UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

Sistema de Informação para Vigilância Epidemiológica

projecto realizado na

Maxdata

por

David Alexandre Fernandes da Silva

PROJECTO

VERSÃO PÚBLICA

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Arquitectura, Sistemas e Redes de Computadores

2012

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

Sistema de Informação para Vigilância Epidemiológica

projecto realizado na

Maxdata

por

David Alexandre Fernandes da Silva

PROJECTO

VERSÃO PÚBLICA

Trabalho orientado pelo Prof. Doutor Carlos Jorge da Conceição Teixeira
e co-orientado pelo Doutor Paulo Jorge Paiva de Sousa

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Arquitectura, Sistemas e Redes de Computadores

2012

Agradecimentos

Antes de mais, quero agradecer ao Doutor Paulo Sousa pela oportunidade que me deu ao permitir que estagiasse na sua empresa.

Quero agradecer igualmente ao Professor Doutor Carlos Teixeira pelo apoio, ajuda e cooperação prestada no decorrer do estágio curricular.

Quero agradecer a todos os funcionários da Maxdata, com especial atenção aos companheiros de desenvolvimento, Paulo Sousa, Alexandre Correia, Cláudio Vicente, Nuno Castro, Nuno Ferreira, que me ajudaram imenso ao longo do decorrer do estágio.

Quero agradecer aos meus amigos mais chegados, Bruno Branco e Brito (*Engenheiro*), Bruno Correia (*CTT*), Rui Almeida (*Ruizinho*), Hugo Silva (*Balsas*), João Casais (*Casais*), João Cruz (*JC*), Pedro Martins (*Basa*), Rui Ferreira (*Bento*) e Sara Patrocínio pela amizade, ajuda, e momentos de lazer que proporcionaram no decorrer na Licenciatura e Mestrado.

Por fim, mas de todo não menos importante, agradeço à minha família e a minha mais que tudo, Andreia Filipa, pelo amor, carinho, apoio e paciência (que é bem precisa!) ao longo de todos estes anos.

Um grande obrigado.

Resumo

Este documento detalha vários pontos fulcrais inerentes ao trabalho desenvolvido no âmbito do Projecto em Engenharia Informática (PEI), integrado no 2ª ano do Mestrado em Engenharia Informática da Faculdade de Ciências da Universidade de Lisboa (FCUL), em particular, estão reflectidos diversos tópicos introdutórios ao tema em estudo, assim como uma descrição sucinta do projecto desenvolvido, bem como conhecimentos e conceitos adquiridos.

Este PEI focou-se essencialmente na área da saúde, onde foi proposto desenvolver um sistema de informação para vigilância epidemiológica. Resumidamente, a vigilância epidemiológica consiste na observação continua, sistemática e activa da ocorrência e distribuição de infecções em doentes internados e nos profissionais de saúde que deles cuidam, a fim de identificar as situações em que é possível melhorar a prática da medicina[3]. Para que tal seja exequível, é necessário que a informação seja obtida e analisada, de modo a que inferências possam ser feitas e conclusões possam ser tiradas. Posteriormente às conclusões, serão eventualmente desencadeados determinados comportamentos definidos (por ex. alertas, mensagens) que têm como intuito alertar a ocorrência de situações anómalas.

O objectivo deste PEI foi, portanto, desenvolver um sistema de informação que abranja toda a instituição de saúde, permitindo obter informações que sejam resultado de estudos automatizados e consequentemente constituam um aspecto importante da prevenção e controlo das infecções.

Palavras-chave: Sistemas de informação para a saúde; Vigilância Epidemiológica; Integração de aplicações; Inovação

Abstract

The goal of this document is to present an overview and to detail several key points of the project I was entitled to. This project was part of *"Projecto em Engenharia Informática"* (PEI), which is the 2nd year of the Masters Degree in Computer Science at Faculty of Sciences of the University of Lisbon (FCUL). The topics discussed in this document are introdutory and related to subject under study, including a brief description of the project developed, as well as knowledge and concepts acquired.

The project focused primarily on public health, as it was proposed to develop an information system for epidemiologic surveillance. Essentially, epidemiologic surveillance is an active and continuous observation of occurences and infections that affect patients, as well as the health care professionals that take care of them, in order to identify situations in which it's possible to improve and apply a better medical practice[3]. To achieve this goal, it's necessary to gather and secure information, which will be analyzed in such way that inferences can be made and conclusions can be drawn. Subsequently, those conclusions will eventually trigger certain predefined behaviors (p.e. alerts, messages) that will be notify and warn the occurrence of abnormal situations.

The goal of the project, was to develop and provide an information system for epidemiologic surveillance that covers the entire health institution, in order to obtain that particular information that will be useful in automated studies, providing a key value over the prevention and infection control.

Keywords: Health Information Systems; Epidemiologic Surveillance; Application integration; Innovation

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

A vigilância epidemiológica consiste essencialmente numa observação contínua, activa e sistemática da ocorrência e distribuição de infecções em doentes internados, assim como nos profissionais que destes cuidam. Tal actividade tem como intuito a identificação de situações onde é possível melhorar o serviço prestado, isto é, melhorar a prática da medicina[3]. Para que esta acção final seja de facto exequível, são necessárias definições claras, recolha, análise e interpretação de dados e informação relevante à prevenção e controlo de infecções.

Um sistema de vigilância epidemiológica deve permitir a identificação, detecção e eventualmente a prevenção e controlo de doenças, a nível pessoal e colectivo, devendo abranger toda a instituição médica, de modo a possibilitar a obtenção de informações que sejam a expressão da ecologia infecciosa e que, pelas suas repercussões, constituam um aspecto importante na prevenção e controlo de infecções. No entanto, é também não só desejável, como essencial, que exista um contacto mais próximo da equipa de controlo de infecções com todos os serviços, de forma a que ocorra mais frequentemente e de forma natural a tomada de conhecimento de problemas que ocasionalmente ocorrem.

Para além da utilidade já referida, este tipo de análise e manipulação de informação, providencia uma óptima fonte de dados estatísticos, desde estirpes, mapeamento de doenças em regiões, faixas etárias, grupos étnicos, etc. Existe portanto uma ínfima lista de possibilidades estatísticas que poderá ser utilizada não só pelo tratamento da informação da vigilância, mas também pelos próprios profissionais de saúde nas suas investigações e estudos.

1.2 Maxdata

A Maxdata desenvolve soluções informáticas para a área de saúde há mais de 30 anos, estando presente nos maiores hospitais nacionais em várias áreas e laboratórios, nomeadamente, requisição electrónica, patologia clínica, anatomia patológica e imunohemoterapia (banco de sangue e transfusões).

A Maxdata providencia apoio técnico e suporte aos sistemas desenvolvidos por esta, incluindo a instalação dos mesmos. Paralelamente, está sempre presente um espírito inovador, dando primazia ao desenvolvimento das alterações requeridas por cada organização, generalizando-as, contribuindo para uma mais fácil integração das tecnologias de informação em proveito da qualidade, redução de custos, comodidade e segurança. A Maxdata tem em curso um processo de internacionalização ambicioso, com o objectivo de tornar o seu *software* uma referência a nível mundial. Neste contexto, tem vindo a efectuar um grande investimento para potenciar e agilizar a divulgação e adequação do seu software ao mercado internacional.

1.3 Objectivos

Este documento descreve o trabalho desenvolvido no âmbito do Projecto em Engenharia Informática (PEI), integrado no 2º ano do Mestrado em Engenharia Informática da Faculdade de Ciências da Universidade de Lisboa (FCUL).
O objectivo principal do PEI foi desenvolver um sistema de informação para Vigilância Epidemiológica, que se pode dividir em objectivos de menor dimensão para que este seja atingido.

- Desenvolver mecanismos que permitam detectar e avisar a ocorrência de situações consideradas anómalas num determinado contexto (gerar alertas).
- Desenvolver mecanismos que permitam controlar e melhorar a prevenção no controlo de infecções (planear acções).
- Desenvolver mecanismos que permitam rastrear e acompanhar os diferentes estados de um tratamento de um paciente (visualizar eventos e casos do paciente).

Adicionalmente, o PEI permitiu ao aluno o enquadramento no mundo do trabalho, onde adquiriu conhecimentos, assimilou e desenvolveu hábitos de trabalho inerentes à experiência profissional que se desenvolveu ao longo da estadia na empresa, fortalecendo deste modo o seu crescimento a nível pessoal, social e profissional.

1.4 Estrutura do documento

Este documento, tratando-se de um relatório final, tem como objectivo providenciar uma descrição do trabalho realizado durante o presente ano lectivo, assim como das tecnologias e conceitos adquiridos. O documento encontra-se divido nos seguintes capítulos:

- **Capítulo 2** - Enquadramento:
  É apresentada uma descrição sucinta da aplicação desenvolvida, assim como os 4 *P's* inerentes à gestão do projecto.

- **Capítulo 3** - Análise e Desenho:
  São enumerados os requisitos do projecto, assim como uma descrição sobre a sua arquitectura e os diferentes modelos utilizados nas suas camadas.

- **Capítulo 4** - Implementação:
  São apresentadas as diferentes tecnologias utilizadas em cada camada da arquitectura, assim como casos de uso relativos a cenários de utilização simples e complexos.

- **Capítulo 5** - Resultados:
  São apresentados os resultados dos testes unitários executados à aplicação.

- **Capítulo 6** - Conclusão e trabalho futuro:
  É finalizado o relatório com uma síntese do projecto e do ano lectivo, assim como o trabalho ainda por realizar no âmbito da aplicação desenvolvida.

