UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

DESENVOLVIMENTO DE SOLUÇÃO MULTI-CANAL PARA INSTITUIÇÃO FINANCEIRA DE SEGUROS

Hugo Jorge das Neves

PROJETO

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Sistemas de Informação

2013

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

DESENVOLVIMENTO DE SOLUÇÃO MULTI-CANAL PARA INSTITUIÇÃO FINANCEIRA DE SEGUROS

Hugo Jorge das Neves

PROJETO

Trabalho orientado pelo Prof. Doutor Pedro Miguel Frazão Fernandes Ferreira e coorientado por Ricardo Alexandre Dias Castanheira

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Sistemas de Informação

2013

Agradecimentos

Em primeiro lugar quero agradecer aos meus pais, não pelas palavras de incentivo e muito menos pelas reprimendas quando as coisas correram mal. Agradeço sim pelo exemplo que me deram e continuam a dar todos os dias: um exemplo de trabalho, esforço e determinação. Estas qualidades, que fui adquirindo ao longo dos anos, foram cruciais para conseguir concluir a licenciatura e agora o mestrado, numa área que é muitas vezes frustrante e desgastante e que requer por isso muita força de vontade. Este tipo de agradecimento é extensível à minha família mais próxima que, tal como os meus pais, partilham desta forma de estar na vida. Obrigado mana, primas e tios.

À minha namorada Rute, agradeço por ter aparecido na minha vida na altura certa, quando precisava de uma mudança a vários níveis, tendo sido uma inspiração durante os anos que temos passado juntos.

Quero agradecer também aos meus colegas de universidade, principalmente àqueles que me acompanharam durante várias cadeiras. Obrigado João, Fábio, André, Oliveira e Miguel, por todas as horas que passámos juntos a bater com a cabeça para fazer os projetos. Um agradecimento especial ao Bruno, por ter sido meu colega de grupo em muitas cadeiras e por termos ultrapassado uma catrefada de problemas, bem como por me ter mostrado como ser um melhor estudante da área de informática. Obrigado também a todos os outros que fizeram parte em algum momento da minha vida na faculdade, tenha sido ele relacionado ou não com os estudos.

Em relação ao estágio, tenho que agradecer à Unisys por ter confiado em mim e também a todos os que me ajudaram a integrar no mundo do trabalho e que continuam a ajudar-me todos os dias com os seus conhecimentos. Obrigado em particular ao Diogo Gonçalves, por ser o meu grande pilar durante estes meses. Ajudou-me e defendeu-me mais do que devia e considero-o um exemplo a seguir no mundo do trabalho. A ele lhe devo muito do que tenho aprendido e algumas das oportunidades que tive.

Finalmente, agradeço ao meu coorientador Ricardo Castanheira pela injeção inicial de conhecimentos sobre seguros e pelo apoio e críticas construtivas durante a escrita da tese, e ao meu orientador Pedro Ferreira, pelos comentários e sugestões importantíssimas para a melhoria constante deste relatório final.

Às minhas avós.

Resumo

Este estágio curricular, denominado “Desenvolvimento de solução multi-canal para instituição financeira de seguros”, consistiu na implementação de várias funcionalidades para uma instituição portuguesa de seguros.

As funcionalidades pertencem principalmente a dois projetos que estão em curso na seguradora. O primeiro tem como objetivo rever a comunicação com os clientes, que se traduz, na particularidade deste estágio, no suporte à implementação de documentos de saída (*outputs*), especificamente na alteração da forma como são apresentados os documentos aos clientes. O segundo projeto tem como objetivo a criação de um novo portal, que tem como principais diretrizes ser uniforme a todos os canais da seguradora, sejam eles internet, intranet ou extranet. Dentro deste projeto, é também objetivo migrar grande parte dos serviços de negócio da seguradora para *software* de código aberto (*open source*), bem como configurar ferramentas de produtividade para apoiar o novo paradigma. Os serviços de negócio consistem geralmente em comunicação cliente-seguradora ou mediador-seguradora e são disponibilizados na forma de páginas *web*.

A solução para dar suporte à implementação de *ouputs* passou por criar novos ficheiros relacionados com o documento a melhorar, neste caso da simulação de um seguro casa ou multirriscos, sendo que estes ficheiros representam o que é denominado de objeto de dados e são escritos na linguagem C *Sharp* (C#). Este objeto de dados foi desenvolvido com base em documentação técnica mantida dentro da seguradora.

A solução para dar suporte à migração dos serviços de negócio consiste em usar uma ferramenta que está a ser desenvolvida na seguradora e que, tendo como base *software* de código aberto, serve para redesenhar os ecrãs a serem migrados e para os exportar para código *Java*. A implementação das regras de negócio foi feita nesses ficheiros *Java* que, usando o padrão *Model-View-Controller* (MVC), comunicam depois com as bases de dados e com os serviços *web* da instituição. Quanto às ferramentas de produtividade, foi configurada a ferramenta *Jenkins*, passando agora a serem feitas por esta ferramenta todas as compilações de código.

Palavras-chave: *model-view-controller*, desenho de interfaces, implementação de páginas *web*, ferramentas de produtividade, implementação de objetos *middleware*

Abstract

This traineeship, entitled “Development of a multi-channel solution for a financial insurance institution”, consisted in the implementation of various tasks for a Portuguese insurance company.

These tasks are included in two main projects that are underway at the company. The first one aims to review the communication with customers, which in this traineeship leads to supporting the implementation of new outputs, that is, change how some documents are presented to clients. The second project aims to create a new portal, which has the main goal of being uniform to all channels of the company, whether they are internet, intranet or extranet. This project also aims to migrate much of the company's business services into open source software and to setup productivity tools to support the new paradigm. The business services usually consist of client-company or broker-company communication and are available to the various channels of the company in the form of web pages.

The solution to support the implementation of outputs consisted in the creation of new files related to the document requiring improvements, in this case the house insurace simulation. These files represent what is called a data object and are written in the C Sharp (C#) programming language. This data object was developed on the basis of technical documentation maintained within the company.

The solution to support the migration of business services consisted in using an open source tool that is being developed at the company, in order to redesign the screens to be migrated and to export those screens into Java code. The business rules that define each business service were implemented in that Java code, using the Model-View-Controller (MVC) architecture, which in turn communicates with the databases and web services of the company. Considering the productivity tools, the Jenkins application was configured. All the software is now being compiled by means of that tool.

Keywords: model-view-controller, interface design, web pages implementation, productivity tools, middleware objects implementation

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Enquadramento

Este estágio curricular foi desenvolvido numa empresa multinacional de consultoria de informática, doravante denominada consultora, sendo necessário ter em conta o meio em que essa consultora se insere, já que a consultoria informática caracteriza-se por um ambiente de rápida mudança e imprevisibilidade, dificultando o planeamento das tarefas a executar no estágio. As tarefas do estágio estiveram englobadas em projetos de um dos clientes desta consultora, uma instituição financeira que opera na área dos seguros, doravante denominada seguradora.

1.2 Motivação

De uma forma global, as tarefas que foram feitas durante o estágio englobam-se em dois grandes projetos que se encontram em curso na seguradora.

1.2.1 Revisão da comunicação com o cliente

Os principais objetivos deste projeto são a revisão da comunicação com os clientes da seguradora, de modo a identificar oportunidades de inovação ao nível dos meios, conteúdos e desenhos ou traçados dos documentos (*layouts*). Neste sentido, é necessário fazer alterações na forma como alguns documentos da seguradora são gerados e apresentados aos clientes. Para cumprir este objetivo, algumas tarefas do estágio consistiram em alterar código existente ou implementar novo código, usando *Microsoft Visual Studio* 2010 com a linguagem C#, de modo a dar suporte à emissão de documentos de saída.

1.2.2 Novo Portal

Este projeto consiste em implementar e uniformizar um novo portal de acesso aos colaboradores, parceiros, prestadores e grandes clientes. O novo portal tem como objetivo servir de único ponto de entrada nos sistemas da seguradora, apresentando diferentes desenhos dos documentos e funcionalidades conforme o utilizador e respetivo contexto, esbatendo cada vez mais o conceito de várias aplicações e fazendo a transição para um portal com várias funcionalidades, que variam conforme o contexto em que se apresentam.

Dentro deste projeto, está em curso uma migração da componente de gestão de interfaces para tecnologias de código aberto, de forma a reduzir os custos com *software*. Para concluir esta migração foram definidos vários projetos, um para cada conjunto de módulos atualmente existentes, e as tarefas que fizeram parte deste estágio estão contidas em alguns destes conjuntos de módulos, como por exemplo os serviços de negócio, os pagamentos e recebimentos ou os simuladores.

Os serviços de negócio, que correspondem a serviços *web* disponibilizados aos clientes, aos colaboradores ou aos prestadores, estão atualmente implementados em tecnologias da empresa *Microsoft* e, seguindo a linha geral do projeto, serão migrados para tecnologias de código aberto, como a plataforma de portais e gestor de conteúdos *Liferay* [9].

Os serviços de pagamentos e recebimentos contêm vários módulos, como as funcionalidades de caixa, a prestação de contas, os recibos, ou o resseguro aceite, tendo sido este último alvo de migração durante o estágio. Tal como nos serviços de negócio, estas funcionalidades são concretizadas em serviços *web* disponibilizados nos vários canais da seguradora.

As tarefas do estágio referentes a estes projetos consistiram na migração de alguns destes módulos, desde a camada de apresentação até à comunicação com os servidores da seguradora.

Outro tipo de tarefa foi, tendo em conta a migração a efetuar, o desenvolvimento de ferramentas de produtividade para dar suporte aos novos serviços, já que foi necessário, de forma semelhante aos serviços, encontrar alternativas de código aberto à ferramenta de compilação de *software* que é usada atualmente nos projetos assentes em *software* fechado e de âmbito comercial.

1.3 Objetivos

Os objetivos deste estágio consistiram no desenvolvimento de várias funcionalidades para uma seguradora. A solução existente consiste numa arquitetura com quatro camadas: camada de apresentação, camada de serviços *web*, camada de lógica de negócio, camada de acesso a dados e comunicação com os servidores da seguradora. A camada de apresentação é uma interface gráfica baseada em *Active Server Pages NET* (ASP.NET), estando em curso a migração dessa camada para *Java*, sendo que a participação nessa migração constituiu um dos principais objetivos deste trabalho. Os componentes de negócio são atualmente desenvolvidos em C# com *Microsoft Visual Studio*, sendo que a parte migrada passa a ser desenvolvida em *Java* usando a ferramenta *Eclipse*. Nos serviços migrados para *Java* é usado o *Liferay Portal* como base, integrado com o servidor aplicacional *JavaBeans Open Source Software Application Server* (*JBoss*) [6]. Na migração da aplicação resseguro aceite, para além da conversão dos componentes desenvolvidos em C# para *Java*, foi também objetivo a migração dos componentes de *middleware* em *Visual Basic* 6 para C# com *Microsoft Visual Studio* 2010.

Outro objetivo deste estágio consistiu em participar na implementação de negócio usando C# com *Microsoft Visual Studio* 2010, para dar suporte à emissão de documentos de saída. A emissão de documentos de saída consiste em preparar informação de documentos afetos ao negócio da seguradora, ou seja, com base numa entrada de dados, a emissão de documentos preenche a respetiva saída, de forma a que possa ser gerado e apresentado ao cliente um documento no formato *Portable Document Format* (PDF). Este objetivo tem a particularidade de ser auxiliado por documentos técnicos, que fazem a transição entre os requisitos funcionais e os componentes de fluxo de trabalho (*workflow*) transacional, dando a visão de como as operações estão implementadas. Para além dos documentos técnicos, é necessária também uma análise funcional, em que os seus documentos são o resultado do levantamento de requisitos e descrevem detalhadamente o comportamento esperado para cada operação, bem como o modo de distribuição e arranjo dos elementos gráficos.

Foi também objetivo deste estágio a implementação de novas ferramentas de produtividade, nomeadamente um servidor de compilação para dar suporte às funcionalidades que foram migradas para *Java*. Este servidor foi realizado com base na ferramenta de integração contínua *Jenkins* [7]. Os principais objetivos do uso desta

ferramenta são a compilação do *software* produzido, posterior disponibilização e integração nos serviços da seguradora, assim como ter a possibilidade de se ter um controlo das versões dos ficheiros, podendo também manter um histórico do que os programadores fizeram e quando o fizeram. Desta forma, conta-se com o aumento da produtividade, passando para tarefas automáticas o que é feito manualmente pelas pessoas, reduzindo assim a probabilidade de existirem erros humanos.

O objetivo final do estágio consistiu em fazer testes unitários, isolados, ao código desenvolvido, e testes integrados às operações, após a finalização de todas as funcionalidades. Estes testes foram sendo feitos à medida que se foi concluindo cada uma das funcionalidades previstas, já que o ambiente de desenvolvimento está arquitetado de tal forma que, aliado ao uso de um sistema de controlo de versões, se permita que sejam feitos imediatamente os testes necessários à aplicação. Embora no planeamento dê a entender que os testes apenas são feitos no fim, para além de ser perigoso em termos de calendarização seria também incomportável a nível de entregas do produto ao cliente. Por exemplo, durante o estágio foram terminados os serviços de negócio para o canal internet, enquanto que os serviços a serem disponibilizados na intranet não foram todos concluídos, devido ao sítio na internet estar agendado para uma data anterior ao da intranet. Desta forma, os testes unitários e integrados encontram-se diluídos ao longo do tempo de desenvolvimento das funcionalidades.

1.4 Planeamento

O planeamento que foi definido aquando da entrega do relatório preliminar foi o seguinte.

1. Apresentação do projeto e Preparação Inicial (1 semana). Acolhimento, apresentação do projeto, da equipa, objetivos e enquadramento global.
2. Formação (2 semanas). Formação teórica e formação *On The Job* na metodologia e no sistema de desenvolvimento.
3. Desenvolvimento (1 semana). Participação na implementação de negócio para dar suporte à emissão de documentos de saída em *Microsoft Visual Studio* 2010 em C#.
4. Análise e Desenho (2 semanas). Interpretação dos documentos funcionais e documentação técnica de uma funcionalidade protótipo a desenvolver, e análise da solução existente, tendo em conta que se trata de uma migração tecnológica.

5. Desenvolvimento (3 semanas). Implementação da funcionalidade protótipo em todas as suas componentes: camada de apresentação *web*, camada de negócio e acesso a bases de dados. Tecnologia utilizada: *Java*.
6. Relatório preliminar (1 semana).
7. Continuação de Análise e Desenho (2 semanas). Interpretação dos documentos funcionais e documentação técnica de um agrupamento funcional de operações a desenvolver.
8. Desenvolvimento (6 semanas). Participação na migração de serviços de negócio (inclui componentes visuais) sobre a nova plataforma de desenvolvimento em *Java*.
9. Desenvolvimento (4 semanas). Elaboração das novas ferramentas de produtividade.
10. Análise e Desenho (2 semanas). Para migração da aplicação de resseguro aceite.
11. Desenvolvimento (6 semanas). Migração da aplicação de Resseguro Aceite para interface com o utilizador *Java* e *Middleware* de *Microsoft Visual Basic* 6 para *Microsoft Visual Studio* 2010 em C#.
12. Testes Unitários e Integrados (4 semanas). Testes unitários, isolados, ao código desenvolvido e testes integrados às operações, após a finalização de todas as camadas.
13. Relatório de Estágio (2 semanas). Finalização do relatório de estágio.

Devido ao estágio ter sido feito num cliente da consultora, as tarefas realizadas estiveram sujeitas ao planeamento determinado pelo cliente. Este planeamento é feito normalmente de forma trimestral, pelo que se tornou complicado definir um plano de trabalhos exato para os nove meses do estágio.

Em relação ao ponto 11, durante o estágio apenas participei nas tarefas relativas à migração da interface com o utilizador (IU), tendo sido atribuídas as tarefas da migração de *Middleware* a outras pessoas da equipa, devido principalmente a questões de prazos de entrega e sobreposição temporal com outras tarefas do estágio.

Neste sentido e, fazendo uma confrontação com o plano de trabalho presente no relatório preliminar, pode-se afirmar que este foi, à exceção de um pequeno subconjunto de tarefas das constantes no ponto 11, totalmente cumprido, tendo inclusive sido adicionadas tarefas entre o plano de trabalho da proposta de estágio e o definido para o relatório preliminar, como o ponto 9 do plano supra apresentado.

1.5 Estrutura do documento

Este documento está organizado da seguinte forma:

- Capítulo 2 – Trabalho relacionado, onde são enquadradas, quer a nível de tecnologia quer de uma forma genérica, as várias metodologias usadas.
- Capítulo 3 – O trabalho, onde é descrito o trabalho que foi feito, como foi feito e são mostrados alguns exemplos de implementação.
- Capítulo 4 – Conclusão, onde são expostas as conclusões com base no trabalho feito durante os nove meses de estágio, bem como explicadas as dificuldades sentidas e as expetativas para um trabalho futuro.

Capítulo 2

Trabalho relacionado

A nível interno, a migração que está a ser feita atualmente é um exemplo da mudança recente de paradigma que foi feita na seguradora.

Em relação à criação dos ecrãs da camada de apresentação *web*, há uma mudança ao nível da implementação dos requisitos definidos e respetiva avaliação. Os requisitos consistem no que é especificado pelo cliente para cada ecrã, ou seja, consistem na descrição funcional dos componentes que cada ecrã deve ter, bem como nas ações para cada um deles. Anteriormente, os requisitos para estes ecrãs eram definidos por uma equipa específica (a equipa de análise de requisitos), sendo que os mesmos eram posteriormente enviados para a equipa de desenvolvimento, que desenhava os ecrãs no *software Microsoft Visio*. De seguida, era necessário que os ecrãs fossem avaliados pela equipa de análise de requisitos, levando a que existissem muitos fluxos de comunicação. Com a migração, os ecrãs passam a ser desenhados numa ferramenta colaborativa que está a ser desenvolvida dentro da seguradora. Esta ferramenta pode ser utilizada por qualquer pessoa da equipa de desenvolvimento que esteja incluída no projeto (independentemente das suas qualificações técnicas), o que permite que uma pessoa da equipa de requisitos possa visualizar rapidamente o ecrã desenhado e tenha a possibilidade de, por exemplo, deixar facilmente as suas notas referentes a determinado ecrã ou função.

A Figura 1 demonstra numa pirâmide os fluxos de comunicação que acontecem hierarquicamente entre a equipa de funcionais e a equipa de programadores.

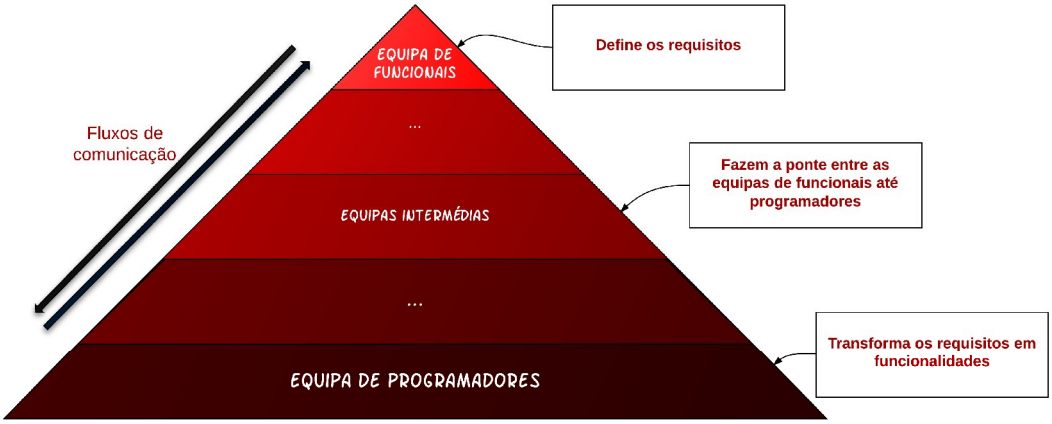


Figura 1: Comunicação hierárquica entre equipas.

Na Figura 2 é apresentada a redução dos fluxos de comunicação que é desejável ao usar a ferramenta colaborativa, não obrigando a que toda a comunicação passe por todas as equipas intermédias entre o topo e a base.

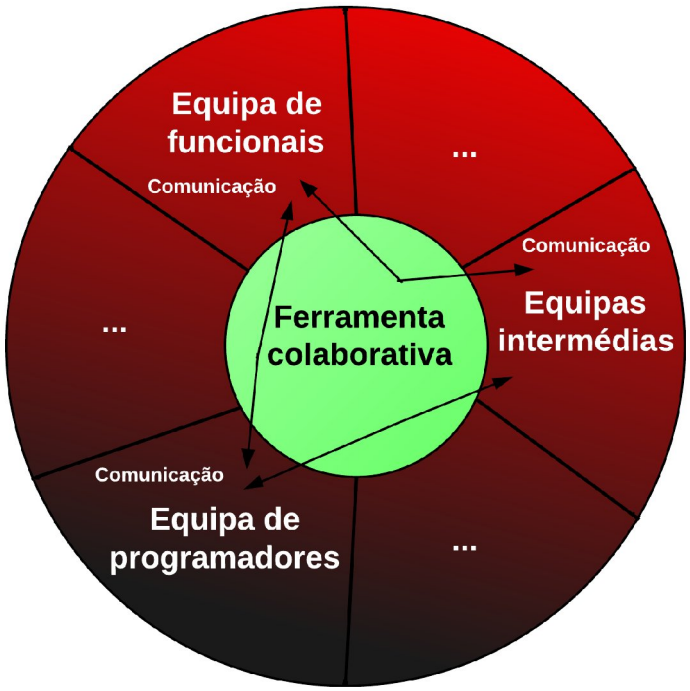


Figura 2: Comunicação direta entre equipas.

