fig. 34 – Imagem da grelha de paciente da consulta externa, Alert® OUTPATIENT

9.3. Metodologia de desenvolvimento

A metodologia usada no desenvolvimento do projecto tem algumas semelhanças com a metodologia SSADM (*Structured Systems Analysis and Design Methodology*). SSADM é uma metodologia em queda de água, o que significa que é constituído por fases, sendo apenas possível passar para a fase seguinte se a anterior estiver terminada. Por ser sustentada na condução de actividades dependentes e, por isso um pouco rígidas, esta metodologia é adequada a projectos com elevadas necessidades de controlo e qualidade. Este é o principal motivo pelo qual os princípios da metodologia são adoptados.

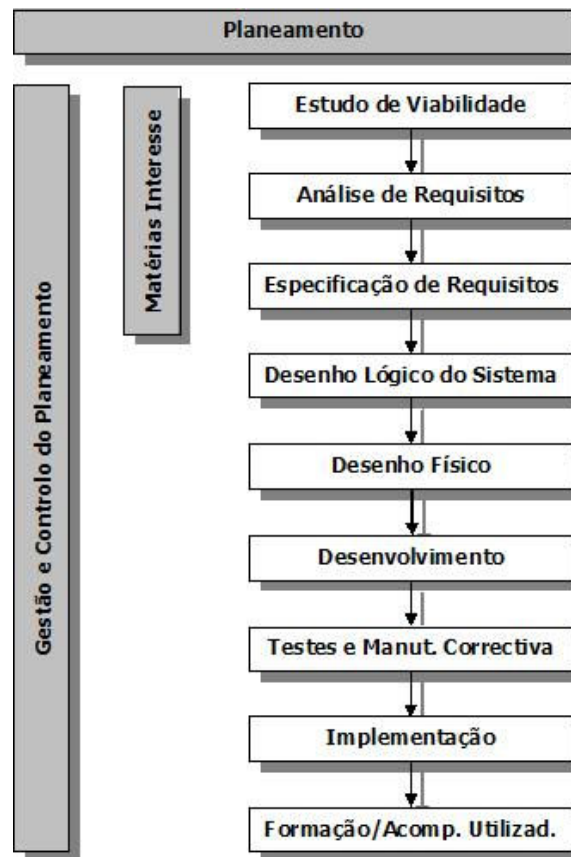


Fig. 35 – Etapas da metodologia SSADM

A etapa Planeamento faz o levantamento das etapas ao longo do projecto, o seu dimensionamento e recursos, o tempo de execução do projecto, para suportar a etapa de Gestão e Controlo do Planeamento, que acompanha todo o projecto.

A etapa Matérias de Interesse visa a previsão de tempo que permite aos recursos envolvidos no projecto adquirem conhecimentos específicos.

O Estudo de Viabilidade pretende determinar se o sistema a desenvolver satisfaz as necessidades dos clientes e se os custos envolvidos são suportáveis, para além de procurar adaptar o produto ao mercado, para aumentar, se possível, o segmento de mercado.

A Análise de Requisitos é baseada na organização dos serviços hospitalares, tendo em atenção as necessidades presentes e futuras dos utilizadores do novo sistema. Nesta fase deve ser ainda tida em conta a articulação do sistema a desenvolver com a restante realidade do Hospital, nomeadamente na identificação das formas de integração com outros sistemas, como os softwares administrativos e interfaces com equipamento electrónico.

Na Especificação de Requisitos é iniciada a definição detalhada das características funcionais e processuais do novo sistema.

O Desenho Lógico do Sistema prevê a validação dos requisitos especificados, concepção do aspecto da aplicação final e definição de diálogos e tratamento de erro.

Na fase de Desenho Físico é definido o desenho da base de dados e criadas as especificações de programa a partir das especificações lógicas do sistema e das especificações técnicas definidas, de forma a lançar as bases do Desenvolvimento.

As fases Desenvolvimento e Teste e Manutenção Correctiva são as fases onde toda especificação elaborada anteriormente, e conhecimentos são utilizados para desenvolver o sistema propriamente dito, seguido de testes e validação de qualidade. São duas fases intimamente ligadas.