Capítulo 2

Enquadramento

2.1 Conceitos

Nesta secção são descritos os conceitos inerentes à aplicação.

- **Origem**: A Origem/serviço corresponde a um departamento dentro da instituição médica.
- **Alarme**: Um alarme é essencialmente uma notificação que marca algum acontecimento anómalo.
- **Acção**: Uma acção consiste numa operação/actividade planeada para um dado profissional de saúde ou um conjunto deles. Um exemplo prático de uma acção é: ”lavar as mãos após um controlo de infecção”.
- **Evento**: Um evento é um conceito abstracto que simplesmente representa um dado acontecimento, uma ocorrência no sistema relevante para a vigilância epidemiológica, como por exemplo, a realização de uma acção.
- **Caso**: Um caso é uma ocorrência, um acontecimento relativo a um dado paciente. No entanto para que tal seja considerado um caso, é necessário que seja cumprido um conjunto de critérios, critérios estes que devem incluir definições para pessoas, espaço, tempo, características clínicas, laboratoriais e epidemiológicas, isto é, determinadas ocorrências numa dada região podem ser consideradas anómalas, no entanto tal poderão não o ser numa outra região.

É de salientar que alguns dos conceitos enumerados são igualmente utilizados pelos profissionais de saúde no seu dia-a-dia.

2.2 A Aplicação

A aplicação *web* desenvolvida trata-se de um módulo de vigilância epidemiológica, perfazendo por si só uma aplicação individual, no entanto encontrar-se integrado na aplicação

web Clinidata®[1].

A aplicação permite aos seus utilizadores usufruir de um sistema onde podem obter informações inerentes à vigilância em toda a instituição hospitalar, nomeadamente casos, pacientes (historial), infecções (confirmar diagnósticos, através da informação inerente às infecções na aplicação) e suas medidas de controlo (melhorar medidas de prevenção e controlo de infecções), com o objectivo de promover a interoperabilidade entre os diversos departamentos e cuidados médicos praticados. É igualmente possível a divulgação de resultados e intervenções médicas (partilha de dados/informação), assim como a consulta de mapas estatísticos, que em última análise, providenciam uma ínfima lista de possibilidades aos utilizadores. Os referidos mapas são configuráveis e definidos pelo utilizador em questão, permitindo obter dados estatísticos e mapeá-los por regiões, faixas etárias, grupos étnicos, etc.

Como mencionado, a aplicação está disponível para toda a instituição médica e consequentemente aos diferentes departamentos que a constituem, tal aspecto é de facto um factor relevante e a favor da aplicação, no entanto requer a utilização de informação e dados gerados por outras aplicações utilizadas nos vários departamentos (se aplicável). O acesso aos dados é tipicamente efectuado por invocação de *WebServices*, existindo portanto uma camada que efectua e trata deste tipo de pedidos (ver camada de negócio - 3.2.2).

2.3 Trabalho relacionado

Nesta secção são apresentadas duas soluções existentes no mercado para Vigilância Epidemiológica, providenciando uma breve descrição destas, enumeração de algumas características e em que diferem em relação à aplicação desenvolvida.

2.3.1 VIGIguard

A aplicação *VIGIguard*[2] tem como objectivo a protecção e activa monitorização de infecções em laboratórios, através das seguintes funcionalidades:

- Vigilância: adaptar à linha de testes de susceptibilidade à realidade epidemiológica local; optimizar o uso de antibióticos; avaliação do impacto de acções preventivas.
- Comunicação: disponibilizar dados de controlo de infecção para equipas de controlo, higienistas, epidemiologistas e clínicos de modo a estes adaptarem o cuidado do paciente ao risco.

[1] Clinidata® - *Software* actualmente a ser desenvolvido pela Maxdata, consistindo numa aplicação web dividida por módulos, formando no seu todo uma solução completa para a requisição, gestão e consulta de resultados de exames clínicos.

[2] `http://goo.gl/jb6Hx` - biomerieux-diagnostics.com

- Alertas: configuração de alertas de acordo com as recomendações nacionais e locais, com alertas exibidos em tempo real.
- Automatização: simplificação na elaboração de estudos estatísticos; manter os dados estatísticos actualizados em tempo real.

Este sistema de informação para a vigilância epidemiológica é desenvolvido pela empresa *bioMérieux Diagnostics*.

2.3.2 Hepic

A aplicação *Hepic*[3] permite gerir todo o processo de vigilância epidemiológica numa unidade de saúde. A arquitectura apresentada na figura 2.1 permite suportar as seguintes funcionalidades:

- Integrações automáticas de dados laboratoriais, farmácia e HIS.
- Detecção de microrganismos alerta
- Envio de alertas em tempo real
- Cálculo de taxas de incidência de infecção
- Produção de estatísticas em tempo real

Este sistema de informação para a vigilância epidemiológica é desenvolvido pela empresa *First*.

2.3.3 Conclusões

Ambas as aplicações têm algumas semelhanças com o sistema de informação para vigilância epidemiológica desenvolvido, nomeadamente as integrações, alertas e estatísticas, no entanto não existe o conceito de Caso, isto é, não há algo que reuna toda a informação inerente a um paciente, assim como não há a noção de Eventos e Acções, não sendo possível visualizar um histórico de ocorrências na unidade de saúde, que por sua vez permitirá melhorar o controlo e prevenção de infecções.

[3] `http://www.first-global.com/pt/pages/hepic.aspx`

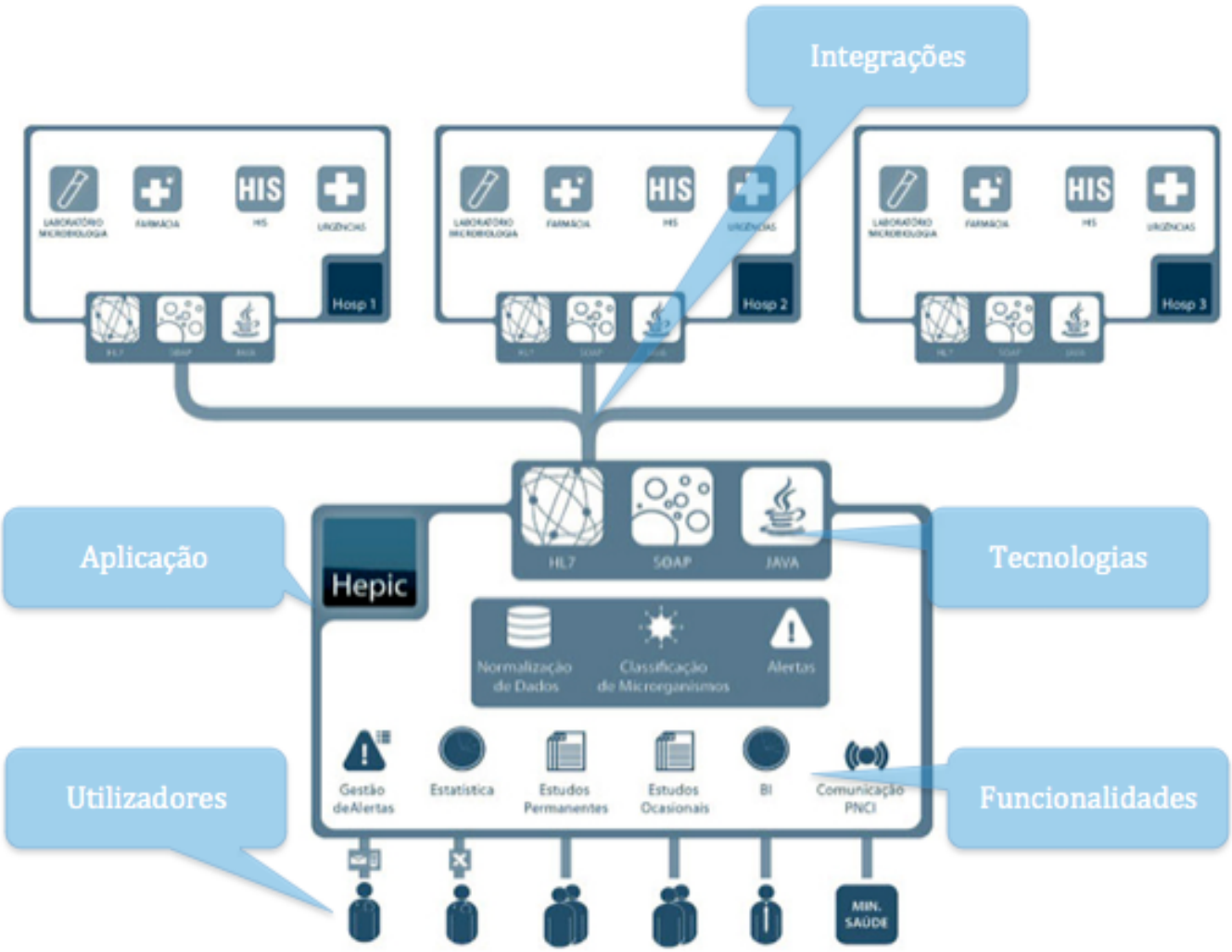

Figura 2.1: Arquitectura da aplicação *Hepic* da *First*.

2.4 Gestão do Projecto - Os 4 P's

Nesta secção são abordados os 4 P's inerentes à gestão de um projecto de *software*. Estes P's constituem as iniciais de 4 palavras-chave intrínsecas a um projecto as iniciais, as **P**essoas (ver 2.4.1), o **P**roduto (ver 2.4.2), o **P**rocesso (ver 2.4.3) e o **P**rojecto (ver 2.4.4).

