Mestrado em Engenharia Informática

Estágio

Relatório Final

CisionPoint 2.0

Tiago António Fernandes Nunes

tnunes@student.dei.uc.pt

12 de Julho de 2011

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

Agradecimentos

À Cision Portugal pela oportunidade proporcionada.

Aos meus orientadores, Eng. Pedro Ladeira e Eng. Carlos Laranjeiro pelos conselhos dados.

Aos elementos da equipa CisionPoint, a Sónia, o Sérgio e o Diogo pela paciência que tiveram e toda a ajuda que prestaram e que para que todo este estágio fosse levado a bom porto.

À Patrícia pela companhia e amor que tanto me motivaram nestes meses.

E por fim, à minha família, especialmente os meus pais e madrinha que desde sempre me deram o apoio necessário para chegar aqui.

Resumo

A Cision é a empresa líder na disponibilização de serviços de planeamento, contacto, monitorização e análise de *media*. Diariamente, é feita a selecção e tratamento de toda e qualquer informação difundida pelos *media*.

Todos os seus serviços são disponibilizados através da plataforma *online* de serviços denominada CisionPoint, acessível em qualquer parte do mundo, via Internet, 24 horas por dia, sete dias por semana.

Este integra todos os serviços da Cision numa plataforma modular completamente personalizada, disponível na Internet e/ou intranet.

Toda a informação disponível no CisionPoint encontra-se segmentada e armazenada numa base de dados regular. O volume de dados é grande e o seu crescimento diário.

Por esta razão existe a necessidade de diariamente serem realizados processos de manutenção na base de dados para que esta esteja sempre na sua melhor performance. Visto que esta manutenção impede o funcionamento normal do resto do sistema, para que o serviço aos clientes não seja afectado, esta manutenção é feita durante a noite.

Contudo, dada a quantidade de informação existente e o seu crescimento diário, os planos de manutenção foram demorando cada vez mais tempo, e começaram a terminar já nas primeiras horas da manhã. Isto resulta em atrasos na disponibilização do serviço aos clientes.

Este estágio pretende implementar medidas de fundo no sistema que resultem na diminuição da duração dos planos de manutenção e que consequentemente possibilitem que o CisionPoint tenha uma maior disponibilidade. Estas medidas centram-se em dois pontos principais, simplificar e reorganizar a estrutura de dados e diminuir a dimensão desta. Assim pretende-se por um lado agilizar o acesso e a gestão da informação e por outro retirar do sistema informação desnecessária que apenas abranda o desempenho deste.

Palavras-Chave

Base de dados, Cision, CisionPoint, *media*, notícias, *online*, performance, *SQL Server*

Glossário

Backup – Processo de criação de cópias de dados que servem para restauro dos valores originais no caso de existir perda de informação (Walters, et al., 2009).

Parsing – Processo de análise de uma sequência de entrada (lida de um ficheiro ou teclado) para determinar a sua estrutura gramatical segundo uma dada gramática (Oxford University Press).

Clipping – Processo de recolha de notícias em jornais, revistas, sites ou outros meios de comunicação, que tenham interesse para quem realiza este processo.

Media – meios de comunicação social e agências de notícias.

Social Media – *Media* baseados nas redes sociais como o *FaceBook* ou o *Twitter*.

Research – Processo de pesquisa de jornalistas e respectivos contactos. No caso particular da Cision, estes contactos são usados posteriormente pelos seus clientes na aplicação *Connect* do CisionPoint para a realização, por exemplo, de *press releases*.

Índice

Lista de Figuras

Lista de tabelas

Capítulo 1
Introdução

1.1. Âmbito do estágio

Este estágio enquadra-se no âmbito da disciplina "Dissertação/Estágio" do Mestrado em Engenharia Informática do Departamento de Engenharia Informática da Universidade de Coimbra. O orientador designado pelo departamento para acompanhar este estágio foi o Eng. Carlos Laranjeiro que fez a ligação entre o Departamento de Engenharia Informática e a empresa em que o estágio decorre, a Cision Portugal. Por parte da Cision Portugal, o Eng. Pedro Ladeira foi o responsável e orientador do estágio, bem como o mediador entre a empresa e o Departamento de Informática da Universidade.

O estágio decorreu nas instalações da Cision Portugal, desde o dia 1 de Setembro de 2010 até ao dia 30 de Junho de 2011, num regime *fulltime*, correspondente a 40 horas semanais presenciais nestas instalações.

O trabalho realizado é parte do projecto CisionPoint, desenvolvido pelo departamento de IT da empresa, no qual o aluno foi integrado.

1.2. Enquadramento

Actualmente e cada vez mais uma empresa para obter sucesso, além de ter que criar um produto ou serviço de qualidade, necessita de cuidar da imagem que transmite ao exterior. É esta imagem que cativa os clientes a procurarem os seus serviços e que lhes transmite uma primeira impressão da empresa e dos seus serviços.

Os *media* são, dada a sua abrangência, um meio primordial de transmissão de informação, e como tal uma influência muito grande na informação a que as pessoas têm acesso todos os dias. Se por um lado, esta influência se reflecte na imagem das empresas de que os *media* falam, por outro lado esta influência pode ser usada pelas empresas a seu favor através da monitorização dos *media* e do planeamento cuidado das suas campanhas de *marketing*.

A Cision é uma empresa que tem como missão ajudar os profissionais de comunicação – independentemente do tamanho do seu negócio – a gerirem esta ligação entre as empresas e os meios de comunicação. Disponibiliza também ferramentas que permitem aos seus clientes planearem a sua mensagem, contactarem o seu público-alvo, monitorizarem a cobertura dos média e analisarem os resultados obtidos nas suas campanhas de *marketing*. Para além da informação que disponibiliza a Cision oferece ainda a capacidade de compreender o significado da mesma de forma adaptada às necessidades das empresas e negócios.

CisionPoint é o portal integrado através do qual os clientes da Cision acedem aos serviços prestados por esta. Neste portal os clientes podem gerir as suas actividades de relações públicas numa aplicação única de uma forma simples e intuitiva. Na prática, o CisionPoint integra várias aplicações destinadas aos vários serviços genéricos da empresa: planear, contactar, monitorizar e analisar. É neste portal, mais especificamente na ferramenta de disponibilização de *clipping*, que este estágio se baseia e é este o serviço que o estagiário se propôs melhorar.

1.3. Objectivos

O objectivo genérico deste estágio é o de criar um sistema com disponibilidade o mais próxima possível das 24 horas por dia de modo a que os serviços de manutenção deste, como por exemplo a realização de *backups* diários da informação, não afectem a qualidade e eficiência do serviço prestado aos clientes. Neste sentido foram definidos pela equipa do CisionPoint os seguintes objectivos intermédios para este estágio:

- Eliminar a replicação de dados e as falhas na integridade destes existentes nas bases de dados de suporte ao CisionPoint;
- Reduzir o peso das chaves primárias das tabelas das bases de dados através da substituição do tipo de dados utilizado, *Guid*, por um tipo de dados adaptado às necessidades actuais do sistema;
- Optimizar os planos de manutenção tendo em conta as alterações realizadas;
- Criar uma estrutura de dados que suporte uma solução de histórico, ou seja, que privilegie a informação mais actual em detrimento de dados mais antigos e por isso menos acedidos;
- Realizar a migração do projecto CisionPoint da *framework* .Net 1.1 para a versão 4.0 (e consequentemente do *Microsof Visual Studio 2003* para a versão 2010) e do *Microsoft SQL Server 2000* para a versão 2008.

Com estes objectivos não é esperado que o tempo de manutenção do sistema passe a ser nulo, mas sim que seja dado um contributo valioso na redução desse tempo utilizando os recursos físicos já existentes.

1.4. Estrutura do documento

Este documento está dividido nos seguintes 8 capítulos.

Capitulo 1 - Introdução

Neste capítulo é apresentado o âmbito em que este estágio se realiza, os seus objectivos, o enquadramento de todo o projecto e um breve resumo da estrutura deste documento.

Capitulo 2 - Estado da Arte

Este capítulo pretende introduzir alguns conceitos que foram utilizados durante este estágio para solucionar os problemas encontrados.

Capitulo 3 - Estado inicial do projecto

Nesta secção é apresentado o estado em que foi encontrado o CisionPoint pelo estagiário. Não é uma descrição exaustiva de todo o sistema, mas apenas dos módulos que estão relacionados directa ou indirectamente com este estágio.

Capitulo 4 - Método de abordagem

Resumo da abordagem feita aos problemas encontrados no estágio.

Capitulo 5 - Trabalho Realizado

Descrição de todo o trabalho realizado no âmbito do estágio.

Capitulo 6 - Descrição dos testes realizados

Descrição dos testes realizados à plataforma com o objectivo de aferir as melhorias conseguidas com as alterações feitas ao sistema durante este estágio.

Capitulo 7 - Resumo e análise dos resultados obtidos

Neste capítulo são mostrados e discutidos os resultados dos testes realizados.

Capitulo 8 - Conclusões e trabalho futuro

Conclusões do trabalho realizado e dos objectivos atingidos e trabalho futuro a realizar neste projecto.

Referências

Listagem das referências consultadas durante o trabalho.

Anexos

Listagem dos documentos anexos a este relatório.

Capítulo 2
Estado da Arte

Neste capítulo pretende-se fazer uma introdução aos principais pontos de acção deste estágio ao nível tecnológico, nos quais se basearam todas as decisões tomadas ao longo deste estágio e consequentemente todo o trabalho realizado.

Em primeiro lugar são abordados alguns conceitos sobre performance e sobre a forma como esta é avaliada. Nas quatro secções seguintes são abordados vários tópicos relacionados com bases de dados e a sua onfiguração. Por fim na sexta secção é abordado o tema da aquisição de informação ou *information retrieval*.

2.1. Performance

A performance dos sistemas de informação é um tema que tem vindo a ter cada vez mais importância principalmente devido à sua utilização cada vez mais generalizada. A dimensão destes sistemas tem também vindo a aumentar e com ela aumenta também a informação que estes gerem. Este aumento provoca por sua vez um crescimento da exigência imposta aos sistemas ao nível da performance fazendo com que este seja cada vez mais um factor a ter em conta em todas as decisões tomadas na gestão dos mesmos.

O aumento da velocidade com que a informação circula faz com que os sistemas de informação tenham que ser cada vez mais eficientes e com que a informação tenha que ser mostrada aos clientes cada vez mais rapidamente. Por outro lado, o aumento da importância da informação para os negócios faz com que a informação que chega aos clientes tenha que ter uma qualidade cada vez melhor e cada vez com menos falhas.

Por estes motivos é necessário que os administradores dos sistemas, utilizando várias métricas, vão monitorizando o estado destes. Algumas das métricas mais utilizadas são a escalabilidade e a disponibilidade que são descritas de seguida:

- **Escalabilidade** - é a capacidade do sistema suportar um aumento de carga sem que este se faça repercutir na performance de todo o sistema e sem que o funcionamento deste seja posto em causa (Bellevue Linux Users Group, 2004).
- **Disponibilidade** - é a relação existente entre o tempo em que o sistema se encontra a funcionar correctamente e todo o seu tempo de vida (Gomes, et al., 2001).

2.2. Armazenamento de dados

Bases de dados são estruturas que têm como objectivo armazenar informação de uma forma organizada e estruturada. Dentro desta estrutura a informação é guardada em objectos que estão por sua vez organizados e agrupados em tabelas.

Estes objectos em que a informação é armazenada podem ser de vários tipos, com o objectivo de se adaptarem da melhor forma possível à informação que guardam. Por exemplo, se quisermos guardar apenas um valor numérico inteiro pode ser utilizado um objecto preparado para receber apenas valores numéricos e portanto mais simples e fácil de utilizar que outros tipos de objecto. Claro que este valor numérico inteiro poderia ser também guardado num objecto que suportasse outro tipo de informação, como valores fraccionários ou texto, não se perdendo a informação, contudo o facto de esses objectos

serem mais complexos aumentaria o espaço que estes ocupam e aumentava a complexidade da sua gestão e utilização. Como tal, é importante durante o *design* de uma base de dados que o tipo de dados utilizado seja definido de acordo com o tipo de informação que se pretende guardar com o objectivo de melhorar a performance do acesso à informação e de reduzir o espaço ocupado por toda a estrutura.

2.3. Restrições de integridade

No caso da base de dados do CisionPoint o espaço ocupado pela informação é um ponto crítico. O aumento constante do tamanho da base de dados agrava progressivamente a performance de todo o sistema. Este é um facto imutável, a informação cresce continuamente e tem que estar disponível para o CisionPoint e para os seus clientes integralmente.

Para além deste aumento, erros na criação e gestão de uma base de dados podem fazer com que a informação deixe de ser exacta e consistente levando também a que se verifique um aumento do espaço desta desnecessariamente. Isto acontece por vários motivos, ou porque informação é replicada, ou porque informação sem qualidade continua a ser armazenada.

Para assegurar a exactidão e consistência dos dados, e consequentemente uma utilização mínima de espaço, são utilizadas restrições de integridade. Estas são definidas na estrutura da base de dados e obrigam a que várias regras sejam aplicadas à informação sempre que esta é inserida, eliminada ou alterada. As restrições de integridade são divididas em quatro tipos (Microsoft Corporation, 2011):

- **Integridade de entidade**
  - Este tipo de restrições é responsável por garantir que um campo chave primária de uma tabela não tem valores repetidos ou não definidos. Garante assim que este campo permite ao utilizador identificar de forma unívoca cada registo da tabela.
- **Integridade referencial**
  - Esta restrição garante que, se uma tabela tem uma referência para um registo de outra tabela, este registo existe necessariamente. Garante ainda que no caso de ser eliminado o registo referenciado acontece uma de duas coisas: ou o registo que tem a referência também é eliminado ou é gerada uma excepção e o registo não é eliminado.
- **Integridade de domínio**
  - Integridade de domínio obriga a que a informação seja armazenada e disponibilizada no formato correcto. Se um campo de uma determinada tabela é do tipo numérico, apenas valores numéricos podem ser guardados nesse campo. Também neste tipo de restrição é definido se um campo pode ter valores a *NULL* ou se tem um valor *default*.
- **Integridade aplicacional ou de negócio**
  - Podem também existir restrições ao nível do negócio, que têm que ser tratadas não ao nível da base de dados mas ao nível aplicacional. Por exemplo, numa tabela que guarde o salário de um funcionário este valor nunca pode ser menor do que zero ainda que ao nível da base de dados valores negativos possam ser guardados.