A fase de Implementação consiste no levantamento topográfico da instituição em causa, e determinação de todas as suas necessidades. Com estes dados faz-se o planeamento da implementação, e de todas as configurações necessárias. Durante a implementação, instala-se a infraestrutura completa o *Alert*® (software+hardware), e fazem-se todas as parametrizações.

Durante a fase Formação e Acompanhamento dos Utilizadores, as equipas de acompanhamento presencial e formação dão formação sobre o *Alert*® aos novos utilizadores e mantêm presença constante na instituição para garantir acompanhamento dos profissionais de saúde na utilização da aplicação, e reportar entraves no abandono do papel. A ALERT actualmente dá grande importância a esta fase, pelo que a equipa de Acompanhamento presencial, actualmente, é bastante numerosa.

10. Anexo B - Glossário

Segue-se a definição de alguma terminologia usada ao longo deste relatório que poderá ser pertinente para a melhor compreensão deste relatório

Episódio

Um episódio corresponde a um período de tempo, durante o qual um paciente recebe tratamento num dado departamento no hospital. Considerando os eventos num ambiente de urgência, um paciente chega à urgência, e é atendido por um administrativo, que regista a sua chegada, o paciente é colocado numa fila de espera. Dessa fila de espera, o paciente a determinada altura é chamado pelo triador, que tipicamente é um enfermeiro. Este regista a queixa do paciente, e consoante a medição de determinados sinais vitais, e relatos do paciente, procede à triagem, da qual obtêm uma cor. Esta cor é determinada pela triagem utilizada, caso efectuada correctamente, e não de livre escolha do triador. O paciente é observado por um médico, podendo este requisitar tratamentos, ou transferir a responsabilidade para outro médico, consoante a gravidade ou especificidade do caso. Se o paciente demonstrar melhorias, o médico pode dar alta médica, que depois é confirmada pelo administrativo. Nesta altura o paciente habitualmente paga a taxa moderadora e abandona o serviço de urgência. Caso o paciente inspire mais cuidados, e não possa abandonar o serviço de urgência, ao fim de 24 horas, desde a sua entrada no departamento, este transita para OBS, que em termos aplicacionais corresponde a um internamento. No cenário indicado anteriormente, existem 2 episódios, um de urgência, e um de internamento, que ocorrem durante a mesma visita ao hospital. No Alert® a informação transita sempre entre episódios, e assim, o tratamento que foi prescrito na urgência é visível a partir da aplicação de internamento.

MCDT

A sigla MCDT significa Métodos Complementares de Diagnostico e Terapêutica. Um MCDT é um exame que se pode realizar a um paciente. Esse exame pode ser um exame de imagem, como um raio-x, análises laboratoriais, outros exames como uma prova de esforço. Estes exames permitem avaliar a condição da efemeridade do paciente, e auxiliar o médico na prescrição de tratamento para o paciente.

11. Anexo C – proposta para gestão de traduções

Para resolver os vários problemas relativos a traduções, foi elaborada uma proposta, mas que fica documentada como anexo, pois não se contextualiza adequadamente nos objectivos iniciais do projecto.

Para resolver o problema das traduções, poder-se-ia utilizar uma aplicação central sobre a qual se geravam e traduziam todos os textos.

11.1. Problemas de gestão de traduções

A listagem de problemas que se segue, embora abranja todos os aspectos das traduções, tem consequências directas no aspecto final do Alert® usado para as demonstrações e clientes: a falta de traduções, traduções dessincronizadas ou de fraca qualidade:

- A equipa de desenvolvimento elabora o modelo de dados e insere conteúdos nas novas tabelas. Muito desse conteúdo deve ter traduzido, embora outro conteúdo, como o clínico, ou a descrição dos edifícios das instituições não deve ser traduzido e versionado. Este tem que ser identificado;
- Não é possível saber em que versão do Alert® surgiu um texto;
- Durante a tradução, a equipa de traduções não tem contexto nem informação suficiente para realizar traduções adequadas e de qualidade;
- Por vezes, aparecem erros ortográficos e sintácticos;
- As traduções são geridas com ficheiros de Excel que exigem bastante mão-de-obra e os cujos dados podem ficar deprecados;
- Por vezes, uma listagem de textos para traduções sobrepõe uma listagem de textos anterior já traduzida previamente;
- Muitas vezes, as listagens para tradução de textos numa língua A não traduzidos numa língua B, inclui textos semelhantes que já foram previamente traduzidos nessa língua B;
- Muitas vezes, a tradução não é revista por consultores;

- A participação de indivíduos nas empresas afiliadas na tradução, é feita recorrendo a ficheiros Excel, e a tradução de uma listagem relativamente grande pode demorar meses, durante os quais esse ficheiro fica desactualizado. Por exemplo: a tradução completa do Alert® para uma nova língua implica a tradução de centenas de milhares de textos;
- Não há um repositório para todas as traduções efectuadas. Estas estão divididas entre o ambiente de desenvolvimento e a máquina de demonstrações comerciais e formação interna;
- Não há controlo de versões sobre cada texto traduzido. Não é possível obter de forma fácil um histórico de edições de um único texto.

11.2. Procedimentos a adoptar

A existência duma aplicação deveria ser acompanhada dos seguintes procedimentos e políticas:

- Todo o modelo de dados de traduções passava para um *schema* numa base de dados própria, para controlar as permissões sobre esta. A aplicação seria implementada neste *schema* também. Em princípio, esta *schema* seria vedada a modificações, e estas só seriam possíveis através do UI;
- A equipa de desenvolvimento utilizaria a aplicação para fazer registo de novos textos a traduzir;
- A equipa de traduções poderia trabalhar directamente sobre esta aplicação, ou então desenvolvia-se uma forma de exportar listagens para estes trabalhar sem acesso directo à aplicação;
- Consultores externos, ou equipas nas empresas afiliadas poderiam aceder facilmente à aplicação, caso seja *web-based*. Adicionalmente, poderiam ver as alterações em tempo real no Alert®, por VPN, ou pela Internet;
- Os administradores da aplicação tratariam de todas as tarefas relativas a scripts, sincronizações, gestão de utilizadores, etc.

11.3. Utilizadores finais

Os utilizadores finais da aplicação seriam:

- A equipa de desenvolvimento – esta teria permissões para:
 - Inserir e alterar textos no ambiente de desenvolvimento, em línguas para as quais tenha permissões. Pela inserção entende a geração de novos códigos identificativos de textos nas tabelas;
 - Exportação/importação de listagens noutros formatos como XML, CSV, Excel, ODF, etc;

- Geração de *scripts*.
- A equipa de traduções, e tradutores nas subsidiárias – esta teria permissões para:
 - Alterar textos já existentes e traduzir textos não traduzidos, em línguas nas quais cada elemento tem permissões para alterar;
 - Criação, gestão, e utilização de dicionários;
 - Exportação/importação de listagens noutros formatos como XML, CSV, Excel, ODF, etc;
 - Aceder a qualquer outra informação, apenas com acesso de leitura.
- Consultores externos – estes teria permissões para:
 - Alterar textos já existentes e traduzir textos não traduzidos, em línguas nas quais cada elemento tem permissões para alterar;
 - Criação, gestão, e utilização de dicionários;
 - Aceder a qualquer outra informação, apenas com acesso de leitura;
- Administradores – estes teria permissões para:
 - Sincronização de diferentes instâncias da aplicação, como por ex, sincronizar uma instância num ambiente próprio de traduções com uma instância no ambiente de demonstrações;
 - Geração de *scripts*.
 - Visualização e alteração de qualquer informação;
 - Reverter modificações efectuadas (*rollback* do histórico);
 - Exportação/importação de listagens noutros formatos como XML, CSV, Excel, ODF, etc;
 - Registo e gestão de utilizadores da aplicação;
 - Definição de grupos de utilizadores e permissões associadas.