2.4.1 As Pessoas

Neste PEI, existiram diversos tipos de pessoas envolvidas, agrupadas nas seguintes funções:

- **Gestor de Projecto**: Cabe a este planear, controlar e coordenar a execução do projecto, actuando igualmente ao longo do desenvolvimento do projecto, nomeadamente efectuando medidas, métricas sobre os recursos gastos por oposição ao

produzido, recrutando recursos necessários se exequível.

Neste PEI encontrou-se envolvido apenas um gestor, gerindo todo o projecto, desde a análise até aos seus recursos.

- **Analistas**: Cabe a estes desenvolver documentos de análise sobre funcionalidades do *software* de acordo com os requisitos. As funcionalidades são descritas de modo a que possam ser interpretadas e posteriormente implementadas pela equipa de desenvolvimento.

  A equipa de analistas neste projecto foi composta por 3 pessoas, sendo que um deles também tinha o papel de gestor do projecto. Os restantes analistas focaram-se exclusivamente na elaboração de documentos de análise.

  A secção *Controlo* tem como intuito detalhar os meta-dados do documento.

  A secção *Registo de alterações* tem como intuito providenciar uma forma de efectuar *tracking* às alterações do documento, úteis para a equipa de desenvolvimento na medida em que é possível constatar as funcionalidades novas a serem desenvolvidas (para versões pós a versão inicial).

  A secção *Contexto* tem como intuito detalhar uma dada funcionalidade presente no documento, havendo portanto uma secção de contexto para cada funcionalidade. Esta secção encontra-se divida em diferentes zonas:

  - Introdução:
    Serve esta zona para providenciar uma introdução e contextualizar o membro (ou os membros) da equipa de desenvolvimento que irá desenvolver a funcionalidade em causa.
  - Modelo de dados:
    Esta zona contém as tabelas do modelo de dados a serem criadas e/ou consultadas no âmbito da funcionalidade.
  - Interface do utilizador:
    Esta zona contém os esboços da interface gráfica a ser desenvolvida, assim como algumas notas junto a estes. Tipicamente estas notas servem de auxilio no desenvolvimento, clarificando de alguma forma uma dada funcionalidade em particular presente no esboço (ou parte deste).

- **Equipa de desenvolvimento**: É da responsabilidade das pessoas pertencentes a esta equipa, o desenvolvimento do *software* de acordo com os documentos de análise produzidos pela equipa de analistas, assim como a geração de ideias e soluções (por exemplo através de *brainstorming*) para colmatar adversidades encontradas durante o desenvolvimento.

A equipa de desenvolvimento neste projecto é composta por 4 pessoas, onde o aluno está incluído.

- **Equipa de testes**: Aos elementos pertencentes a esta equipa cabe a execução de testes unitários exaustivos de determinadas funcionalidades, assim como a execução de testes integrados e de sistema. Findados os testes, cabe a esta equipa reportar erros, incoerências ou simplesmente emitir sugestões.

  Esta equipa foi composta por 5 elementos, pelo que nenhum destes elementos faz parte da equipa de análise ou desenvolvimento.

- **Cliente**: Definem os requisitos, sugerem ideias, providenciam linhas de pensamento e casos de uso para a aplicação a ser desenvolvida.

  Este projecto teve como cliente inicial o Centro Hospitalar de Lisboa Norte (CHLN), composto pelo Hospital de Santa Maria[4] e pelo Hospital Pulido Valente[5]. No entanto, tratando-se de um produto, será comercializado e instalado em outras unidades de saúde que manifestem interesse na aplicação.

- **Utilizadores**: Em última análise, estes são de facto as pessoas que usufruem e desfrutam das funcionalidades disponibilizadas na aplicação. Neste PEI, os utilizadores da aplicação foram os enfermeiros, médicos, técnicos e administrativos da Comissão de Controlo de Infecção Hospitalar (CCIH) do CHLN.

É possível observar uma representação gráfica do grupo de pessoas envolvidas no projecto, através do organigrama[6] ilustrado na figura 2.2.

[4] `www.hsm.min-saude.pt`

[5] `http://www.hpv.min-saude.pt/`

[6] Este organigrama representa uma versão bastante simplificada da estrutura da empresa, visto que existem uma série de processos omissos neste documento.

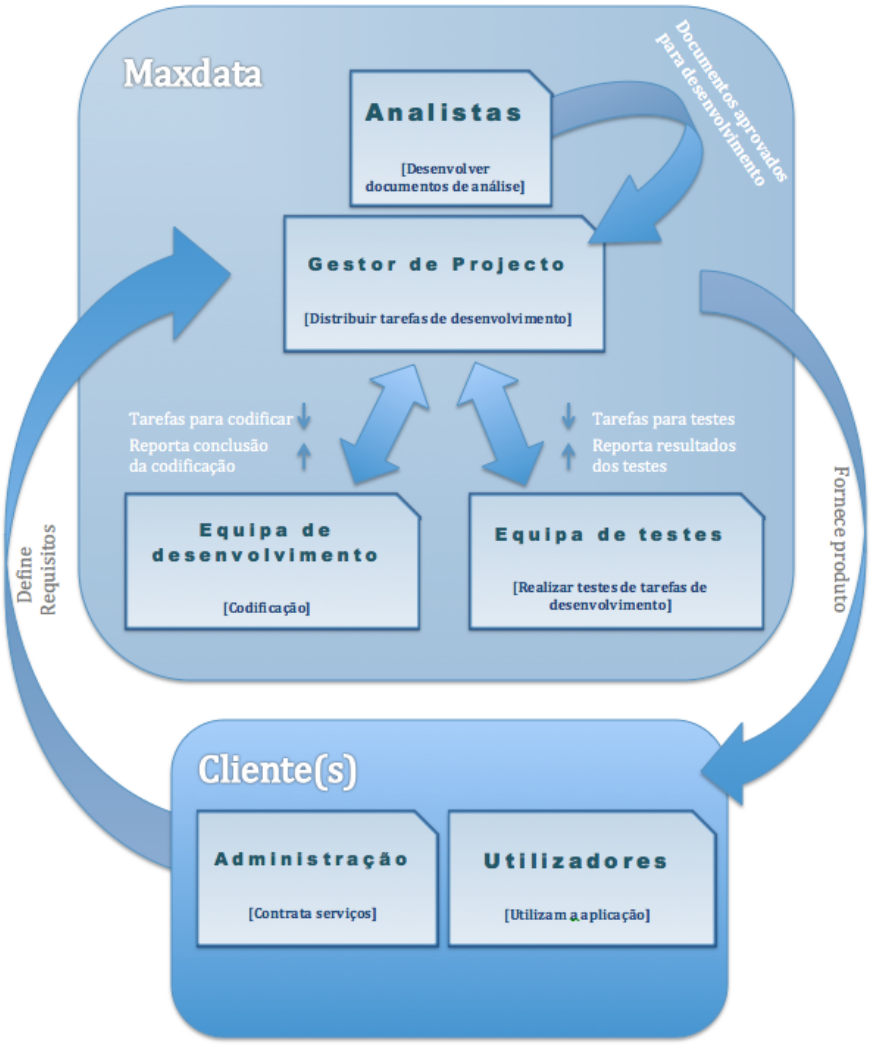

Figura 2.2: Organigrama simplificado da empresa na perspectiva do PEI.

2.4.2 O Produto

O produto encontra-se descrito na secção 2.2.

2.4.3 O Processo

O processo de desenvolvimento da aplicação foi baseado no modelo de desenvolvimento ágil[7], no entanto difere um pouco deste. No modelo ágil, cada iteração visa as seguintes fases:

[7]O método ágil é constituído por um conjunto de metodologias que visam minimizar o risco através do desenvolvimento de *software* em curtos períodos/iterações, onde cada iteração possui única e exclusivamente as tarefas necessárias para implementar determinada funcionalidade.

1. Planeamento
2. Análise de requisitos
3. Codificação
4. Testes
5. Documentação

No modelo de desenvolvimento utilizado, existe um ciclo com *1* a N iterações entre a fase de codificação e testes, assim como entre a fase de análise de requisitos e codificação (no entanto a ocorrência de iterações entre estas últimas fases é rara), isto é, ao serem implementadas e concluídas novas funcionalidades na aplicação, estas são submetidas para testes, onde o resultado destes ditará o eventual retorno a uma nova fase de implementação (nomeadamente para correcção de erros encontrados).
É possível constatar as diferenças entre os modelos visados na figura 2.3.