2.4. Data Partitioning

Apesar de as soluções anteriores reduzirem efectivamente a dimensão da estrutura de dados, quando a estrutura de dados engloba uma grande quantidade de informação, por melhor que se optimizem os tipos de dados utilizados ou outros parâmetros a pesquisa continua a ser feita num bloco muito grande de informação cuja performance piora de forma directamente proporcional ao aumento desta. Uma das soluções possíveis para este problema é a divisão desse bloco de informação em blocos mais pequenos cuja manutenção e manuseamento seja mais simples. A este processo é dado o nome de *data patitioning*.

Data partitioning é uma técnica utilizada no *design* de estruturas de dados (Schumacher , 2010). Esta utiliza a filosofia *divide and conquer* e consiste, tal como foi dito acima, em separar os dados em vários blocos que são tratados independentemente e a que é dado o nome de partições. Assim é possível em primeiro lugar que a pesquisa da informação nos vários blocos possa ser feita simultaneamente, ou seja em paralelo, tirando partido de vários processadores ou até de várias máquinas. Por outro lado o facto de a pesquisa ser feita em blocos mais pequenos agiliza o processo de pesquisa, pois passa a ser necessária a leitura de menos registos em cada conjunto de dados para que os resultados pretendidos sejam atingidos.

Existem genericamente dois tipos de particionamento:

Particionamento vertical em que cada partição armazena um dado conjunto de colunas de uma tabela. Assim a informação de cada registo desta fica distribuída pelas várias partições da base de dados (Schumacher , 2010). Isto possibilita por exemplo que se as partições se encontrarem em dispositivos físicos diferentes, a leitura dos dados de um registo possa ser feita em paralelo e portanto de uma forma mais rápida.

Em contradição existe também o **particionamento horizontal**, em que a informação de cada registo não é separada pelas várias partições mas sim os registos integralmente (Schumacher , 2010). Em cada partição existe assim um conjunto de registos independente. Assim apesar de a leitura de um registo singular não ser acelerada, passa a existir a possibilidade de serem lidos vários registos em simultâneo, caso estejam em partições diferentes, acelerando desta forma a devolução de resultados.

A selecção dos registos a colocar em cada partição pode ser feita de várias formas, dependendo das necessidades do sistema. As formas de selecção mais conhecidas são a selecção por intervalos de valores, selecção por lista de registos ou selecção utilizando uma função de *Hash*.

Na selecção por intervalo de valores, cada partição tem um intervalo de valores definido e pertencem a essa partição todos os registos que tenham, um determinado campo, com valor compreendido nesse intervalo. Por exemplo, numa base de dados que registe os dados dos clientes de uma hipotética loja, na partição X serão guardados todos os registos dos clientes com idade compreendida então 0 e 30 anos e na partição Y serão guardados os restantes registos, ou sejam aqueles cujo campo idade tenha valor superior a 30 anos. Este tipo de selecção é utilizado quando o campo do registo pelo qual é feita a selecção tem valores contínuos como valores numéricos ou datas.

A selecção por lista de valores é idêntica à anterior com a diferença de que, não é definido um intervalo em que os valores estarão compreendidos, mas sim os valores específicos que devem pertencer à partição. Um exemplo deste tipo de selecção seria a selecção para uma partição de todos os clientes com o campo localidade igual a "Coimbra". Os restantes

pertenceriam a outra partição. Esta selecção é utilizada apenas quando o campo selector é constituído por valores discretos e previamente definidos.

Por fim, na selecção por função de *Hash*, como o próprio nome indica a partição a que um registo pertence é feita a partir de uma função de *Hash* aplicada a um dos seus campos. O objectivo desta selecção é espalhar pelas várias partições os registos com o valor deste campo idêntico, possibilitando assim que as pesquisas obtenham resultados do maior número de partições possível. Isto proporciona a leitura de registos em paralelo e com isto a maior velocidade na pesquisa dos dados.

É claro possível fazer combinações destas três formas de selecção tirando partido das várias vantagens destas.

Qualquer um dos caminhos seguidos na partição dos dados tem vantagens a dois grandes níveis. Ao nível da performance do acesso aos dados, devido à redução da quantidade de informação que tem que ser lida para que os resultados sejam apurados e ao nível da complexidade de todo o sistema, a estrutura de gestão dos dados passa a ter que gerir apenas pequenos e mais simples conjuntos de dados em vez de trabalhar com um grande e complexo conjunto de dados.

2.5. *Backups* de base de dados

Tal como já foi dito antes neste documento, nos tempos que correm a informação é cada vez mais um bem importante e valioso. No caso particular da Cision esta tem ainda mais valor devido à importância que tem para todo o negócio da empresa. Todos os dias na Cision é recolhida, tratada e disponibilizada aos clientes uma grande quantidade de informação. Dado que o negócio da Cision passa sobretudo pela disponibilização desta informação é crucial que esta nunca seja perdida e que seja mantida nas melhores condições.

Qualquer sistema de armazenamento está propenso a falhas, acidentes ou catástrofes. Se a informação estiver em papel o papel pode arder ou molhar-se e a informação fica perdida. No caso de se tratar de uma *cassete VHS* a fita pode-se deteriorar. Várias coisas podem também acontecer a um disco rígido de um computador fazendo com que a informação que contem seja perdida.

O facto de o disco ficar danificado é imutável, mas existe forma de prevenir que a informação se perca definitivamente. Se a informação que está no disco danificado estiver replicada num segundo disco ou noutra estrutura de armazenamento, o disco danificado pode ser reparado ou substituído e a informação restituída ao disco inicial a partir da estrutura de dados que estava em duplicado.

Esta cópia da informação pode ser criada de várias formas dependendo dos riscos que o sistema corre e de quão crítica é a informação copiada.

No caso de a informação estar contida numa base de dados este processo pode ser realizado utilizando ferramentas disponibilizadas pelo próprio motor de base de dados. Uma destas ferramentas e também uma das mais utilizadas é a ferramenta de criação de *backups* (Walters, et al., 2009). Esta permite ao administrador do sistema guardar toda ou parte da informação contida na base de dados num ou em vários ficheiros. Estes ficheiros podem ser depois utilizados para repor o estado da base de dados no momento da criação do *backup* no caso da base de dados se perder ou ficar corrompida.

Este processo tem duas contrapartidas mais relevantes e contraditórias. Em primeiro lugar no caso de ocorrer um problema na base de dados algum tempo depois da realização do

backup, sendo o backup utilizado para repor a informação, todas as alterações feitas no intervalo entre o *backup* e a reposição seriam perdidas. Por esta razão seria proveitoso realizar *backups* com mais frequência possível. Por outro lado, o *backup* é um processo pesado do ponto de vista da performance, facto que se agrava com o aumento da dimensão da base de dados. Tendo em conta este facto, é importante que os *backups* não sejam realizados com demasiada frequência. É por estas duas razões importante que seja encontrado o equilíbrio ideal entre a quantidade de informação que é passível de ser perdida e a frequência com que são realizados os *backups*.

Devido a esta dificuldade e com o objectivo de adaptar o melhor possível os processos de *backup* ao negócio em causa existem várias formas de realizar *backups*: *backup* completo, *backup* diferencial e *backup* de ficheiros ou grupos de ficheiros (Walters, et al., 2009).

***Backup* completo** tal como o nome indica consiste na criação de uma cópia integral da base de dados. Este tipo de *backup* é mais demorado que os restantes, contudo tem a vantagem de a partir do ficheiro criado ser possível reconstruir toda a base de dados com toda a informação existente no momento da realização do *backup*.

Um ***backup* diferencial** consiste em criar apenas um *backup* da informação que foi alterada desde a realização do último *backup* completo. Isto faz com que este processo seja realizado mais rapidamente devido à dimensão muito menor da informação a copiar. Em contrapartida utilizando este tipo de *backup* é necessário que exista um *backup* completo logo à partida e toda a informação, do completo e do diferencial, tem que ser reunida no momento em que seja necessário recuperar o sistema o que aumenta a complexidade e a duração deste processo.

Estes dois tipos de *backup* têm em comum uma elevada necessidade de espaço em disco para poderem guardar toda a informação existente na base de dados.

Logicamente, numa base de dados pode existir informação prioritária e mais importante e informação menos crítica e que não carece de tantos cuidados de segurança. Por este motivo existem os ***backups* de ficheiros ou grupos de ficheiros**. Com este tipo de *backup* é possível que apenas seja feito o *backup* de alguma informação da base de dados e não de toda a base de dados. Isto faz com que por um lado o espaço ocupado em disco pelo ficheiro de *backup* seja menor, por outro lado dado que a quantidade de informação é menor a realização do backup é feita mais rapidamente.

2.6. Information Retrieval

Information Retrieval é o processo de pesquisar e/ou seleccionar informação de natureza estruturada ou não, dentro de um conjunto alargado de informação (Manning, et al., 2008).

O cerne deste processo é logicamente a forma como é realizada a pesquisa. Consideremos o exemplo simples da realização da pesquisa de duas palavras num conjunto de documentos de texto. A forma mais óbvia de efectuar a pesquisa seria percorrer os documentos um a um e para cada documento comparar cada palavra deste documento com as palavras pesquisadas. Este seria um processo simples para a pesquisa de uma palavra num conjunto relativamente pequeno de documentos.

Logicamente, do ponto de vista de um qualquer utilizador de um hipotético sistema que realize esta pesquisa o ideal seria que os resultados fossem obtidos instantaneamente ou que fossem obtidos no menor espaço de tempo possível. Contudo, se a pesquisa fosse feita num conjunto de documentos muito grande e/ou com um conjunto muito maior de palavras o

processo seria proporcionalmente mais demorado. Por esta razão ao longo dos anos este tema tem vindo a ser estudado tendo sido encontrados vários métodos de pesquisa que têm uma performance melhor que o método descrito a cima.

Dadas as variadas utilidades da *Information Retrieval* a escolha do processo de pesquisa utilizado depende também de outros factores periféricos à pesquisa.

Em primeiro lugar é relevante a forma como a informação é obtida, se é adquirida no momento da pesquisa através de serviços externos ao sistema ou se por outro lado é obtida e armazenada pelo sistema previamente, antes de ser feita a pesquisa.

Por outro lado também a finalidade e o tipo de resultados pretendidos pelas pesquisas podem influenciar o método de pesquisa utilizado. Por exemplo, se apenas for necessário saber a quantidade de documentos que têm uma dada palavra a abordagem será diferente para um caso em que seja importante mostrar destaque para documentos que tenham a referida palavra num maior número de vezes.

No caso particular do CisionPoint as pesquisas são realizadas no portal de duas formas. A pesquisa simples, em que o utilizador insere um conjunto de palavras e espera obter todos os artigos em que no titulo ou no corpo do texto têm as palavras inseridas e uma pesquisa avançada que para além da pesquisa no texto existente na pesquisa simples possibilita ao utilizador a definição de outros parâmetros para filtrar os resultados, como o tipo de *media* pretendido ou um intervalo de datas específico. Estas duas pesquisas devem devolver toda a informação necessária para o preenchimento da grelha de resultados do portal.

No início do estágio a informação do CisionPoint estava armazenada principalmente numa base de dados Sql Server. Contudo, dada a complexidade e especificidade da realização de pesquisas de texto, a realização das pesquisas do portal era feita utilizando uma instância Lucene em que estavam indexados todos os artigos presentes na base de dados.

Lucene é uma biblioteca de *Information Retrieval* originalmente desenvolvida em Java, contudo neste momento está já disponível num vasto conjunto de linguagens de programação (The Apache Software Foundation, 2011).

Esta biblioteca possibilita em primeiro lugar a indexação de informação. A informação é armazenada em campos (*fields*) de texto que por sua vez estão agrupados em documentos. Por exemplo, armazenando informação sobre artigos, cada artigo será representado por um documento no Lucene e cada um desses documentos terá vários campos que guardam a informação do respectivo artigo como o título, a data e o texto do artigo propriamente dito. No momento da indexação cada um dos campos tem três atributos que indicam a forma como estes devem ser indexados: *stored*, *indexed* e *tokenized*. O primeiro indica que o texto do campo deve ficar disponível para ser devolvido no resultado das pesquisas. O segundo indica se o campo deve ficar disponível para ser pesquisado. E o atributo *tokenized* indica se o texto, antes de ser indexado, deve ser analisado e separado em *tokens*.

Também no Lucene existem ferramentas de pesquisa destes dados indexados. Com estas ferramentas podem ser elaboradas desde as pesquisas mais simples, de uma palavra num determinado campo por exemplo, até pesquisas mais elaboradas que utilizem por exemplo operadores booleanos ou definição de intervalos de valores. Nesta pesquisa é ainda possível definir o resultado pretendido. Um utilizador pode não necessitar de toda a informação indexada sobre o artigo pesquisado ou pode querer os resultados ordenados por um determinado campo. Tudo isto pode ser definido na pesquisa feita ao Lucene.

Apesar de o Lucene trazer muitas vantagens, tem também algumas limitações. Por este motivo os seus criadores decidiram desenvolver o SolR.

SolR é uma plataforma de pesquisa baseada na biblioteca Lucene. Como tal faz uso do potencial e ferramentas de pesquisa do Lucene acrescentando-lhe algumas novas funcionalidades e optimizações.

Ao contrário do Lucene, o SolR tem suporte para outros tipos de dados além dos dados de texto, como valores numéricos ou datas, possibilita a pesquisa de informação de um modo facetado, a definição de *highlighting* nos resultados, entre outras possibilidades.

No SolR é ainda possível definir com precisão os campos que compõem os documentos indexados, bem como a forma como estes são indexados. Isto é feito através de um *schema* em que são definidos os tipos de dados utilizados no índice e os campos que cada documento indexado terá. Assim todos os documentos existentes num dado índice terão que ter a estrutura definida no *schema* deste. Cada um destes campos tem ainda definidos vários atributos, à semelhança do que acontecia no Lucene, que definem a forma como estes devem ser indexados e pesquisados. O SolR acrescenta também alguns novos atributos aos três existentes no Lucene. Na tabela seguinte são descritos os mais comuns destes atributos.