11.4. Requisitos gerais

Para resolver os problemas enunciados, a aplicação deverá suportar as seguintes funcionalidades

Criação de filtros/pastas

A aplicação deverá suportar a criação hierárquica de filtros. Estes filtros seriam estruturados tal e qual como uma árvore de pastas num sistema de ficheiros, e a cada filtro estariam associados um conjunto de textos. Os filtros usufruiriam das seguintes propriedades:

- Podem-se definir regras de inclusão e/ou exclusão em cada filtro para este incluir automaticamente textos, como por exemplo, incluir automaticamente todos os textos cujo código começa por FOO;
- Textos podem ser atribuídos manualmente pelo utilizador a um filtro;
- O mesmo texto pode aparecer em vários filtros;
- Um filtro pode ter vários filtros-pai.

Exemplo:

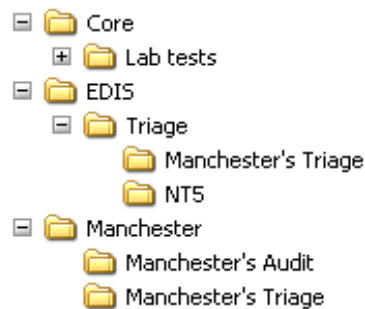


Fig.13 – exemplo da hierarquia de filtros

Um filtro teria várias propriedades, como versão em que foi criado, data de criação, criador, e conjuntos de imagens associadas às funcionalidades ou contextos aos quais o filtro se aplica.

Os filtros podiam agrupar conteúdo clínico (análises, medicação, etc.), aplicacional (cabeçalhos, *multichoices*, etc.), e institucional (salas, camas, instituições, etc.). Apenas o aplicacional seria traduzido.

Adicionalmente dever-se-ia definir também uma estrutura anexa para conter desenhos do Alert®, provenientes do design, devidamente agrupadas hierarquicamente, por pastas, da mesma forma que os textos. Estes desenhos forneceriam algum contexto. A cada filtro podia-se associar várias pastas com imagens, e vice-versa.

Criação de novos textos

Associada à criação de filtros, e para a equipa de desenvolvimento, a aplicação obviamente suportaria a criação de novos textos. O(s) código(s) que identificariam unicamente um texto seria especificados pelo utilizado assim como a língua. Esta introdução de textos poderia ser efectuada sobre um interface estilo grelha, como numa folha de cálculo, gerava entradas de histórico, e automaticamente assumia uma versão do Alert® por defeito, onde o texto seria introduzido, embora o utilizador pudesse definir outra versão.

Listagem e alteração de textos, pesquisas *locks*

A aplicação forneceria listagens de textos, por filtros. O utilizador poderia escolher que línguas pretende ver e editar. Dentro de um filtro, poder-se-ia aceder a uma listagem de todos os textos associados a esse filtro e filhos.

Após obter uma listagem para edição, a base de dados marcaria os registos devolvidos, para os quais se tem permissões de alteração, como bloqueado (*lock*). Posteriores modificações a esses registos por outros indivíduos não seriam permitidas, assim como conseguir um *lock*. O *lock* teria um *timeout*, e o tradutor receberia notificações periodicamente para renovar os *locks*, caso contrário havia *timeout*, e outro colega poderia trabalhar sobre os mesmos texto.

O utilizador poderia fazer pesquisas por código, texto, língua, versão, etc, dentro do contexto de um filtro.

Correcção ortográfica

Ortografia correcta é indispensável. A aplicação poderia fazer validação ortográfica, e os dicionários, usados para essa validação, deveriam ser editáveis, para se incluir terminologia adequada.

Histórico de alterações

Cada alteração efectuada aos textos presentes na aplicação gerava histórico. O histórico deveria conter a seguinte informação:

- Identificação única do texto traduzido, constituída pela identificação da língua e código(s);
- Quem alterou;
- Como foi efectuada essa alteração: edição manual, sincronização entre várias instâncias da aplicação, uso de dicionário, importação de listagem;
- Que tipo de alteração é: *insert*, *update*, *delete*.

Os administradores da aplicação deveriam ter a hipótese de reverter modificações, baseando-se no histórico.