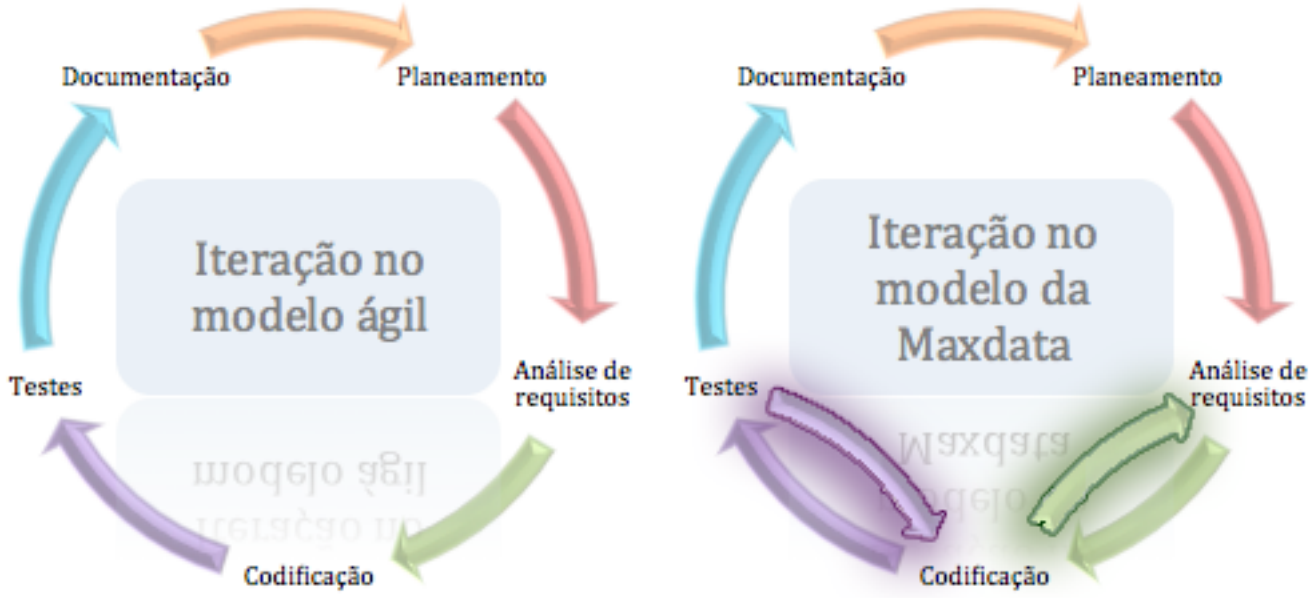

Figura 2.3: Estrutura das iterações no modelo ágil e no modelo utilizado na Maxdata.

2.4.4 O Projecto

O PEI descrito neste documento foi dividido em 3 fases, descritas nas subsecções seguintes.

A tabela 2.1 apresenta a duração definida para cada fase.

Fase	Data de inicio	Data de fim	Duração
1	Outubro 2011	Novembro 2011	2 meses
2	Novembro 2011	Maio 2012	6 meses
3	Maio 2012	Junho 2012	1 mês

Tabela 2.1: Tabela de prazos para as fases planeadas

2.4.4.1 1ª fase: Setembro - Outubro de 2011

Nesta fase foi recebida formação por parte dos responsáveis da Maxdata, lida a documentação sobre as tecnologias a serem utilizadas no desenvolvimento do projecto proposto, assim como as tecnologias utilizadas na empresa de um modo geral. Para além de referida leitura, houve igualmente uma vertente prática, na medida em que foram desenvolvidos exercícios de adaptação e integração nas respectivas tecnologias. Foi também nesta fase que se enquadrou a elaboração do relatório preliminar.

2.4.4.2 2ª fase: Novembro de 2011 - Maio de 2012

Esta fase correspondeu à concepção e desenvolvimento do sistema de informação para vigilância epidemiológica, onde o seu desenvolvimento se encontra dividido em várias camadas descritas no capítulo 4.

Nesta fase foi desenvolvida a primeira de três etapas de desenvolvimento planeadas para o projecto, a única dentro do âmbito do PEI. No entanto é possível observar as três etapas planeadas na tabela 2.2.

Etapa	Descrição	Data de inicio	Data de fim	Duração
1	Gestão de casos e mapas estatísticos	Janeiro 2012	Maio 2012	5 meses
2	Integração com aplicações departamentais	—	—	—
3	Integração com aplicações externas	—	—	—

Tabela 2.2: Etapas planeadas para o projecto

Como é possível constatar na figura 2.2, a primeira etapa corresponde ao desenvolvimento da gestão de casos e mapas estatísticos, essencialmente correspondem as principais funcionalidades da aplicação, que incluem:

- Ecrãs de configuração da Vigilância Epidemiológica
- Gestão de Acções
- Gestão de Eventos
- Gestão de Casos

- Mecanismo de automatização de casos
- Integração com a aplicação ClinidataXXI
- Mapas estatísticos dinâmicos

Os ecrãs de configuração da VE correspondem simplesmente a ecrãs auxiliares que facilitam a manipulação de dados de configuração, tal como é como dos tipos de estudo, tipos de acções, etc. Estes dados de configuração por sua vez são utilizados pelos ecrãs de gestão descritos de seguida.

A gestão de acções permite aos utilizadores visualizar as diversas acções planeadas e/ou realizadas pelos diferentes profissionais de saúde, assim como para si próprio, podendo marcar estas como realizadas ou mesmo cancelá-las providenciando a devida razão. As acções são o mecanismo utilizado para ajudar e prevenir a ocorrência de infecções, através de um planeamento delineado para os profissionais de saúdes, que salvo excepções devidamente justificadas, terão que cumprir o seu planeamento.

A gestão de eventos permite visualizar todas as ocorrências registadas que são relevantes para a vigilância epidemiológica, desde resultados de análises, acções realizadas a respostas a questionários. Essencialmente neste ecrã de gestão é exibido o rastreio de uma ou mais ou mais ocorrências.
A gestão de eventos irá permitir o rastreamento e auditaria do controlo de infecções, na medida em que possuem toda a informação necessária inerente à vigilância epidemiológica.

A gestão de casos resume-se essencialmente ao acompanhamento de pacientes, que possibilita a visualização de um histórico completo de um dado paciente, casos associados a este (abertos ou fechados), infecções, tratamentos, alarmes, essencialmente todos os eventos registados para o paciente estão disponíveis. Este âmbito alargado em relação ao paciente é conseguido através da inclusão de diferentes ecrãs de gestão num só (a gestão de casos), isto é, a gestão de casos para além das suas funcionalidades únicas, engloba outros ecrãs dentro de si, herdando consequentemente as suas funcionalidades. No entanto os *ecrãs* integrados são restringidos ao caso/paciente em estudo, visto que, e como já mencionado, o foco de qualquer caso é o paciente.

O mecanismo de automatização de casos é um sistema que executa operações não-triviais de forma autónoma - tipicamente sem intervenção de um responsável/profissional de saúde. Das operações efectuadas por este, encontram-se a capacidade de inferir/(re)abrir e fechar casos. Este mecanismo é semelhante a um motor de inferências, onde a fonte da sua informação está localizada em mapas estatísticos já existentes, formados por outras

aplicações. O cruzamento e análise da informação obtida irá eventualmente desencadear indicadores de infecções em determinadas situações, podendo ou não originar um novo caso ou até a reabertura de um caso (fechado) de um dado paciente.

Por sua vez, os mapas estatísticos são os responsáveis por apresentar os dados contidos no sistema aos profissionais de saúde/utilizadores da aplicação, sendo gerados dinamicamente à medida de quem os requisita, isto é, são personalizáveis. Mapas estatísticos neste tipo de área são essenciais, permitindo observar comportamentos anómalos e actuar perante estes, como por exemplo, actuar perante a detecção de surtos.

Das funcionalidades descritas, apenas a Gestão de Casos (ou pelo menos grande parte desta) não foi desenvolvida pelo aluno.

Todas as tarefas que foram desenvolvidas no contexto desta fase podem ser observadas através de o mapa de Gantt presente na figura 2.4. É possível verificar que este só tem inicio em Fevereiro, tal facto deve-se à existência de uma etapa que antecedeu a etapa descrita na sub-subsecção 2.4.4.2. Esta etapa, denominada de etapa zero, consistiu no desenvolvimento da aplicação base que serviu, e servirá de suporte para todos os módulos a ser integrados nesta, tal como o módulo da vigilância epidemiológica.

2.4.4.3 3ª fase: Maio - Junho de 2012

Esta fase correspondeu à elaboração do relatório final.

Apesar de todo o planeamento prévio efectuado pela Maxdata, a realidade foi um pouco diferente, existindo algum desfasamento entre o planeamento feito e o executado na prática, derivado a alguns factores internos.
Inicialmente, o planeamento efectuado descriminava três etapas na segunda fase deste PEI, no entanto como se pôde constatar na sub-subsecção 2.4.4.2, apenas a primeira etapa foi de facto consumada. As restantes duas etapas irão decorrer fora do âmbito do PEI, no entanto o seu objectivo pode ser consultado na secção 6.2.

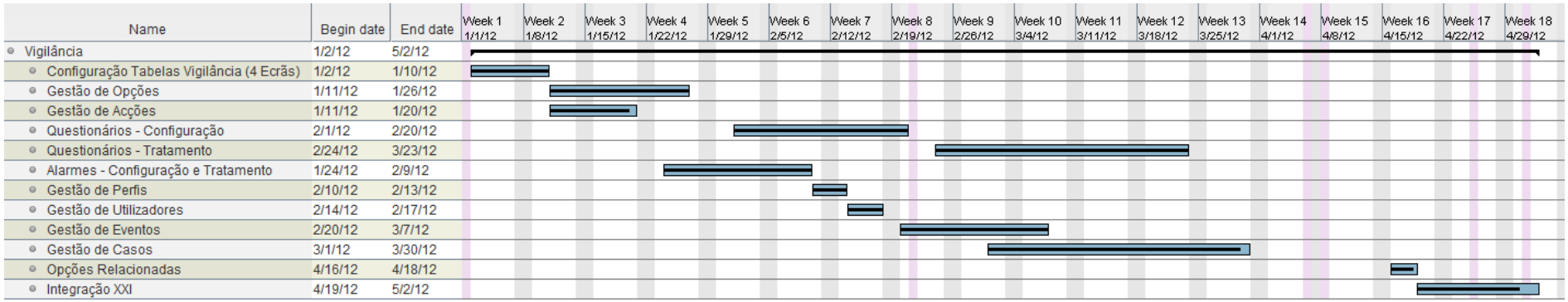

Figura 2.4: Mapa de Gantt referente às tarefas desenvolvidas no âmbito da 2ª fase do PEI.