Tabela 1: Atributos dos campos dos índices SolR

Atributo	Descrição
Type	Neste atributo é definido o tipo de dados indexado neste campo
Default	Valor utilizado neste campo sempre que este não seja preenchido
Indexed	Verdadeiro se o campo em questão deve estar disponível para ser pesquisado
Stored	Se for verdadeiro o campo ficará disponível para ser integralmente devolvido no resultado de uma pesquisa
Compressed	Define se a informação guardada no campo deve ser comprimida
Multivalued	Verdadeiro se este campo pode ter mais do que um valor
omitNorms	Este campo permite que as normas associadas a este campo possam ser ignoradas com o objectivo de melhorar a performance da indexação do mesmo

As novas possibilidades que o SolR abre à realização de pesquisas podem também ser uma vantagem para a performance desta tarefa, razão pela qual foi abordada no âmbito deste estágio.

Capítulo 3
Estado inicial do CisionPoint e problemas identificados

O ponto de partida para este estágio foi a plataforma CisionPoint tal como é disponibilizada actualmente para os clientes da Cision Portugal. Dado que este sistema já sofreu várias alterações ao longo da sua existência e dada a sua dimensão, não é prático que seja feita uma descrição exaustiva do mesmo neste capítulo. Como forma de dar ao leitor o conhecimento necessário para compreender as necessidades deste estágio e as suas implicações, mostram-se aqui apenas as características do CisionPoint às quais foram apontados problemas no âmbito do estágio ou características que estão directa ou indirectamente relacionadas com esses problemas.

No início deste estágio o CisionPoint encontrava-se na versão 1.0 sendo objectivo do estágio levar este portal até à versão 2.0. Por questões de *marketing* esta nova versão foi também apelidada de CisionPoint 24 por ter como ambição estar disponível para os clientes 24 horas por dia.

3.1. Planos de manutenção da base de dados

Quando este estágio começou, o problema fundamental apontado ao CisionPoint era a duração dos planos de manutenção da base de dados. Estes planos realizam várias operações necessárias ao seu bom funcionamento, como optimização de tabelas e índices, verificação de integridade, realização de *backup* de dados, entre outras e têm necessidade de ser realizados na sua maioria diariamente. Estes planos, enquanto decorrem, impedem o bom funcionamento de todo o sistema devido à sobrecarga criada na base de dados e são por isso normalmente realizados durante o período nocturno. Esta interrupção nocturna do sistema até à data não trazia problemas. Contudo, devido ao aumento da quantidade de informação presente na base de dados, a duração destes planos começou a ser demasiado extensa, passando a afectar o normal funcionamento da empresa e do serviço prestado aos clientes nas primeiras horas do dia. Para além deste facto, actualmente, o período nocturno já não pode ser considerado um período em que o sistema pode estar parado, porque apesar de em Portugal ser de noite e os clientes normalmente não acederem ao portal neste horário, podem existir clientes a aceder ao portal noutra parte do mundo com um fuso horário diferente.

Por estas duas razões mostrou-se necessário diminuir ao máximo a duração dos planos de manutenção, tentando aproximar o tempo de disponibilidade do CisionPoint das 24 horas diárias.

3.2. Organização e estrutura da base de dados

Dada a constante mudança da área de negócio da Cision, bem como o seu crescimento, as necessidades desta estão permanentemente em mutação. Todas as semanas existem requisitos novos para a plataforma CisionPoint que logicamente vão sendo validados e satisfeitos à medida que aparecem. Isto faz com que o CisionPoint se vá alterando e crescendo à medida das necessidades dos clientes e impede que este crescimento seja feito de um modo tão estruturado e optimizado como seria ideal. Este tipo de abordagem é necessária mas acarreta algumas consequências ao nível da estrutura e organização de todo o projecto. As bases de dados que suportam o sistema não são excepção. A constante alteração dos requisitos do sistema implicou logicamente constantes modificações na

estrutura e funcionamento das bases de dados utilizadas, levando a que várias tabelas deixassem de ser úteis ou alguns dos seus campos deixassem de ser utilizados. Dada a complexidade da sua eliminação da base de dados, estes continuavam a ser preenchidos no momento em que os dados eram importados para a base de dados, aumentando a complexidade do processo e aumentando o tamanho ocupado pela informação. Existiam também algumas relações do tipo chave estrangeira que não estavam formalizadas com restrições de integridade. Este facto, juntamente com erros simples e inócuos existentes no sistema, foi criando problemas ao longo do tempo na integridade referencial dos dados, que se repercutiam na performance de todo o sistema.

3.3. Tipo de dados das chaves primárias

O CisionPoint no momento da sua implementação inicial não foi construído para o fim que tem actualmente. A plataforma começou por ser uma estrutura que seria usada por todas as empresas do grupo Cision. Foi, por isso, dimensionada para uma quantidade de utilizadores e de informação muito maior do que tem actualmente na Cision Portugal. Este facto é evidente no tipo de dados que foi escolhido para as chaves primárias, *Global Unique Identifier*, que tem uma gama de valores muito superior à quantidade de registos que existe nas tabelas utilizadas pelo CisionPoint. Este facto cria um peso evidentemente desnecessário tanto no espaço que cada chave ocupa, como no processamento que esta requer na sua utilização, peso esse que poderia ser evitado utilizando outro tipo de dados com tamanho e complexidade menores.

3.4. Sistema de histórico

A Cision recolhe informação dos *media* desde 2006, que disponibiliza sob várias formas aos seus clientes através do CisionPoint. Toda esta informação foi sendo guardada ao longo dos anos na base de dados referida anteriormente. Ou seja, todos os artigos recolhidos desde esse ano até aos artigos da última semana estão guardados numa mesma tabela na base de dados. Só os artigos da última semana estão numa tabela desnormalizada, para que a sua consulta seja feita mais rapidamente. Sempre que é pesquisada informação que não seja da última semana os dados pretendidos são seleccionados entre toda a informação de todos os anos. Esta informação está sempre a crescer, o que faz com que estas pesquisas na base de dados sejam cada vez mais demoradas atrasando o serviço disponibilizado aos clientes, atraso este que em alguns casos começa a ser insuportável do ponto de vista do negócio.

3.5. Tecnologias

No início deste estágio a plataforma encontrada estava implementada utilizando a *framework .Net 1.1* e as bases de dados de apoio a esta utilizavam o motor de base de dados *Microsoft SQL Server* na versão 2000. Um dos objectivos deste estágio é o de realizar a migração destas tecnologias para as versões mais recentes.

Para além destas tecnologias, neste projecto são ainda utilizadas outras tecnologias como:

- *C#*
- *XML*
- *Javascript*
- *SQL*
- *Lucene/SolR*

Capítulo 4
Trabalho Realizado

Considerando a dimensão e complexidade da aplicação que este estágio aborda e considerando que vários dos objectivos propostos podem ser realizados de uma forma relativamente independente uns dos outros, foi decidido pela equipa do CisionPoint abordar os objectivos do estágio individualmente ou em conjuntos pequenos, não perdendo nunca de vista o objectivo final.

Assim foi realizado cada um dos passos enumerados de seguida de uma forma integral, ou seja, só começou a ser executado o passo seguinte depois de o anterior ter sido dado como concluído.

- Reestruturação da base de dados e alteração dos tipos de dados das chaves das bases de dados, bem como de outras estruturas dependentes destas, tal como procedimentos e funções;
- Adaptação do motor de importação às alterações feitas;
- Adaptação do portal CisionPoint às alterações feitas;
- Criação de uma estrutura de dados de histórico;

Logicamente, no início deste estágio, toda a estrutura de dados era desconhecida para o estagiário. Por esta razão, antes que qualquer trabalho fosse realizado, foi realizada uma reunião com um dos elementos da equipa em que foi explicada a estrutura e organização das principais tabelas da base de dados ao estagiário. Esta explicação não tinha como objectivo dar um conhecimento profundo da estrutura de dados, mas apenas introduzir os principais conceitos do sistema e as principais relações existentes entre a informação.

Neste ponto o estagiário poderia ter realizado um estudo de todo o sistema, todos os seus componentes e relacionamentos para conhecer realmente o seu funcionamento e limitações. Contudo rápido ficou patente que, dada a dimensão e complexidade de todos os componentes que constituem o CisionPoint, este estudo seria algo demoroso, complexo e sobretudo neste ponto do estágio, desnecessário. Por esta razão, tal como aconteceu com o desenvolvimento dos vários objectivos, também o conhecimento do sistema foi sendo obtido de forma particionada à medida que ia sendo requerido para o desenvolvimento das fases enunciadas em cima.

Nas secções seguintes pretende-se descrever o trabalho que foi realizado ao longo das várias fases de desenvolvimento. À excepção da secção 4.8, "Migração da plataforma", o trabalho foi realizado pela ordem encontrada neste capítulo. A secção "Migração da plataforma" foi realizada, dadas as suas características, de uma forma transversal e simultânea ao resto das tarefas do projecto.

4.1. Criação da base de dados CisionPoint24

Como foi apontado no início do estágio, o grande *bottleneck* do sistema é a base de dados, visto que todos os serviços se baseiam na informação obtida desta. É portanto importante que o CisionPoint24 corrija e melhore a estrutura da base de dados de modo a que tanto os processos de manutenção como a prestação de serviços aos clientes sejam agilizados.

Sendo a base de dados o suporte de toda a aplicação e o principal foco de intervenção deste estágio, torna-se o ponto de partida óbvio para a transformação que se pretende efectuar no CisionPoint.

A análise das alterações pretendidas na base de dados levou-nos à conclusão de que a forma mais eficiente de efectuar este processo seria simplificando cada passo destas, alterando e reconstruindo um módulo da plataforma de cada vez.

Foi por isto criada uma nova instância de base de dados, a que foi dado o nome CisionPoint24, com o objectivo de substituir a instância usada nas versões anteriores do CisionPoint, a O4K. A partir desta última foram gerados *scripts* de criação de todas as suas tabelas e vistas para que estas pudessem ser reconstruídas nas novas instâncias.

A instância CisionPoint24 é constituída por duas bases de dados com os nomes CisionPoint24 e ObserverCommon24, à semelhança da instância O4K que é também composta por duas bases de dados, O4K e ObserverCommon. Esta separação física dos dados não tem relevância no âmbito deste estágio, é por esta razão ignorada daqui para a frente.

Estes *scripts* foram então a base para a realização das alterações estruturais da base de dados. As alterações foram realizadas neles de modo a que a criação de toda a estrutura de dados fosse feita na nova instância já com as alterações implementadas.

Como forma de simplificar a leitura deste documento o par de bases de dados CisionPoint24 e ObserverCommon24 é genericamente referido como uma única base de dados, CisionPoint24.

4.2. Reestruturação da base de dados

Tal como explicado acima, com a constante alteração das necessidades do negócio e consequentemente da aplicação, a base de dados foi sendo adaptada ao longo do tempo para que essas necessidades fossem sendo satisfeitas. Assim, foram sendo adicionadas ou removidas tabelas e restrições que fizeram com que a estrutura desta fosse sendo alterada. Dada a dimensão desta estrutura e a complexidade das relações que existem entre a informação, quando é feita uma alteração é difícil controlar todas as vertentes que esta afecta, levando a erros de construção e de organização da base de dados. Isto leva, por exemplo, a que seja violada a integridade referencial, a que sejam replicados dados ou a que simplesmente deixem de ser usados os dados, continuando a ocupar espaço em disco desnecessariamente.

Os índices existentes na base de dados foram ignorados na criação dos *scripts* com o objectivo de, depois de realizadas todas as optimizações, ser feita uma nova análise, para que os índices utilizados estejam adaptados a esta nova estrutura e às suas necessidades.

Selecção das tabelas a eliminar da base de dados

A análise pormenorizada de todas as tabelas e das relações que estas têm entre si, revelaram a existência de tabelas que não eram no momento usadas ou que poderiam ser suprimidas com o objectivo de melhorar a performance do sistema. Esta análise foi realizada tabela a tabela através da observação das suas características, relações e dos valores que estas contêm no ambiente de produção.

Durante esta análise foram registados os vários critérios que foram sendo usados para justificar a eliminação das tabelas. Estes critérios de eliminação foram aplicados em toda a base de dados com o objectivo de eliminar todas as tabelas que não são nem se prevê que venham a ser utilizadas pela aplicação.

Estes critérios são os enumerados de seguida:

- ***AutomatedDashboard* deixou de ser usado** – tabelas de apoio à aplicação *AutomatedDashboard*, aplicação que não é utilizada no momento;
- **Tabela sem registos** – tabela sem dados e que não é usada em actividades temporárias como é o caso da importação de dados;
- **Tabela temporária** – tabela usada para armazenar dados específicos, pontuais e estáticos. Na sua maioria são tabelas aparentemente usadas durante análises pontuais ou de testes e que acabaram por ficar esquecidas no final destas análises;
- **Tabela não utilizada** – tabelas que pela sua natureza e utilidade, e devido a adaptações realizadas na base de dados antes deste estágio, deixaram de ser utilizadas pela aplicação;
- **Tabela deixou de ser utilizada** – tabela que deixou de ser utilizada devido à reestruturação da base de dados realizada no âmbito deste estágio;
- **Tabela com dados desactualizados** – tabelas com dados temporais que pela sua desactualização evidenciam que a aplicação deixou de necessitar destes;
- **Tabela com poucos registos e com dados sem sentido** – tabelas com número relativamente baixo de registos em que os valores destes não fazem sentido no contexto da base de dados. Aparentemente os poucos registos que estas tabelas têm são fruto de testes.
- **Tabela cuja referência foi eliminada** – tabelas de ligação em que uma das tabelas que tem o objectivo de ligar foi eliminada durante a optimização actual da base de dados;
- **Tabela com dados repetidos** – tabela que contém dados que existem já numa outra tabela idêntica;
- **Tabela com relações redundantes** – tabelas cuja existência provoca violações na integridade dos dados;
- **Tabela eliminada por questões de performance** – tabela eliminada durante a reestruturação da base de dados com o objectivo de simplificar e optimizar o acesso à informação e assim melhorar a performance da base de dados.