A visualização do histórico deveria ser visualizável em vários formatos (com paginação):

- Por filtro;
- Por datas;
- Por revisão;
- Por utilizador;
- Por texto único;
- Por língua;
- Histórico completo;
- Outros critérios de pesquisa customizáveis.

Geração de *scripts*

Usufruindo do histórico e da hierarquia por filtros, facilmente se poderia escolher através da interface da aplicação textos para os quais seriam gerados *scripts* para inclusão

nos *scripts* de versão do Alert®. Estes seriam corridos numa base de dados, preferencialmente a base de dados de testes, para fazer validação final dos *scripts* e traduções.

Sincronização entre várias instâncias

Como o trabalho de geração e tradução de textos é feitos por equipas diferentes, estas deveriam trabalhar em ambientes diferentes consoantes os seus objectivos.

- A equipa de desenvolvimento deveria trabalhar sobre o ambiente de desenvolvimento para criação de novos textos, ou correcções ao conteúdo existente;
- A equipa de traduções, e consultores externos, deveriam trabalhar sobre uma máquina estável, com um versão (quase) fechada.

Sincronizar os dois ambientes iria permitir partilhar toda a informação relativamente a filtros, histórico, imagens e textos

Geração e manutenção de dicionários

Um dicionário, no caso das traduções, consiste num conjunto de textos únicos, com traduções únicas noutras línguas. À partida, os dicionários podem conter ícones e textos não alfabéticos (só números ou pontuação).

A criação dum dicionário passaria pela escolha dum conjunto de códigos, para os quais a aplicação geraria uma listagem de todos os textos únicos nesse domínio, numa ou em várias línguas. Ao introduzir novo conteúdo na aplicação numa língua, o uso de dicionários iria permitir traduzir automaticamente esse novo conteúdo para outras línguas. Este processo poderia ser automático ou manual. O uso de dicionários permitia fazer traduções automáticas.

Os dicionários deverão ser 100% editáveis pelo utilizador.

Como o custo de correr um dicionário contra a BD de traduções seria muito grande (dependendo da quantidade de registos envolvida), esta tarefa podia ser assíncrona.

11.5. Questões técnicas

A aplicação basear-se-ia nalguns pressupostos, para garantir extensibilidade e flexibilidade:

- Que podem haver vários grupos de utilizadores, com permissões diferentes;
- Que existe um conjunto de tabelas arbitrário, que contém traduções, sendo cada texto unicamente identificado por um conjunto de colunas numa tabela que definem um código, e uma língua também unicamente identificada por uma ou mais colunas, nessas mesmas tabelas;
- Que podem haver várias instância da aplicação a correr, como uma num ambiente de desenvolvimento, e outra num ambiente próprio para traduções;

- Que podemos ter textos em diferentes alfabetos (latino, cirílico, arábico, etc) pelo que a base de dados terá que guardar textos numa codificação que suporte múltiplos alfabetos, como *Unicode*. Toda a transmissão de texto terá que ser efectuada obrigatoriamente em *Unicode*, por exemplo, UTF-8 ou UTF-16;
- Que a aplicação pode ser acedida através de linhas com pouca largura de banda pelo que a quantidade de informação partilhada entre cliente e servidor importa;
- Que poderá haver uma aplicação para gestão de conteúdo clínico, que terá que interoperar com esta. Esta interface idealmente deverá ser extrínseca à base de dados;
- Que qualquer que sejam os resultados obtidos, a aplicação terá que ser mantida e constantemente evoluída.

11.6. Observações

Uma aplicação central para traduções com os requisitos sugeridos resolveria muitos dos problemas actuais. Os problemas apresentados neste documento seriam resolvidos ou minimizados pelos requisitos propostos. No entanto isto implicaria mudanças nos procedimentos actuais, preferencialmente para melhor.

Uma das principais alterações que esta aplicação proporcionaria seria o fim dos ficheiros Excel para transportar textos. Actualmente os ficheiros são um dos principais problemas, pois é difícil perceber que conteúdos têm, ficam rapidamente deprecados, e por vezes incompatíveis, e requerem bastante tempo para gerir.