Esta redução no custo de comunicação entre equipas é uma das mais importantes vantagens de uma ferramenta colaborativa, o que, aliado a outras vantagens deste tipo de *software* (como a junção de múltiplas perspetivas e especialidades, por exemplo), leva a que o trabalho seja mais eficiente [13].

Esta nova forma de trabalho representa, de uma maneira mais genérica, uma mudança na metodologia que é usada. Na metodologia anterior era necessário que alguém definisse, não só os requisitos, mas um documento completo com as funcionalidades a implementar. Isto passava normalmente pela criação de vários casos de uso, de vários exemplos, sendo que estes exemplos eram bastante rústicos e pouco dinâmicos para o cliente. Depois da avaliação e aceitação desses requisitos, as funcionalidades entravam em produção e raras vezes eram alteradas. Com uma metodologia mais ágil, a equipa de programadores pode mostrar facilmente ao cliente aquilo que interpreta dos seus requisitos, levando não só a que estes sejam muito mais completos e corretos, mas também possibilitando eventuais alterações durante a produção. Esta metodologia está muito mais de acordo com as rápidas mudanças do ambiente de produção informático, dando a possibilidade ao cliente de ver o seu sistema ser constantemente melhorado, de uma forma incremental.

Outra mudança que está a acontecer é ao nível do tipo de *software* a ser usado. Nomeadamente, está-se a passar do uso de *software* fechado e de âmbito comercial para *software* de código aberto e acesso livre. Por exemplo, o código da camada de apresentação era feito em C# e está a ser migrado para *Java*, eliminando a dependência de tecnologias comerciais da empresa *Microsoft*. Da mesma forma, as ferramentas *Microsoft Office* foram substituídas pelo conjunto *LibreOffice*, entre outros exemplos. Analisando as razões que levaram à mudança, pode-se perceber que no *software* de código aberto existe uma comunidade muito maior, mais pessoas a desenvolver e a manter de forma documentada, bem como custos bastante menores, quando comparados com o *software* fechado e de âmbito comercial. A documentação também costuma ser bastante mais fácil de encontrar e de expandir mas, por outro lado, o suporte que é dado aos utilizadores é muitas vezes de menor qualidade, o que pode aumentar o tempo e custo de produção [8][23]. Quanto ao tema da segurança, devido à popularidade das soluções de código aberto, pode existir, no contexto desta seguradora e deste trabalho, uma desvantagem em relação a *software* fechado e de âmbito comercial. Este tema é no entanto discutível, podendo salientar-se a vantagem que quanto mais gente tiver acesso ao código, mais facilmente serão encontradas e resolvidas falhas de segurança no mesmo.

Um conceito importante para a realização deste trabalho é a utilização do padrão de desenho MVC, já que os ficheiros *Java* que servem de base para a implementação

das regras de negócio associadas às páginas *web* estão separados consoante este paradigma. O padrão MVC procura separar a apresentação (*display)* e a introdução de dados da lógica de controlo e isolar a camada de negócio da camada de apresentação. A arquitetura do padrão é apresentada na Figura 3.

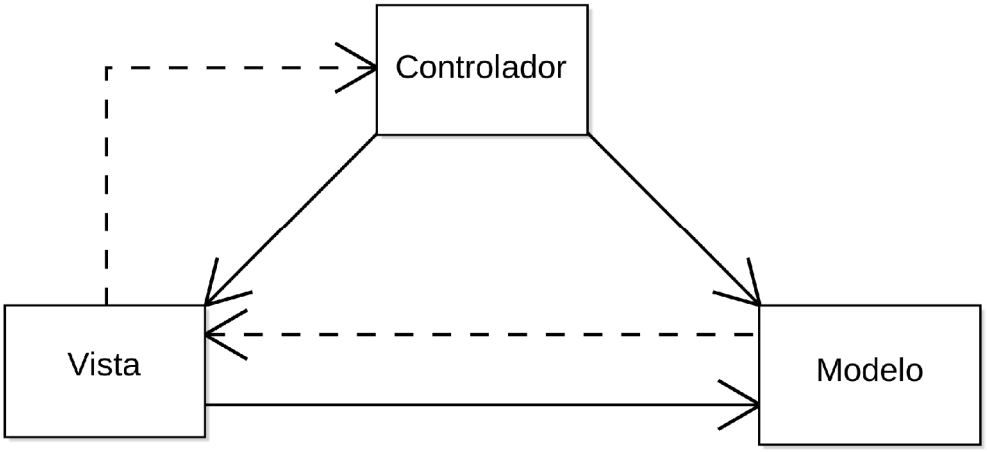


Figura 3: Arquitetura do padrão MVC.

No padrão MVC, a vista apresenta o modelo num determinado estado e é responsável por capturar os eventos gerados por objetos da IU e passá-los ao controlador. Por exemplo, numa página *web* de pesquisa é capturado o evento de clique de um botão "Pesquisar" e é chamado um método do controlador que lhe está associado, passando-lhe como parâmetros os dados existentes no objeto da IU que são necessários para se efetuar a lógica da pesquisa. O controlador disponibiliza métodos para serem invocados pelos vários eventos existentes na vista. Estes métodos criam componentes de negócio (*Business Components*) e invocam métodos dos mesmos. A principal razão da existência do controlador é separar o fluxo de trabalho associado a uma funcionalidade da IU da própria funcionalidade [14].

Uma alternativa ao MVC é o padrão *Model-View-Presenter* (MVP). Neste padrão, que se pode considerar como uma versão evoluída do MVC, a vista recebe os eventos da IU e chama o apresentador quando necessário. O apresentador é responsável por atualizar a vista com os novos dados que são gerados pelo modelo. A arquitetura do padrão é apresentada na Figura 4.

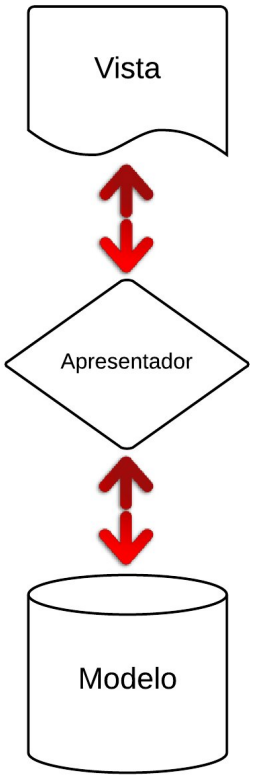


Figura 4: Arquitetura do padrão MVP.

A componente modelo pode ser vista como uma interface para os dados. Qualquer parte de uma página que precise de dados tem que passar pela interface ou pelos métodos definidos pelo programador que está a manter o modelo. Contrariamente ao MVC, no padrão MVP o modelo assume-se quase como um modelo de domínio, comunicando apenas com o apresentador e estando completamente separado da vista.

A vista tem o mesmo significado e funções que no padrão MVC, ou seja, é uma interface passiva que mostra os dados que vêm do modelo e canaliza os comandos do utilizador (eventos) para o apresentador, para que este possa atuar sobre os dados.

O componente apresentador, como já referido, atua como um intermediário entre os outros dois componentes e faz a dissociação entre eles. Toda a lógica e regras de negócio que são precisas para responder a uma ação do utilizador estão implementadas no apresentador. Pode-se fazer uma analogia entre este componente e o controlador no MVC, mas tendo o apresentador muito mais funções, separando de forma absoluta a vista do modelo. Esta diferença tem a vantagem de ser possível fazer testes unitários ao código muito mais facilmente, assim como dar total liberdade ao desenhador da interface, para que este possa fazer as páginas sem se preocupar com as chamadas aos métodos ou implementações das regras de negócio da aplicação. No entanto, ao

programar a camada do apresentador, é necessário mais trabalho manual de associação entre as propriedades do modelo e da vista [15][20][22].

A Tabela 1 demonstra as principais diferenças entre o padrão MVP e o padrão MVC.

MVP	MVC
A vista é completamente passiva, todas as interações com o modelo passam pelo apresentador.	A vista tem alguma funcionalidade e pode comunicar com o modelo diretamente.
O apresentador faz a associação manual de propriedades entre uma vista e um modelo.	O controlador pode mudar de vista sempre que necessário.
Suporte a testes unitários avançado.	Suporte a testes unitários limitado.

Tabela 1: Principais diferenças entre o padrão MVP e o padrão MVC.

Capítulo 3

O trabalho

Tal como especificado nas secções 1.2 e 1.3 deste relatório, os objetivos deste estágio enquadraram-se em dois tipos de projetos que estão a decorrer na seguradora: um que corresponde a uma revisão da comunicação com os clientes da seguradora, e outro que consiste na migração de várias funcionalidades da camada de gestão de interfaces para *software* de código aberto. Dado que se tratam de tecnologias bem distintas e até de camadas separadas dentro da seguradora (embora sejam do mesmo gabinete) irei abordar separadamente as funcionalidades que foram feitas para cumprir os objetivos de cada projeto.

A Figura 5 demonstra a arquitetura, camadas e subsistemas que constituem o sistema de informação da seguradora, com especial destaque para aquelas onde foram desempenhadas funções: *Middleware* e Camada de Gestão de Interfaces (CGI).

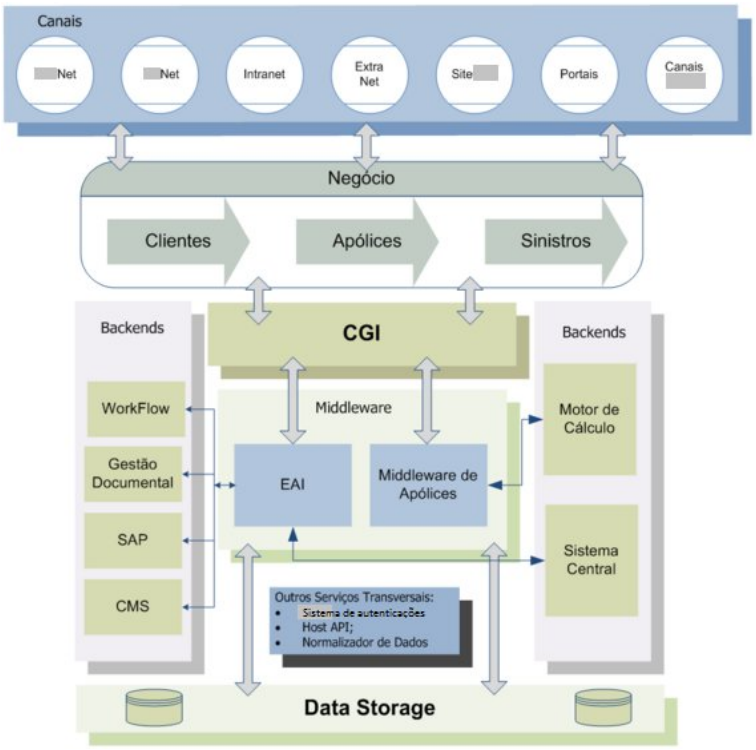


Figura 5: Arquitetura do sistema de informação da seguradora.

3.1 Suporte para emissão de documentos de saída

As primeiras tarefas que foram feitas na seguradora, depois de ter sido apresentado à equipa e integrado nas várias componentes aplicacionais que constituem a camada de *Middleware*, corresponderam à participação no projeto da revisão de comunicação com os clientes.

Este projeto, tendo como objetivo a melhoria da comunicação com o cliente, definiu como requisitos que vários documentos teriam de ser alterados. Refere-se como exemplo o PDF que é gerado, impresso e entregue a um cliente quando este faz uma simulação para um seguro. Neste caso em particular, a tarefa consistiu em alterar o documento da simulação de seguro casa. Para concretizar esta tarefa, foi necessário recorrer a alterações ao projeto "Motor de Impressão", que está englobado na camada de *Middleware*.

O Motor de Impressão serve para gerar vários tipos de objetos de dados, sendo que, cada objeto de dados corresponde a um documento diferente. Depois de gerar um objeto de dados a partir de um contrato que esteja em sessão na aplicação, o Motor de Impressão é responsável por enviar esse objeto de dados para outra secção da seguradora que gera os PDFs. Após ser gerado o PDF correspondente, é enviada ao

Motor de Impressão uma hiperligação para esse ficheiro, que se encarrega de enviar essa hiperligação à aplicação *web*, de modo a que o documento seja apresentado ao utilizador do sistema. Este fluxo de ações é exemplificado na Figura 6.

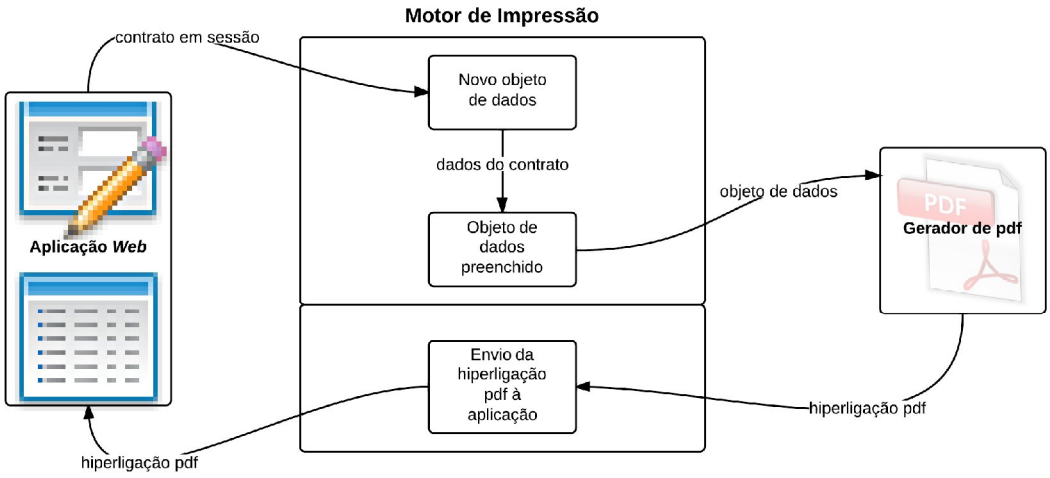


Figura 6: Vista geral do Motor de Impressão.

Um objeto de dados é representado por um conjunto de ficheiros codificados em C#, que definem o conteúdo de determinado documento, ou seja, as suas propriedades. Para este objeto de dados é definida uma estrutura em *Extensible Markup Language* (XML) [12], que servirá para indicar a estrutura do documento, bem como o valor de cada uma das suas propriedades, quando este estiver preenchido. Esta estrutura XML é definida com base em anotações nos ficheiros C#, o que torna o objeto de dados passível de ser serializado [18], já que o Motor de Impressão recebe ficheiros XML da aplicação *web* e, é com base nesse formato que faz as suas ações.

Inicialmente, foi feito um pequeno programa para perceber melhor a linguagem XML em geral, como funciona a sua estrutura em árvore e como se podem obter e alterar alguns nós, em particular. De seguida, foi estudado o mapeamento que é feito entre os documentos técnicos e o código efetivamente escrito no Motor de Impressão. Estes documentos técnicos servem para se perceber facilmente que algoritmos aplicar, de modo a implementar determinada funcionalidade, bem como para perceber a estrutura do objeto de dados.

3.1.1 Estrutura XML

No caso do objeto correspondente a uma simulação de seguro multirriscos, o documento técnico especifica a estrutura XML que o objeto de dados terá que ter, bem

como as entradas e as saídas das suas funções. Esta documentação técnica foi escrita por outro elemento da equipa, de modo a acelerar a minha integração no projeto. A estrutura XML seguinte foi retirada do documento de especificação técnica e corresponde à interface do novo objeto de dados. De modo a facilitar a leitura, foram omitidas (com reticências) secções desta estrutura que não têm interesse para esta secção do relatório. A estrutura XML completa está disponível no Anexo I.

```
<Input>
    <NrContrato />
    <DtConsulta />
    <Accao />
    <CodOpcao Desc="" />
    <CodFraccionamento Desc="" />
</Input>
<Output>
    <Cabecalho>
      ...
    </Cabecalho>
    <TabInfoRodape>
      ...
    </TabInfoRodape>
    <PrecoDestacar>
      ...
    </PrecoDestacar>
    <VantagensTranquilidade>
      ...
    </VantagensTranquilidade>
    <DadosTomador>
      ...
    </DadosTomador>
    <DadosSimulacao>
      ...
    </DadosSimulacao>
    <DadosObjecto>
      <Label />
      <LabelVariosObjectosSeguros />
      <NrObjectosSeguros />
      <ObjectoSeguro />
    </DadosObjecto>
    <DadosMediadorPontoVenda>
      ...
    </DadosMediadorPontoVenda>
    <DetalhePremio>
      ...
    </DetalhePremio>
    <ValorSimulacao>
      ...
    </ValorSimulacao>
    <DetalheObjectoSeguro>
      <LstLocalRisco>
        <LocalRisco>
          <DadosLocalRisco>
            <OSLocalRisco />
            <TipoImovel />
            <MoradaLocalRisco />
            <PostalLocalRisco />
```

```
            <TipoHabitacao />
            <IndPlacaCimentoEntrePisos />
            <AnoConstrucao />
            <IndExisteAlarme />
            <NumeroAssoalhadas />
            <AreaBruta />
          </DadosLocalRisco>
          <LstOSMR>
            <OSMR>
              <IDOS />
              <CoberturasOS>
                ...
              </CoberturasOS>
            </OSMR>
          </LstOSMR>
        </LocalRisco>
      </LstLocalRisco>
    </DetalheObjectoSeguro>
    <IndRevComCliente />
</Output>
```

As secções desta estrutura que estão destacadas são as que tiveram que ser implementadas efetivamente, já que as restantes estavam, ou incluídas noutros métodos, ou eram similares a outro objeto de dados feito para o seguro automóvel, do qual foi necessário replicar alguma da implementação comum. A ideia principal é que, para uma entrada com a estrutura definida neste XML, a saída terá os atributos que estão dentro da etiqueta *output*.

3.1.2 Arquitetura

Em termos de arquitetura, na implementação começa-se por definir as interfaces e as propriedades de cada objeto de dados. Neste caso, isto correspondeu a um ficheiro chamado IDadosOSMR, que contém: os dados do objeto seguro e os respetivos detalhes; os locais de risco e os seus detalhes; e, coberturas inerentes.

O esquema apresentado na Figura 7 mostra a localização dos ficheiros desenvolvidos nos módulos do Motor de Impressão.

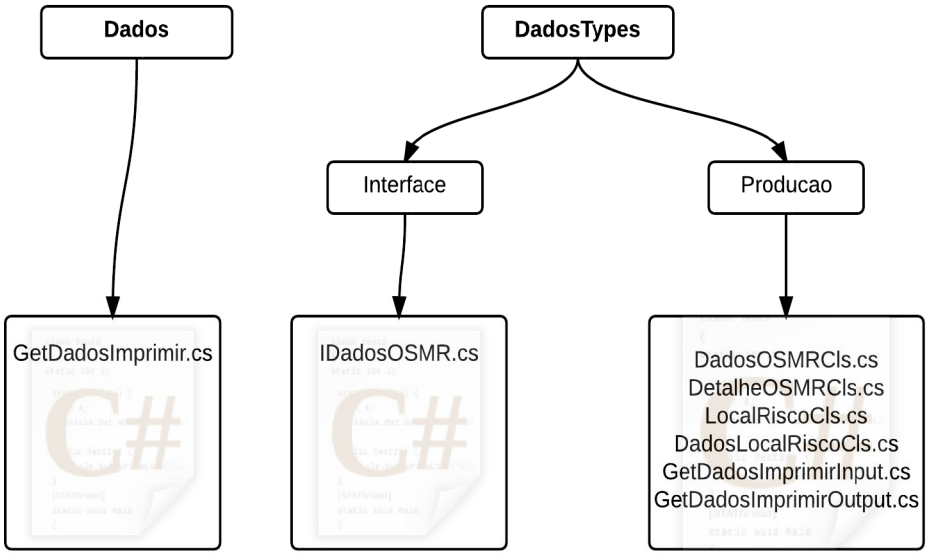


Figura 7: Integração dos ficheiros desenvolvidos na arquitetura existente.

O módulo "Dados" corresponde à implementação propriamente dita e o ficheiro GetDadosImprimir contém objetos do tipo GetDadosImprimirInput e GetDadosImprimirOutput que, por sua vez, têm as propriedades da entrada e da saída, respetivamente. A entrada é obtida através da sessão e a saída é preenchida nesta classe, que será abordada com mais detalhe na secção 3.1.3.

O módulo "DadosTypes" corresponde à definição das interfaces e propriedades de cada objeto de dados.

A Tabela 2 exemplifica o mapeamento entre os ficheiros criados e a estrutura XML definida em 3.1.1:

Ficheiro C#	Etiqueta XML
DadosOSMRCls.cs	OSMR
DetalheOSMRCls.cs	DetalheObjetoSeguro
LocalRiscoCls.cs	LocalRisco
DadosLocalRiscoCls.cs	DadosLocalRisco

Tabela 2: Relação entre os ficheiros C# e a estrutura XML.

No Anexo II deste relatório encontra-se um diagrama de classes dos ficheiros desenvolvidos, apresentando uma visão mais formal e completa da arquitetura de ficheiros.

Depois de definir os atributos que cada ficheiro tem, a relação entre eles, as anotações XML (para a serialização) e o mapeamento correto entre os ficheiros e a

especificação XML, procede-se à codificação do ficheiro GetDadosImprimir, que faz a implementação.