Capítulo 3

Análise e Desenho

Neste capítulo são descritos os requisitos definidos para a aplicação pelos analistas com base nos requisitos do cliente, assim como a arquitectura e opções de desenho tomadas para o desenvolvimento da aplicação.

3.1 Requisitos

Neste secção são listados os requisitos para a aplicação, que têm como principio definir restrições, funcionalidades, interfaces que permitem cobrir o âmbito da aplicação de modo a que o seu objectivo possa ser cumprido.

3.1.1 C Requisitos

Os C-requisitos constituem um *input* para a concepção e desenho da aplicação, este são os requisitos definidos pelo cliente. Essencialmente são identificados os pontos críticos da aplicação, desde funcionalidades, restrições, limitações e dependências. Em suma, é como que definido o domínio do problema para o projecto em causa.

1. Perspectiva do produto:
   O produto tem de definir e implementar uma série de mecanismos que permitam efectuar e melhorar o controlo de infecções hospitalares.

2. Funcionalidades:

   (a) Possibilitar a gestão de acções

   (b) Possibilitar a gestão de eventos

   (c) Possibilitar a gestão de casos

   (d) Permitir a consulta e definição de mapas estatísticos

   (e) Gerar alarmes/alertas para situações/comportamentos definidos como anómalos

   As funcionalidades enumeradas encontram-se detalhadas na sub-subsecção 2.4.4.2.

3. Restrições e dependências:

   (a) Navegador *Web* (com suporte *HTML5*): O produto como referido depende da presença de um navegador *web* no sistema, sendo portanto esta uma das restrições da solução. No entanto, na área da saúde, na grande maioria dos casos já existem navegadores instalados nos sistemas. Em relação ao suporte para *HTML5*, este é desejável, visto que o comportamento da aplicação nos navegadores que o suportam é de uma maior fiabilidade, derivado às características inerentes à mesma, nomeadamente o suporte para *drag and drop* nos seus componentes, assim como tirar partido do *canvas*[1] para desenhar grafos e/ou gráficos estatísticos em tempo real.

   (b) Bases de Dados: Em relação aos dados, estes podem ser armazenados em diferentes bases de dados relacionais, pelo que a aplicação assenta sobre uma camada de abstracção (ver secção 4.1) que trata dos diferentes pedidos e instruções consoante o SGBD configurado, portanto a aplicação deve suportar um grande número de SGBDs, incluindo os mais usados na unidade de saúde.

   (c) Segurança: Derivado da natureza da aplicação, esta manipulará com frequência dados privados, cuja confidencialidade, integridade e direitos de acesso são bastante valorizados. A disponibilidade é também um aspecto a ter em conta, visto que o âmbito da aplicação a torna vital para o bom funcionamento e operação dos serviços em que se irá incluir.

3.1.2 D Requisitos

Os D requisitos por sua vez apresentam restrições, características mais intrínsecas à aplicação, sendo este definidos pelos analistas/programadores. Face aos C-requisitos, estes como que efectuam a transição entre o domínio do problema e o domínio da solução. No entanto, sem entrar detalhadamente na sua implementação.

1. Interfaces externas: São necessárias interfaces externas, para que outras aplicações/serviços possam usufruir dos serviços prestados pela aplicação através da invocação de *Web-Services*.

2. Desempenho: Serão manipulados tipicamente quantidades avultadas de dados, pelo que a aplicação deverá realizar pesquisas e efectuar operações sobre o modelo de dados de forma rápida e eficaz, de forma a garantir um desempenho aceitável.

3. Padrões: Existem alguns padrões e normas a serem aplicados na aplicação, como é o caso do HL7[2] utilizado na integração/comunicação com equipamentos, assim

[1] `http://www.w3schools.com/html5/html5_canvas.asp`

[2] HL7 - acrónimo usado para referenciar os *standards* HL7 desenvolvidos pela *Health Level 7*, com o intuito de padronizar a linguagem médica

como modelo *Model-View-Presenter* que deverá permitir a reutilização da lógica do cliente em diferentes interfaces gráficas a serem disponibilizadas (ver subsecção 3.2.3).

3.2 Modelos - Arquitectura

A aplicação desenvolvida, tal como já referido trata-se de uma aplicação Web, seguindo uma arquitectura Web[3] padronizada, desde os *URIs* até aos protocolos de comunicação usados.

A aplicação encontra-se essencialmente divida em três grandes camadas, a camada de dados (ver subsecção 3.2.1) onde assenta o modelo de dados, a camada de negócio (ver subsecção 3.2.2) onde se encontra a lógica inerente ao negócio, e a camada cliente (ver subsecção 3.2.3) onde é desenvolvida a interface rica disponibilizada ao utilizador.

É possível observar a interacção entre as diferentes camadas aplicacionais no diagrama de fluxo da figura 3.1.

De seguida são descritas com mais detalhe cada uma das camadas aplicacionais.

3.2.1 Camada de dados

O modelo de dados utilizado trata-se de um modelo relacional, que permite uma representação lógica e consistente da informação inerente à aplicação. A informação é armazenada em bases de dados relacionais, nomeadamente *Oracle Server* e *PostgreSQL* (ver secção 4.1).

As tabelas do modelo relacional implementado encontram-se categorizadas, existindo as seguintes categorias:

1. *Tabelas de configuração*: São tabelas responsáveis por guardar todo o tipo de informação de configuração, informação esta que pode ir desde parâmetros da aplicação até a dados usados pelas *tabelas de dados*.
   Tipicamente estas tabelas têm um tamanho reduzido, tendo um crescimento logarítmico relativamente ao crescimento das tabelas de dados.
   Este tipo de tabela pode ser identificado pelo prefixo **TB** no seu nome.
   Exemplo: Opções - *TbwOption*, Tipos de Evento - *TbvEventType*.

2. *Tabelas de dados*: São tabelas responsáveis por guardar dados oriundos das funcionalidades principais da aplicação.
   Este tipo de tabelas podem atingir dimensões muito elevadas, dependendo do tipo de dados armazenados e do tamanho do cliente.

[3] http://www.w3.org/standards/webarch/

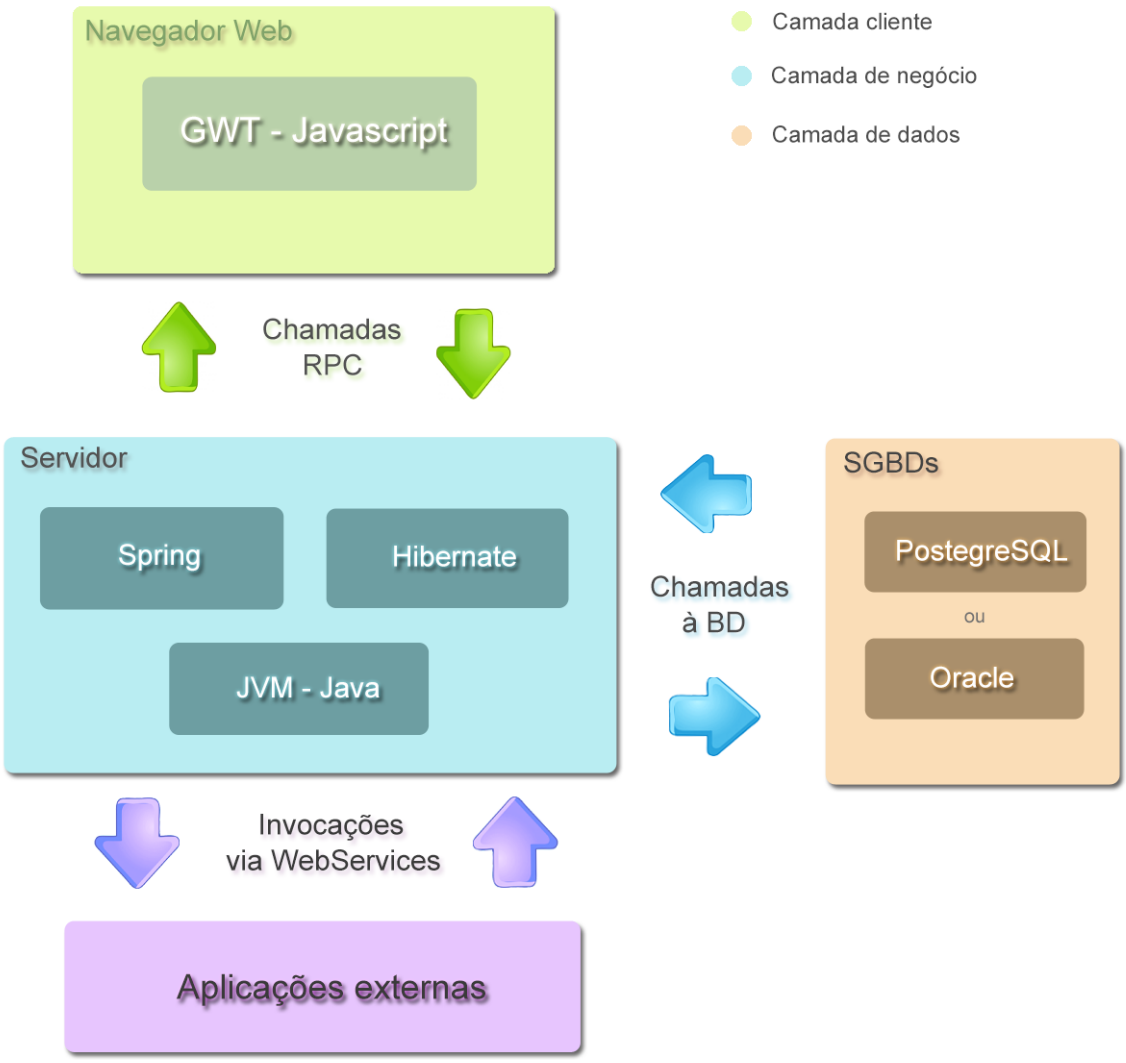

Figura 3.1: Diagrama de fluxo entre as diferentes camadas aplicacionais.