Na análise realizada, foram seleccionadas para serem eliminadas 114 tabelas das duas bases de dados usadas pela aplicação, CisionPoint24 e ObserverCommon24, que estão enumeradas no anexo A deste documento. Neste está registado, além do nome das tabelas seleccionadas, o critério da sua eliminação.

A eliminação de todas estas tabelas foi realizada através da alteração dos *scripts* de criação mencionados anteriormente. Este método foi escolhido para que o processo de eliminação seja feito de uma forma mais metódica e simples, com o objectivo de guardar os vários estados da alteração da base de dados de uma forma mais perene e permitir que caso algum erro seja encontrado seja fácil recuperar informação perdida.

Principais módulos da base de dados afectados pela eliminação das tabelas

A eliminação das tabelas fez não só com que o espaço ocupado pela base de dados fosse reduzido, mas também com que toda a estrutura das relações das tabelas fosse alterada. Para ilustrar as principais alterações realizadas e denotar a redução e simplificação do número de tabelas e relações existentes na base de dados foram agrupadas algumas tabelas em vários diagramas que podem ser consultados no anexo B deste documento.

Selecção dos campos das tabelas a eliminar

À semelhança das tabelas, há também alguns campos destas que não têm utilidade no contexto actual do sistema. O número destes campos aumentou também com a eliminação de tabelas da base de dados porque todos os campos que referenciavam campos das tabelas apagadas deixaram de ser úteis. Por esta razão foi também realizada uma análise da base de dados ao nível dos campos das tabelas, observando tabela a tabela todos os seus campos, com a finalidade de identificar e eliminar todos aqueles que não serão úteis na nova versão da base de dados. Neste processo foram identificados trinta e quatro (34) campos nestes termos que estão listados no anexo A deste documento.

Alteração do tipo de dados das chaves das tabelas

Depois de seleccionadas as tabelas que serão realmente úteis para o funcionamento do CisionPoint foi realizada a mudança do tipo de dados utilizado nas chaves identificadoras das tabelas. Estas tinham inicialmente o formato *uniqueidentifier* mas tornou-se evidente que seria vantajoso que este fosse alterado para o tipo numérico (*numeric*). Esta conclusão foi obtida através da análise comparativa feita a uma das tabelas principais do CisionPoint, a tabela *Articles*, que como o nome indica contém todos os artigos mostrados na plataforma.

Esta tabela contém no momento cerca de 10.2 milhões de artigos provenientes dos anos compreendidos entre 2006 e 2010, sendo que 3.3 milhões dos quais pertencem ao ano de 2009. Considerando que cada variável do tipo *GUID* necessita de 16 *bytes* para ser guardada e que cada variável numérica apenas precisa de 9 *bytes* (Microsoft Corporation, 2010), as colunas com identificadores das tabelas passam aproximadamente a ocupar metade do tamanho actual. Esta diminuição afecta logicamente o espaço ocupado pelas tabelas e pelos seus índices, o que é uma das preocupações da realização deste estágio. Esta diminuição de espaço depende de caso para caso de acordo com vários factores. Por exemplo uma tabela que tenha poucos campos que sejam chaves será menos afectada por esta alteração, já no caso dos índices, que na sua maioria são feitos sobre chaves de tabelas, a diminuição de espaço ocupado é normalmente muito elevada visto que o que estes guardam são estes campos chave.

Com esta mudança no tipo de dados das chaves das tabelas, além de a base de dados passar a ocupar menos espaço, pretende-se ainda com os campos numéricos simplificar o tratamento desses valores. Tratando-se de valores numéricos, operações como comparações e ordenações tornam-se mais simples e rápidas. Por exemplo, quando se pretende um determinado artigo com um dado identificador, mesmo com a utilização de índices é necessário realizar varias comparações entre o identificador dado e identificadores presentes na base de dados. Obviamente comparar dois valores numéricos é bastante mais simples do que comparar dois *Guid's* devido à diferença de tamanho e complexidade destes tipos, logo a pesquisa tornar-se-á mais leve e mais rápida.

Os *Guid's* que são utilizados até ao momento têm uma gama de aproximadamente 34×10^{37} valores, o que é um número muito superior ao número de artigos que é previsível que

existam nos próximos anos. Por outro lado, utilizando valores numéricos com uma precisão de 18 algarismos, a gama de valores possíveis é de 10^{18} valores, o que é um valor 420 vezes maior que o número de artigos existente actualmente na tabela. Os *int's* têm uma amplitude de valores de 10^{9} e os *smallint's* de 65.536. O tamanho dos Guid's é claramente exagerado tendo em conta as necessidades da empresa, ao passo que a amplitude dos *smallint* é já à partida insuficiente. Os valores *int* resolviam este problema a curto prazo, contudo devido ao aumento constante da quantidade de informação recolhida, a sua capacidade corria o risco de ser rapidamente atingida. A utilização de numéricos é portanto a opção mais adaptada à dimensão actual das tabelas deixando ainda muito espaço de manobra para que estas possam crescer a longo prazo. Podemos considerar, por isso, que esta opção é viável e que é a mais adaptada a este problema.

Para efeitos de comparação, estão na Tabela 2 registados os valores do espaço ocupado pelos dados e pelos índices das tabelas de maior dimensão e que por isso têm um maior peso na base de dados. Foi ainda simulada a redução do tamanho das chaves de 16 *bytes* para 9 *bytes* com o objectivo de estimar a redução do tamanho do espaço ocupado pelas tabelas. É assim apresentada, na coluna da direita na tabela, a percentagem de espaço esperada para as tabelas utilizando numéricos no lugar de *GUID's*. Esta tabela está ordenada pelo campo de espaço ocupado pelos dados.

Tabela 2: Número de registos e espaço ocupado pelas principais tabelas do CisionPoint

Nome da tabela	Número de registos	Tamanho de Dados (Kb)	Tamanho do Índice (Kb)	Cálculo de espaço (%)
Articles	10.150.695	50.654.008	16.596.336	99
Clippings	50.609.763	13.937.928	19.529.200	70
Observer MappingKeys	67.276.010	4.448.472	4.971.376	82
ClippingProfiles	66.706.516	2.997.808	15.888	48
ClippingTopics	66.140.730	2.972.360	15.864	48
FactArticle	2.676.898	2.859.512	14.916.424	89
ArticlesLucene Full	10.013.450	1.191.560	563.456	90
UserAccess ClippingHistory	8.992.536	818.904	2.184	61
Files	6.317.250	802.696	322.864	91
TableData	523.733	659.120	347.136	88
ProjectClippings	12.191.366	658.744	1.560.880	57
ClippingFiles	6.286.375	613.352	1.369.728	40

Tal como foi já dito antes, mais do que as tabelas, esta mudança de chaves vai afectar o tamanho dos índices. Nos índices do tipo *Clustered*[1] a redução de tamanho vai ser obviamente idêntica à redução do tamanho das tabelas visto que estes não têm uma estrutura própria mas são implementados na estrutura da tabela. Nos índices *Non-Clustered*[2] a redução vai ser muito acentuada visto que nestes são guardadas quase exclusivamente as chaves *Guid*. Por exemplo, o índice *IX_Clippings* da tabela *Clippings* que no princípio ocupava aproximadamente 4,5 Gb passará a ocupar cerca de 50% desse espaço.

Propriedades das chaves das tabelas e restrições de integridade

Nas chaves primárias das tabelas foi ainda adicionada a propriedade *Identity* que torna estas chaves em valores sequenciais atribuídos pelo motor de base de dados. Com esta propriedade pretende-se simplificar a criação de chaves primárias e a inserção de novos registos nas tabelas. Visto que as tabelas têm índices Clustered nas chaves primárias, sendo a chave primária sequencial, cada novo registo é inserido sempre no final da tabela, não sendo necessário reajustar a tabela ou o índice desta em cada inserção feita.

A criação das chaves estrangeiras foi também revista com o objectivo de implementar restrições de integridade referencial na base de dados. Com isto pretende-se evitar que existam referências na base de dados para dados que já foram apagados desta. Por exemplo, para evitar que exista um artigo na tabela *Articles* com o ID de um jornalista que não existe na tabela *Journalists*. Para resolver este problema foram criadas novas restrições em algumas tabelas e alteradas as restrições de chaves estrangeiras já existentes, para que a integridade referencial dos dados seja obrigatória. Este processo foi realizado tabela a tabela definindo para cada um dos campos chave estranjeira as restrições necessárias (propriedade *Check*).

Adaptação dos procedimentos e funções da base de dados às alterações realizadas

Com estas alterações na estrutura das tabelas, foi necessário adaptar os procedimentos e as funções presentes na base de dados. Por exemplo, todos os procedimentos que inseriam novos registos nas tabelas utilizavam a função *NEWID()* para criar uma nova chave para o registo (*Guid*) e inseriam esta juntamente com os dados na base de dados. Com as alterações efectuadas deixou de ser necessário criar uma nova chave manualmente. Passou simplesmente a ser necessário inserir os dados na base de dados, que trata da criação da chave necessária devido à propriedade *Identity*.

Foram ainda identificados alguns procedimentos com problemas de integração com a nova estrutura da base de dados ou que já não eram utilizados pela aplicação. Devido a este facto foram eliminados dos *scripts* de criação da base de dados. Os procedimentos eliminados estão listados no anexo A deste documento.

4.3. Motor de importação de dados

Como já foi explicado anteriormente, o CisionPoint é uma plataforma integrada que disponibiliza aos clientes da Cision vários dos seus serviços. O cerne destes serviços é a base

[1] Um índice clustered faz com que os dados de uma tabela sejam armazenados fisicamente ordenados por um determinado campo. Assim em vez de ser criada uma nova estrutura para guardar todos os dados do índice, é utilizada a própria estrutura de dados principal (Microsoft Corporation, 2011).

[2] Ao contrário dos índices clustered, os índices non-clustered são armazenados separados da estrutura de dados principal, fazendo com que seja ocupado muito mais espaço em disco (Microsoft Corporation, 2011; Microsoft Corporation, 2011).

de dados CisionPoint onde é guardada a informação que é disponibilizada aos utilizadores. Esta informação é proveniente de várias fontes, como por exemplo a imprensa escrita ou a internet, e chega à Cision em vários formatos desde papel a ficheiros *PDF*. Logicamente, esta informação não pode ser disponibilizada directamente no CisionPoint, precisa primeiro de passar por alguns processos para que seja normalizada. Por exemplo, um jornal que seja disponibilizado em papel é digitalizado para o formato *PDF*, são recortadas as várias notícias digitalmente que depois são interpretadas automaticamente de modo a serem associadas a tópicos e a clientes. A informação até este ponto foi sendo guardada numa base de dados a que é dado o nome de base de dados de produção. Quando o tratamento de uma notícia é dado como concluído, é necessário então que esta informação seja migrada para a base de dados do CisionPoint. Para isto é criado um ficheiro *XML* com toda a informação sobre a notícia. Este é depois lido por um motor de importação que tem a função de realizar a inserção da informação na base de dados do CisionPoint para que fique disponível aos clientes. Este fluxo de informação está resumido na ilustração seguinte.

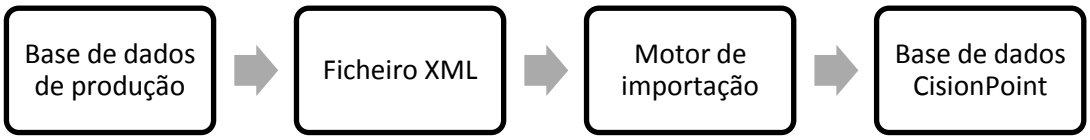

Ilustração 1: Caminho seguido pela informação até chegar ao CisionPoint

Depois de criada uma nova estrutura para a base de dados, tal como explicado no tópico anterior deste documento, foi necessário testar as alterações feitas com o objectivo de verificar se as alterações produziam o efeito esperado no tamanho das bases de dados e na sua performance ou se não eram suficientes e necessitavam de ser repensadas.

Para a realização destes testes foi necessário logicamente que existissem dados nas tabelas, para que o espaço ocupado por estas pudesse ser avaliado. Apesar de o método mais rápido para popular a base de dados fosse copiar as tabelas uma a uma directamente de uma base de dados já existente, faz parte dos processos da empresa a utilização do motor de importação de dados já explicado. Como tal, este poderia ser usado para a população pretendida da base de dados. Este seria o processo natural de inserção de informação na base de dados CisionPoint.

Contudo, dadas as alterações já realizadas na base de dados, para que o motor pudesse ser usado, teria que ser adaptado para a nova estrutura da base de dados. Esta era uma adaptação algo trabalhosa, que na fase inicial do projecto era perfeitamente dispensável, visto que os dados poderiam ser copiados manualmente. Contudo a longo prazo e mantendo-se as alterações realizadas, teria que ser feita para que o sistema funcione com normalidade. Foi, por esta razão, decidido realizar já este passo da implementação do trabalho.

Houve genericamente duas alterações na base de dados que fizeram com que o motor de importação existente necessitasse de ser adaptado. As chaves das tabelas deixaram de ser do tipo *uniqueidentifier* para serem do tipo *numeric* e as chaves primárias deixaram de ser criadas pelo motor de importação para serem criadas pela base de dados (chaves passaram a ser incrementais).

Isto fez com que o motor de importação tivesse que ser alterado em toda a sua extensão, desde o *parsing* do ficheiro *XML* até aos procedimentos que inserem a informação na base de dados CisionPoint. As alterações realizadas são descritas nos tópicos seguintes.

Alteração dos identificadores da informação no motor de importação

O motor de importação, à semelhança de todo o projecto está construído para funcionar com uma base de dados em que as chaves das tabelas são do tipo *uniqueidentifier*. Visto que a ferramenta *.Net* não inclui este tipo de dados, é utilizada uma classe de nome Guid (abreviatura de *Global Unique Identifier*) para representar as variáveis deste tipo. Uma vez que a nova base de dados deixou de ter *uniqueidentifier's* que foram substituídos por valores numéricos, no motor de importação a utilização da classe *Guid* deixou de fazer sentido. Foram então substituídas as instâncias da classe *Guid* por variáveis numéricas do tipo *long*. Esta alteração era indispensável dada a incompatibilidade natural destes dois tipos de dados.