3.1.3 Regras de implementação

As regras de implementação, também especificadas no documento técnico, foram codificadas no ficheiro GetDadosImprimir. Como estas regras estão expressas através de algoritmos genéricos, apresentam a vantagem de serem quase independentes da linguagem usada, desde que essa linguagem suporte o que está definido; bem como, de se ter a documentação guardada e facilmente acessível em qualquer altura. Tem, no entanto, a desvantagem de, para algoritmos mais complicados, ser difícil conseguir-se ter uma definição final dos mesmos sem recorrer primeiro à programação. A Figura 8 exemplifica um excerto da especificação para o objeto de dados casa.

```
// Obtem os dados da simulação
1  GetDados(#input, #output)

// Preenche o output comum a todos os produtos
2  FillDadosSimulacaoBase(#output, #input)

Nota: Fica a faltar o preenchimento das estruturas especificas dos produtos multirriscos: DadosObjecto e DetalheObjectoSeguro

// Seleccionamos o contrato seleccionado
3  Definir #contratoSelecionado com o nº de contrato seleccionado pelo utilizador, isto é, seleccionar o contrato correspondente à
   opção/fraccionamento recebido por input

// Descritivos a imprimir no desenho
4  #output.DadosObjecto.Label = "Dados do Objecto Seguro / Pessoa Segura"
5  #output.DadosObjecto.LabelVariosObjectosSeguros = "Nº de Locais de Risco Seguros:"

6  Construir um dicionário (#dicLocaisRiscoOS) com a relação entre os locais de risco e os objectos seguros
       6.1    Key: Local de Risco (Tipo de dados: Prd.Types.Apolice.LocalRiscoCls)
       6.2    Value: Lista de Objectos Seguros (Tipo de dados: List<ObjectoSeguroCls>)
7  Percorrer os objectos seguros do contrato seleccionado (#contratoSelecionado.LstObjectoSeguro) e agrupar os objectos seguros (#OSLoop) de
   cada NrLocalSeguro (#OSLoop.LocalRisco.NrLocalRisco)

// Nota Importante: 1 Local de Risco pode conter 1 ou 2 Objectos Seguros
       Exemplo1: O Local de Risco "Av Liberdade 1250-149 Lisboa" com os objectos Seguros Imóvel e Recheio
       Exemplo2: O Local de Risco "Av Liberdade 1250-149 Lisboa" com o objecto Seguro Imóvel
       Exemplo3: O Local de Risco "Av Liberdade 1250-149 Lisboa" com o objecto Seguro Recheio

   // Percorrer os Locais de Risco ...
8  Para cada #localRiscoLoop em #dicLocaisRiscoOS
       // Validação ...
       8.1    Se no #localRiscoLoop não existe o objecto seguro Imóvel nem Recheio, Então
         8.1.1  Lança um BusinessError "O Local de Risco " + #localRiscoLoop.NrLocalRisco + " não tem imóvel nem recheio definido."

       // Para cada #localRiscoLoop criar um Local de Risco (#newLocalRisco) a adicionar ao #output.DetalheObjectoSeguro.LstLocalRisco
       8.2    Se o #localRiscoLoop contem os objectos seguros Imóvel e Recheio, Então
         8.2.1  #newLocalRisco.DadosLocalRisco.OSLocalRisco = "Imóvel e Recheio"
       8.3    Caso contrário, se o #localRiscoLoop contem apenas o objecto seguro Imóvel, Então
         8.3.1  #newLocalRisco.DadosLocalRisco.OSLocalRisco = "Imóvel"
       8.4    Caso contrário
         8.4.1  #newLocalRisco.DadosLocalRisco.OSLocalRisco = "Recheio"
```

Figura 8: Excerto de regras de implementação.

Foi a partir de regras de implementação deste género que o objeto de dados para a simulação do seguro multirriscos foi implementado, recorrendo ao C# com *Microsoft Visual Studio* 2010, que são as ferramentas mais usadas na camada de *Middleware*. A especificação completa relativa ao objeto de dados casa está disponível no Anexo III deste relatório.

Na Figura 9 é possível visualizar como é feita a implementação de um objeto de dados genérico. Tendo como objetivo o preenchimento de uma saída XML, o objeto de dados é implementado a partir da documentação técnica, com base tanto na estrutura XML como nas regras de implementação.

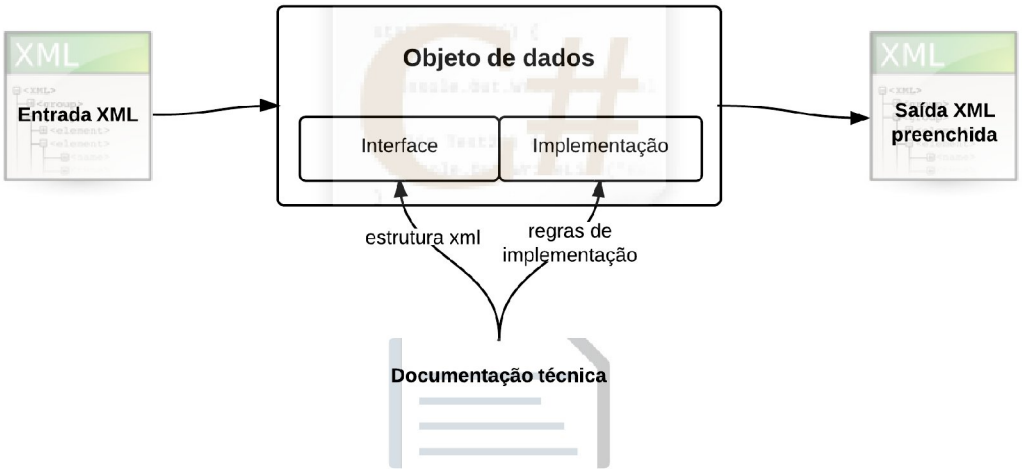


Figura 9: Implementação do objeto de dados.

De modo a testar a implementação, foi usada uma ferramenta de visualização de registos de eventos (*logs*) que tem a possibilidade de, a partir de determinada entrada em XML e usando o código gerado, nos mostra o XML de saída. Dessa forma foi possível ir testando se a implementação estava a preencher corretamente os dados de uma simulação, ou seja, se estava a ser corretamente preenchida a saída.

Depois da conclusão destes testes, foi-me apresentado o sítio na intranet que é usado internamente para fazer o controlo de versões, bem como o servidor onde são feitas as compilações de todos os projetos. De modo a disponibilizar as alterações efetuadas, o processo consiste em copiar as *Dynamic-link librarys* (DLLs) alteradas para a máquina de compilações, compilar lá o projeto e, se tudo estiver bem, registar as alterações no sítio da intranet para controlo de versões: que ficheiros foram alterados, as suas versões, o que foi feito e, no âmbito de que projeto.

Esta metodologia de disponibilização das alterações feitas é chamada de promoção dentro da seguradora, consistindo na disponibilização do código feito, para os vários ambientes, do produto desenvolvido. Cada um destes ambientes contém as suas restrições de integridade e serve para controlar melhor que versão de código está presente nessa fase de desenvolvimento. Ao fazer-se e testar-se a promoção de alterações feitas para um ambiente superior, está-se a garantir que o código nesse

ambiente funciona da maneira esperada, integrado com todos os outros projetos relacionados. A Figura 10 demonstra os vários ambientes existentes, bem como a ordem correta de promoção de código.

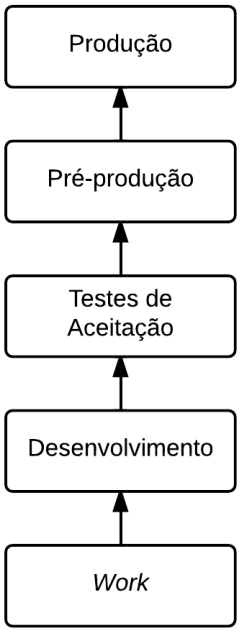


Figura 10: Promoções entre ambientes de desenvolvimento.

O ambiente de *work* corresponde ao trabalho feito nos computadores dos programadores, enquanto que o ambiente de produção corresponde ao produto final que é apresentado ao cliente.

No Anexo IV deste relatório encontra-se um exemplo de documento que foi gerado com base na implementação do objeto de dados para a simulação casa.

3.2 Migração dos serviços de negócio

Para fazer a migração da camada de gestão de interfaces para *software* de código aberto, é preciso, entre outros, disponibilizar os serviços de negócio existentes, depois de migrados para a nova plataforma, com algumas alterações estruturais, funcionais e de desenhos ou traçados dos documentos. Com a migração será possível ter mais informação e controlo da utilização dos respetivos serviços.

Os serviços de negócio são disponibilizados em vários canais e aplicações, ou seja, estão disponíveis tanto aos clientes como aos mediadores e colaboradores, nos canais internet, intranet e extranet, e vão desde requisições de impressos até à pesquisa

de lojas na página da internet, passando pelo pedido de contacto por parte dos mediadores.

A solução proposta para migrar estes serviços passa por recriar os ecrãs e o respetivo código nas novas ferramentas de desenvolvimento e pela sua integração no servidor aplicacional *JBoss*, que servirá de base a estas aplicações. As migrações tecnológicas que estão em curso, e que são relevantes no contexto das tarefas a realizar no estágio, estão representadas na Figura 11. O código C#, *Active Server Pages* (ASP) e ASP.NET, que era integrado no *Internet Information Services* (IIS), está a ser convertido em código *JavaServer Pages* (JSP) e *Java*, integrado no *JBoss*. Ao nível dos sistemas de gestão de conteúdos, que integram os portais *web* com o servidor aplicacional e respetivas aplicações, está-se a passar do uso de um sistema personalizado (desenvolvido anteriormente por outra empresa) para o portal *Liferay*.

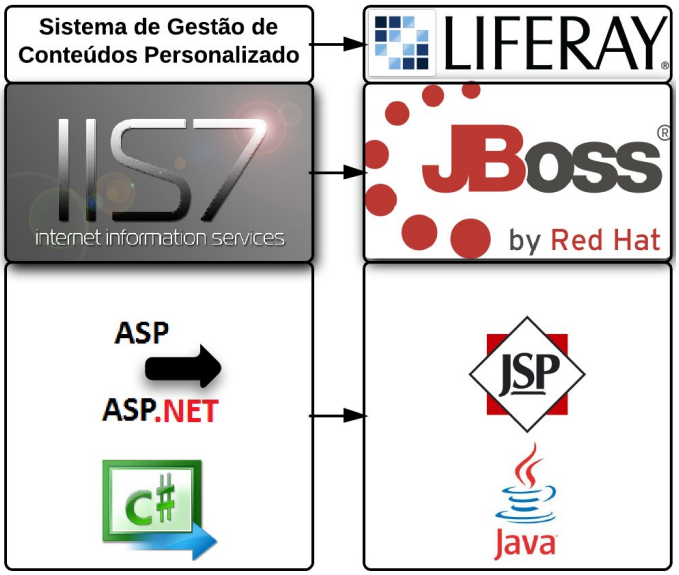


Figura 11: Migrações tecnológicas em curso.

3.2.1 Redesenho dos ecrãs

O primeiro passo para migrar os serviços de negócio consiste em redesenhar os ecrãs que os compõem. Para isso, é usada uma aplicação que está a ser desenvolvida na seguradora, como alternativa ao *Microsoft Visio*, o qual era usado anteriormente. O uso desta aplicação traz todas as vantagens de uma ferramenta de código aberto e mitiga algumas das desvantagens mais comuns, como a falta de suporte, já que a equipa que está a desenvolver esta ferramenta está a trabalhar em conjunto com a equipa responsável pelos serviços de negócio, para além de ter sido criado um projeto no

BugNET [1] que, ajuda não só a manter um histórico dos problemas reportados, como permite uma interação mais célere entre os membros das equipas.

Usando esta ferramenta, para desenhar uma página basta arrastar os componentes que ela deve conter para o ecrã principal, implementando de seguida os fluxos de informação e transições que existem na mesma. Os fluxos de informação consistem em determinar a navegação entre as páginas, bem como as ações que despoletam cada transição e as informações que são passadas de uma página para outra. Pode-se desenhar num fluxo, por exemplo, que ao clicar num botão de pesquisa, existe uma transição para outra página, normalmente de resultados, em que a informação passada é a que foi pesquisada e a informação recebida na página seguinte é a dos resultados obtidos.

Nas Figura 12 e 13 está ilustrada a ferramenta de desenho, tanto para a criação de páginas como para a criação de fluxos.

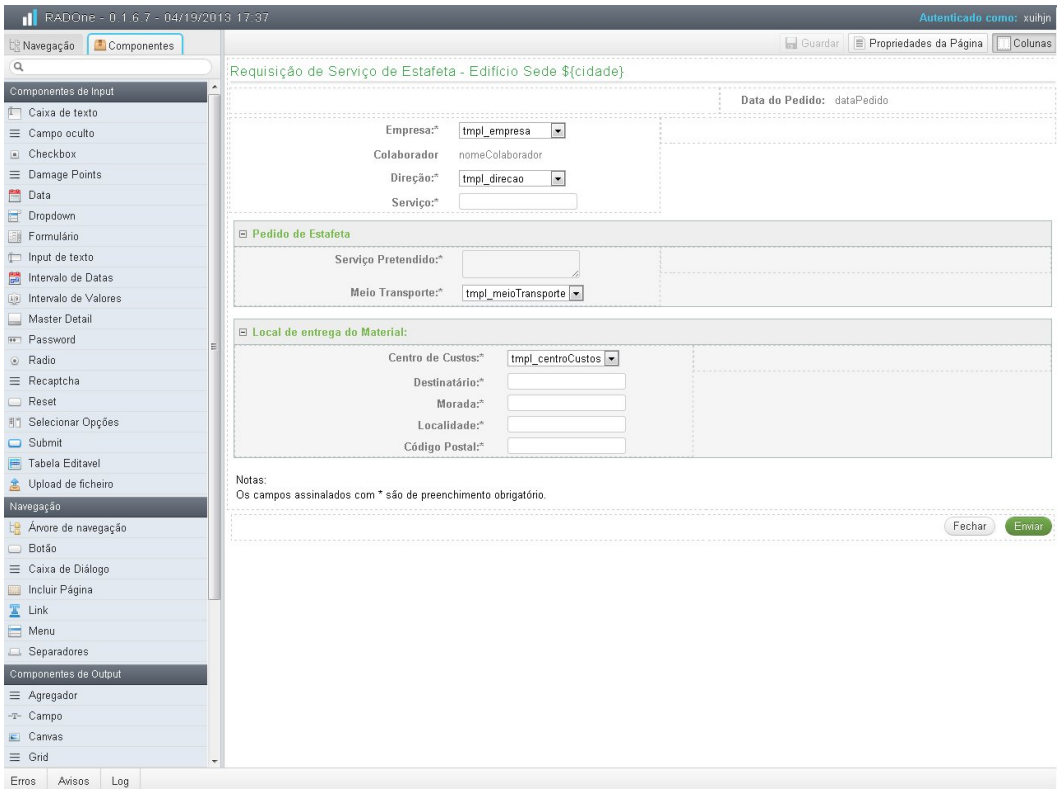


Figura 12: Exemplo de criação de uma página na ferramenta de desenho.

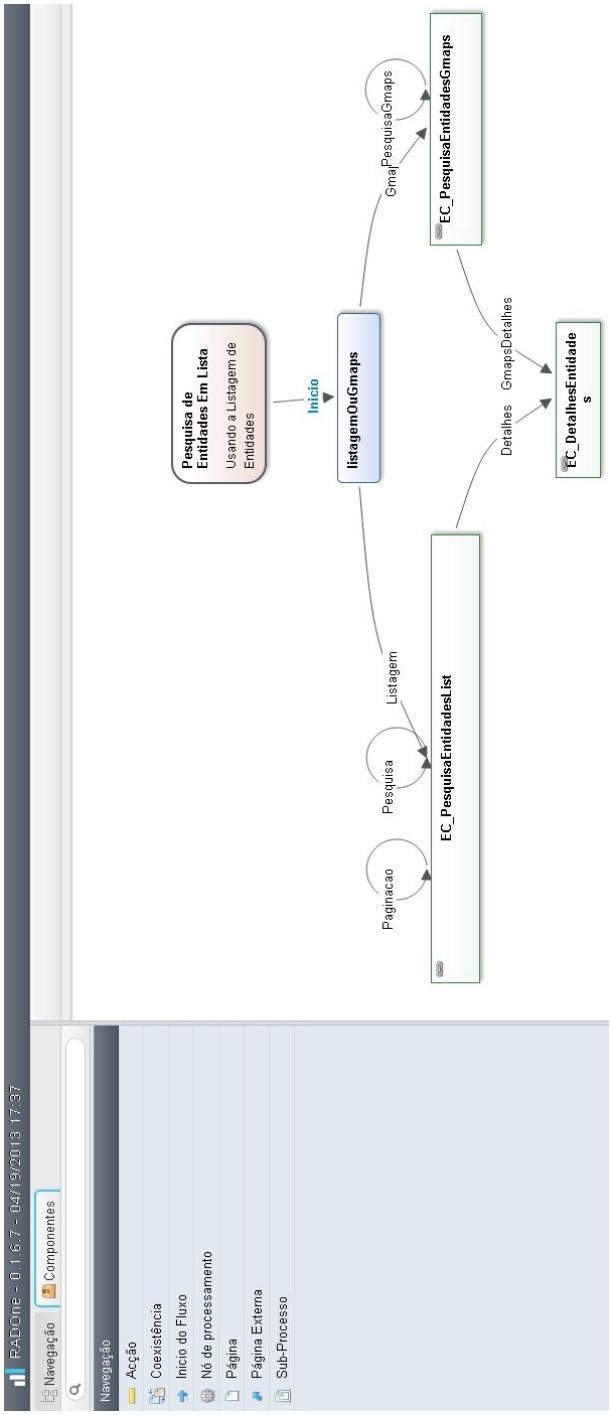


Figura 13: Exemplo de um fluxo e suas transições na ferramenta de desenho.

No redesenho dos ecrãs que fiz, tentei sempre manter o visual o mais aproximado possível dos ecrãs originais, excetuando as características que requerem alterações ou que não são ainda possíveis de implementar com a ferramenta de desenho. É importante salientar que a ferramenta de desenho ainda está em desenvolvimento e existem ainda alguns problemas na utilização da mesma, diminuindo a velocidade de implementação

mas, levando a uma melhor compreensão da ferramenta, reduzindo problemas similares no futuro.

Ao redesenhar um módulo que engloba as requisições de brindes, serviços e vários tipos de impressos, foi tomada a opção de reutilizar algumas partes da página, já que muitas outras páginas deste módulo partilham essas partes. Da mesma forma, o serviço de negócio das pesquisas de entidades também foi concretizado de uma forma altamente reutilizável. Tendo em conta que, o módulo da pesquisa de entidades contém as pesquisas de lojas, oficinas e clínicas (por enquanto), foram usadas as mesmas páginas de pesquisas e de detalhes para os vários tipos de entidade. Com esta opção de desenho e estruturação, poupou-se tempo de desenvolvimento, espaço de armazenamento e, também, se tornou possível que uma futura mudança em alguma destas páginas seja rapidamente propagada para todas as outras que reutilizem as partes comuns do ecrã. No entanto, esta última vantagem também tem os seus pontos negativos, no sentido de ter menos flexibilidade para a personalização de uma das páginas, assim como a manutenção ter que ser feita com mais cuidado, devido à propagação de alterações para as várias páginas.

3.2.2 Exportação para *Java*

Depois de estarem criados os ecrãs na ferramenta de desenho, o passo seguinte corresponde à exportação desse projeto para *Java*, neste caso o projeto dos serviços de negócio. Esta exportação é feita automaticamente pela ferramenta e cria um projeto *Java* que, neste caso, é depois manipulado no *Integrated development environment* (IDE) *Eclipse*.

Este projeto é constituído por duas pastas na raiz, uma aplicacional com os *Data Access Objects* (DAOs) e outra com as vistas das páginas. Um DAO é uma camada (representada por um objeto) que disponibiliza uma interface abstrata de acesso a uma base de dados, ou a outro mecanismo de persistência, expondo métodos públicos que são depois invocados por outra camada aplicacional [2]. Esta forma de organização assenta no padrão MVC, detalhado no capítulo de trabalho relacionado.

Depois de ter o projeto exportado para *Java* e de se terem corrigido algumas configurações de sistema, as páginas que foram criadas estão quase passíveis de ser visualizadas na *web* local, bastando para isso compilar o projeto completo, o que leva a que este seja empacotado em dois ficheiros, um para a parte aplicacional e outro para a

parte das vistas, e que seja feito automaticamente o carregamento deste projeto no servidor aplicacional *JBoss*. Esta ação de carregamento num servidor aplicacional consiste em publicar os pacotes criados dentro do contexto em que se está a correr o servidor, o que neste caso se traduz numa cópia de ficheiros para as pastas locais do *JBoss*. A partir desse momento é possível visualizar as páginas que foram definidas na *web* local, acedendo ao portal *Liferay* que está instalado localmente em cada máquina e que serve para gerir conteúdos, criando depois uma página no ambiente de testes e adicionando lá a aplicação correspondente aos serviços de negócio. No entanto, cada um destes serviços de negócio não está ainda completo nesse momento, faltando implementar as regras de negócio, ou seja, o que cada página faz quando se invoca alguma ação na página, como clicar num botão, por exemplo.

3.2.3 Implementação das regras de negócio

De uma maneira geral, as páginas correspondentes aos serviços de negócio são compostas por um formulário e por um botão de "enviar". Na maior parte dos casos, é necessário "enviar" para a seguradora os dados do formulário (fazendo o envio de um *email*) e guardar na base de dados um registo da invocação desse serviço.