Este tipo de tabela por ser identificado pelo prefixo **DT** no seu nome.
Exemplo: Um Evento disparado - *DtwEvent*

3.2.2 Camada de negócio

É nesta camada que é desenvolvida a lógica do negócio, encarregue dos processos e rotinas que realizam operações sobre o modelo de dados. Como é possível observar no diagrama de fluxo presente na figura 3.1, esta camada é o ponto de entrada do servidor, isto é, toda a comunicação efectuada por um dado cliente para com o servidor aplicacional tem como destino esta camada (a camada cliente nunca comunica com a camada de dados directamente). O cliente pode inclusive ser uma aplicação externa que pretende comunicar com a nossa aplicação (a roxo na figura 3.1).

Em aplicações para a área da saúde, é bastante comum existirem integrações com aplicações externas. Na arquitectura definida para a aplicação, é nesta camada que é feita essa integração. Em termos mais técnicos, são disponibilizados *WebServices* para que terceiros os possam consumir, nomeadamente para exportação de dados (por parte da aplicação externa). No entanto o contrário também é uma situação válida, ou seja, a aplicação actua como fornecedora de informação (exportação para aplicações externas).

Em qualquer um dos casos acima descritos (comunicação oriunda de um canal intra ou inter-aplicação), a camada de negócio é a única que comunica com a camada de dados, permitindo assim um maior controlo sobre o modelo de dados, informação que é exportada de e para a aplicação, assim como permite guardar informação de auditoria necessária para efectuar *tracking* de erros e/ou eventuais adulterações de dados, ou seja, não existem aplicações externas que acedam directamente às bases de dados relacionais (p.e.: usando *dblinks*[4]).

3.2.3 Camada Cliente

A camada cliente foi desenvolvida com base no padrão *Model-View-Presenter* (*MVP*). O modelo *MVP* é um padrão de desenho que deriva do padrão de *software MVC*[5] usado tipicamente para desenvolver interfaces gráficas. O modelo *MVP* encontra-se dividido em 3 componentes:

1. ***Model***
   Essencialmente representa o modelo dados do domínio, os objectos a serem manipulados. Define uma interface com os dados a serem manipulados/exibidos.

[4] É um objecto de *SGBDs* que permite executar operações e consultas num *SGBD* remoto.
[5] *Model-view-controller* - modelo de desenvolvimento usado para isolar a lógica (*controller*) da interface gráfica (*view*), permitindo desenvolver, editar e testar separadamente cada parte.

2. ***View***
   A *view* define a interface gráfica onde são exibidos os dados do *model*, cabe também a esta disparar eventos mediante comandos/operações executadas pelo utilizador.

3. ***Presenter***
   Actua sobre o *model* e *view*, isto é, através de repositórios obtém os dados do *model*, injectando-os na *view*. Tal injecção pode ser precedida por uma formatação prévia dos dados.
   É no *presenter* que se situa a lógica inerente ao contexto definido para a *view*, isto é, todas as operações que a *view* realiza sobre o modelo são executadas no *presenter*. A comunicação entre a *view* e o *presenter* é tipicamente efectuada através de eventos (*view* $\rightarrow$ *presenter*) e contractos definidos (*presenter* $\rightarrow$ *view*).

É possível observar o fluxo de interacção entre *model*, *view* e *presenter* na figura 3.2, onde interacções do utilizador na *view* desencadeiam acções a serem executadas no *presenter*, que actua sobre o modelo de dados.

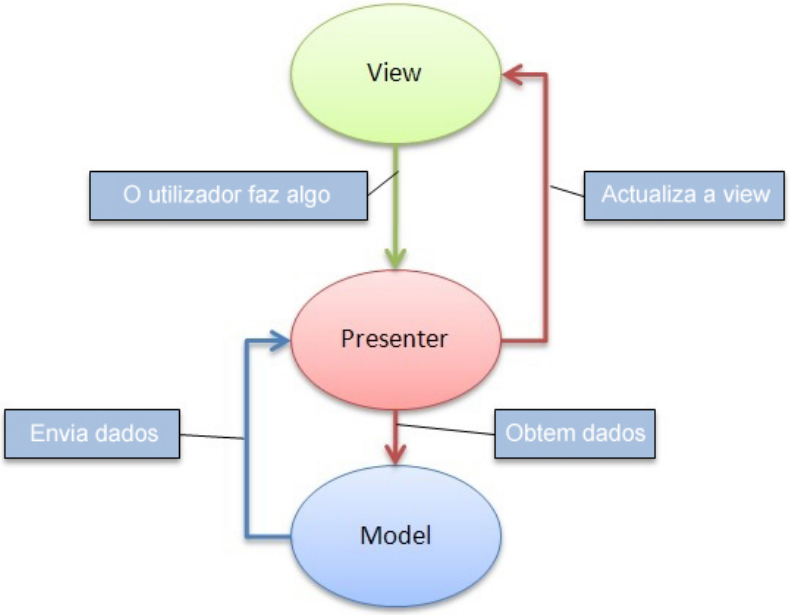

Figura 3.2: Diagrama de fluxo entre os diferentes componentes do *MVP*.

Como já referido, a comunicação entre a *view* e o *presenter* é efectuada por eventos e contratos, no entanto para que o *presenter* obtenha o modelo de dados é necessário que este comunique com o repositório de dados, tal é efectuado através de *RPCs* assíncronas (ver secção 4.3), que derivado das suas características, permitem que a *view* continue dis-

ponível e fluída mesmo durante a execução de pedidos.

Esta arquitectura apresenta algumas vantagens face ao tradicional (e confuso) modelo de desenvolvimento, de entre estas, destacam-se duas grandes vantagens:

1. Reaproveitamento da lógica em diferentes *views*, isto é, é possível desenvolver diferentes *views* (ex: *mobile UI*, *standard UI*) partilhando a lógica inerente a estas, evitando código em duplicado.

2. Execução de testes unitários[6]independentes do ambiente gráfico. Isto é, os testes têm como alvo os *presenters* (*versus views*), abstraindo desta maneira todos factores inerentes à *view*, nomeadamente a tecnologia e componentes utilizados.

A maior desvantagem desta arquitectura prende-se com o facto de tipicamente haver mais linhas de código por funcionalidade desenvolvida, devido à separação existente entre os diferentes componentes *model*, *view* e *presenter*.

[6]Codificados através de ferramentas apropriadas para a tarefa, tal como *jUnit* - `http://www.junit.org`.

Capítulo 4

Implementação

Nesta secção são descritas as tecnologias utilizadas no desenvolvimento do sistema de informação para vigilância epidemiológica, assim como a razão da sua adopção, discriminando as ditas tecnologias por camada aplicacional: dados (4.1), negócio (4.2) e cliente (4.3). Existe no entanto uma tecnologia que é partilhada pelas camadas, esta tecnologia é o *Java*.

- *Java*[1]:
  Java é uma das tecnologias mais utilizadas em larga escala mundialmente. O seu código é compilado para *bytecode* e executado por uma máquina virtual, a *JVM*[2], fornecendo uma camada de abstracção que nos garante uma independência de plataforma onde corre (*"write once, run anywhere"*[3]). Outro ponto a favor desta tecnologia, é a segurança que nos fornece face a outras tecnologias mais primitivas (como *C*/*C++*). Para além de não haver a necessidade de controlar a escrita em memória, a *JVM* por si só actua como uma *sandbox*[4], permitindo no entanto definir e refinar as políticas de segurança a serem aplicadas durante a execução de uma aplicação na *JVM*.

  Esta tecnologia é usada para desenvolver o *core* da aplicação, que se encontra na camada de negócio (ver 3.2.2), no entanto é também utilizada na camada do cliente, com algumas restrições que são descriminadas na secção 4.3.

[1] Java - `www.oracle.com/technetwork/java/`

[2] JVM - *Java Virtual Machine*, é um *software* que fornece um ambiente de execução para o *bytecode* java compilado, assim como uma camada de abstracção entre o *software* e *hardware*.

[3] *Slogan* criado pela *Sun Microsystems* para ilustrar os benefícios do *Java*, fazendo alusão ao suporte para multi-plataforma.

[4] Sandbox - Essencialmente, é um mecanismo de segurança que permite controlar o acesso e uso de recursos do sistema por parte de uma aplicação.

4.1 Camada de dados

A camada de dados encontra-se implementada com base em dois sistemas de gestão de bases de dados, estes foram escolhidos pelos analistas da Maxdata tendo em conta alguns factores chave: desempenho, balanceamento de carga, gestão de recursos, custos de licenciamento, entre outros.