Quando é gerado um novo ficheiro *XML* pela produção, o motor de importação faz o *parsing* desse ficheiro e coloca a sua informação em objectos *.Net* com estrutura idêntica aos registos das várias tabelas da base de dados. Por exemplo, cada *XML* de um artigo tem a informação deste artigo, do respectivo *clipping*, dos autores do artigo, dos tópicos a que o artigo está associado e da restante informação do artigo. Com essa informação é então criada uma instância da classe *ArticleData*, que representa o artigo, e respectivas instancias para as classes *ClippingData*, *JournalistData* e *TopicData* que representam o *clipping* do artigo, autores e tópicos. O mesmo acontece para o resto da informação contida no ficheiro *XML* original.

Visto que os ficheiros *XML* pretendem representar registos da base de dados, à sua semelhança tinham também identificadores do tipo *Guid*. Estes foram alterados para o tipo *long* para que não existisse incompatibilidade de dados.

Alteração da forma de atribuição de chaves primárias aos registos da base de dados

Na base de dados original as chaves primárias eram *Guid's* que eram criadas fora da base de dados e só depois eram inseridos nesta ao mesmo tempo que os restantes dados do registo que identificam.

O motor de importação, quando fazia o *parsing* de um elemento do ficheiro *XML*, se esse elemento ainda não existisse na base de dados, criava uma chave primária aleatória para ele. Essa chave era posteriormente inserida na base de dados como chave primária do registo inserido.

Na base de dados CisionPoint24, tal como já foi dito, as chaves primárias passaram a ser definidas incrementalmente pela base de dados. Quando é inserido um novo registo numa tabela o motor de base de dados atribui um ID ao registo que devolve à aplicação que inseriu o registo.

Dado que, com as alterações feitas, os identificadores só são atribuídos à informação quando esta é inserida na base de dados, para que não existam falhas nas referências dos registos inseridos na base de dados entre o *parsing* e a inserção nas tabelas, a ordem pela qual a informação é lida do ficheiro *XML* e pela qual é inserida na base de dados teve que ser alterada.

Por exemplo, um *clipping* passou a só poder ser inserido na tabela *Clippings* depois de inserido o respectivo artigo/editorial, porque só neste momento é que o motor de importação conhece o seu identificador, valor que é necessário para preencher o campo *articleid/editorialid* da tabela *Clippings*.

Também devido a estas alterações o *parsing* do *XML* teve que ser modificado. Deixaram de ser criadas neste as chaves aleatórias para serem inseridas na base de dados. Estes campos

passaram a ser preenchidos com o valor *"-1"* até ao momento em que a informação que representam seja inserida na base de dados.

Adaptação dos procedimentos utilizados pelo motor de importação

Como acontece com o motor de importação, os procedimentos que este usa para inserir informação na base de dados estavam preparados para tratar os identificadores das tabelas como *uniqueidentifier* recebendo-os por parâmetro. Tiveram por isso que ser adaptados à nova realidade do sistema. Os parâmetros de entrada e saída destes procedimentos que eram do tipo *uniqueidentifier* passaram também a ser do tipo *numeric*, bem como todas as variáveis utilizadas deste tipo.

Estes procedimentos têm praticamente todos a mesma estrutura e cada um deles insere um ou vários registos numa dada tabela da base de dados. Começam por verificar se já existe algum registo nessa tabela com a informação que desejamos inserir. Na versão anterior do motor de importação esta verificação era feita com base nas chaves primárias das tabelas. Na nova versão do motor, visto que estas chaves ainda não estão atribuídas no momento desta verificação os procedimentos foram alterados para passarem a identificar a informação pelo seu identificador na base de dados de produção.

Por exemplo, para verificarmos se um artigo já existe na base de dados passou a ser pesquisado um registo na tabela *Articles* com o campo *oldKey* (campo com o identificador do artigo na base de dados de produção) igual ao identificador de produção que é passado por parâmetro ao procedimento.

Visto que o motor de importação precisa dos identificadores dos dados que insere na base de dados para os referenciar noutras instâncias, os procedimentos que invoca para fazer essas inserções passaram a ter que devolver esses mesmos identificadores, recém criados, ao motor de importação.

Por exemplo, um *media* é inserido e o seu novo identificador é devolvido. É então preenchido o campo *mediaId*, que até ao momento tinha o valor *"-1"*, no objecto *mediaInstance* associado a este *media*. De seguida, quando o *mediaInstance* é inserido na base de dados já tem a referência correcta para o *media* respectivo. Se esta atribuição não fosse realizada, era gerada uma excepção na base de dados no momento da inserção do *mediainstance* na base de dados por violar a integridade referencial desta ao tentar inserir um valor no campo *media* que não existe na tabela de *medias*.

4.4. Adaptação do portal CisionPoint

Logicamente, dadas as alterações feitas na estrutura da plataforma, o modo como o portal é construído para ser mostrado ao utilizador teve também que ser adaptado.

Para além das páginas disponibilizadas a todos os utilizadores registados o CisionPoint possui ainda várias zonas que são disponibilizadas apenas a utilizadores privilegiados, como administradores e editores de notícias, em que é feita toda a configuração dos serviços prestados aos clientes e da informação que é mostrada aos seus utilizadores. Por este motivo foram definidos os vários módulos em que se divide o portal que foram abordados um a um e que estão listados na Tabela 3.

A adaptação do portal consistiu principalmente na alteração do tipo de dados que é recebido da base de dados e na adaptação dos procedimentos da base de dados que são chamados para a construção do portal.

Este processo não teve nunca como objectivo alterar qualquer funcionalidade presente na versão anterior mas sim fazer com que o sistema, tendo uma camada de dados diferente, mostrasse os resultados aos seus utilizadores da mesma forma que já fazia antes. Obviamente alguns erros e possíveis optimizações detectados foram sendo corrigidos e implementados.

Testes funcionais

A par da adaptação das várias secções do portal e com o objectivo de a apoiar foram sendo realizados testes funcionais nestas secções com o objectivo de validar gradualmente as alterações feitas.

No fim da adaptação do portal estar completa foi realizado um teste de integração a todo o portal com o objectivo de identificar problemas funcionais do sistema que ainda não tenham sido encontrados. Os principais problemas encontrados neste teste estavam relacionados com funcionalidades partilhadas por diferentes secções do portal e em que ao serem realizadas alterações foram provocados erros em secções diferentes. Este teste estendeu-se por quase todas as secções do portal. De fora desta fase de adaptação inicial e de testes ficaram as secções relacionadas com a realização de pesquisas e a utilização dos filtros de notícias na página principal do portal. Estas foram testadas posteriormente tal como será explicado no capítulo seguinte deste documento.

Tabela 3: Secções do portal CisionPoint

Tipo de utilizador	Secção do portal
Gestor de companhias	Gestão de clientes Preferências de companhia
Gestor de clientes	Gestão de utilizadores Configuração de notificações Gestão de conteúdos Monitorização
Utilizador simples	Página *default* Imprimir *PressBook* *E-mail* *Newsletter* *RSS* *AudioClipp* Pesquisa avançada Tradução Detalhes de artigos
Utilizador editor	Gestão de notícias Gestão de *media* Administração de conteúdos

4.5. Índice Solr e pesquisas

Lucene

No portal CisionPoint é possível, do ponto de vista do utilizador, realizar dois tipos de pesquisas de notícias. Uma pesquisa simples que pesquisa um determinado texto no título e

no corpo das notícias do cliente e uma pesquisa avançada em que podem ser pesquisadas notícias segundo vários parâmetros como a data de publicação e de produção ou o nome do seu autor.

Para que estas pesquisas fossem realizadas de uma forma mais célere e para que estas não afectassem os restantes acessos à base de dados era utilizada a biblioteca *Lucene*. Esta biblioteca é direccionada para a realização de pesquisas através da construção de um índice com toda a informação necessária à pesquisa e que está optimizado para esta tarefa. No sistema encontrado no início deste estágio as pesquisas feitas neste índice devolviam à aplicação os identificadores dos *clippings* resultantes da pesquisa. Estes *ID's* eram depois usados pela aplicação para retirar directamente da base de dados, a informação necessária para o portal CisionPoint. Isto retirava o peso da pesquisa propriamente dita da base de dados, mas a informação continuava a ser servida por esta.

Os índices criados pela biblioteca Lucene, como foi descrito no *Estado da Arte,* indexam genericamente documentos, que podem ter vários campos definidos. Cada um desses campos tem várias propriedades que o caracterizam e que caracterizam a sua utilização. No caso do índice utilizado no CisionPoint cada documento do índice representa um artigo ou um editorial e tem os campos que estão listados na tabela abaixo.

Tabela 4: Campos existentes no índice Lucene existente no CisionPoint

Nome do campo	Descrição
ArticleLuceneId	Identificador do artigo/editorial no *Lucene*
ArticleId	Identificador do artigo/editorial na base de dados
Headline	Título do artigo/editorial
BodyText	Texto do artigo/editorial
CustomerIds	Lista dos clientes que visualização este artigo
ProductionDate	Data de tratamento da notícia no departamento de produção
PublicationDate	Data de publicação da notícia
MediaId	Identificador do Media onde o artigo foi publicado
MediaTypeId	Identificador do tipo do *media* do artigo
TopicIds	Lista dos tópicos a que este artigo está associado
ClippingIds	Lista dos *clippings* a que o artigo está associado
CodeSearchType	
MediaName	Nome do Media onde o artigo foi publicado
MediaTypeName	Nome do tipo do Media do artigo
JournalistName	Nome do jornalista que publicou a notícia
Country	País em que o artigo foi publicado
CategoryId	Identificador da categoria em que o artigo se insere

Este índice era povoado a partir da base de dados CisionPoint por uma aplicação que corria várias vezes por hora. Assim os artigos ficavam disponíveis para serem pesquisados pouco tempo depois de entrarem no CisionPoint.

Dadas as alterações realizadas na estrutura da base de dados da plataforma, o índice existente ficou inviável. Em primeiro lugar com a alteração do tipo de dados de campos da base de dados a maioria dos campos do índice deixaram de ser compatíveis com a informação proveniente da base de dados. Depois a reestruturação realizada na base de dados, fez com que alguns dos campos do índice deixassem de fazer sentido por já não existir na base de dados a informação que estes guardam.

Por estes motivos e à semelhança dos vários módulos do CisionPoint também a aplicação que realiza a indexação e todo o índice Lucene ficaram desajustados na nova versão do CisionPoint.

SolR

Dadas as vantagens que o SolR apresenta em relação ao Lucene apresentadas no capítulo *Estado da Arte* e o facto de a estrutura existente ter que ser praticamente toda ajustada, foi decidido pela equipa CisionPoint que a nova versão da plataforma utilizasse o SolR e não apenas o Lucene.

Ao contrário do Lucene, no SolR é definida previamente a estrutura que os documentos indexados têm através de um *schema*. Neste *schema* é definido cada um dos campos do documento bem como as suas propriedades. Estas propriedades incluem o nome do campo, o *tokenizer* e o *analyzer* a que a informação do campo são sujeitos quando indexados e pesquisados respectivamente, o tipo de informação que o campo guarda, a cardinalidade do campo, entre outras.

Como já foi dito, o índice existente no CisionPoint apenas devolvia à aplicação os identificadores dos *Clippings* que resultavam da pesquisa feita e cabia à aplicação tirar a informação sobre cada *clipping* da base de dados. Este método era utilizado devido à inexistência de uma ferramenta de paginação no Lucene e como tal, o resultado das pesquisas realizadas podiam ter um volume demasiado grande para serem devolvidos de uma só vez. Tendo já o SolR suporte para paginação este passo pode ser simplificado e a informação pode ser enviada directamente do índice para o portal. Este facto foi tido em conta também na definição do novo índice, principalmente na definição de qual a informação que teria que ser apenas indexada ou que teria também que ser guardada no índice de forma a ser disponibilizada.

Como base para a construção deste índice, foram então definidos os campos que o índice devia conter bem como as respectivas propriedades e foi criado o ficheiro *schema.xml* de acordo com esta definição. Partindo da análise dos campos que existiam foram então definidos os seguintes campos para o novo índice.

Tabela 5: Campos existentes no índice SolR criado para a nova versão do CisionPoint

Nome do campo	Descrição
Id	Identificador incremental do artigo no SolR
text	Texto com o título e o corpo do texto para a realização da pesquisa
ArticleId	Identificador do artigo na base de dados
Headline	Título do artigo/editorial
Bodytext	Texto do artigo/editorial
URL	*URL* do artigo, caso este seja um artigo *Web*
AEV	Valor monetário calculado para o artigo
IsBodyTextCorrected	*Flag* que indica se o texto já está corrigido manualmente
MediaId	Identificador do *media* onde o artigo foi publicado
MediaName	Nome do *media* onde o artigo foi publicado
MediaTypeId	Identificador do tipo do *media* do artigo
MediaTypeName	Nome do tipo do *media* do artigo
Journalist	Nome do jornalista que publicou a notícia
ClippingId	Lista dos *clippings* a que o artigo está associado
IDA	Identificador do artigo na base de dados de produção
CodSearchType	Código de pesquisa atribuído ao artigo
Country	País em que o artigo foi publicado
CategoryId	Identificador da categoria em que o artigo se insere
CategoryName	Nome da categoria em que o artigo se insere
PublicationDate	Data de publicação da notícia
ProdutionDate	Data de tratamento da notícia no departamento de produção
CreationDate	Data de importação da notícia para o CisionPoint
ClippingsCustomer	Lista com os *ID's* dos *Clippings* e respectivos clientes
Customer	Lista dos clientes que recebem este artigo
PT_Show	Lista dos clientes a quem este artigo é mostrado
Topic	Lista dos tópicos a que este artigo está associado
TopicsCustomer	Lista que associa os tópicos do artigo aos clientes
Comment	Lista dos comentários feitos ao artigo

CommentsCustomer	Lista que associa os comentários aos clientes que os realizaram
Folder	Lista das pastas em que o artigo está
UserFolder	Lista que associa as pastas aos utilizadores que as criaram
EditorialId	Identificador do editorial na base de dados
EditorialClippingId	Lista que associa o editorial aos respectivos *clippings*
FileId	Lista com os identificadores dos ficheiros
File	Lista com os caminhos para os ficheiros do artigo

De acordo com esta estrutura e com o sistema já existente foram criadas duas aplicações distintas. Uma aplicação que realiza a indexação dos novos artigos do CisionPoint e um Web Service que será utilizado pelo portal CisionPoint para realizar as pesquisas no índice e devolver os respectivos resultados para que sejam mostrados. A integração de ambas as aplicações com o SolR foi feita utilizando a *API SolrNet*.