Hibernate

Para guardar a invocação de um serviço na base de dados recorreu-se à ferramenta *Hibernate* [4], assim como à sua funcionalidade de fazer engenharia reversa. O *Hibernate* é uma ferramenta que faz o mapeamento entre uma base de dados relacional e/ou tipos de dados *Structured Query Language* (SQL) com objetos *Java*. O objetivo do *Hibernate* é aliviar o programador de grande parte do trabalho inicial de persistência de dados, eliminando a necessidade de configuração manual usando SQL e *Java Database Connectivity* (JDBC). Assim, usando o *Hibernate*, a grande vantagem que se obtém é a de poder ser alterado em qualquer altura o sistema de base de dados, não sendo preciso alterar o código implementado. Esta vantagem é portanto vital para a migração que se está a fazer, já que se poderá continuar a usar a base de dados antiga para alguns serviços (assente em *SQL Server*), ao mesmo tempo que se vai migrando a informação para a nova base de dados (assente em *Oracle Database*). Esta migração está representada na Figura 14.

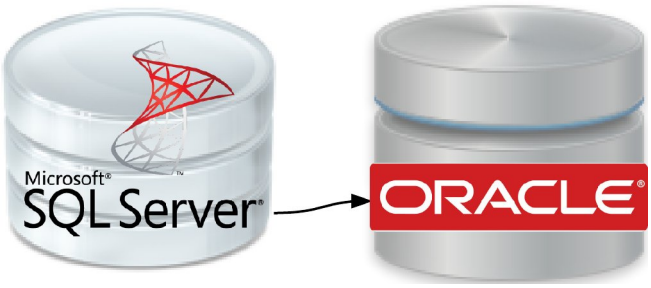


Figura 14: Migração de bases de dados.

A engenharia reversa, por seu lado, consiste em criar classes *Java* a partir de uma base de dados relacional, de forma automática, podendo ainda assim definir configurações personalizadas. Usando a engenharia reversa, foi criado um conjunto de classes que foram mapeadas diretamente de tabelas da base de dados. Entre estas tabelas está a que guarda o registo de uma invocação a cada serviço de negócio, assim como outras de suporte. Desta forma, quando se estão a implementar as regras de negócio, basta apenas instanciar um objeto do tipo correspondente a essa tabela e persistir o objeto, já que o *Hibernate* comunica com a base de dados e insere nessa tabela uma linha correspondente a esse objeto *Java*. Este objeto é preenchido nessa altura (cada propriedade corresponde a uma coluna na tabela associada) com as informações do pedido que foi feito. Para esse efeito foi criado um ficheiro *Java* correspondente a um DAO comum, que contém o método de persistência e que pode ser usado por qualquer um dos serviços de negócio.

Para dar suporte aos serviços de negócio desenvolvidos durante o estágio, teve que se replicar, não só a tabela onde são guardadas as informações dos pedidos, como outras tabelas de suporte que são necessárias para o funcionamento correto dos serviços.

A título de exemplo, vários serviços que foram migrados correspondem a requisições de material diverso ou de impressos. Os materiais que podem ser requisitados são mantidos em base de dados e estavam, anteriormente, em *SQL Server*, sendo mostrada ao utilizador (nas páginas *web*) uma tabela com os materiais passíveis de serem requisitados para cada serviço, em particular. A Figura 15 demonstra uma destas tabelas.

Designação dos Impressos	N.º Modelo	Versão	Quantidade
KIT Abertura Conta		Set.2011	
Folheto Vantagem		fev.2013	
Folheto Conta		mar.2012	
Folheto Vantagem Negócios AT		fev.2013	

Figura 15: Exemplo de tabela de requisição de impressos.

Para concretizar os serviços que usam a tabela de materiais, foi necessário migrar as tabelas que contêm a relação entre os serviços de negócio e os materiais de cada serviço, bem como programar o acesso à base de dados de *SQL Server*, primeiro para testes, e de *Oracle* depois, contando com isso com o acesso facilitado do *Hibernate* para ajudar na transição para o novo sistema de gestão de base de dados (SGBD).

Para além da migração das tabelas, foram também recriados procedimentos armazenados no novo SGBD. Procedimentos armazenados são sub-rotinas compostas por conjuntos de instruções SQL que são normalmente usados para fazer trabalho repetitivo e extenso de uma forma automática, acedendo para isso à base de dados [16]. Um exemplo de procedimento armazenado migrado foi o de obtenção do número de requisição atual, número esse que é mantido na base de dados e varia consoante o tipo de serviço. Esse valor serve para se manter um registo do número de requisições que são feitas pelos utilizadores e é também adicionado ao assunto do *email* que é enviado.

Envio de *email*

Para enviar o *email* com as informações do serviço de negócio, foi necessário analisar o código existente e perceber que texto é enviado, para quem, por quem, e se tem algumas condições especiais. Depois de ter estes dados, foi experimentada uma solução de envio de *email* usando o *JavaMail* [5], tendo sido implementada com sucesso. No entanto, foi-me explicada a existência de uma base de dados com várias tabelas que decompõem as informações do envio de um *email* (ou de carta, por exemplo).

De uma maneira simplificada, a cada serviço de negócio estará associada uma regra e, cada regra pode ser dividida em várias partes (de forma a facilitar a compreensão e a reutilização), sendo que cada parte terá um bloco de texto; texto esse que pode corresponder ao campo “Assunto” ou ao conteúdo do *email*, por exemplo. Estas regras são parametrizadas na base de dados para depois, no código de cada serviço de negócio, poder ser chamado um serviço *web* que comunica com a base de dados,

obtém a regra correspondente e respetivas informações que necessita para enviar o *email*, e procede ao seu envio.

Para alguns serviços de negócio foi então analisado o código existente (que ainda não foi migrado e que pode conter restrições de negócio) - de modo a perceber o que é necessário enviar por *email* - e foram parametrizadas as regras correspondentes a esses serviços na base de dados. Para criar essas regras foi escrito um conjunto de instruções em código SQL (por serviço de negócio) que insere algumas linhas na base de dados, com as informações necessárias. Após estarem parametrizadas na base de dados estas regras, resta apenas invocar a chamada ao serviço *web* com os argumentos necessários. Esse serviço *web* aceita que seja personalizado o corpo do *email* ou o assunto do mesmo, por exemplo, assim como permite que sejam definidas variáveis no corpo da mensagem, de forma a que possa ser genérico e que possa conter os dados de cada pedido (como o nome ou a morada de quem fez o pedido, por exemplo).

Independentemente de guardar o registo na base de dados e/ou de enviar um *email*, é sempre necessário analisar a solução existente, de modo a implementar todas as regras de negócio associadas a cada serviço. Estas regras baseiam-se, normalmente, em restrições, ações ou validações a fazer, dependendo dos dados que são preenchidos na página do serviço. Na Figura 16 são exemplificadas as regras de negócio associadas à generalidade dos serviços, assim como os fluxos de informação que são passados entre os vários sistemas com que interagem os serviços de negócio.

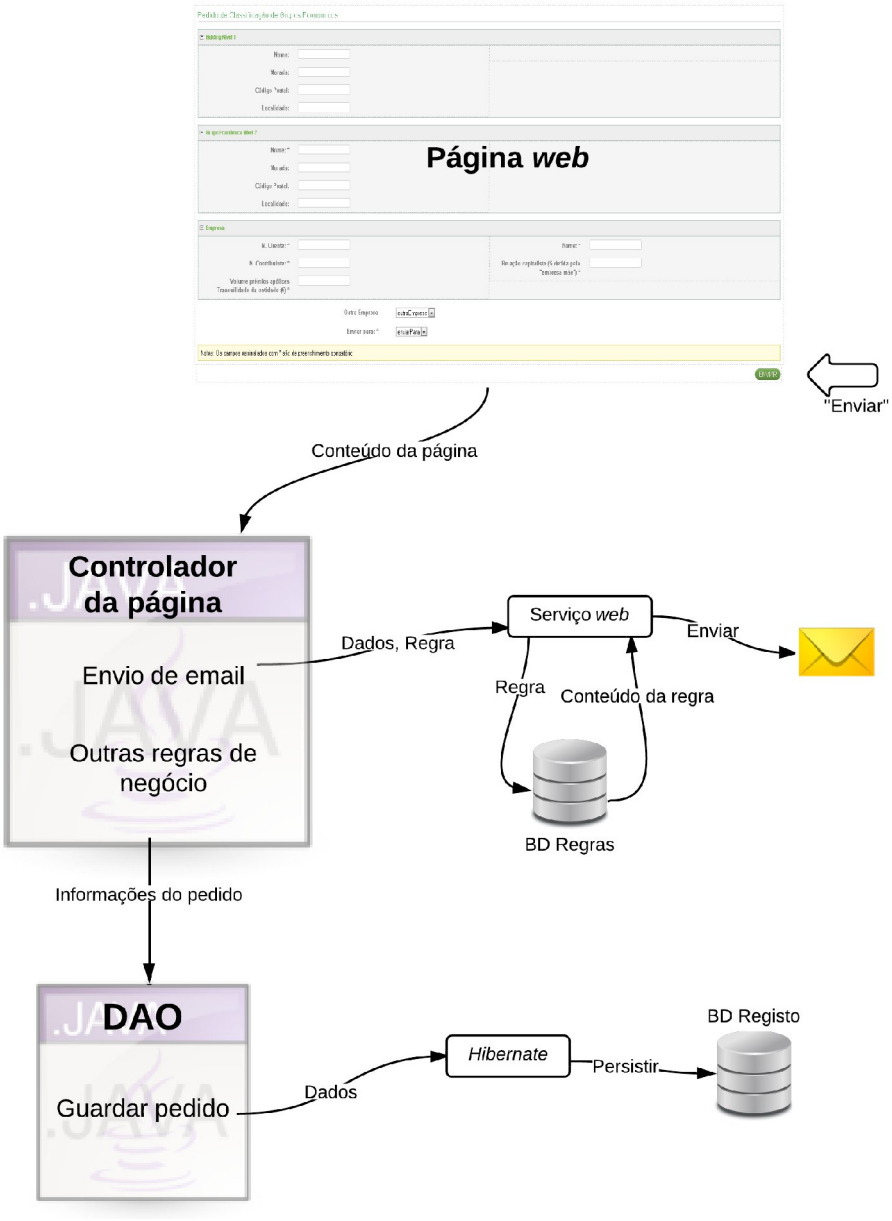


Figura 16: Regras de negócio associadas à generalidade dos serviços.

3.2.4 Exemplo - Pesquisa de Entidades

Um dos serviços de negócio migrados e implementados durante o estágio foi o da pesquisa de entidades. Este serviço consiste em disponibilizar aos utilizadores uma pesquisa pelas várias entidades associadas à seguradora, nomeadamente lojas, oficinas e clínicas. Quanto ao modo de apresentação de resultados, é possível visualizar as entidades ordenadas numa lista ou num mapa baseado em *Google Maps* mas, independentemente do modo de apresentação, a pesquisa funciona de forma comum: ao

utilizador são apresentados vários critérios de pesquisa (como o nome de entidade ou o distrito, por exemplo) que depois é feita com base nesses critérios.

Todas as entidades passíveis de serem pesquisadas foram previamente carregadas numa base de dados que o *Liferay* disponibiliza, definindo para cada uma delas as suas informações, incluindo coordenadas *Global Positioning System* (GPS) ou uma imagem.

Ao iniciar uma pesquisa, o controlador do ecrã obtém do mesmo os critérios que foram pesquisados, definindo logo à partida o tipo de entidade a filtrar, consoante a página de onde foi feita a pesquisa. Mesmo existindo três páginas diferentes, uma para cada tipo de entidade, o ecrã que é apresentado é sempre o mesmo, variando de forma dinâmica o que distingue umas entidades das outras.

Depois de obtidos os critérios de pesquisa, é feita uma invocação à base de dados do *Liferay*, que devolve as entidades correspondentes. A forma como estes resultados são depois apresentados, bem como a interação que permitem, é descrita de seguida.

Listagem

A listagem consiste numa lista ordenada que apresenta algumas informações sobre cada um dos resultados. Esta lista é paginada e pode ser reordenada pelo utilizador, tendo para isso sido implementada uma ordenação com base nos dois campos possíveis de ordenação (nome e local) e na ordem (ascendente ou descendente). Embora as interrogações à base de dados do *Liferay* permitam ordenação, à data de migração deste serviço a ordenação não suportava caracteres especiais (como acentos ou cedilhas), o que impossibilitava uma ordenação perfeita. Esta limitação levou a que tenha sido implementada uma ordenação manual, que não é a opção mais eficiente, já que todas as entidades são obtidas sempre, independentemente do número de página de pesquisa, sendo depois ordenadas e filtradas para se mostrarem apenas as da página corrente. No entanto, esta opção revelou-se necessária, devido aos custos temporais potencialmente elevados que seriam necessários para melhorar a ordenação automática. A juntar a estas desvantagens, atualmente existem cerca de 700 entidades pesquisáveis, que mantém o tempo de ordenação relativamente residual. Independentemente disso, devido a esta não ser a opção mais escalável, mal seja disponibilizada passar-se-á a usar a ordenação automática do *Liferay*.

Na Figura 17 é apresentado um exemplo do ecrã de resultados de uma pesquisa em modo listagem.

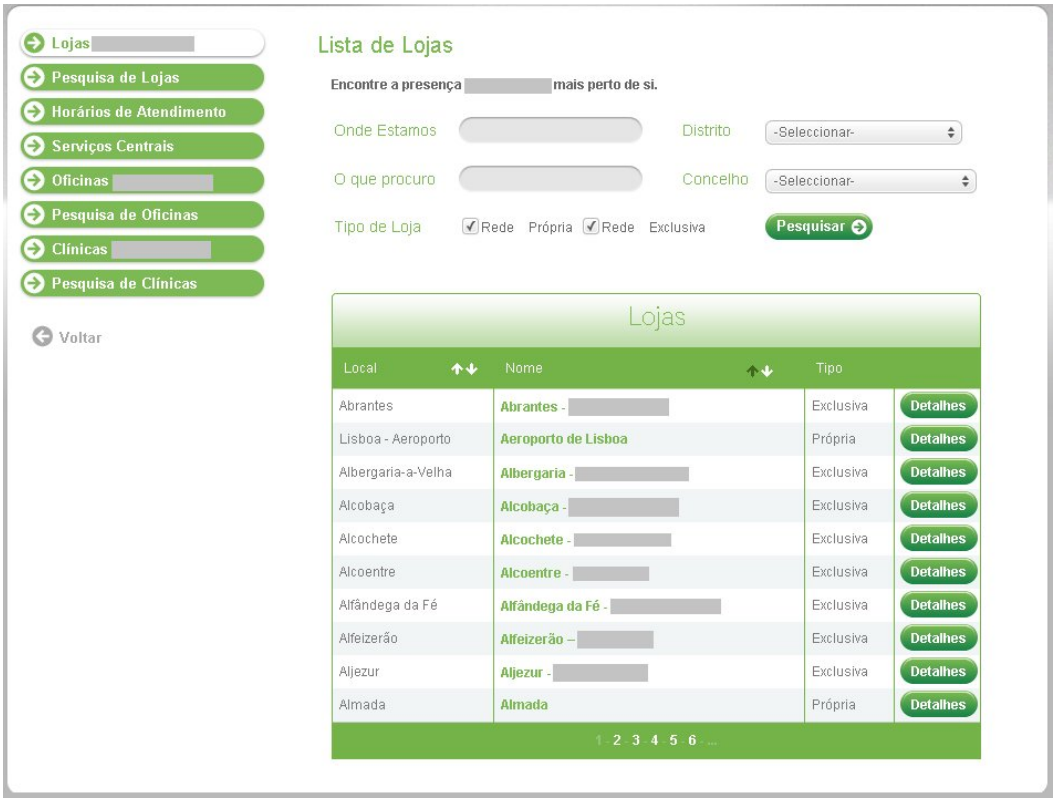


Figura 17: Exemplo de Pesquisa de Entidades - Listagem.

Mapa

O mapa consiste numa apresentação de resultados baseada em *Google Maps*, sendo apresentado um mapa com um marcador para cada entidade. A implementação que foi feita recorre à *Application Programming Interface* (API) do *Google Maps*, que oferece a possibilidade de incluir em qualquer página um mapa disponibilizado pela *Google*, permitindo também a personalização do aspeto do mapa, incluindo, por exemplo, o ícone do marcador ou o nível de *zoom* pré-definido [3].

Ao carregar a página que contém o mapa, os resultados da pesquisa efetuada são adicionados a um objeto *JavaScript Object Notation* (JSON), que é depois recebido e tratado pelo código *JavaScript* que foi definido na página e que é executado assim que esta é carregada. O objeto JSON recebido consiste numa lista de objetos, cada um correspondendo a uma entidade e tendo várias informações sobre as entidades retornadas pela pesquisa. A localização de cada entidade é adicionada ao mapa, com base nas coordenadas GPS, adicionando-se um marcador para cada entidade. O ícone que é apresentado nos vários marcadores foi personalizado e varia consoante o tipo de

entidade e, ao clicar num marcador é disponibilizada uma janela com algumas informações sobre essa entidade.

Na Figura 18 é apresentado um exemplo do ecrã de resultados de uma pesquisa em modo mapa.

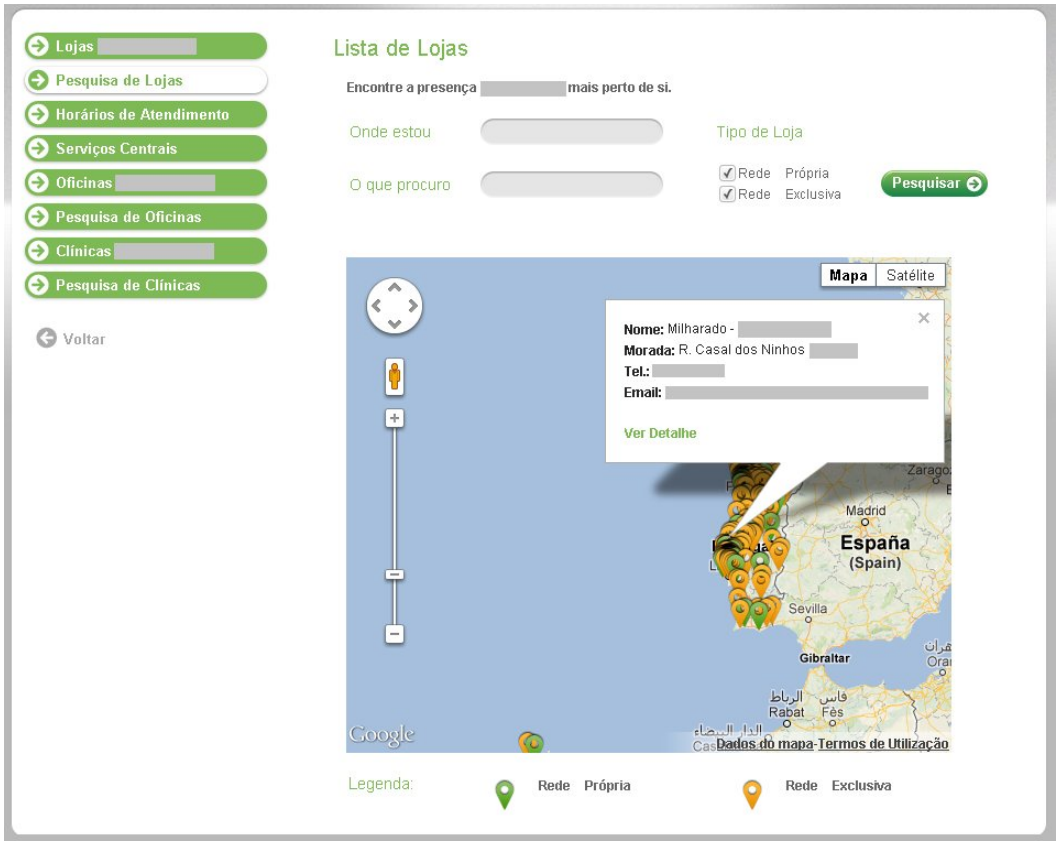


Figura 18: Exemplo de Pesquisa de Entidades - Mapa.

Detalhes

Independentemente do modo de pesquisa de entidades que é usado, é disponibilizada uma página que contém todos os detalhes de cada entidade. O utilizador pode aceder a esta página pela listagem ou pelo mapa, tendo sido criadas para isso hiperligações nessas páginas que dão a possibilidade ao utilizador de ver mais detalhes sobre uma entidade em que esteja interessado.

A página de detalhes apresenta grande parte das informações disponíveis na base de dados *Liferay* sobre uma entidade e contém, para além de detalhes textuais como nome ou *email*, uma imagem e um mapa *Google* apenas com a localização da entidade em questão, de modo a dar uma informação textual completa, suportada por informação visual. Tal como nas outras páginas, a página de detalhes é reutilizada e foi criada

apenas uma para todas as entidades, recebendo as informações de cada entidade pelas páginas de pesquisa.

A Figura 19 apresenta uma página de detalhes com informação sobre uma das entidades.

Figura 19: Exemplo de Pesquisa de Entidades - Detalhes.

3.2.5 Exemplo - Registo de Cliente

Estando englobado nos serviços de negócio, o registo de cliente foi um dos módulos migrados durante o estágio. Este módulo consiste na disponibilização aos clientes da seguradora de um registo de clientes, de modo a que estes possam aceder, depois de registados, a um portal interno da seguradora que apresenta várias funcionalidades relacionadas com a gestão de conta de cada cliente.