Esta camada assenta ainda sobre uma *framework* de abstracção, removendo grande parte da especificidade inerente aos *SGBDs* usados, visto que tipicamente estes diferem em alguns aspectos, nomeadamente na interpretação das interrogações (*queries*) elaboradas, assim como nas funções/funcionalidades suportadas.

4.1.1 *SGBDs* - Sistemas de Gestão de Bases de Dados

Neste projecto, os sistemas de gestão de bases de dados utilizados foram o *Oracle SQL Server*[5] e o *PostgreSQL Server*[6]. Ambos são relativamente semelhantes, no entanto existem algumas diferenças, quer a nível de configuração, comportamento, quer a nível de sintaxe e desempenho. Tipicamente o SGBD *Oracle* tem maior suporte e desempenho, no entanto as suas licenças são extremamente dispendiosas, e por vezes, simplesmente não compensa a sua utilização face às diferentes realidades existentes em diferentes clientes.

4.1.2 *Framework Hibernate*

O *Hibernate*[7] é um biblioteca *ORM*[8] desenvolvida primeiramente para Java, com o intuito de fornecer uma *framework* que permitisse mapear objectos pertencentes ao *modelo do domínio* em objectos equivalentes no respectivo *modelo relacional* inerente a bases de dados, providenciando uma camada de persistência transparente, isto é, permite representar tabelas em classes *java* que lhes são equivalentes, reflectindo e persistindo as alterações sobre esses objectos (*POJOs*[9]) na respectiva tabela.

O *hibernate* tem a sua própria linguagem para efectuar queries, denominada de *HQL*[10], permitindo construir *queries* com uma estrutura semelhante ao típico *SQL*, com a diferença de estas serem executadas sobre objectos mapeados pelo *Hibernate*.

Em suma, o *hibernate* fornece uma camada de abstracção entre a aplicação e e base de dados relacional e, tal como o *Java*, a mudança de plataforma (ou neste caso de *SGBD*) tipicamente requer apenas alterações a nível de configuração. Tais factores são importantíssimos, quando a realidade do negócio implica algum grau de adaptabilidade.

[5] Oracle - `http://www.oracle.com/technetwork/database/enterprise-edition/`
[6] PostgreSQL - `http://www.postgresql.com/`
[7] Hibernate - `http://www.jboss.com/products/hibernate/`
[8] ORM - Object-relation mapping
[9] POJOs - *Plain Old Java Objects*
[10] HQL - *Hibernate Query Language*

Neste projecto, o *hibernate* foi usado para desempenhar exactamente o descrito. Mapear *POJOs* nas suas respectivas tabelas, abstraindo a complexidade e sintaxe dos SGBDs usados.

4.2 Camada do negócio

Nesta camada reside toda a lógica inerente ao negócio, isto é, esta camada representa o núcleo da aplicação, possuindo diversos serviços[11] divididos por funcionalidades. Na grande maioria dos casos, estes serviços possuem dependências que os ligam a outros serviços. Para auxiliar a gestão de serviços é utilizado o *Spring*.

4.2.1 *Spring Framework*

O *Spring*[12] é uma *framework* desenhada e desenvolvida para *Java*, sendo constituída por diversos módulos que oferecem uma gama de serviços abrangente. De seguida são enumerados e brevemente descritos os módulos de maior relevância para a aplicação desenvolvida no âmbito do PEI descrito neste documento.

- *Dependency Injection*:
  É um dos módulos *core* do *spring*, é de sua responsabilidade a gestão de objectos e do seu ciclo de vida. Tal é efectuado providenciando mecanismos que permitem a configuração e gestão desses objectos através de reflexão[13].
  Os objectos criados pelo referido módulo são denominados de *Beans*, estes são configurados através de *XML* onde são guardadas as suas definições e dependências. Esta abordagem permite a configuração e personalização de uma aplicação de uma forma simples e consistente, suportando uma integração com grande parte dos ambientes de produção *Java*.

- *Data Access*:
  Este módulo fornece suporte para diferentes *drivers* e *frameworks* que lidam com bases de dados, como *JDBC*[14] e o já mencionado *Hibernate*, sendo este último de facto o utilizado neste projecto.
  É de salientar que o *spring* não oferece nenhuma *API* comum, abstracção para aceder às bases de dados de forma transparente e independente do *driver*/*frawework* utilizado, em vez disso, oferece o acesso directo às *APIs* suportadas.

[11] Um serviço é constituído por uma série de métodos expostos para que possam ser invocados remotamente pela aplicação interna (*RPCs*) e/ou aplicação externa (*WebServices*).

[12] `http://www.springsource.org/`

[13] Reflexão - Capacidade de um *software* observar e modificar a sua estrutura e comportamento em tempo de execução.

[14] JDBC - Consiste numa *API* desenvolvida para *java* que define métodos que permitem um dado cliente aceder e manipular a bases de dados.

4.2.2 *Java*

Esta tecnologia é utilizada em conjunção com o *Spring* para disponibilizar os serviços a serem invocados pela camada aplicacional cliente. Estes serviços podem ser de vários tipos, desde serviços de impressão, até serviços de acesso à camada de dados. Será com o auxilio do *Java* que a integração com outras aplicações/serviços será efectuada, tipicamente esta integração será desenvolvida através de *Web Services*[15] disponibilizados pelas referidas aplicações/serviços.
Futuramente, caso seja necessário fornecer dados a outras aplicações/serviços, quer seja de foro estatístico ou simplesmente dados referentes a pacientes, estes serão disponibilizados igualmente por *Web Services*.

4.2.3 *Apache CFX*

O *Apache CFX*[16] constitui uma *framework* para *Web Services*, na medida em que implementa uma série de protocolos[17] que podem ser usados *out of the box*.
Esta *framework* é usada em conjunção com o Spring para expor métodos/serviços situados na camada de negócio em *Web Services* que podem ser invocados por aplicação internas à Maxdata ou externas (integrações).

4.3 Camada cliente

A camada cliente foi implementada sobre uma *framework* da *Google*, o *Google Web Toolkit*[18] (*GWT*). O *GWT* é uma ferramenta *open-source* que permite desenvolver *RIA's*[19] numa linguagem de programação de alto nível, em particular, o *Java*. Na base desta ferramenta está o seu compilador, que interpreta o código *Java* e o traduz o mesmo para *HTML* e *JavaScript*, que por sua vez é interpretado nativamente pelo *browser*.

O *GWT* suporta a tradicional arquitectura cliente-servidor, onde a comunicação é feita feita através de *RPCs*[20] assíncronas, seguindo a metodologia definida para a Web 2.0 através da utilização de *AJAX*[21].

O *GWT* encontra-se dividido em diferentes componentes, dos quais destacam-se 4 como sendo os seus principais:

[15] Web Services - é uma solução utilizada na integração de sistemas e na comunicação entre aplicações distintas.
[16] `http://cxf.apache.org/`
[17] SOAP, XML/HTTP, RESTful HTTP, CORBA.
[18] https://developers.google.com/web-toolkit/overview
[19] *Rich Internet Applications*
[20] Chamada a procedimentos remotos
[21] *Asynchronous Javascript and XML*

Compilador Tal como já referido, este componente é a base da ferramenta, tendo como função traduzir o código *Java* em código *HTML* e *JavaScript*, para que o *Browser* o possa interpretar e executar correctamente. Este efectua ainda diversos processos de optimização, de entre eles:

- *Pruning*: remove classes, interfaces, métodos e propriedades não referenciadas.
- *Type tightening*: substitui tipos genéricos por tipos em concreto quando possível.
- *Method Call Tightening*: actualiza as chamadas a métodos polimórficos, evitando a determinação do método a invocar em tempo de execução.
- Remoção de código morto: remove código inatingível e logicamente irrelevante.
- *Method Inlining*: substitui chamadas a métodos que não referenciem outros métodos externos que não possuem mais que 2-3 linhas pelo corpo do próprio método.

Modo desenvolvimento Este componente permite aos programadores correr e executar código/aplicações *GWT* numa *JVM*, isto é, o código não é compilado para *JavaScript*. No entanto é necessário um *plugin*[22] instalado no *browser*, para que o código *Java* seja correctamente interpretado pelo navegador *web*.
O modo desenvolvimento, tem a grande vantagem de permitir testar e depurar o código desenvolvido de uma forma muito mais rápida do que compilando para *HTML* e *JavaScript*.

Biblioteca de emulação do *JRE*[23] Cabe a este componente providenciar uma implementação em *JavaScript* de grande parte das classes base disponibilizadas em ambiente *Java* pertencentes ao pacote *java.lang* e *java.util*.

Biblioteca Web UI Este componente disponibiliza uma série de *widgets*[24], assim como interfaces e classes que podem ser usadas para a criação de componentes gráficos.

Um resumo da arquitectura implementada e desenvolvida pela *Google* para o *GWT* pode ser observada na figura 4.1.