Devido às diferenças no processo de indexação entre o Lucene e o Solr a primeira aplicação foi construída de raiz baseada na lista de campos descritos em cima na tabela. Esta consiste na pesquisa na base de dados de artigos que ainda não tenham sido indexados (identificados pelo campo *indexed* da tabela *Articles)* e preenchimento de um objecto previamente definido com a informação desses artigos. Este objecto previamente definido tem uma estrutura idêntica à estrutura dos documentos do índice SolR, o que permite que, depois de completamente preenchido, a API SolrNet consiga indexar directamente esta informação, passando o artigo assim a estar disponível nas pesquisas feitas no índice.

No caso da segunda aplicação foi possível realizar a adaptação da estrutura que já existia, devido às semelhanças existentes entre a sintaxe das *querys* Lucene e SolR. Na construção das querys foram apenas necessárias algumas alterações relacionadas com as alterações feitas aos tipos de dados e pequenas alterações relacionadas com diferenças entre a sintaxe dos dois motores de pesquisa. Foi também necessário mudar a forma como os resultados são devolvidos, dado que na nova versão já não são devolvidos apenas os identificadores dos artigos mas sim toda a informação destes.

Pesquisas

Criado o índice SolR foi então possível acabar a adaptação do portal à nova estrutura de dados. Para isto foi apenas necessário adaptar as páginas em que são realizadas as pesquisas para utilizarem o novo *Web Service* referido em cima. Este processo já era realizado anteriormente através de um *Web Service* que realizava as pesquisas no índice Lucene, contudo como já foi descrito, o portal deixou de receber do índice apenas os identificadores dos artigos resultantes, mas toda a informação destes. Assim deixaram de ser necessários acessos posteriores à base de dados para obter esta informação, sendo assim agilizado este processo. Assim do ponto de vista do portal foi apenas necessário adaptar a estrutura existente para os requisitos do serviço e para tratar de modo diferente os resultados deste.

4.6. Implementação da estrutura de histórico

Um dos objectivos deste estágio era a criação de uma estrutura de dados que separasse a informação que é acedida mais frequentemente, ou seja, as noticias mais recentes, da informação que por ser mais antiga é acedida com menos frequência. O objectivo desta

alteração era o de, retirando a maior parte da informação da estrutura de dados principal, facilitar e acelerar o acesso à informação mais recente ainda que para isso a performance de pesquisas em informação mais antiga seja afectada.

No decorrer deste estágio as regras do negócio foram alteradas, deixando de ser uma necessidade a disponibilização, no CisionPoint, de notícias com mais de treze (13) meses. Dada esta alteração, foi decidido pela equipa do CisionPoint que deveriam existir duas estruturas de dados independentes, uma que servia as notícias mais actuais, dos últimos 13 meses, ao CisionPoint e outra que teria todas as notícias armazenadas até à actualidade e em que o tempo de pesquisas deixaria de ser um factor crítico. Estas duas estruturas de dados serão referidas neste capítulo e por razões de simplicidade, como estrutura de dados do CisionPoint e estrutura de dados de Histórico.

Dados do CisionPoint

Esta estrutura é constituída por três elementos distintos:

- Portal CisionPoint;
- Base de dados CisionPoint;
- Índice SolR com os artigos da base de dados em cima.

Quando um utilizador navega no portal é pedida a informação de que necessita ao servidor Web onde se encontra instalado o portal que por sua vez pesquisa informação na base de dados ou no índice SolR. O diagrama seguinte ilustra a organização destes elementos.

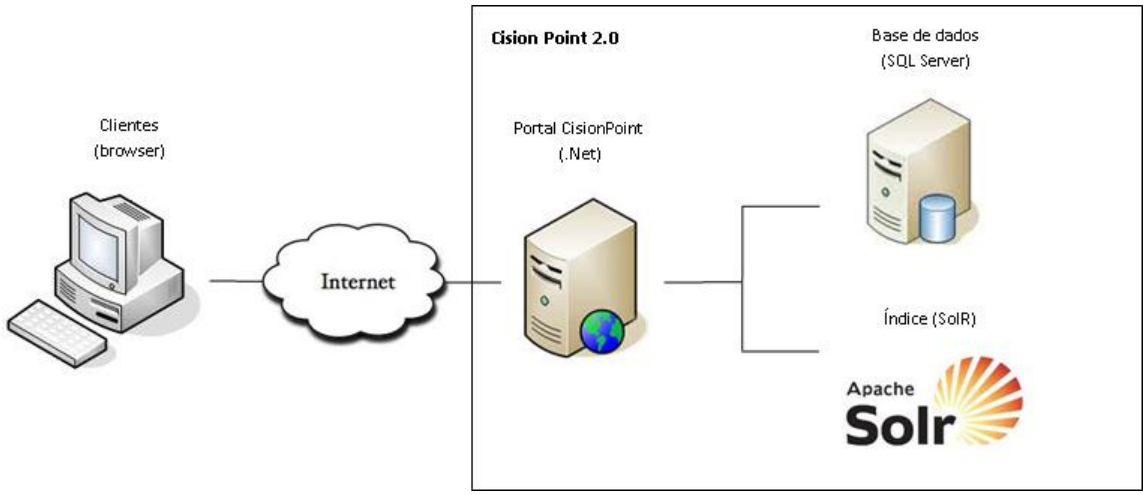

Ilustração 2: Estrutura de dados do CisionPoint

Tanto na base de dados como no índice *Sol*R apenas estará informação dos últimos 13 meses. Diariamente, tal como foi já explicado no capítulo sobre importação, os artigos são importados da produção para esta base de dados, de onde são depois indexados no *Sol*R. Periodicamente, de semana a semana ou mensalmente, são eliminados da base de dados e do índice os artigos que passem a ter uma idade superior a 13 meses, para que esta estrutura mantenha sempre um tamanho controlado e relativamente pequeno.

Mantendo o tamanho da base de dados, mantem-se também controlado o tempo de duração dos planos de manutenção da base de dados e da realização de *backups* da informação. Também por ser uma quantidade de informação muito mais pequena estes processos são realizados muito mais rapidamente minimizando o tempo em que estes afectam a disponibilidade do sistema.

Dados de Histórico

A par da estrutura descrita antes, para que seja disponibilizada a informação sobre os artigos com mais de 13 meses foi construída uma nova estrutura de dados.

A pensar nisto foi desenvolvida por outros elementos da equipa a aplicação *CP Desktop* que, à semelhança do portal CisionPoint, utiliza a base de dados o4k (existente no início deste estágio) para obter a informação necessária para pesquisar e mostrar notícias e para criar *PressBooks* e *NewsLetters* para os clientes. Esta aplicação seria instalada do lado dos clientes da Cision e colocaria algumas das actividades que implicam processamento e ocupação de espaço em disco sob a responsabilidade dos clientes. Esta aplicação tem como alvo o sistema que existe actualmente em produção, e não a nova versão do CisionPoint.

A base de dados a que esta aplicação acede tem, à semelhança da existente actualmente em produção, todos os artigos que já foram importados para o CisionPoint, e obviamente os mesmos problemas de performance.

Por este motivo, no seguimento do trabalho realizado no CisionPoint e utilizando as soluções apresentadas no estado da arte deste documento em relação ao particionamento de informação, foram implementadas na estrutura de histórico as soluções que inicialmente foram pensadas para a estrutura principal do CisionPoint.

Neste sentido, para que esta estrutura de dados fosse escalável foi então decidido particionar a informação em vários índices SolR, estruturalmente idênticos ao criado para a nova versão do CisionPoint, e a partir dos quais passariam a ser realizadas todas as pesquisas.

Esta decisão deveu-se a vários factores. Em primeiro lugar, a criação de vários índices distintos faz com que a informação seja particionada de uma forma simples e isolada. Tendo vários índices distintos, mesmo que algum problema ocorra num desses índices, os restantes continuam a funcionar plenamente e a reposição de um backup do índice danificado é realizada mais rapidamente. Em segundo lugar dado que nesta estrutura, salvo raras excepções, não se pretende apagar qualquer informação mas sim inserir e pesquisar artigos pelos seus campos, um índice SolR é uma opção bem mais adaptada que uma base de dados *SQL Server* dadas as suas características mencionadas no Capítulo 2 Estado da Arte. Por fim, desta forma o tamanho da informação em que as pesquisas são feitas seria limitado e apesar de o tempo de resposta poder ser maior em alguns casos em média os resultados seriam substancialmente melhores porque com o aumento dos dados aumenta também o número de índices e não a dimensão destes, evitando a degradação da performance destes. Inicialmente o objectivo é que exista um índice por ano, mas com o tempo e experiência este intervalo poderá e deverá ser ajustado.

Do ponto de vista prático esta solução é muito semelhante à estrutura do CisionPoint, o que facilitou o seu desenvolvimento. Nesta estrutura os artigos estão inicialmente numa base de dados de onde são indexados. São depois pesquisados através de um *Web Service* que constrói e executa a *query* SolR semelhante ao existente na estrutura do CisionPoint. Nesta nova abordagem, dado que a indexação e pesquisa deixam de ser feitas apenas num índice mas sim em vários foi apenas necessário adaptar a aplicação de indexação e o *Web Service* de pesquisas já existentes do CisionPoint para este novo facto de forma a seleccionarem o(s) índice(s) onde pretendem indexar ou pesquisar os artigos. A organização destes elementos está esquematizada no diagrama seguinte.

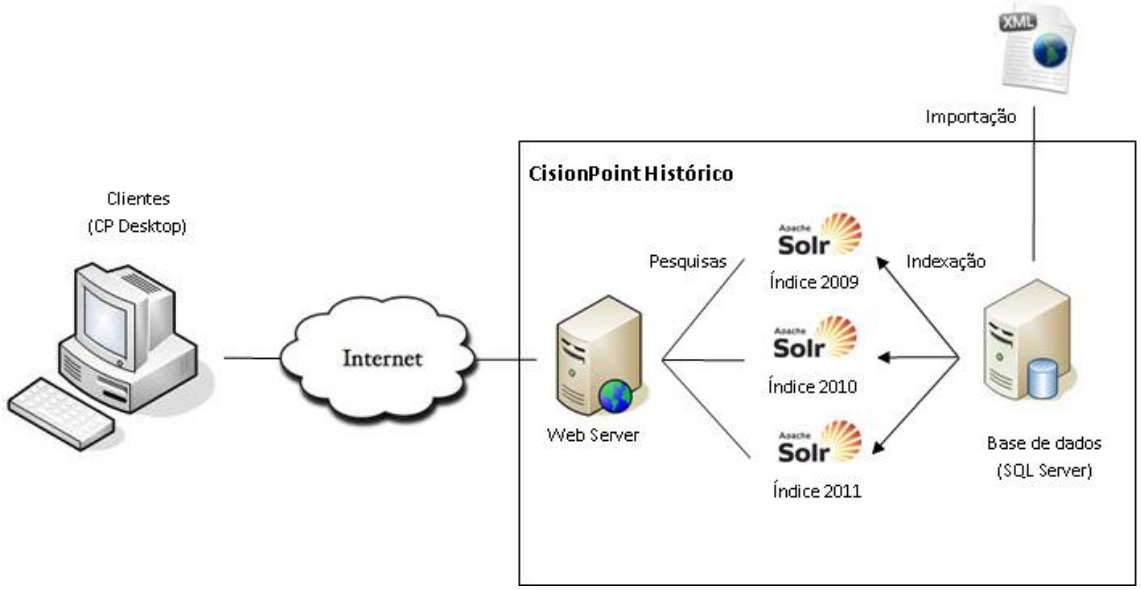

Ilustração 3: Estrutura de dados de histórico

Nesta abordagem ao contrário do que acontece no CisionPoint, a base de dados não é utilizada no funcionamento do *CP Desktop*, servindo apenas de base da indexação de artigos e de *backup* para o caso de ocorrer algum problema com um ou vários índices.

No âmbito deste estágio não foram realizadas quaisquer alterações à aplicação *CP Desktop*, foram apenas desenvolvidas as ferramentas para que os artigos presentes possam ser indexados de forma particionada e para que possam ser realizadas pesquisas nos vários índices (aplicação de indexação e *Web Service* de pesquisas referidos acima). Simplificando, no âmbito do estágio foram apenas desenvolvidas as funcionalidades contidas no rectângulo da Ilustração 3.

4.7. Alteração do plano de *backup* da base de dados

Tendo por base o estudo realizado na elaboração do capítulo *Estado da Arte* foi identificada uma forma de reduzir a duração da realização diária de *backups*. No sistema existente em produção estava a ser realizado todos os dias um *backup* completo da base de dados CisionPoint, o que é logicamente desnecessário. Depois de identificado este problema foi de imediato corrigido no sistema existente em produção reduzindo consideravelmente o tempo de realização dos *backups* diários. Para que não fosse realizado o *backup* completo dos dados diariamente e para que não exista o risco de serem perdidos dados críticos no caso de ocorrer uma falha no sistema de armazenamento, passou a ser feito apenas um *backup* diferencial diário durante a semana, backup este que é realizado num período muito menor do que acontecia até agora. Ao fim de semana é depois realizado um *backup* integral e eliminados os vários *backups* que tenham sido feitos durante a última semana.

Esta alteração no sistema existente em produção, devido às regras de segurança existentes na empresa e criticidade da informação envolvida, não foi realizada pelo estagiário directamente.

4.8. Migração da plataforma

A migração da plataforma tem duas vertentes, migração das bases de dados *SQL Server* da versão 2000 para a versão 2008 e migração da aplicação da *framework .Net* 1.1 para a versão 4.0.

Uma vez que a base de dados foi criada de raiz, a migração das bases de dados para a versão 2008 foi praticamente transparente. Já a migração da *framework .Net* necessitou de mais atenção. A solução do projecto que estava implementada no *Microsoft Visual Studio 2003* necessitou de ser convertida para que pudesse ser aberta na versão 2010 deste *IDE*. Foi ainda necessário alterar algumas referências dos projectos para colocar o sistema de novo funcional.