O desenvolvimento do módulo de registo de cliente passou pela implementação de três fluxos de informação principais: registo, ativação e entrada no sistema.

Registo

O fluxo de informação correspondente ao registo contém os ecrãs que permitem aos clientes da seguradora registarem-se como utilizadores autenticados do portal *web*. Por cliente, entenda-se qualquer pessoa que tenha um seguro feito na seguradora, tendo consigo o respetivo número de apólice, bem como o número de cliente associado.

Ao iniciar este fluxo de informação, é apresentado ao utilizador o ecrã que é demonstrado na Figura 20.

Figura 20: Exemplo de Registo de Cliente - Formulário de registo.

Após a inserção pelo utilizador dos seus dados pessoais são feitas algumas validações do lado do cliente, como o número de identificação fiscal (NIF) e a data (recorrendo a *Java*Script), ou o componente de *ReCaptcha* (recorrendo a um serviço externo). O componente de *ReCaptcha* consiste num desafio visual ou auditivo de segurança, que foi desenvolvido como variante e para servir de alternativa ao *Captcha*, um componente similar mas implementado de forma diferente. À semelhança do componente em que se baseia, o *ReCaptcha* tem como principal objetivo verificar que, os utilizadores de determinada página na internet são humanos. Esta verificação tornou-se necessária devido à massificação pela internet de conjuntos de instruções em código maliciosos que tentam inundar de pedidos uma determinada página, levando a que os sítios na internet fiquem instáveis ou demorem mais tempo que o habitual, a responder aos pedidos feitos pelos utilizadores [10].

Depois de serem validados os dados necessários, o pedido é enviado para o servidor com os dados do formulário. É nessa altura que é feita uma validação do número de cliente e do número de apólice, recorrendo a um serviço *web* disponibilizado para o efeito. Se essa validação não tiver sucesso, é enviado para o ecrã essa informação; caso contrário, o serviço devolve a morada do utilizador, que está registada

na base de dados da seguradora e está associada ao par número de cliente/número de apólice.

Tendo em conta que a ativação do registo será feita posteriormente recorrendo a um código enviado para a morada do utilizador, é mostrado, no ecrã seguinte do fluxo, a morada que foi devolvida pelo serviço *web*, de modo a que o utilizador possa confirmar a localização. Esse ecrã está demonstrado na Figura 21.

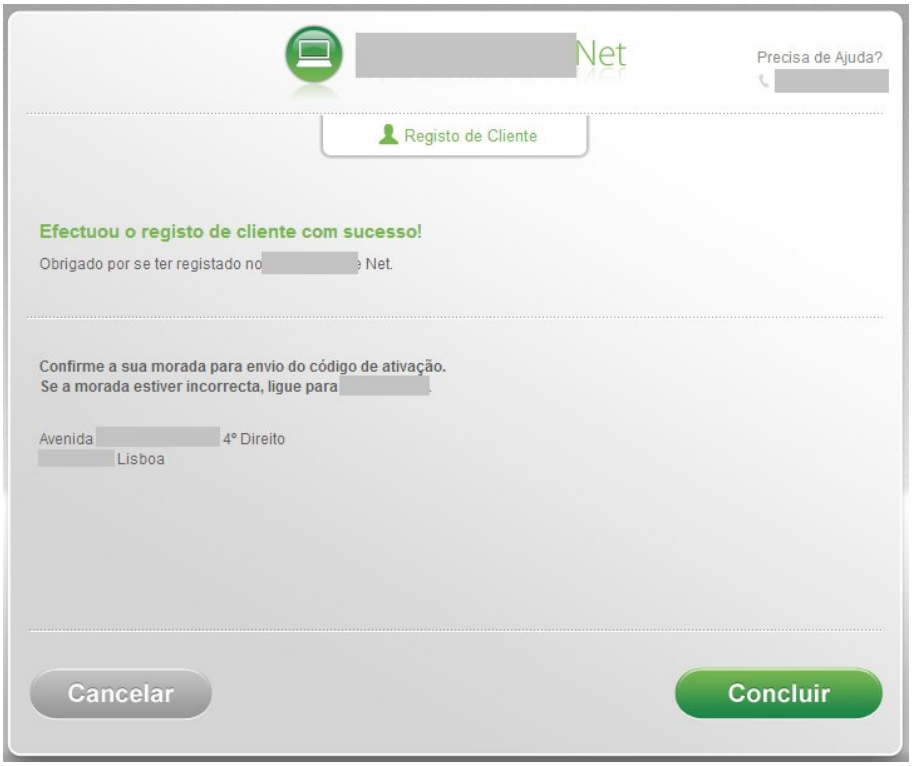


Figura 21: Exemplo de Registo de Cliente - Confirmar morada.

Ao confirmar a morada é despoletado um serviço *web* que regista o utilizador na base de dados do sistema e põe em marcha os meios necessários para ser enviada uma carta para a morada apresentada no ecrã, carta essa que contém um código de ativação.

Ativação

O fluxo de ativação do registo de cliente inicia-se após o utilizador receber na sua morada uma carta com um código de ativação. Ao entrar no sítio da internet da seguradora é possível, analogamente ao que fez para se registar, iniciar o fluxo de ativação.

No ecrã inicial, apresentado na Figura 22, o utilizador insere o endereço de correio eletrónico com o qual pretende que seja associada a sua conta, bem como a palavra-chave pretendida e o código que recebeu por carta.

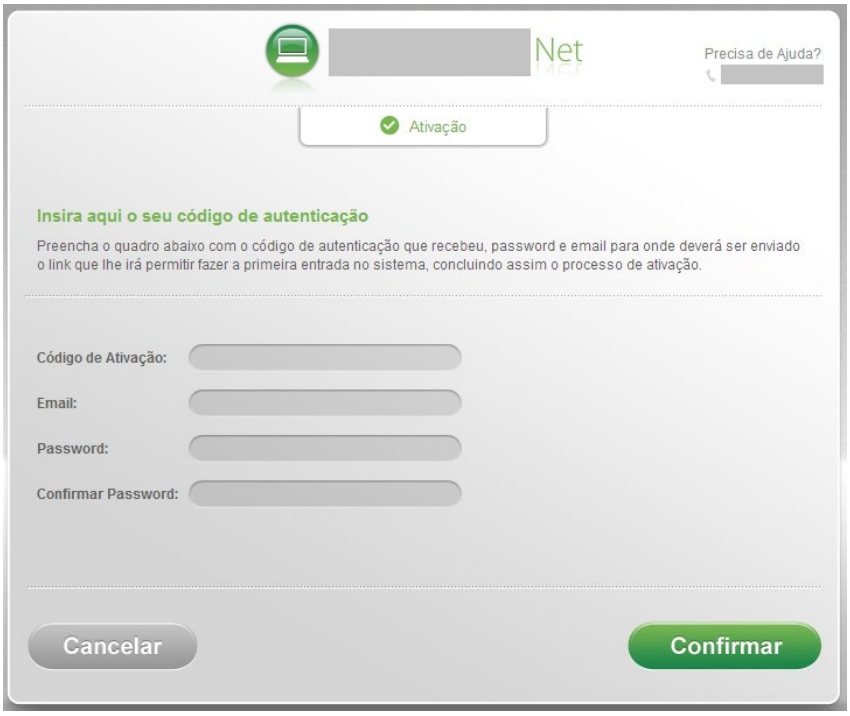


Figura 22: Exemplo de Registo de Cliente - Formulário de ativação.

Ao continuar o fluxo de ativação, é invocado outro serviço *web* que procede à pré-ativação do registo de cliente, associando o endereço de correio eletrónico e palavra-chave pretendidos ao resto das informações do cliente. Caso este último passo seja executado com sucesso, é enviado um *email* para o utilizador, contendo uma hiperligação que permite ao utilizador ativar a sua conta. Ao clicar nessa hiperligação, é aberta uma janela no navegador e é invocado outro serviço *web* que confirma que todos os dados estão coerentes, fazendo efetivamente a ativação do registo. Se tudo correr bem, é apresentada uma mensagem de sucesso ao utilizador, indicando que este pode a partir desse momento fazer a entrada no sistema.

Entrada no sistema

Para entrar no sistema, o cliente (depois de registado e ativado) apenas precisa de aceder ao sítio na internet da seguradora e, usando o endereço de correio eletrónico e a palavra-chave com que se registou, preenchê-los no ecrã de autenticação.

De modo a manter a privacidade do utilizador, é usada uma cifra no campo da palavra-chave, sendo enviado para o servidor a palavra-chave já cifrada. A cifra é feita com base no algoritmo *Rivest Shamir Adleman* (RSA) e concretizada com recurso a *JavaScript*.

Na Figura 23 é possível visualizar o ecrã de entrada que foi criado e implementado.

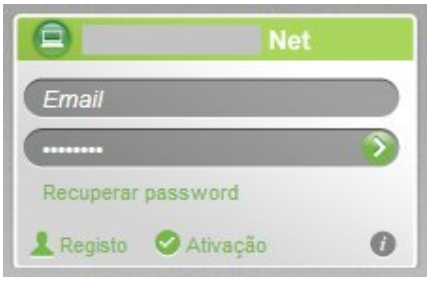


Figura 23: Exemplo de Registo de Cliente - Formulário de entrada no sistema.

Após as credenciais de acesso serem verificadas, o utilizador é automaticamente reencaminhado para o portal de clientes, o qual contém vários detalhes relacionados com a sua conta, como os dados pessoais ou a informação de apólices subscritas.

Caso o cliente não se recorde da sua palavra-chave, tem a possibilidade de receber uma nova no endereço de correio eletrónico com que se registou, usando a funcionalidade recuperar *password*, disponível para acesso a partir do ecrã de entrada. Esta funcionalidade foi concretizada recorrendo novamente a um serviço *web* que comunica com a base de dados da seguradora, de modo a criar, associar uma nova palavra-chave ao cliente e enviá-la para o endereço de correio eletrónico registado.

3.2.6 Síntese dos serviços de negócio migrados

Devido a terem sido migrados vários serviços de negócio de alguns tipos diferentes, durante o estágio, é apresentado na Tabela 3 um resumo da descrição de cada um deles, bem como o seu estado de migração, podendo ser vista de uma maneira geral a síntese dos serviços de negócio que foram migrados. De modo a proteger a confidencialidade da seguradora foram alterados os nomes de alguns serviços (por exemplo Requisição de impressos “4”).

Nome do serviço	Resumo	Estado da migração
Pedido de classificação de grupos económicos	Formulário de pedido com registo em base de dados e envio de *email*.	Implementação a 95%.
Pesquisa de entidades	Pesquisa de mediadores, oficinas e clínicas convencionadas. Visualização em lista ou em mapa e detalhes de cada entidade.	Migrado.
Centro de contacto *online*	Formulário de pedido com registo em base de dados e envio de *email*.	Ecrã redesenhado, falta implementação.
Proposta e avaliação da rede de prestadores	Vários formulários de avaliação de serviços prestados, com registo em base de dados e envio de *email*.	Implementação a 50%.
Serviço de recrutamento de agentes	Formulário de recrutamento de novos mediadores, com registo em base de dados e envio de *email*.	Migrado.
Requisição de brindes	Formulário de requisição com registo em base de dados e envio de *email*.	Implementação a 80%.
Requisição de serviço de estafeta Lisboa	Formulário de requisição com registo em base de dados e envio de *email*.	Implementação a 95%.
Requisição de serviço de estafeta Porto	Formulário de requisição com registo em base de dados e envio de *email*.	Implementação a 95%.
Requisição de viatura de aluguer para colaboradores	Formulário de requisição com registo em base de dados e envio de *email*.	Implementação a 95%.
Requisição de material 1	Formulário de requisição com registo em base de dados e envio de *email*.	Implementação a 80%.

Nome do serviço	Resumo	Estado da migração
Requisição de impressos 1	Formulário de requisição com registo em base de dados e envio de *email*.	Implementação a 80%.
Requisição de impressos 2	Formulário de requisição com registo em base de dados e envio de *email*.	Implementação a 80%.
Requisição de impressos 3	Formulário de requisição com registo em base de dados e envio de *email*.	Implementação a 80%.
Requisição de material de apoio à venda	Formulário de requisição com registo em base de dados e envio de *email*.	Implementação a 80%.
Requisição de impressos de seguros de saúde	Formulário de requisição com registo em base de dados e envio de *email*.	Implementação a 70%.
Requisição de impressos 4	Formulário de requisição com registo em base de dados e envio de *email*.	Implementação a 80%.
Registo de cliente	Registo e ativação de conta no portal de clientes. Entrada no sistema e recuperação de palavra-chave.	Migrado.
Alocação de negócio	Pedido de contacto após simulação de seguro, com pesquisa de mediadores relacionados e envio de *email*.	Migrado.
Simulador multirriscos	Simulação de seguro casa, estabelecimento e condomínio.	Implementação a 95%.

Tabela 3: Síntese dos serviços de negócio migrados.

3.3 *Jenkins*

Tendo em conta a migração que se está a fazer e os pressupostos descritos em 1.2.2 e 1.3, algumas tarefas do estágio recaíram na necessidade de encontrar, instalar e configurar uma ferramenta de compilações automáticas que desse suporte ao *software* que passou a ser produzido em tecnologias de código aberto.

Seguindo as sugestões da equipa de trabalho e, como era preciso encontrar inicialmente a ferramenta a usar, as opções consideradas foram o *Hudson* e o *Jenkins*. Estas sugestões foram feitas com base em experiências anteriores de utilização, seguindo-se depois uma breve investigação dos prós e contras das duas opções. Na realidade, ambas as ferramentas se revelaram bastante válidas, sendo até muito parecidas, devido principalmente a terem sido criadas pelas mesmas pessoas, que trabalhavam na empresa *Sun*. O *Hudson* foi o primeiro a ser criado, por isso ganhou bastante popularidade e começou a ser bastante utilizado. No entanto, passado algum tempo, questões corporativas e legais - compra da *Sun* por parte da *Oracle* - levaram a que os principais responsáveis pelo projeto do *Hudson* o deixassem para trás, à inteira responsabilidade da *Oracle*, criando assim uma alternativa chamada *Jenkins*.

Depois de fazer alguma investigação [11][17], concluiu-se que as principais diferenças entre o *Hudson* e o *Jenkins* são as que estão representadas na Tabela 4.

	Hudson	***Jenkins***
Equipa de desenvolvimento	Trabalhadores da *Oracle*	Criadores originais
Desenvolvimento de *plugins*	Muito poucos programadores	Grande número de programadores
Comunidade	Pequena e pouco ativa	Grande e bastante ativa
Atualizações	Poucos lançamentos	Vários lançamentos
Atualizações estáveis	Quase todas	Versões trimestrais
Documentação	Elevada e profissional	Razoável
Estabilidade	N.º de *bugs* em aberto razoável	N.º de *bugs* em aberto elevado
Auditoria	Auditado na *Oracle*	Não auditado
Quantidade de *plugins*	Tendência a estagnar	Tendência a aumentar

Tabela 4: Principais diferenças entre *Hudson* e *Jenkins*.

Embora tenha sido uma decisão difícil de tomar, se se juntar a estas diferenças a forte aposta que o *Hudson* faz na integração com *Maven* (que não é usado na seguradora), foi escolhido o *Jenkins*, principalmente pela quantidade de *plugins* disponível, que dá uma profundidade à ferramenta em termos de funcionalidades que, de outra forma não teria. Para além disso, a documentação de menor qualidade acaba por não ser relevante, já que é compensada pela maior comunidade de utilizadores, que muitas vezes oferece uma dinâmica maior do que documentação estática escrita num certo ponto de desenvolvimento.

3.3.1 Configuração

Após ter sido feita a escolha pelo *Jenkins*, foi necessário proceder à sua instalação e configuração inicial. O *Jenkins* integra-se com o servidor aplicacional *JBoss* da mesma forma que as outras aplicações, bastando depois aceder à página do mesmo para ver todas as informações e configurações. Na Figura 24 está um exemplo da página principal que é visualizada, quando se acede ao *Jenkins*.

Figura 24: Página inicial do *Jenkins* com permissões limitadas.

Inicialmente começou-se por configurar a aplicação, nomeadamente as opções globais e a integração com o *Java*, assim como a instalação de *plugins* que se revelaram necessários devido à sua utilidade ou à ajuda de integração com os processos de disponibilização de *software* que existem na seguradora.

Segurança

Para além das configurações iniciais, foi necessário ativar um mecanismo de segurança que o *Jenkins* oferece e que assenta em grupos de utilizadores e papéis. Esta configuração revelou-se essencial, não tanto pela segurança externa mas, principalmente, pela necessidade de haver um registo de compilações por parte dos utilizadores, bem como haver libertação de trabalho por parte das poucas pessoas que podiam compilar os projetos, inicialmente. Assim, cada programador pode executar as suas compilações e subidas de *software*, ficando tudo guardado nos registos do *Jenkins*; quem executou o quê, a que horas, se falhou, o porquê da falha, entre outras informações.

Para implementar a segurança, foram criados manualmente os vários utilizadores necessários, um para cada pessoa que trabalha no gabinete e que precisa de ter ou poderá precisar de ter acesso, no futuro, ao *Jenkins*. Foi escolhida a criação de utilizadores de forma manual mas, existem outras alternativas, como por exemplo, a integração com *Lightweight Directory Access Protocol* (LDAP) e com *Active Directory* (AD). No entanto, nenhuma delas se revelou atrativa, já que o número de pessoas que poderá ter acesso ao *Jenkins* é relativamente pequeno e a sua manutenção está bastante facilitada, contrapondo com uma integração mais complexa com as tecnologias citadas acima.

Depois da criação dos utilizadores foram definidos vários papéis, ou seja, as permissões associadas, como por exemplo: ver só algumas tarefas, poder executar só algumas delas, poder configurar algo, cancelar uma subida, entre outros. Finalmente, depois de associar cada utilizador ao seu papel, o que garante automaticamente que terá apenas as permissões definidas para esse papel, foi necessário definir padrões nos nomes das tarefas. Esta padronização dos nomes das tarefas levou a que a manutenção da segurança esteja bastante mais simplificada, ou seja, se for criada uma nova tarefa basta apenas dar-lhe o nome padrão que se quer corresponder com o papel em questão. A correspondência entre o nome da tarefa e o papel é concretizada com base em expressões regulares.

Na Figura 25 é possível ver um exemplo da configuração de segurança que foi implementada no *Jenkins*.

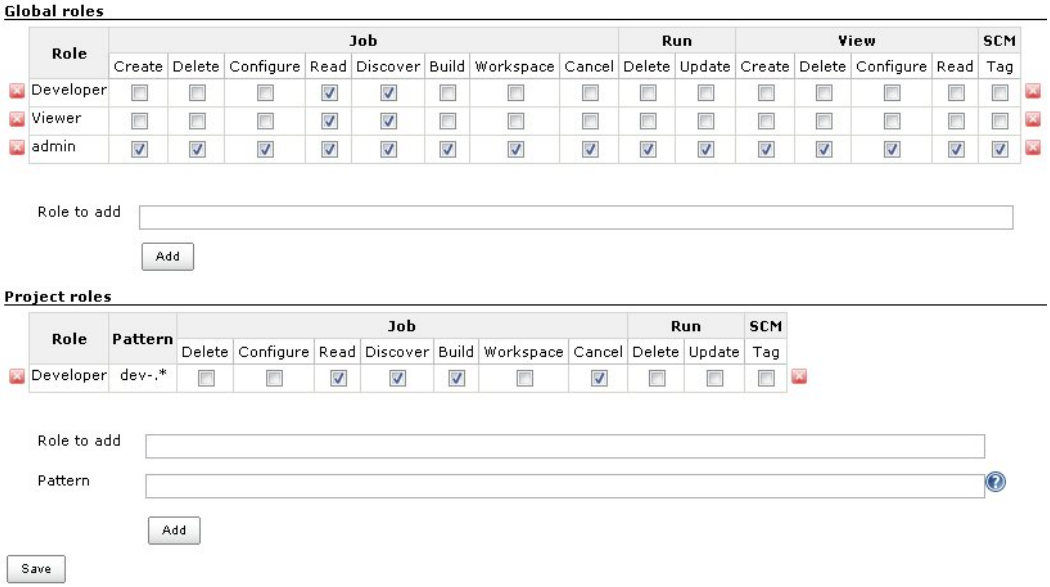


Figura 25: Excerto de configuração de papéis na segurança do *Jenkins*.

3.3.2 Implementação

De um modo geral, o principal objetivo com o uso do *Jenkins* é a compilação e disponibilização automática do *software* produzido nos vários ambientes de desenvolvimento. Para atingir este objetivo foi necessário criar várias tarefas no *Jenkins*, em que cada uma executa a sua sequência de ações.

Ao criar e configurar cada uma destas tarefas tentou-se, dentro do possível, a automatização e reutilização das mesmas, assim como a redução ao máximo da intervenção humana, de forma a mitigar os erros cometidos pelas pessoas. A forma de configurar estas tarefas foi altamente baseada nas melhores práticas de carregamento e promoção de *software*, tendo recorrido várias vezes a um guia bastante completo que contém várias informações sobre a utilização do *Jenkins* [21]. Uma das tarefas principais consiste em fazer uma atualização e promoção do código desenvolvido para ambientes superiores.