[22] *GWT Developer Plugin*
[24] Componente visual de uma interface gráfica do utilizador

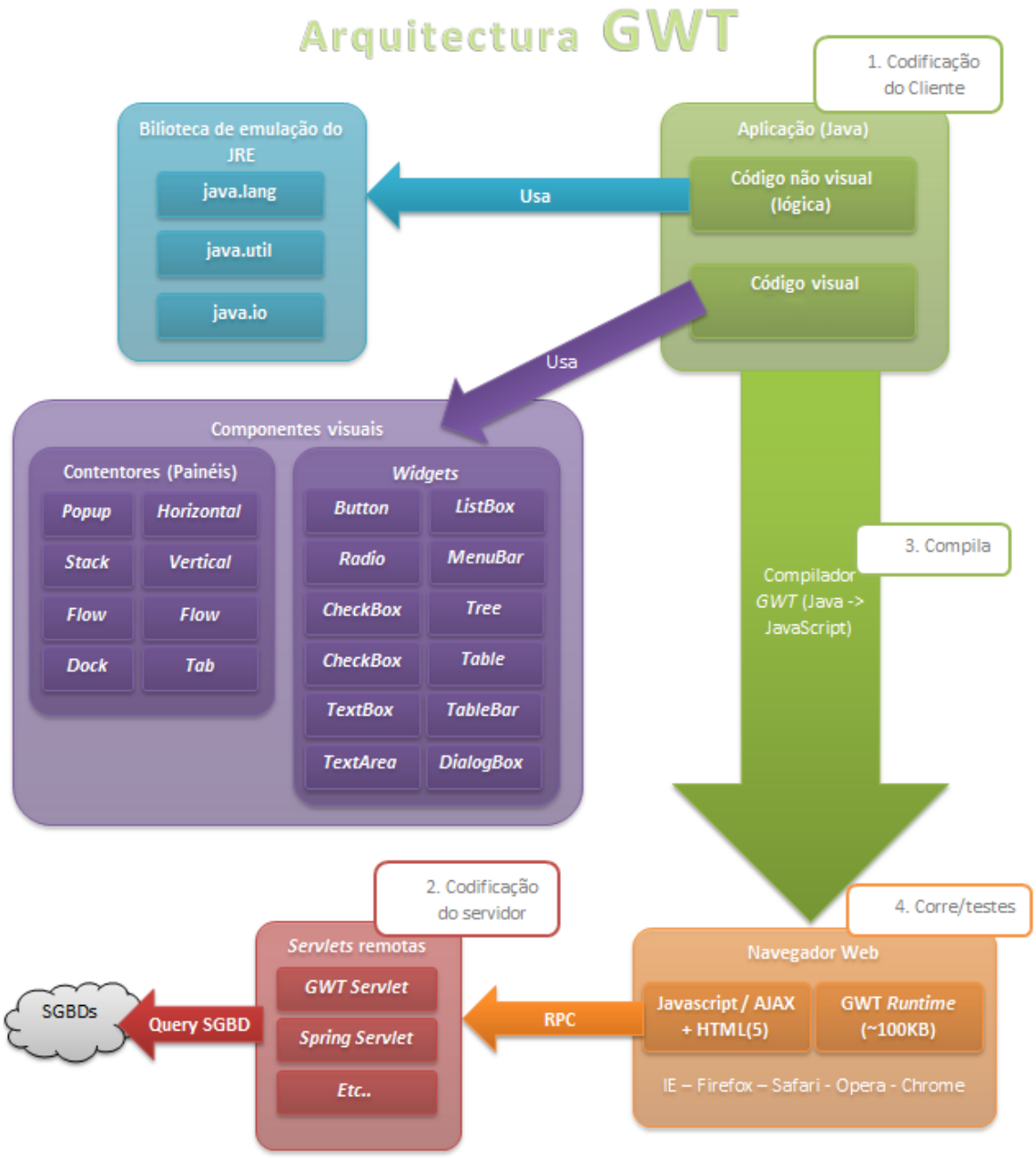

Figura 4.1: Arquitectura do *GWT*.

Para além das características já mencionadas, o *GWT* possui ainda suporte para:

- Gestão de histórico no *browser*
- Integração com *JUnit*[25]
- Internacionalização (*i18n*) e localização (*L10n*)
- Código nativo *JavaScript* embebido em código *Java*

[25] *JUnit* é uma ferramenta *open-source* que permite a criação de testes unitários automatizados.

Em comparação com *RIA's* desenvolvidas em *JavaScript* puro, as *RIA's* baseadas em *GWT* para além das vantagens inerentes à codificação numa linguagem de alto nível, são mais fáceis de testar (através dos mencionados testes unitários), erros de codificação são tipicamente descobertos sem a necessidade de efectuar testes (automáticos e/ou manuais) derivada da verificação feita pelo compilador, desde simples imagens, *CSS*, até à lógica das *views*.

Capítulo 5

Resultados

Neste capítulo são apresentados os resultados práticos obtidos até à data de conclusão do PEI, visto que, embora o projecto se encontre num estado avançado, este ainda carece de algum desenvolvimento.

Os resultados finais após as iterações efectuadas em cada uma das tarefas envolvidas no desenvolvimento da aplicação são os seguintes:

- Em causa estiveram 9 tarefas.
- Foram despendidas 126.53 horas de análise, 616.48 horas de desenvolvimento e 18.42 horas de testes.
- Em média, foram reportados 5 erros relacionados com a interface gráfica por funcionalidade.
- Em média, foi reportado 1 erro relacionado com a lógica do negócio.
- Em média, foram efectuadas 6 sugestões por funcionalidade.

A tabela 5.1 apresenta os resultados agrupados por funcionalidade.

Funcionalidade	Tempo de análise	Tempo de desenvolvimento	Tempo de testes	Erros de UI	Incoerências de dados	Sugestões
Ecrãs de Configuração da VE	16h	57h20m	6h45m	3	0	7
Gestão de Acções	32h30m	138h50m	7h8m	6	1	6
Gestão de Eventos	28h45m	148h50m	4h30m	4	1	5
Gestão de Casos	49h15m	271h20m				
Total:	126h30m	616h20m	18h23m	13	2	18

Figura 5.1: Tabela com os resultados das diferentes funcionalidades.

Capítulo 6

Conclusão e Trabalho Futuro

Neste capítulo são tiradas as ilações sobre os tópicos discutidos e apresentados ao longo deste documento, não esquecendo o trabalho que terá lugar fora do âmbito deste PEI.

6.1 Conclusão

Neste documento foi apresentado um sistema de informação para vigilância epidemiológica, cujas características permitirão aos seus utilizadores melhorar gradualmente o controlo e prevenção de infecções. Tal foi constatado após uma reunião efectuada com o cliente sobre o sistema de informação a 12 de Abril de 2012, onde foi realizada uma demonstração da aplicação desenvolvida até à data. Na altura, esta encontrava-se a cerca de 15% do fim da 1ª etapa (subsecção 2.4.4.2). No entanto, o *feedback* foi extremamente positivo, não tendo havido qualquer pedido de alteração.

Do ponto de vista do aluno, este PEI permitiu o enquadramento no mundo do trabalho, onde desenvolveu hábitos e rotinas de trabalho, adquiriu conhecimentos e conceitos inerente à área de estudo, assim como a nível técnico, nomeadamente ao nível da metodologia (subsecção 2.4.3) e arquitectura (secção 3.2) de desenvolvimento. Esta última, sendo inclusive utilizada e aconselhada pela *Google*[1][2], provou ser trabalhosa, no entanto, os seus frutos serão colhidos num futuro próximo, ao ser implementado o suporte para dispositivos móveis na aplicação.

[1] `https://developers.google.com/web-toolkit/articles/mvp-architecture`
[2] `https://developers.google.com/web-toolkit/doc/latest/DevGuideMvpActivitiesAndPlaces`

6.2 Trabalho Futuro

Tal como mencionado na sub-subsecção 2.4.4.2, inicialmente a segunda fase do PEI tinha três etapas planeadas, no entanto só foi de facto consumada a primeira das três. As duas restantes etapas são os próximos passos antes da aplicação ser entregue ao cliente.

- **2ª etapa - Integração com aplicações departamentais**

  Nesta segunda etapa será efectuada a integração com outras aplicações departamentais[3], contendo informação especifica a ser recolhida e utilizada na investigação dos casos e infecções. Terão que ser igualmente definidas regras para regular e controlar o comportamento das inferências efectuadas com base nestas novas fontes, tal é requerido para garantir a qualidade dos resultado, um último caso, da própria aplicação.

- **3ª etapa - Integração com aplicações externas**

  Nesta etapa final, similarmente à fase anterior, é planeada a integração com aplicações externas, no entanto esta integração tem com objectivo a exportação de dados relativos a casos finalizados e dados adquiridos em aplicações do PNCI[4].

  Esta fase tem como objectivo principal, diminuir a quantidade de informação que é introduzida manualmente nesta rede de informação. No entanto, por vezes tal não é exequível (campos de preenchimento obrigatório a submeter não inferidos ou apenas conhecidos de fonte externa), portanto a inserção manual de informação será igualmente suportada.

Inerente à aplicação está ainda o suporte necessário para o bom funcionamento desta, a curto e longo prazo. Este suporte pressupõe actualizações de versão da aplicação, onde são incluidas novas funcionalidades (por ventura pedidas pelo próprio cliente), assim como correcções detectadas ao longo da execução da aplicação em ambiente de produção.

Faz igualmente parte dos planos da Maxdata, implementar suporte para dispositivos móveis na aplicação. Sendo esta uma das razões da adopção da arquitectura MVP.

[3] Farmácia, Cirurgias, etc.

[4] PNCI - Programa Nacional de Prevenção e Controlo da Infecção associada aos Cuidados de Saúde (despacho nº 14178/2007)

Bibliografia

[1] Maxdata Informática. *Documentos internos*. não publicado.

[2] Madhusudhan Konda. *Just Spring*. O'Reilly Media, first edition, July 2011.

[3] Haley RW, Gaynes RP, Aber RC, and Bennet JV. *Hospital Infections*. third edition, 1992.