Esta migração permite ao estagiário utilizar as ferramentas mais actuais e melhor capacitadas para a realização do projecto e permite também ao CisionPoint tirar partido das melhorias implementadas nestas versões mais recentes do *.Net* e do *SQL Server*.

Logicamente estas vantagens vão repercutir-se também em futuros desenvolvimentos realizados no CisionPoint pela sua equipa.

Capítulo 5
Descrição dos testes realizados

Depois de implementadas as alterações planeadas tornou-se necessária a realização de testes com o objectivo de aferir as melhorias obtidas. Por esta razão e para que as principais alterações realizadas fossem abarcadas por estes testes foram então realizados três tipos de testes distintos:

- Testes ao tempo de importação de artigos;
- Testes à navegação no portal;
- Testes ao espaço ocupado pela informação nas bases de dados;

Logicamente para que pudessem ser avaliadas as melhorias conseguidas no sistema foi necessário utilizar um termo de comparação. Assim a par do sistema que foi estabelecido para a realização dos testes à nova versão da plataforma, foi criado um sistema com a versão existente em produção do CisionPoint, ou seja, com as versões do portal, base de dados e índices existentes no início deste estágio.

Os testes realizados estão descritos nas subsecções deste capítulo.

5.1. Comparação do tempo de importação

Em primeiro lugar foram realizados testes com o objectivo de medir o tempo necessário para importar um novo artigo para o CisionPoint. Para a realização destes testes foram usados dois servidores com características diferentes. Um servidor de base de dados e um servidor *Web*. Estes servidores serão a partir daqui, por razões de simplicidade, referidos como *DBserver* e *WebServer* respectivamente.

As características destas máquinas são as seguintes:

- ***WebServer* (servidor *Web*)**
  - *Microsoft Windows Server 2008 Standard Edition com Service Pack 2*
  - 4 Processadores *Intel Xeon MP*, a correr cada um a 1.9 *GHz* com 8 *Gb* de *RAM*
  - *Internet Information Services 7.0*
- ***DB server* (servidor de base de dados)**
  - *Microsoft Windows Server 2003 Standard Edition com Service Pack 2*
  - 4 Processadores *Intel Xeon*, a correr cada um a 3.0 *GHz* com 2 *Gb* de *RAM*
  - *Microsoft SQL Server 2008 Developer Edition*

Ambiente inicial dos testes

É pretendido que estes testes sejam realizados num ambiente tanto quanto possível, igual ao ambiente em que a base de dados estará sujeita quando for colocada em produção. Além disso é necessário que o ambiente em que os testes são realizados seja igual para todos os testes realizados em ambas as bases de dados, para que os seus resultados sejam comparáveis. Como tal, procurou-se definir de forma esquemática o estado do sistema em que os testes seriam iniciados, para que este mesmo estado pudesse ser reestabelecido no início de cada realização destes testes.

Os testes foram, para ambas as bases de dados, realizados nos servidores descritos acima que trabalharam exclusivamente neste processo.

No ambiente de produção, quando os artigos são importados para a base de dados, esta contem já uma grande quantidade de artigos, o que faz com que tenha comportamentos diferentes do que se estivesse vazia. Por outro lado, dada a elevada quantidade de artigos existentes em produção e o facto de este ser um sistema crítico, foi impossível copiar toda esta informação para a nova versão. Por esta razão, ambas as bases de dados foram populadas inicialmente com os mesmos 386.850 artigos associados a 987.720 *clippings*. Sendo o objectivo destes artigos criar algum peso na base de dados este foi o conjunto maior de artigos que foi possível agrupar para a realização de testes sem que qualquer sistema de produção fosse afectado.

Como forma de simular não só a quantidade, mas também a variedade de artigos existentes no ambiente de produção, procurou-se obter todos os artigos correspondentes a um período de tempo consecutivo e que abarcassem os ciclos temporais que afectam a produção de notícias, como o ciclo semanal e mensal. Por outro lado, o processo de criação de ficheiros para a importação é um processo critico em termos de performance, pois só pode ser realizado na base de dados usada pelo departamento de produção da empresa.

Por este motivo, e para evitar prejudicar os processos normais da empresa foi utilizado um conjunto de ficheiros que estavam já criados. Estes ficheiros correspondiam a um período alargado de tempo. Dado que o objectivo era o de encontrar uma amostra que representasse a totalidade dos artigos importados num intervalo de tempo, estes ficheiros foram analisados e concluiu-se que os artigos dos meses de Fevereiro e Março cumpriam este requisito. Foram escolhidos por isto os artigos deste intervalo para a realização dos testes.

Assim é ainda assegurado por exemplo que a proporção de artigos dos vários tipos de *media* é realista. Ou seja, que na base de dados de teste existem por exemplo muito mais artigos de Internet do que artigos de Rádio tal como acontece na realidade.

Descrição dos testes

Este teste consistiu na importação de um conjunto de artigos seleccionados de acordo com a sua data de produção. Ou seja, foi seleccionado um dia em que foi produzida uma quantidade de artigos acima da média. Foram então utilizados todos os artigos produzidos nesse dia. O dia escolhido foi o dia 26 de Abril de 2011 que por ser o último dia de um fim-de-semana prolongado teve uma quantidade de notícias ligeiramente superior ao habitual, por reunir, além dos artigos do dia, aqueles que dada a sua baixa urgência, não foram tratados durante o fim-de-semana e se acumularam durante este período. Escolhendo um dia com um número de artigos superior ao normal por um lado é testado um caso pior que a média mas que acontece na realidade, por outro lado com o aumento de notícias recolhidas é esperado que este caso excepcional venha a ser habitual daqui a algum tempo e portanto uma opção desejável.

Neste teste foram então importados 21.258 ficheiros *XML*, correspondentes a 19.553 artigos diferentes e registados os tempos de importação de cada um destes.

Apesar desta selecção, os artigos escolhidos foram depois analisados qualitativamente de forma informal segundo vários critérios com o objectivo de compreender se esta selecção se enquadra no perfil de artigos que é importado diariamente. Alguns desses critérios foram por exemplo o tipo de *media* em que o artigo foi publicado ou a quantidade de tópicos que um artigo possui.

Os resultados desta análise enquadraram os artigos no perfil habitual dos artigos existentes diariamente a entrar no CisionPoint. Apesar de terem sido produzidos mais artigos neste dia, a proporção dos tipos de *media* a que estes correspondem é idêntica à que se tem registado diariamente e não se registou nenhum factor extraordinário para que a qualidade das noticias pudesse ser de alguma forma anormal.

Como forma de simular o ambiente de produção, em que os artigos são importados continuamente e em que a utilização de cache é natural, foi considerado um período de *Warm-Up* de 100 artigos cujos resultados foram ignorados. Depois de alguns testes de importação, os valores registados foram analisados manualmente. Nos resultados analisados dos vários testes feitos, apenas os primeiros 10 a 15 resultados de cada teste mostravam valores muito diferentes da média dos resultados obtidos. Contudo, dado que o número de artigos era muito mais elevado considerou-se que ignorar 100 destes resultados não seria relevante para os resultados obtidos e que assim ficaria coberta alguma anomalia que pudesse ocorrer na inicialização dos testes. Assim foi estabelecido este valor como período de *Warm-Up*.

5.2. Comparação da performance do portal CisionPoint

Nesta fase dos testes pretende-se testar a performance dos acessos ao portal CisionPoint simulando o acesso de vários utilizadores em simultâneo a várias funcionalidades do portal tal como acontece no dia-a-dia da plataforma.

Este processo foi realizado em vários passos independentes descritos nas secções seguintes.

Escolha da ferramenta de testes

Existem no mercado várias ferramentas que permitem a realização deste tipo de testes. A generalidade destas ferramentas utiliza a mesma abordagem para realizar os testes. Esta consiste em, numa primeira fase, gravar a navegação de um utilizador no portal. Com esta gravação é possível então que o processo seja repetido automaticamente. Este método faz com que seja possível criar testes realistas em todas as funcionalidades acedidas pelos utilizadores no portal de uma forma simples e rápida.

Após alguma pesquisa, foi escolhida para a realização desta simulação a ferramenta de testes disponibilizada pelo *Microsoft Visual Studio 2010* versão *Ultimate* por ser menos dispendiosa (versão *trial*) e por ter todas as capacidades necessárias para a realização dos testes pretendidos.

Requisitos de testes

Os testes aqui apresentados são o culminar de uma série de testes exploratórios. Os valores apresentados nos parâmetros dos testes são o resultado da repetição e afinação de cada um dos testes. Por um lado era pretendido que os testes tivessem uma carga considerável no sistema, por outro lado, esta carga não poderia ser de tal modo elevada ao ponto de provocar um número elevado de erros na execução dos testes (por exemplo por ser excedido o valor de *timeout* dos pedidos, ou por o sistema não conseguir construir as repostas devidamente) ao ponto de a maiorias destes falharem. Para que os valores apresentados fossem encontrados foram realizados vários testes preliminares. Nestes foram monitorizadas várias variáveis como a percentagem de utilização dos processadores e memória das máquinas envolvidas ou a utilização dos índices de texto.

Para que estes factores fossem controlados de uma forma metódica foram definidos os seguintes requisitos que devem ser cumpridos pelos testes realizados e pelos seus resultados. Cada teste aqui referido consiste na repetição/iteração de um conjunto de passos.

Tabela 6: Requisitos de testes

Identificador	Requisito
REQ_01	Número médio de iterações falhadas (ou seja, iterações que não chegaram ao seu término ou cuja resposta da parte do servidor não foi a pretendida) deve ser inferior a 20% do total de testes executado
REQ_02	Percentagem de utilização do processador da máquina de testes deve ser inferior a 90% durante a realização de todo o teste
REQ_03	Todos os computadores envolvidos na realização dos testes são dedicados a estes testes exclusivamente
REQ_04	*Software* de segurança da máquina de testes (antivírus e *firewall*) e outras aplicações dependentes da rede (*Outlook*, clientes de *instant messaging* e outros) devem estar, durante os testes, desligados ou desactivados
REQ_05	Utilizadores virtuais não iniciam os testes em simultâneo mas de forma temporalmente esparsa. Idealmente, com o objectivo de iniciar testes ao longo do tempo, o ultimo utilizador a iniciar deve começar o teste imediatamente antes de o primeiro terminar o seu primeiro teste (esta medida não deve ser tomada como assertiva, mas sim como uma orientação para a afinação da velocidade com que são iniciados os testes)
REQ_06	As máquinas utilizadas nos testes encontravam-se ligadas através da rede interna da empresa (para que a velocidade da rede não seja um *bottleneck* para os testes)

Criação do ambiente inicial de testes

Com o objectivo de simular o mais fielmente possível o ambiente existente em produção, foram utilizadas nestes testes as duas bases de dados populadas na fase de testes anterior com 386850 artigos. Tendo em conta que o portal depende em muito da informação da base de dados, a utilização de uma base de dados com mais informação ajudará a evidenciar os resultados das alterações realizadas no sistema e mais concretamente na base de dados.

Ambas as bases de dados estão instaladas no servidor referido em cima como *DBserver*.

No servidor *WebServer* estão alojadas as duas versões do portal CisionPoint. Dado que estas utilizam *frameworks .Net* diferentes, estão instaladas em *pools* de aplicações diferentes, uma com a *framework 1.1* e outra com a *framework 4.0*.

Foram também criados dois índices, um para cada versão a testar, que são utilizados na realização de pesquisas no portal. Para a versão mais antiga do CisionPoint foi criado um índice Lucene idêntico ao existente no início deste estágio na plataforma. Para a versão CisionPoint desenvolvida neste estágio foi criado um índice SolR tal como foi descrito já neste relatório.

Estes índices estão também a correr no *WebServer*, evitando assim que o servidor *web* tenha que aceder a estes através da rede.

Por fim, todos estes testes foram levados a cabo a partir de um computador pessoal com o Visual Studio referido em cima. Por questões de simplicidade, neste documento este computador é referido como emulador de clientes e tem as seguintes características:

- Emulador de clientes
    - *Microsoft Windows Vista Business* com *Service Pack 2*
    - *Intel Core 2 Duo*, a correr cada um a 2.4 *GHz* com 2 *Gb* de *RAM*
    - *Microsoft Visual Studio Ultimate Edition (trial version)*

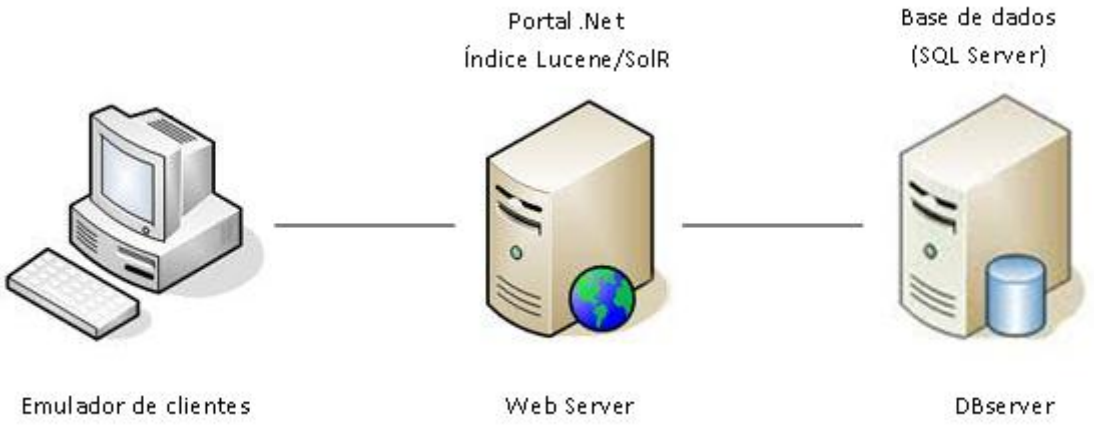

Ilustração 4: Arquitectura de testes

Abordagem realizada aos testes

Mais à frente estão descritos os testes que foram realizados ao nível do portal. Contudo, para que estes fossem elaborados muitos outros foram levados a cabo. O processo de desenvolvimento destes testes foi por um lado um processo de aprendizagem progressiva da ferramenta utilizada e das suas capacidades, por outro lado um processo de afinação constante dos parâmetros utilizados e das várias configurações do sistema. Dada a complexidade deste e a variedade de componentes que incorpora, só foi possível chegar aos resultados aqui apresentados com a repetição exaustiva dos testes e a experimentação dos vários parâmetros destes. Os testes apresentados são o culminar de todo este processo.