Promoção

A promoção de *software* consiste, neste contexto em particular, na cópia dos artefactos resultantes da compilação de código dos projetos desenvolvidos com recurso

à ferramenta de desenho. Estes artefactos consistem em dois pacotes de ficheiros que são, depois, carregados no servidor aplicacional *JBoss*, de modo a ficarem disponíveis para serem utilizados, como explicado em 3.2.2. Para fazer uma promoção de *software*, esta cópia de ficheiros para o ambiente em que se pretende testar o código seria suficiente. No entanto, de forma a oferecer mais controlo e segurança, foi implementado um sistema que recorre a etiquetas do *Team Foundation Server* (TFS) e a ficheiros de versão, sendo estes colocados dentro dos pacotes compilados. O TFS é um sistema de controlo de versões e uma plataforma colaborativa que permite o desenvolvimento de *software* de uma forma ágil e controlada, mantendo para isso um histórico de alterações do código desenvolvido.

Etiquetas TFS

As etiquetas do TFS funcionam como um marcador que se coloca numa versão dum ficheiro, ou seja, quando se aplica uma etiqueta a um ficheiro, a etiqueta irá conter esse ficheiro, na versão em que foi aplicada. Depois de aplicada, é possível, em qualquer altura, obter todos os ficheiros que foram marcados com uma determinada etiqueta, num determinado momento. Se algum desses ficheiros tiver sido alterado entretanto, a etiqueta não se altera, guardando sempre a informação original de quais as versões dos ficheiros que foram marcados. Esta funcionalidade torna-se útil na medida em que, permite voltar a uma versão de ficheiros que foi marcada como importante, num dado momento.

Tipicamente, quando se quer fazer uma subida de *software*, o programador grava as suas alterações de código no TFS. A partir desse momento o código alterado fica disponível no ambiente de *work*. Imaginando o caso mais comum, ou seja, que se pretende disponibilizar essas alterações para o ambiente seguinte (desenvolvimento), o programador precisa então de, manualmente, criar a etiqueta de *work* no TFS, associada ao projeto em que está a trabalhar, marcando assim essa versão do código como a versão estável do ambiente de *work*. A criação desta etiqueta faz-se com o intuito de, posteriormente, obter o código que está marcado como estável em *work*, compilá-lo e copiar os pacotes resultantes para o ambiente de desenvolvimento, marcando nessa altura (com a respetiva etiqueta) o código como estável no ambiente de desenvolvimento. Este trabalho de criação de etiquetas foi automatizado no *Jenkins* com a configuração de uma tarefa, dando a possibilidade ao utilizador de escolher o projeto e a versão do código sobre a qual será aplicada a etiqueta que deseja. Dessa forma, é

possível em qualquer altura criar uma etiqueta sobre um projeto, associada a um ambiente.

Ficheiros de versão

Para além da criação de etiquetas, o que permite ter uma segurança ao nível da recuperação de código numa determinada altura, foi implementado também um sistema de controlo de versões bastante simples, recorrendo a ficheiros de versão.

Ao compilar e promover *software* com a tarefa principal do *Jenkins*, criada para esse efeito, é criado e adicionado a cada pacote de ficheiros um ficheiro de versão que contém um identificador único de versão, para essa compilação. Esse identificador único é formado pela concatenação de várias informações como o nome do projeto, a data de compilação ou o número de compilação dessa tarefa no *Jenkins*, entre outros.

Ao conter um ficheiro com um identificador único de versão, é possível depois verificar em que versão, ou seja, em que compilação, foram criados os pacotes de código que estão em qualquer um dos ambientes.

Tarefa de promoção

Com base nos conceitos explanados anteriormente, a principal tarefa de promoção foi implementada recorrendo a um conjunto de passos, ou de ações.

Inicialmente é obtido, a partir do TFS, o código que está marcado com a etiqueta do ambiente origem, ou seja, o código que se quer promover. Estando o *Jenkins* instalado numa máquina de compilações, o código é obtido para essa máquina. De seguida, o código é compilado nessa máquina, criando os pacotes que serão mais tarde carregados no servidor aplicacional *JBoss*.

Após estar terminada a compilação, o ficheiro de versões é criado com base nessa compilação e é adicionado a cada um dos pacotes de código. Estando prontos nessa altura, os pacotes são então copiados para uma máquina temporária, máquina essa que tem uma estrutura dividida por ambientes e, quando se está a promover *software* para desenvolvimento (por exemplo), a tarefa do *Jenkins* copia os pacotes para a pasta do ambiente de desenvolvimento presente nessa máquina.

A passagem dos pacotes presentes na máquina temporária para uma máquina final é executada sincronamente por uma tarefa externa ao *Jenkins*. A máquina final tem instalado o *JBoss* e é responsável pela disponibilização dos conteúdos promovidos aos utilizadores, nos vários ambientes.

Se todos os passos executados pela tarefa de *Jenkins* correrem bem, a tarefa termina com a marcação da etiqueta do ambiente destino, ou seja, marca esse código com a etiqueta do ambiente que recebeu as atualizações, indicando que esse ambiente será estável, já que a subida só foi feita depois de se testar no ambiente inferior. Para além da etiqueta principal, é também criada uma etiqueta por compilação, oferecendo mais uma forma de controlo do que foi compilado, em que versão de código e quando foi compilado.

A principal sequência de ações, que foi implementada em várias tarefas no *Jenkins* e que demonstra a base para a promoção do *software* migrado que tem sido desenvolvido com as novas ferramentas de código aberto, está esquematizada na Figura 26, assim como a cópia de pacotes que é feita pela tarefa síncrona exterior ao *Jenkins*. O esquema exemplifica o processo de promoção de *work* para desenvolvimento mas os passos são iguais para uma promoção entre outros quaisquer ambientes.

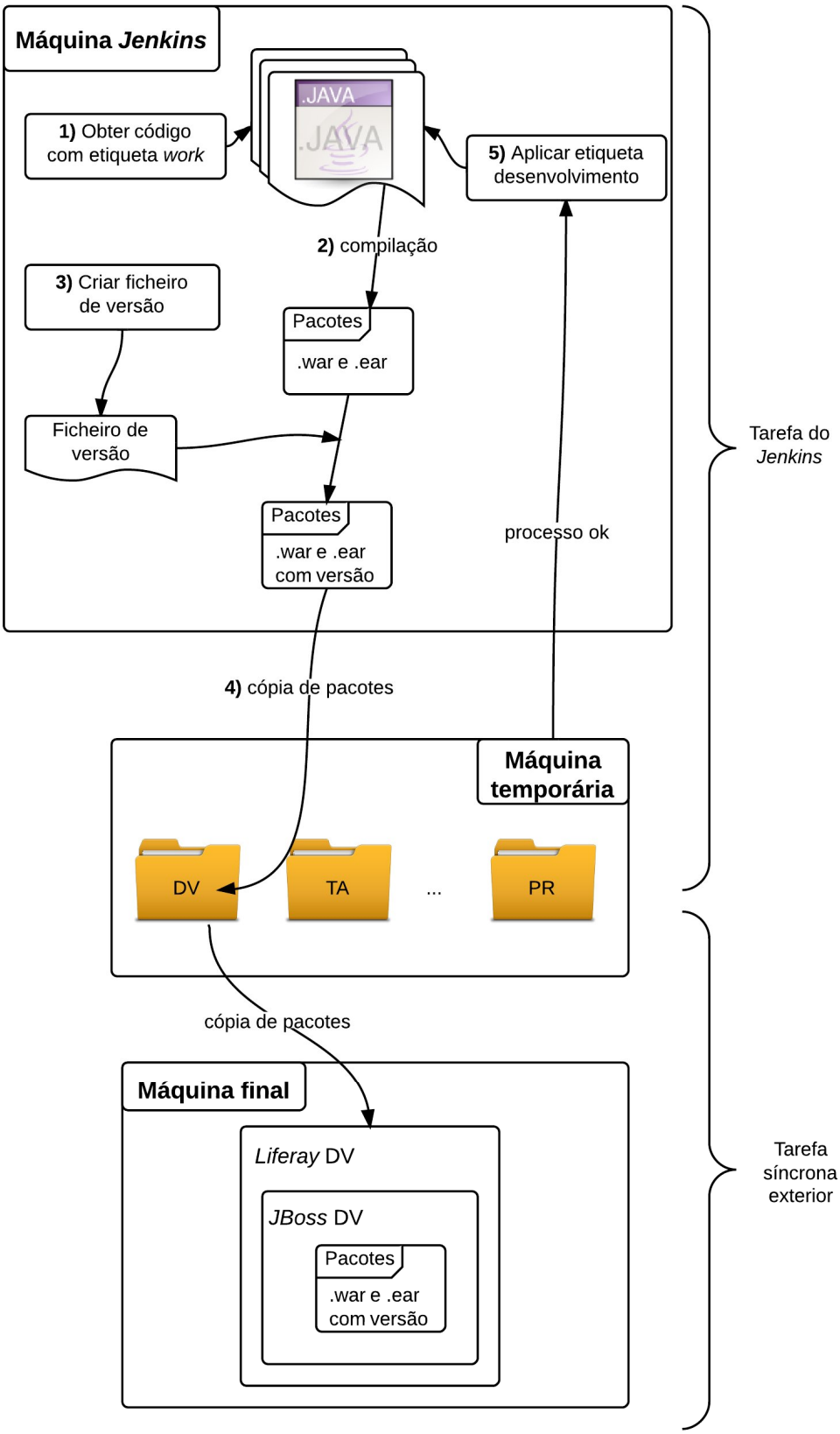


Figura 26: Exemplo de ações executadas no processo de promoção.

3.4 Migração do resseguro aceite

A aplicação “Resseguro Aceite” é constituída, à semelhança dos serviços de negócio, por um conjunto de páginas *web* que são disponibilizadas aos utilizadores, neste caso apenas no canal intranet. Tal como descrito em 1.4, as tarefas realizadas no estágio passaram pela migração da camada de gestão de interfaces para *Java*, recorrendo para isso à nova ferramenta de desenho de código aberto que está a ser desenvolvida na seguradora. Neste sentido, foram redesenhados todos os ecrãs da aplicação - cerca de uma dezena - e implementadas com sucesso todas as regras de negócio associadas a um deles, o ecrã de balancetes. Devido a este ecrã fornecer informação sobre relatórios financeiros, foi necessário, à semelhança do que foi feito com o *Jenkins*, testar uma aplicação alternativa para a apresentação de relatórios, que assentasse em tecnologias de código aberto e que se integrasse com *Java*.

3.4.1 Geração de relatórios

Para responder aos requisitos necessários, foi testada a ferramenta de geração de relatórios da *JasperReports*, a *iReport*. Esta ferramenta permite criar relatórios que contenham gráficos, imagens ou sub-relatórios, para além de suportar diversas fontes de dados, como o acesso direto a uma base de dados, ficheiros no formato XML ou no formato *Comma-separated Values* (CSV), entre outros. Para além disso, permite também exportar o resultado dos relatórios para vários formatos, incluindo PDF, *HyperText Markup Language* (HTML) ou texto simples.

Ao ser compilado um relatório que tenha sido desenhado na *iReport*, é criado um ficheiro com extensão “.jasper”. Este ficheiro permite criar uma relação entre o desenho do relatório em si com uma fonte de dados escolhida, exportando depois o relatório final no formato pretendido [19].

A Figura 27 representa a arquitetura *JasperReports* que dá suporte ao desenho, geração e exportação de relatórios.

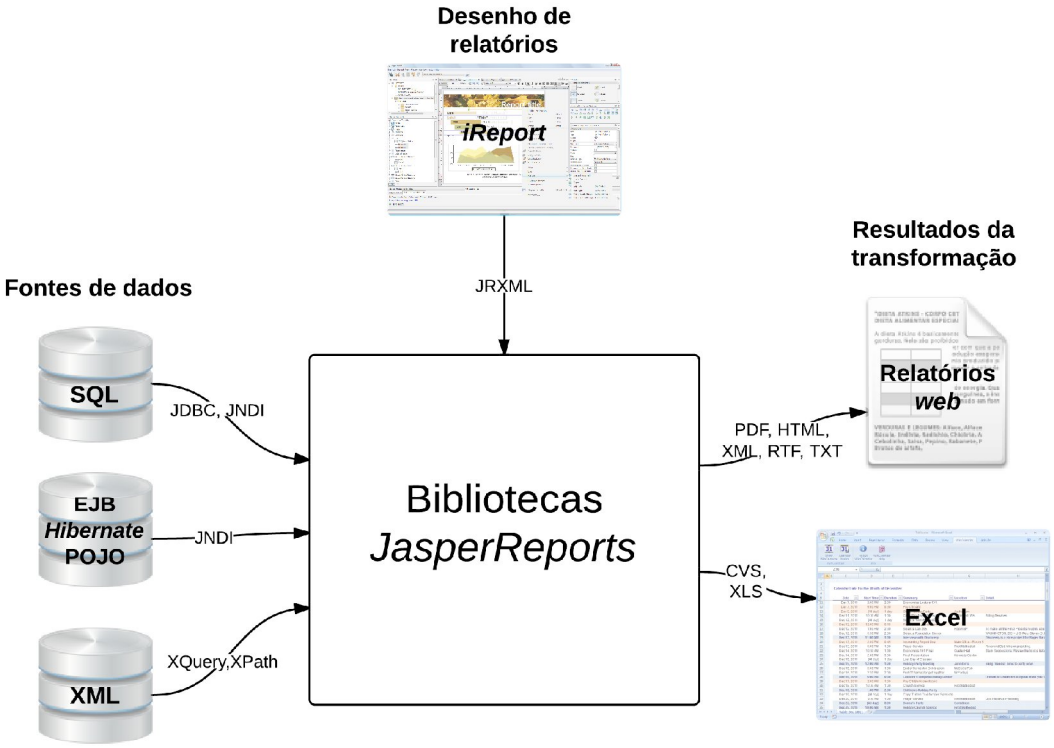


Figura 27: Arquitetura *JasperReports*.

Durante o uso da ferramenta, foram desenhados na *iReport* vários relatórios previamente implementados com outras tecnologias. Estes relatórios têm como fonte de dados um conjunto de informações estruturado em XML, aplicam as transformações necessárias e produzem um ficheiro em formato PDF, com o conteúdo da fonte de dados XML apresentado da forma desenhada. Nos relatórios migrados foram usados com sucesso várias funcionalidades da *iReport*, como a utilização de sub-relatórios para a inclusão de partes comuns entre vários (para cabeçalhos, por exemplo) ou as somas totais e parciais, por página.

Na Figura 28 é possível visualizar a implementação em *iReport* de um dos relatórios que foram migrados. De salientar a propriedade "*Query Text*" do relatório, onde se define qual o caminho a percorrer no XML de origem para encontrar cada um dos itens que irão ser discriminados na secção de detalhes do relatório.

Figura 28: Exemplo de relatório desenhado em *iReport*.

No Anexo V e no Anexo VI deste relatório estão exemplificados: uma das estruturas de dados XML que foi usada como fonte de dados para o relatório acima e um ficheiro PDF resultante da aplicação do relatório a essa fonte de dados. Devido à grandeza do conteúdo das fontes de dados XML usadas, os anexos supracitados representam um relatório com pouca informação, dando a possibilidade de verificar facilmente o mapeamento entre a fonte de dados e o ficheiro resultante. Para colmatar essa escassez, está representado no Anexo VII um ficheiro PDF resultante da aplicação de outro relatório a uma fonte de dados mais rica.

3.4.2 Ecrã de balancetes

As principais funcionalidades do ecrã de balancetes consistem na disponibilização aos utilizadores de vários tipos de relatórios financeiros relacionados com a sua conta de utilizador. Este ecrã é constituído por um conjunto de abas, cada uma com um relatório, por uma opção de escolha do mês e do ano para os quais se pretende ver as informações e pelas funcionalidades de exportação e de impressão. A exportação consiste em disponibilizar ao utilizador o relatório em formato *Excel Binary File* (XLS) e a impressão apresenta ao utilizador o relatório em formato PDF, ambas referentes à aba e data escolhidas.

Na Figura 29 é possível ver um exemplo do ecrã de balancetes que foi implementado.

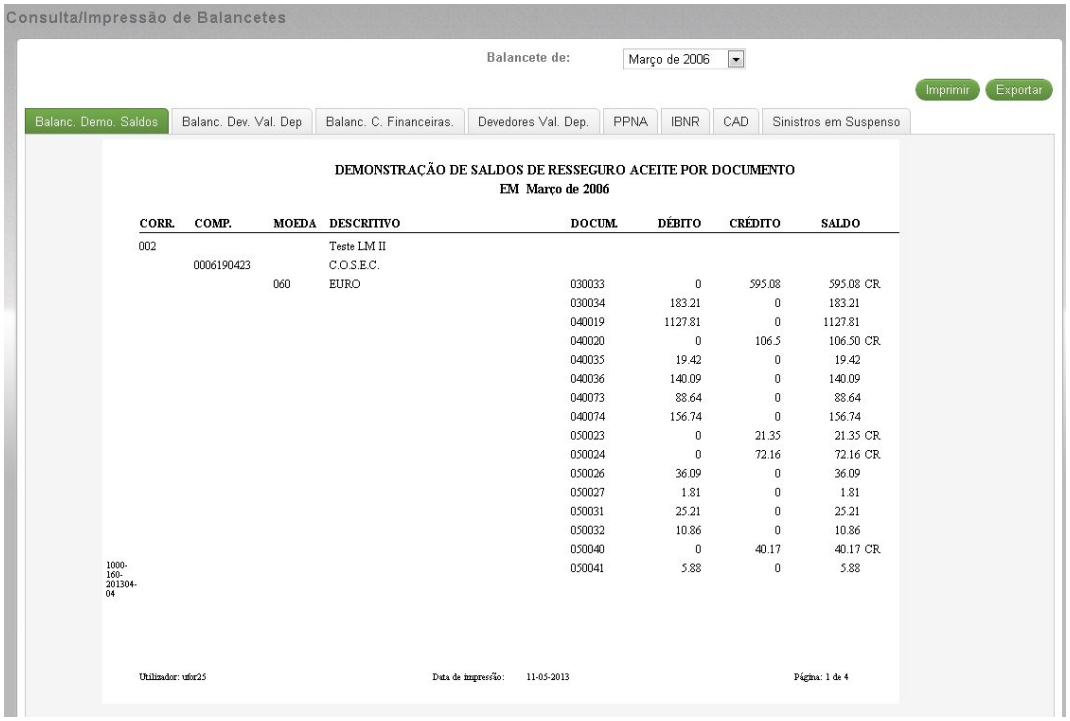


Figura 29: Ecrã de balancetes.

O processo de disponibilização de cada relatório começa no desenho do próprio relatório na ferramenta *iReport*, dando origem a um ficheiro resultante da compilação desse relatório desenhado. Os ficheiros de compilação dos vários relatórios são depois adicionados ao projeto *Java* que contém a aplicação Resseguro Aceite. Já no ecrã de balancetes, é despoletado um conjunto de ações sempre que o utilizador faz uma das seguintes tarefas: muda de aba; escolhe uma data diferente; clica num dos botões de imprimir ou exportar. Ao executar cada uma destas ações, é chamado um serviço *web* que recebe como argumentos a aba (ou seja, o nome do relatório) e a data escolhidas e devolve um conjunto de informações em XML. Este conjunto de informações é depois enviado para uma função utilitária, juntamente com o tipo de formato que se pretende obter (HTML, PDF ou XLS). Esta função utilitária recorre a bibliotecas de *JasperReports* para *Java* e aplica o relatório escolhido aos dados XML recebidos, devolvendo de seguida, para o controlador do ecrã, o resultado dessa transformação no formato pretendido. O controlador é então responsável por apresentar o relatório final na vista do ecrã, no caso do HTML, ou por enviar para o ecrã o ficheiro PDF ou XLS.

O processo implementado no ecrã de balancetes está esquematizado na Figura 30.

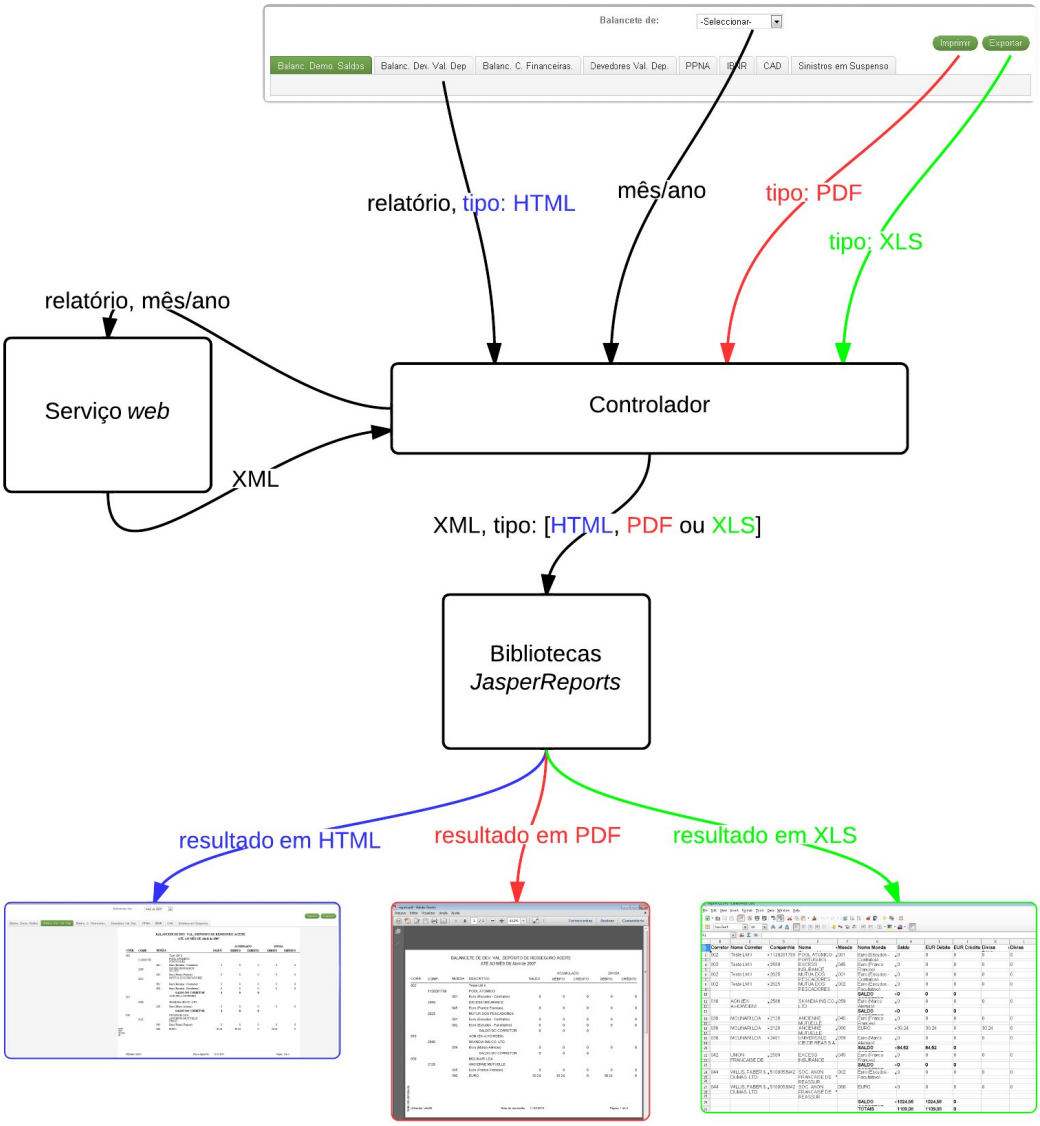


Figura 30: Processo implementado no ecrã de balancetes.

Capítulo 4

Conclusão

4.1 Estágio

Na formação inicial que tive foi-me transmitido, primeiramente, conhecimento sobre o negócio em que se insere o cliente da consultora onde foi realizado o estágio. Neste caso, como não tinha muitos conhecimentos na área, foram-me apresentados os conceitos gerais do negócio de seguros, para entender melhor o ambiente em que se insere a seguradora. Nas primeiras semanas também me foi transmitido conhecimento sobre os vários módulos existentes no *Middleware* e na Camada de Gestão de Interfaces. Este conhecimento permitiu-me perceber, inicialmente, como estão desenhadas as arquiteturas destas camadas, assim como o propósito de cada módulo e as suas componentes, para poder assim ter uma integração mais rápida com os restantes elementos.

Em relação a metodologias de trabalho existem algumas diferenças quando comparadas com as metodologias de trabalho seguidas na faculdade. Na faculdade é habitual haver grupos de poucas pessoas enquanto que, nesta seguradora, cada equipa é normalmente constituída por oito pessoas. Ainda assim, a maior diferença está no relacionamento entre equipas já que, muitas vezes o trabalho de uma pessoa depende do trabalho de uma pessoa de outra equipa; logo, tem que existir comunicação entre os elementos das várias equipas. Este foi um dos aspetos mais difíceis na integração inicial na seguradora, devido a não ter anteriormente nenhuma experiência de trabalho na área de informática e, como tal, não conhecer alguns conceitos que são importantes no meio empresarial. Outra diferença na metodologia de trabalho consiste na integração do código desenvolvido. Na faculdade esta vertente não é muito focada mas, para um produto que tenha várias pessoas a desenvolver, é essencial, já que permite que cada pessoa desenvolva o seu código sem preocupações de conflitos com trabalho feito por

outra pessoa, podendo ser feita, posteriormente, a integração. Neste sentido, é usada na seguradora a ferramenta de controlo de versões TFS, com a qual nunca tinha trabalhado anteriormente e, portanto, não tinha aprofundado o método de trabalho usando uma ferramenta de controlo de versões, pelo que houve um período de aprendizagem do uso da mesma.

Quanto ao uso do padrão MVC, nunca o tinha utilizado em ambiente académico, principalmente pela natureza dos projetos ser algo diferente do que foi feito durante o estágio. Pelas funcionalidades que fui desenvolvendo durante o estágio, concluí que este padrão se adequa bem a qualquer tipo de implementação de páginas *web*, separando de forma simples e eficaz as componentes visuais das componentes funcionais das aplicações. Por exemplo, é possível estar a desenvolver uma funcionalidade para uma página e outra pessoa da equipa trabalhar nos acessos que a funcionalidade dessa página faz à base de dados, não havendo a preocupação, quer de um quer de outro, em afetar negativamente o trabalho do outro. Como foi explicado no Capítulo 2, na comparação entre MVC e MVP, verifiquei que o MVP seria uma alternativa válida. Devido a vários problemas que, muitas vezes, são encontrados já em ambiente de testes - não detetados durante o desenvolvimento - o suporte a testes unitários avançado do MVP poderia ser bastante útil para a melhoria da qualidade dos produtos desenvolvidos. No entanto, é um facto que a comunicação estrita do MVP de cada um dos componentes apenas com o componente adjacente poderia ser uma limitação, em alguns casos. Por outro lado, esta limitação, responsável por um maior trabalho inicial, poderia compensar no futuro, em termos de escalabilidade e de manutenção de código. A migração que está a ser feita poderia ser uma boa altura para mudar para o padrão vizinho mas, compreende-se que, devido ao sucesso que o MVC tem tido e, também, à forte utilização no sistema de informação da seguradora, a alteração seria talvez demasiado custosa.

Relativamente às dificuldades sentidas, uma das maiores foi a dimensão dos projetos em que fui inserido. Cada um destes projetos tem uma grande quantidade de subprojectos, tendo cada um deles centenas de ficheiros de código. Isto leva a que, quando se altera ou adiciona alguma funcionalidade, seja preciso ter noção das repercussões que essa mudança pode ter nos projetos relacionados, tendo para isso que se ter um conhecimento alargado dos projetos. Esta barreira dificultou-me o trabalho, já que representa uma diferença abismal quando comparada com os projetos que foram feitos em ambiente académico. No entanto, essa foi uma das razões que me levou a

querer fazer um estágio fora da Faculdade de Ciências da Universidade de Lisboa (FCUL) já que, quanto mais cedo me inteirasse do ambiente empresarial, mais rapidamente conseguiria adaptar-me ao mesmo. Outra das principais dificuldades foi a aprendizagem e configuração das várias ferramentas e tecnologias a usar, já que comecei a usar a maior parte delas pela primeira vez.

De uma forma global, fiquei bastante satisfeito com as oportunidades que me foram dadas durante o estágio, com o trabalho que pude desenvolver durante o mesmo e com os seus resultados. Usei grande parte das tecnologias e metodologias de trabalho pela primeira vez e tive a oportunidade de aprender muito durante os nove meses do estágio, para além de ter reforçado e melhorado os conhecimentos que já detinha.

4.2 Trabalho

Em relação à emissão de documentos de saída e à migração de resseguro aceite, a implementação foi concluída. Na migração dos serviços de negócio foram concluídos vários, do início ao fim, incluindo tanto os serviços de envio de formulários como serviços bastante abrangentes - a nível de tecnologias - da pesquisa de entidades e do registo de clientes. Neste campo ainda há trabalho para continuar a ser feito, nomeadamente na finalização de alguns serviços, na continuação da migração dos simuladores de seguros, que começaram a ser implementados perto do final do estágio, e na conclusão de outros serviços cujo prazo de entrega não ocorreu durante a duração do estágio.

Relativamente ao *Jenkins* é possível dizer que foi investigado, planeado e aplicado com sucesso, sendo hoje a base de todas as promoções de *software* que assentam na nova plataforma de gestão de interfaces e estando a ser usado por todos os colaboradores da seguradora que estão integrados na equipa de desenvolvimento da plataforma supracitada. A introdução do *Jenkins* acelerou o processo de promoção de *software* e tem ajudado a mitigar os erros humanos que aconteciam quando se compilava e copiava manualmente os pacotes de código, dando também a oportunidade de se ter um registo detalhado sobre o que foi feito, por quem e a que horas. No entanto, o número de tarefas que são feitas automaticamente por esta ferramenta pode crescer ainda mais, levando a um desenvolvimento de *software* mais eficaz e com cada vez mais automatismos.

Expandindo o que foi explicado no ponto 1.4, o plano de trabalho foi cumprido na totalidade, à exceção de parte de um ponto, tendo sido implementado por colegas de equipa, devido a questões temporais de entrega ao cliente. Para além do plano inicial, houve ainda espaço para a implementação de vários relatórios com a ferramenta *iReport*, assim como para a migração de um conjunto de simuladores multirriscos para a nova plataforma de gestão de interfaces.

Bibliografia

[1] [...] BugNET Project. (2012). *BugNET Project.* Disponível em: http://www.bugnetproject.com. Último acesso em 20 de junho de 2013.

[2] [...] CodeFutures Corporation. (2012). *Data Access Object.* Disponível em: http://www.codefutures.com/data-access-object. Último acesso em 20 de junho de 2013.

[3] [...] Google Developers. (2013). *Google Maps API*. Disponível em: https://developers.google.com/maps. Último acesso em 20 de junho de 2013.

[4] [...] Hibernate. (2012). *Relational Persistence for Java and .NET*. Disponível em: http://www.hibernate.org. Último acesso em 20 de junho de 2013.

[5] [...] JavaMail. (2012). *JavaMail API.* Disponível em: http://www.oracle.com/technetwork/Java/Javamail/index.html. Último acesso em 20 de junho de 2013.

[6] [...] JBoss. (2012). *JBoss Community*. Disponível em: http://www.jboss.org. Último acesso em 20 de junho de 2013.

[7] [...] Jenkins. (2012). *Jenkins CI.* Disponível em: http://jenkins-ci.org. Último acesso em 20 de junho de 2013.

[8] [...] Konfide. (2012). *Software proprietário X Software Open Source.* Disponível em: http://www.konfide.com.br/salvador/artigos/software-proprietario-x-software-open-source. Último acesso em 20 de junho de 2013.

[9] [...] Liferay, Inc. (2012). *Liferay Portal*. Disponível em: http://www.liferay.com/products/liferay-portal/overview. Último acesso em 20 de junho de 2013.

[10] [...] reCAPTCHA. (2013). *What Is reCAPTCHA.* Disponível em: http://www.google.com/recaptcha/learnmore. Último acesso em 20 de junho de 2013.

[11] [...] Stack Overflow. (2011). *How to choose between Hudson and Jenkins?*. Disponível em: http://stackoverflow.com/questions/4973981/how-to-choose-between-hudson-and-jenkins. Último acesso em 20 de junho de 2013.

[12] [...] W3. (2012). *XML*. Disponível em: http://www.w3.org/XML. Último acesso em 20 de junho de 2013.

[13] [...] Wikipédia. (2012). *Software colaborativo.* Disponível em: http://pt.wikipedia.org/wiki/Software_colaborativo. Último acesso em 20 de junho de 2013.

[14] [...] Wikipédia. (2013). *Model–view–controller*. Disponível em: http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller. Último acesso em 20 de junho de 2013.

[15] [...] Wikipédia. (2013). *Model–view–presenter*. Disponível em: http://en.wikipedia.org/wiki/Model_View_Presenter. Último acesso em 20 de junho de 2013.

[16] [...] Wikipédia. (2013). *Stored procedure.* Disponível em: http://en.wikipedia.org/wiki/Stored_procedure. Último acesso em 20 de junho de 2013.

[17] Friedrich, M.. Matthias Friedrich's Blog. (2011). *Hudson vs. Jenkins Revisited.* Disponível em: http://blog.mafr.de/2011/12/27/hudson-vs-jenkins-revisited. Último acesso em 20 de junho de 2013.

[18] kooriyoo. MSDN. (2008). *Controlling XML Serialization Using Attributes* . Disponível em: http://msdn.microsoft.com/en-us/library/2baksw0z%28v=vs.80%29.aspx. Último acesso em 20 de junho de 2013.

[19] Sadik, H.. DZone. (2008). *Java Reporting With Jasper Reports - Part 2.*Disponível em: http://*Java*.dzone.com/articles/*Java*-reporting-part-2. Último acesso em 20 de junho de 2013.

[20] Schall, D.. Darron Schall. (2004). *MVC vs. MVP*. Disponível em: http://archive.darronschall.com/weblog/2004/06/mvc-vs-mvp.html. Último acesso em 20 de junho de 2013.

[21] Smart, J. F. (2011). *Jenkins: The Definitive Guide*. Sebastopol: O'Reilly Media, Inc..

[22] T.Emmatty, J.. Code Project. (2011). *Differences between MVC and MVP for Beginners.* Disponível em: http://www.codeproject.com/Articles/288928/Differences-between-MVC-and-MVP-for-Beginners. Último acesso em 20 de junho de 2013.

[23] Wehr, L.. Mashable. (2011). *Closed or Open Source: Which CMS is Right for Your Business?.* Disponível em: http://mashable.com/2011/04/05/best-cms-for-business. Último acesso em 20 de junho de 2013.

Anexos

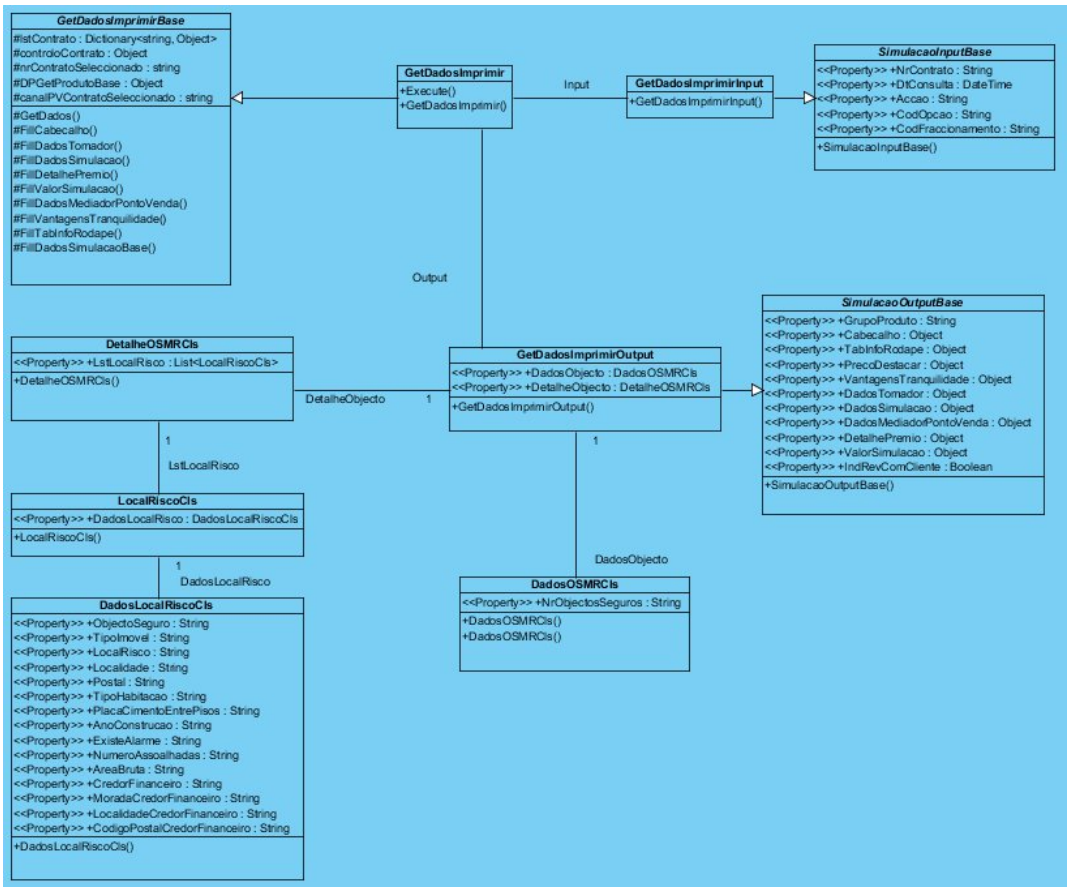
Anexo I Estrutura XML do objeto de dados

```
<Input>
  <NrContrato />
  <DtConsulta />
  <Accao />
  <CodOpcao Desc="" />
  <CodFraccionamento Desc="" />
</Input>
<Output>
  <Cabecalho>
    <DescProduto />
    <CodProduto />
    <NrSimulacao />
    <DtSimulacao />
    <ProdutoOpcao />
    <DtTarifa />
    <NrContrato />
    <CodEstadoSimulacao />
    <CodOpcao />
    <CodFraccionamento />
  </Cabecalho>
  <TabInfoRodape>
    <LabelSimulacao />
    <LabelTarifa />
    <DescontoComercial />
    <VendaCruzada />
    <CartaoT />
    <CrossSegment />
    <CodProtocolo />
    <AccaoMarketing />
    <Bancarizacao />
    <DescontoAgrupamento />
    <DescontoEmpregadoLogo />
    <DescontoProfissionalSeguros />
    <LabelInfoComercial />
  </TabInfoRodape>
  <PrecoDestacar>
    <PremioTotalDestacar />
    <DescFraccionamentoDestacar />
  </PrecoDestacar>
  <VantagensTranquilidade>
    <GrupoProdutoActual />
    <GrupoProdutoPropor />
  </VantagensTranquilidade>
  <DadosTomador>
    <NomeTomador />
```

```
  <numerocontribuinte />
  <MoradaTomador />
  <CodPostalTomador />
  <BICCTomador />
  <TipoClienteTomador />
  <numerocliente />
  <DtNascimentoTomador />
  <CAETomador />
  <TelefoneTomador />
  <EmailTomador />
</DadosTomador>
<DadosSimulacao>
  <NrSimulacao />
  <DtSimulacao />
  <DtInicio />
  <Duracao />
  <DtVencimentoAnual />
  <ModalidadePagamento />
  <Fraccionamento />
  <ValidadeSimulacao />
</DadosSimulacao>
<DadosObjecto>
  <Label />
  <LabelVariosObjectosSeguros />
  <NrObjectosSeguros />
  <ObjectoSeguro />
</DadosObjecto>
<DadosMediadorPontoVenda>
  <numeromediador />
  <NomeCobrador />
  <MoradaCobrador />
  <PostalCobrador />
  <NrPortaCobrador />
  <TelefCobrador />
</DadosMediadorPontoVenda>
<DetalhePremio>
  <PremioTotalSemDescontos />
  <DescontoComercialPremio />
  <DescontoVendaCruzada />
  <DescontoBancarizacao />
  <CustosFraccionamento />
  <PremioTotalSeguradora />
  <TotalCargas />
  <PremioTotal />
</DetalhePremio>
<ValorSimulacao>
  <LstValorFraccionamento>
    <ValorFraccionamento>
      <DescFraccionamento />
      <PremioTotal />
      <PremioPrimeiroRecibo />
      <PremioRestantesRecibos />
      <PremioDanosProprios />
    </ValorFraccionamento>
  </LstValorFraccionamento>
</ValorSimulacao>
<DetalheObjectoSeguro>
  <LstLocalRisco>
    <LocalRisco>
      <DadosLocalRisco>
        <OSLocalRisco />
```

```
            <TipoImovel />
            <MoradaLocalRisco />
            <PostalLocalRisco />
            <TipoHabitacao />
            <IndPlacaCimentoEntrePisos />
            <AnoConstrucao />
            <IndExisteAlarme />
            <NumeroAssoalhadas />
            <AreaBruta />
          </DadosLocalRisco>
          <LstOSMR>
            <OSMR>
              <IDOS />
              <CoberturasOS>
                <HeaderCoberturasOS>
                  <DescHeaderCoberturas />
                  <MarcaHeaderCoberturas />
                  <ModeloHeaderCoberturas />
                  <MatriculaHeaderCoberturas />
                  <DescOpcaoHeaderCoberturas />
                </HeaderCoberturasOS>
                <DescOpcaoCoberturasOS />
                <LstCoberturas>
                  <Cobertura>
                    <DescCobertura />
                    <IncluidoCobertura />
                    <CapitalCobertura />
                    <FranquiaCobertura />
                  </Cobertura>
                </LstCoberturas>
                <LstClausulaCV>
                  <ClausulaCV>
                    <CodigoClausulaCV />
                    <IndTextoTodoBoldClausulaCV />
                    <TextoClausulaCV />
                  </ClausulaCV>
                </LstClausulaCV>
                <LstLegenda>
                  <Legenda />
                </LstLegenda>
              </CoberturasOS>
            </OSMR>
          </LstOSMR>
        </LocalRisco>
      </LstLocalRisco>
    </DetalheObjectoSeguro>
    <IndRevComCliente />
  </Output>
```

Anexo II Diagrama de classes do objeto de dados



Anexo III Regras de implementação do objeto de dados

// Obtem os dados da simulação
#lstContrato – lista de contratos simulados (Nr Opções Seleccionadas * Nr Fraccionamentos (Anual, Semestral, Trimestral, Mensal))
#nrContratoSeleccionado – contrato "escolhido" no popup de impressão (CodOpcao + CodFraccionamento) a ser impresso no output
#DPGetProdutoBase – Definição das caracteristicas do produto multirrisco simulado: Casa, Casa Prestigio, etc...

1 GetDados(#input, #output)

// Preenche o output comum a todos os produtos
2 FillDadosSimulacaoBase(#output, #input)

Nota: Fica a faltar o preenchimento das estruturas especificas dos produtos multirriscos: DadosObjecto e DetalheObjectoSeguro

// Seleccionamos o contrato seleccionado
3 Definir #contratoSelecionado com o nº de contrato seleccionado pelo utilizador, isto é, seleccionar o contrato correspondente à opção/fraccionamento recebido por input

// Descritivos a imprimir no desenho
4 #output.DadosObjecto.Label = "Dados do Objecto Seguro / Pessoa Segura"
5 #output.DadosObjecto.LabelVariosObjectosSeguros = "Nº de Locais de Risco Seguros:"

6 Construir um dicionário (#dicLocaisRiscoOS) com a relação entre os locais de risco e os objectos seguros
6.1 **Key**: Local de Risco (Tipo de dados: Prd.Types.Apolice.LocalRiscoCls)
6.2 **Value**: Lista de Objectos Seguros (Tipo de dados: List<ObjectoSeguroCls>)
7 Percorrer os objectos seguros do contrato seleccionado (#contratoSelecionado.LstObjectoSeguro) e agrupar os objectos seguros (#OSLoop) de cada NrLocalSeguro (#OSLoop.LocalRisco.NrLocalRisco)

// Nota Importante: 1 Local de Risco pode conter 1 ou 2 Objectos Seguros
Exemplo1: O Local de Risco "Av Liberdade 1250-149 Lisboa" com os objectos Seguros Imóvel e Recheio
Exemplo2: O Local de Risco "Av Liberdade 1250-149 Lisboa" com o objecto Seguro Imóvel
Exemplo3: O Local de Risco "Av Liberdade 1250-149 Lisboa" com o objecto Seguro Recheio

// Percorrer os Locais de Risco ...
8 Para cada #localRiscoLoop em #dicLocaisRiscoOS
// Validação ...
8.1 Se no #localRiscoLoop não existe o objecto seguro Imóvel nem Recheio, Então
8.1.1 Lança um BusinessError "O Local de Risco " + #localRiscoLoop.NrLocalRisco + " não tem imóvel nem recheio definido."

// Para cada #localRiscoLoop criar um Local de Risco (#newLocalRisco) a adicionar ao #output.DetalheObjectoSeguro.LstLocalRisco
8.2 Se o #localRiscoLoop contem os objectos seguros Imóvel e Recheio, Então
8.2.1 #newLocalRisco.DadosLocalRisco.OSLocalRisco = "Imóvel e Recheio"

8.3 Caso contrário, se o #localRiscoLoop contem apenas o objecto seguro Imóvel, Então
8.3.1 #newLocalRisco.DadosLocalRisco.OSLocalRisco = "Imóvel"
8.4 Caso contrário
8.4.1 #newLocalRisco.DadosLocalRisco.OSLocalRisco = "Recheio"

8.5 #newLocalRisco.DadosLocalRisco.MoradaLocalRisco = #localRiscoLoop.Morada.MoradaFormatada
8.6 #newLocalRisco.DadosLocalRisco.PostalLocalRisco = Utils.FormatarCodPostal(#localRiscoLoop.Morada.CodPostal)
8.7 #newLocalRisco.DadosLocalRisco.TipoHabitacao = #localRiscoLoop.CodTipoHabitacao
8.8 #newLocalRisco.DadosLocalRisco.AnoConstrucao = #localRiscoLoop.CodEscalaoIdadeImovel
8.9 #newLocalRisco.DadosLocalRisco.NumeroAssoalhadas = #localRiscoLoop.NumeroAssoalhadas
8.10 #newLocalRisco.DadosLocalRisco.AreaBruta = #localRiscoLoop.AreaBruta

// Invocamos o método BODP para descodificar o tipo de imóvel
8.11 #newLocalRisco.DadosLocalRisco.TipoImovel = #getCaractNovaOfertaMultiRisco2007Output.CodTipologiaImovel

// Verifica se tem o AD da placa de cimento através #contratoSelecionado.LstAD.GetAD e da seguinte forma:
8.12 #newLocalRisco.DadosLocalRisco. IndPlacaCimentoEntrePisos = "Não" // Assume inicialmente que o indicador está a false

8.12.1 Se devolveu #placaCimentoAD, Então
8.12.1.1 #newLocalRisco.DadosLocalRisco.IndPlacaCimentoEntrePisos = "Sim"

// Preencher o indicador do alarme
8.13 Se o local de risco não contem o objecto seguro Recheio, Então
8.13.1 #newLocalRisco.DadosLocalRisco.IndExisteAlarme = "Não"
8.14 Caso contrário // Verifica se tem o AD do alarme através #contratoSelecionado.LstAD.GetAD e da seguinte forma:
8.14.1 #newLocalRisco.DadosLocalRisco.IndExisteAlarme = "Não" // Assume inicialmente que o indicador está a false
8.14.1.1 Se devolveu #alarmeAD, Então
8.14.1.1.1 #newLocalRisco.DadosLocalRisco.IndExisteAlarme = "Sim"

// Constroi as coberturas do local de risco (#newOSMR) para cada objecto seguro associado a adicionar #newLocalRisco.LstOSMR
8.15 Para cada #OSLoop em #dicLocaisRiscoOS[#localRiscoLoop]
8.15.1 FillCoberturas(#OSLoop, #lstClausulasOS, #newOSMR, #input)

9 Se o # dicLocaisRiscoOS contem os objectos seguros Imóvel e Recheio, Então
9.1 #output.DadosObjecto.ObjectoSeguro = "Imóvel e Recheio"
10 Caso contrário, se o # dicLocaisRiscoOS contem apenas o objecto seguro Imóvel, Então
10.1 #output.DadosObjecto.ObjectoSeguro = "Imóvel"
11 Caso contrário
11.1 #output.DadosObjecto.ObjectoSeguro = "Recheio"

Anexo IV Documento da simulação casa

2ª VIA

Simulação
Seguro Multirisco

Valor anual (acresce 5,45 € de custo de apólice)
88, 55 €

2ª Via gerada em 24-11-2012

Dados do Cliente
Nome:
Tipo de Cliente: Individual
N.º Cliente: 1100351148
Data de nascimento: 1974-11-09
Telefone:

Dados da Simulação
N.º Simulação: 030031132101/2FA
Data da Simulação: 2012-11-23
Data de início: 2012-11-24
Duração: 1 Ano e Seguintes
Data de vencimento anual: 24 de novembro
Modalidade de pagamento: Débito em conta
Periodicidade de pagamento: Anual
Validade da Simulação: 60 dias

Dados do Objeto Seguro
Objeto Seguro: IMÓVEL
Morada: AV GEN NORTON MATOS ESTADIO LUZ ESTADIO LUZ LISBOA
Ver detalhe no verso.

Dados do Mediador / Ponto de Venda
Nome/Ponto de Venda:

Morada:

Telefone:

CASA
VALOR F0

Detalhe do Valor a Pagar

Prémio de tarifa	83,27 €
Custos de fracionamento	0,00 €
Total seguradora	**83,27 €**
Impostos e taxas	10,73 €
Valor a pagar na primeira anuidade	**94,00 €**

Valor da Simulação

Periodicidade de pagamento	Valor a pagar na 1ª anuidade	Valor do 1º recibo	Valor dos restantes recibos
Anual	**94,00 €**	94,00 €	-
Semestral	**96,00 €**	50,48 €	45,52 €

Vantagens

Casa
- Indemnização do recheio até 100% do capital, com base no valor dos bens novos.
- Imóvel seguro pelo valor de reconstrução.
- Coberturas base muito alargadas e uma vasta oferta de coberturas opcionais.

Auto
- Assistência em viagem na hora, 24h/dia, recebendo uma indemnização em caso de atraso do reboque.
- Complemento de indemnização em caso de perda total (em danos próprios).
- Excelentes bonificações, desde o primeiro ano sem acidentes.

Cliente fidelizado
- Possuir todos os seguros permite-lhe simplificar a sua gestão e um maior controlo no seu orçamento.
- Tem vantagens financeiras, como a atribuição de descontos até 10% na compra de outros seguros.
- Somos uma empresa com forte implantação no mercado e uma grande solidez financeira.

9000-NDS-20120-01

Campanha Promocional

Não dispensa a consulta de informação pré-contratual e contratual legalmente exigida. Oferta válida até 2012-12-31.

Simulação
Seguro Multirisco

Detalhe do Objeto Seguro

Objeto Seguro: IMÓVEL
Local de risco: AV GEN NORTON MATOS ESTADIO LUZ ESTADIO LUZ
LISBOA
1500-313 LISBOA
Ano de construção do Imóvel: >=1930 e <1960
N.º de assoalhadas: 3
Área bruta de construção(m2): 232,00

Tipo de Imóvel: Vivenda
Tipo de Habitação: Principal
Imóvel construído com placa de cimento entre pisos: Sim
Habitação protegida com alarme com ligação à polícia e/ou central de alarmes e/ou telemóvel do segurado: Não

Opção selecionada: **CASA - VALOR F0**

Coberturas, Capitais e Franquias

Morada: AV GEN NORTON MATOS ESTADIO LUZ ESTADIO LUZ
Código Postal: LISBOA
Localidade: 1500-313 LISBOA

		IMÓVEL	
COBERTURAS	**INCLUÍDO NA SIMULAÇÃO**	**CAPITAIS / FRANQUIA**	
Atos grevistas	Sim	(1)	Não aplicável
Actos vandalismo	Sim	(1)	Não aplicável
Inundações/tempest./aluimentos	Sim	(1)	Não aplicável
Aluimentos de terras			
Inundações			
Tempestades			
Danos por água	Sim	(1)	Não aplicável
Incêndio, raio e explosão	Sim	(1)	Não aplicável
RC familiar/proprietário	Sim	50 000,00 €	Não aplicável
RC proprietário			
Danos no imov.p/furto ou roubo	Sim	5 000,00 €	Não aplicável
Honorários de técnicos	Sim	5 000,00 €	Não aplicável
Pesquisa de avarias	Sim	5 000,00 €	Não aplicável
Outros riscos	Sim	(1)	Não aplicável
Choque de veículos terrestres		(1)	
Demolição e remoção escombros		5 000,00 €	
Danos por fumo ou calor		5 000,00 €	
Derrame acidental de óleo		(1)	
Danos de carácter estético		5 000,00 €	
Proteção jurídica		5 000,00 €	
Privação temporária uso local		5 000,00 €	
Queda de aeronaves		(1)	
Quebra e queda de antenas		5 000,00 €	
Quebra, queda painéis solares		5 000,00 €	
Quebra louças sanitárias		5 000,00 €	
Quebra vidros / pedras mármore	Sim	5 000,00 €	Não aplicável
Riscos elétricos	Sim	5 000,00 €	Não aplicável
Assistência ao domicílio	Sim	(2)	Não aplicável
Acidentes pessoais	Não	Opcional	Não aplicável
Fenómenos sísmicos	Não	Opcional	Opcional
Perda de rendas	Não	Opcional	Não aplicável
Reconstituição de jardins	Não	Opcional	Não aplicável
Respons. civil piscinas	Não	Opcional	Não aplicável

2

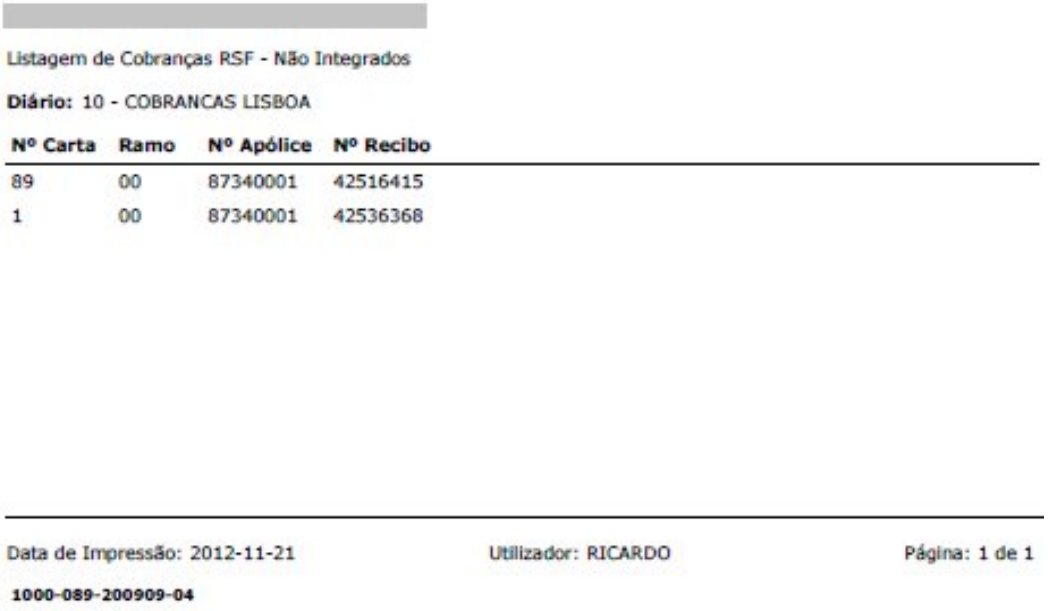
9000-NDS-20120-01

Anexo V Fonte de dados XML

```
<?xml version="1.0" encoding="UTF-8"?>
<Function ID="CxBOConsCobrancasRSF.BOSel.Execute">
   <ReturnValue>0</ReturnValue>
   <Output>
      <IdModelo>089</IdModelo>
      <ModeloCanal>10</ModeloCanal>
      <ModeloVariante>«vazio»</ModeloVariante>
      <Status>
         <Diario>10</Diario>
         <NrFolhaCaixa>022</NrFolhaCaixa>
         <Ano>2009</Ano>
         <DataMovimento>2009-09-08</DataMovimento>
         <StatusCaixa>1</StatusCaixa>
         <StatusCaixaUtilizador>1</StatusCaixaUtilizador>
         <BibEcras>10</BibEcras>
         <BibParamConta>10</BibParamConta>
         <AcessoFrontOffice>S</AcessoFrontOffice>
         <AcessoBackOffice>S</AcessoBackOffice>
         <AcessoRSF>S</AcessoRSF>
         <AcessoGestao>S</AcessoGestao>
         <AcessoAdministracao>S</AcessoAdministracao>
         <AcessoContabilidade>S</AcessoContabilidade>
         <AcessoCheques>S</AcessoCheques>
         <AcessoMediadores>S</AcessoMediadores>
         <NomeUtilizador>RICARDO</NomeUtilizador>
         <UtilizadorMed>MEDIAD</UtilizadorMed>
         <Tesouraria>A</Tesouraria>
         <AberturaRSF>3</AberturaRSF>
         <StatusCheques>1</StatusCheques>
         <BibCentroCustos>01</BibCentroCustos>
         <BibCompConta>01</BibCompConta>
         <BibTipoMovimento>01</BibTipoMovimento>
         <BibContP>01</BibContP>
         <Offline>X</Offline>
         <ImportarRec>S</ImportarRec>
         <UtilizadorMultiDiario>1</UtilizadorMultiDiario>
         <Divisa>EUR</Divisa>
         <Conta>0000010</Conta>
      </Status>
      <Deleg>
         <Diario>10</Diario>
         <Delegacao>COBRANCAS LISBOA</Delegacao>
         <DelegAbrev>COBRANCAS LX2</DelegAbrev>
         <Area>3</Area>
         <Localidade>LISBOA</Localidade>
      </Deleg>
      <ListaCobrancaRSF>
         <CobrancaRSF>
            <Diario>10</Diario>
            <Ano>2004</Ano>
            <NrFolhaCaixa>005</NrFolhaCaixa>
            <Utilizador>xunrcc</Utilizador>
            <Carta>89</Carta>
            <NrSeq>64</NrSeq>
            <Ramo>00</Ramo>
            <NrApolice>87340001</NrApolice>
            <NrRecibo>42516415</NrRecibo>
```

```
            <TipoApolice>«vazio»</TipoApolice>
            <ValorRecibo>87.33</ValorRecibo>
            <DataMovimento>2004-11-17</DataMovimento>
            <CodBanco>«vazio»</CodBanco>
            <NrCheque>«vazio»</NrCheque>
            <ValorCheque>«vazio»</ValorCheque>
            <NrCobrador>0000084</NrCobrador>
            <DataImpressao>1900-01-01</DataImpressao>
            <SituacaoRecibo>A</SituacaoRecibo>
            <StatusEnvio>«vazio»</StatusEnvio>
            <Reconciliado>*</Reconciliado>
            <NrMensagem>39600000776</NrMensagem>
            <Transaccao1>1737846110</Transaccao1>
            <Transaccao2>«vazio»</Transaccao2>
            <Valor>0</Valor>
            <DateTime>2005-03-23 9:02:26</DateTime>
            <ErroEnvio>'Notice' inexistente na tabela W5N0</ErroEnvio>
            <TotalCheques>«vazio»</TotalCheques>
            <TotalRecibos>«vazio»</TotalRecibos>
            <Diferenca>«vazio»</Diferenca>
         </CobrancaRSF>
         <CobrancaRSF>
            <Diario>10</Diario>
            <Ano>2004</Ano>
            <NrFolhaCaixa>005</NrFolhaCaixa>
            <Utilizador>xunrcc</Utilizador>
            <Carta>1</Carta>
            <NrSeq>79</NrSeq>
            <Ramo>00</Ramo>
            <NrApolice>87340001</NrApolice>
            <NrRecibo>42536368</NrRecibo>
            <TipoApolice>«vazio»</TipoApolice>
            <ValorRecibo>91.91</ValorRecibo>
            <DataMovimento>2004-11-17</DataMovimento>
            <CodBanco>«vazio»</CodBanco>
            <NrCheque>«vazio»</NrCheque>
            <ValorCheque>«vazio»</ValorCheque>
            <NrCobrador>0000089</NrCobrador>
            <DataImpressao>1900-01-01</DataImpressao>
            <SituacaoRecibo>«vazio»</SituacaoRecibo>
            <StatusEnvio>«vazio»</StatusEnvio>
            <Reconciliado>*</Reconciliado>
            <NrMensagem>39600000775</NrMensagem>
            <Transaccao1>4627846110</Transaccao1>
            <Transaccao2>«vazio»</Transaccao2>
            <Valor>«vazio»</Valor>
            <DateTime>2005-03-23 9:02:27</DateTime>
            <ErroEnvio>'Notice' inexistente na tabela W5N0</ErroEnvio>
            <TotalCheques>«vazio»</TotalCheques>
            <TotalRecibos>«vazio»</TotalRecibos>
            <Diferenca>«vazio»</Diferenca>
         </CobrancaRSF>
      </ListaCobrancaRSF>
      <NomeListagem>RSFNaoIntegrados</NomeListagem>
   </Output>
</Function>
```

Anexo VI Documento gerado por bibliotecas *JasperReports*



Listagem de Cobranças RSF - Não Integrados

Diário: 10 - COBRANCAS LISBOA

Nº Carta	Ramo	Nº Apólice	Nº Recibo
89	00	87340001	42516415
1	00	87340001	42536368

Data de Impressão: 2012-11-21 Utilizador: RICARDO Página: 1 de 1

1000-089-200909-04

Anexo VII Documento gerado por bibliotecas *JasperReports* - II

Listagem de Cobranças RSF - Integrados

Diário: 10 - COBRANCAS LISBOA

Nº Carta	Ramo	Nº Apólice	Nº Recibo	Valor Recibo	Nº Cobrador	Banco	Nº Cheque
108	00	87340001	42544285	2374,13	0000087	07	5023270543
107	00	87340001	42630255	89,07	0000062	21	4300000111
106	00	87340001	42638345	297,95	0000075	35	0332889803
105	00	87340001	42430970	1604,81	0000004	18	7200000022
104	00	87340001	42601565	2333,89	0000056	10	6580841718
103	00	87340001	42659527	341,38	0000022	33	5955145181
102	00	87340001	42615188	54,77	0000014	07	5822762000
101	00	87340001	42528342	239,53	0000089	10	6668643812
99	00	87340001	42612132	1385,59	0000077	07	1325528645
98	00	87340001	42597015	665,26	0000073	45	0496600268
97	00	87340001	42658761	68,51	0000062	33	4347525346
96	00	87340001	42563350	151,69	0000057	35	3759820526
95	00	87340001	42649475	177,63	0000068	36	5095336697
94	00	87340001	42555092	298,43	0000082	35	5510857493
93	00	87340001	42534934	115,92	0000051	35	9833916233
92	00	87340001	42649844	45,33	0000098	35	9513601468
91	00	87340001	42002954	245,93	0000004	35	6929418911
90	00	87340001	42603241	275,53	0000072	33	3877733959
89	00	87340001	42511939	517,4	0000084		
89	00	87340001	42517976	6,68	0000084	33	7575055983
88	00	87340001	42644050	35,81	0000053	36	4812538593
87	00	87340001	42507328	28,33	0000084		
87	00	87340001	64182660	36,73	0000096		
87	00	87340001	64182662	50,07	0000096		
87	00	87340001	64182661	50,07	0000096	33	7112598168
86	00	87340001	42617809	68,93	0000014	35	8121604269
85	00	87340001	42525615	643	0000062	10	4172694947
84	00	87340001	42657017	196,2	0000076	35	6131868305
83	00	87340001	42437878	28,21	0000067	35	6922309241
82	00	87340001	42616945	72,29	0000096	07	8967669921
81	00	87340001	42625917	72,29	0000096	07	6467669913
80	00	87340001	42587516	282,45	0000076	35	9119665815

Total: 12.853,81

Data de Impressão: 2012-11-21 Utilizador: RICARDO Página: 1 de 1

1000-089-200909-04