Definição dos casos de uso a testar

Tendo em vista a caracterização dos clientes do CisionPoint e da sua actividade neste portal, desde o início deste estágio que tem vindo a ser registada informação sobre as principais acções que os utilizadores realizam.

Partindo da observação desta informação e com a ajuda e conhecimento dos demais elementos da equipa do CisionPoint foi possível analisar o comportamento dos utilizadores na navegação no CisionPoint e identificar vários padrões recorrentes para os vários tipos de utilizador. Por exemplo, clientes como agências de comunicação acedem principalmente a detalhes de artigos e com muita frequência, já outros utilizadores diariamente consultam algumas notícias e por fim criam um *pressbook* e enviam. Tendo em conta esta informação foi então possível estabelecer vários perfis de utilizador genéricos. Estes perfis estão descritos na Tabela 7. Logicamente existirão utilizadores com perfis diferentes dos aqui apresentados,

contudo estes são os que se evidenciam nos dados registados e na experiência partilhada pelos restantes elementos da equipa.

Tabela 7: Perfis de utilizador do CisionPoint

Identificador	Perfil
PRF_1	***Login* – Navega por vários tópicos - Abre várias notícias** Este perfil consiste em realizar, em primeiro lugar, o *login* no portal para obter acesso à página inicial do CisionPoint. Depois disto, o utilizador consulta quatro dos tópicos que tem disponíveis e no último destes acede às páginas de detalhes de quatro notícias.
PRF_2	***Login* - Abre várias notícias - Selecciona as notícias abertas - Cria *PressBook*** Neste perfil o utilizador começa por realizar os mesmos passos existentes no perfil 1 seguidos da criação de um *Pressbook* com as quatro notícias consultadas.
PRF_3	**Utilizador abre várias notícias a partir de uma *Newsletter* recebida** Este perfil representa o tipo de utilizador que recebe uma *NewsLetter* proveniente do CisionPoint e que acede aos detalhes de quatro notícias dessa *Newsletter*. Este acesso é realizado sem que o utilizador se autentique.
PRF_4	***Login* - Realiza pesquisa avançada - Consulta resultados** Este perfil tem por objectivo representar o grupo de utilizadores que utiliza com frequência a ferramenta de pesquisa avançada do portal. O utilizador realiza o *login* no portal e abre a janela de pesquisa avançada. Selecciona o intervalo de datas de dia 1 de Março a dia 1 de Abril, preenche o campo de pesquisa de texto com a palavra *'twitter'* e carrega no botão de pesquisa. Este processo termina quando os resultados da pesquisa são mostrados ao utilizador. Estes parâmetros de pesquisa foram escolhidos por resultarem num número de resultados substancial para que o peso destes tenha algum impacto nos resultados dos testes realizados.

Casos de teste

Com base nos perfis de utilizador apresentados em cima foram então definidos os vários cenários de teste a implementar e a realizar para ambas as versões do sistema, o4k e CisionPoint24.

Em cada cenário de teste foram definidos diferentes valores para vários dos parâmetros existentes. Estes valores foram estabelecidos, tal como já foi descrito antes, com base na realização de testes experimentais e tendo em vista os requisitos estabelecidos para os testes e as limitações dos recursos disponíveis. Na tabela seguinte é apresentada uma descrição dos parâmetros utilizados na realização dos testes.

Tabela 8: Parâmetros de testes

Campo	Descrição
# inicial de utilizadores	Número de utilizadores simultâneos com que o teste é iniciado
# final de utilizadores	Numero máximo de utilizadores simultâneos que o teste comporta. O número de utilizadores deve crescer até este valor.
Aumento	Aumento do número de utilizadores a registar com a frequência e modo de crescimentos definidos nos parâmetros seguintes
Frequência do aumento	Intervalo de tempo entre cada aumento.
Modo de crescimento	O aumento pode-se verificar de dois modos: instantâneo ou contínuo. No primeiro em cada intervalo de tempo são colocados os utilizadores correspondentes a esse aumento a realizarem os testes respectivos simultaneamente. No modo contínuo os utilizadores vão começando os seus testes ao longo do intervalo de tempo em que se deve registar o aumento (esta diferença é ilustrada na Ilustração 5).
# de iterações realizadas	Número de vezes que o processo a ser testado é realizado. Depois de terminadas estas repetições é dado o teste como concluído.

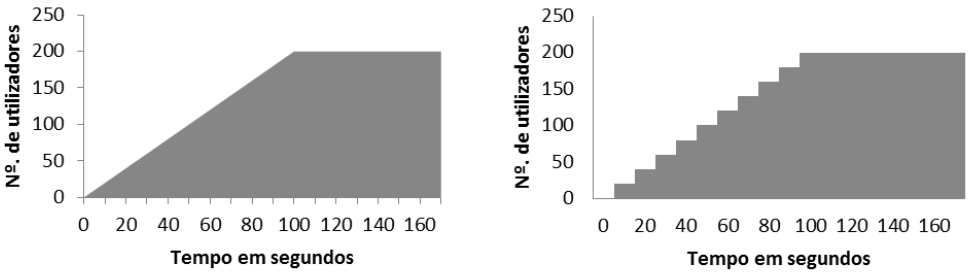

Ilustração 5: Modos de crescimento do número de utilizadores: continuo e instantâneo

Estes dois gráficos ilustram em primeiro lugar a diferença entre o modo de crescimento contínuo (à esquerda) e o modo de crescimento instantâneo (à direita). A diferença prática entre estes dois modos é que, no primeiro como os utilizadores são distribuídos mais uniformemente ao longo do tempo também a utilização de recursos é distribuída ao longo deste. Além disto o modo contínuo é mais realista pois no dia-a-dia os utilizadores não entram no CisionPoint sincronizadamente mas sim de uma forma aleatória.

Nas secções seguintes estão descritos os cenários de testes referidos acima.

Descrição do caso de teste CT01

Tabela 9: Descrição do caso de teste CT01

<table>
<tr><th colspan="2">Especificação do caso de teste</th></tr>
<tr><td>Identificador</td><td>CT01</td></tr>
<tr><td>Objectivo</td><td>Este teste tem como objectivo simular o comportamento de ambas as versões do sistema quando são utilizadas por 200 utilizadores com perfil de utilizador PRF_01 em simultâneo.</td></tr>
<tr><td colspan="2">Input

Cenário do teste tem os seguintes parâmetros:

<table>
<tr><th>Campo</th><th>Valor</th></tr>
<tr><td># Inicial de utilizadores</td><td>10</td></tr>
<tr><td># Final de utilizadores</td><td>200</td></tr>
<tr><td>Aumento</td><td>10</td></tr>
<tr><td>Frequência do aumento (s)</td><td>10</td></tr>
<tr><td>Modo de crescimento</td><td>Contínuo</td></tr>
<tr><td># de iterações realizadas</td><td>1000</td></tr>
</table>

Sequência realizada em cada iteração:

Em cada iteração deste teste um utilizador virtual realiza uma vez o caso de uso PRF_01 (ver Tabela 7).</td></tr>
<tr><td>Output</td><td>Resumo dos valores registados durante o teste para ambas as versões do sistema. O processo deve estar de acordo com os requisitos definidos na Tabela 6.</td></tr>
</table>

Descrição do caso de teste CT02

Tabela 10: Descrição do caso de teste CT02

<table>
<tr><th colspan="2">Especificação do caso de teste</th></tr>
<tr><td>Identificador</td><td>CT02</td></tr>
<tr><td>Objectivo</td><td>Este teste tem como objectivo simular o comportamento de ambas as versões do sistema quando são utilizadas por 200 utilizadores com perfil de utilizador PRF_02 em simultâneo.</td></tr>
<tr><td colspan="2">Input</td></tr>
<tr><td colspan="2">Cenário do teste tem os seguintes parâmetros:

<table><tr><th>Campo</th><th>Valor</th></tr><tr><td># inicial de utilizadores</td><td>10</td></tr><tr><td># final de utilizadores</td><td>200</td></tr><tr><td>Aumento</td><td>10</td></tr><tr><td>Frequência do aumento (s)</td><td>10</td></tr><tr><td>Modo de crescimento</td><td>Contínuo</td></tr><tr><td># de iterações realizadas</td><td>1000</td></tr></table>Sequência realizada em cada iteração:

Em cada iteração deste teste um utilizador virtual realiza uma vez o caso de uso PRF_02 (ver Tabela 7).</td></tr>
<tr><td>Output</td><td>Resumo dos valores registados durante o teste para ambas as versões do sistema. O processo deve estar de acordo com os requisitos definidos na Tabela 6.</td></tr>
</table>

Descrição do caso de teste CT03

Tabela 11: Descrição do caso de teste CT03

<table>
<tr><th colspan="2">Especificação do caso de teste</th></tr>
<tr><td>Identificador</td><td>CT03</td></tr>
<tr><td>Objectivo</td><td>Este teste tem como objectivo simular o comportamento de ambas as versões do sistema quando são utilizadas por 200 utilizadores com perfil de utilizador PRF_03 em simultâneo.</td></tr>
<tr><td colspan="2">Input</td></tr>
<tr><td colspan="2">Cenário do teste tem os seguintes parâmetros:

<table>
<tr><th>Campo</th><th>Valor</th></tr>
<tr><td># inicial de utilizadores</td><td>10</td></tr>
<tr><td># final de utilizadores</td><td>200</td></tr>
<tr><td>Aumento</td><td>10</td></tr>
<tr><td>Frequência do aumento (s)</td><td>10</td></tr>
<tr><td>Modo de crescimento</td><td>Contínuo</td></tr>
<tr><td># de iterações realizadas</td><td>1000</td></tr>
</table>
Sequência realizada em cada iteração:

Em cada iteração deste teste um utilizador virtual realiza uma vez o caso de uso PRF_03 (ver Tabela 7).</td></tr>
<tr><td>Output</td><td>Resumo dos valores registados durante o teste para ambas as versões do sistema. O processo deve estar de acordo com os requisitos definidos na Tabela 6.</td></tr>
</table>

Descrição do caso de teste CT04

Tabela 12: Descrição do caso de teste CT04

<table>
<tr><th colspan="2">Especificação do caso de teste</th></tr>
<tr><td>Identificador</td><td>CT04</td></tr>
<tr><td>Objectivo</td><td>Este teste tem como objectivo simular o comportamento de ambas as versões do sistema quando são utilizadas por 200 utilizadores com perfil de utilizador PRF_04 em simultâneo.</td></tr>
<tr><td colspan="2">Input</td></tr>
<tr><td colspan="2">
Cenário do teste tem os seguintes parâmetros:
<table>
<tr><th>Campo</th><th>Valor</th></tr>
<tr><td># inicial de utilizadores</td><td>0</td></tr>
<tr><td># final de utilizadores</td><td>50</td></tr>
<tr><td>Aumento</td><td>10</td></tr>
<tr><td>Frequência do aumento (s)</td><td>30</td></tr>
<tr><td>Modo de crescimento</td><td>Contínuo</td></tr>
<tr><td># de iterações realizadas</td><td>1000</td></tr>
</table>
Sequência realizada em cada iteração:
Em cada iteração deste teste um utilizador virtual realiza uma vez o caso de uso PRF_04 (ver Tabela 7).
</td></tr>
<tr><td>Output</td><td>Resumo dos valores registados durante o teste para ambas as versões do sistema. O processo deve estar de acordo com os requisitos definidos na Tabela 6.</td></tr>
</table>

Descrição do caso de teste CT05

Logicamente cada um dos perfis apresentados tem uma relevância diferente no dia-a-dia do portal, ou por implicarem a realização de uma tarefa mais complexa e pesada para o sistema ou por serem realizados com mais frequência. Mostrou-se por isto importante a criação de um cenário de teste que incluísse os vários perfis de utilizador e em que, tal como acontece no dia-a-dia do CisionPoint, alguns desses perfis fossem mais frequentes que outros.

Foi por isto levada a cabo uma análise da informação existente com o objectivo de definir a importância que cada um dos perfis de utilizador deveria ter neste teste.

Em primeiro lugar foi utilizada a informação disponível sobre acessos realizados no portal da qual puderam ser extraídos os seguintes valores.

Tabela 13: Acessos realizados no CisionPoint

	Contagem	Percentagem
Login	190.962	5%
AdvancedSearch	79.831	2%
ArticleDetail	2.659.451	71%
Email	4.787	0%
LeftMenu	791.518	21%
Newsletter	11.471	0%
PressBook	24.540	1%

Com o objectivo de cruzar esta informação com os perfis de utilizador existente foi criada uma matriz com a quantidade de vezes que cada uma das actividades acima ocorre em cada um dos cenários de teste anteriores (duas das referidas actividades foram omitidas dada a sua baixa relevância e por não estarem contempladas nos perfis de utilizador definidos).

Tabela 14: Acções realizadas pelos vários perfis de utilizador

	Login	AdvancedSearch	ArticleDetail	LeftMenu	PressBook
PRF_01	1	0	4	4	0
PRF_02	1	0	4	0	1
PRF_03	0	0	4	0	0
PRF_04	1	1	0	0	0
Total	3	1	12	4	1
% Relativa	14%	5%	57%	19%	5%

Para além destes factos, outros pontos mais subjectivos tiveram que ser tidos em conta.

Em primeiro lugar, dado que o processo de consulta dos detalhes de um artigo é um processo pouco exigente ao nível do acesso à estrutura de dados do CisionPoint e que estes testes têm por objectivo avaliar as melhorias obtidas nesta mesma estrutura, mesmo sendo este um processo muito importante para a plataforma, achou-se oportuno que neste cenário de teste a sua relevância/frequência fosse menor que aquela que tem na realidade.

Pelo contrário, a navegação no portal através do menu lateral (*LeftMenu*) é uma tarefa exigente ao nível da base de dados por implicar informação de um elevado número de tabelas, razão pela qual a sua influência na distribuição dos testes foi inflacionada.

Dadas estas condicionantes, foi definido o seguinte cenário de teste.

Tabela 15: Descrição do caso de teste CT05

<table>
<tr><th colspan="2">Especificação do caso de teste</th></tr>
<tr><td>Identificador</td><td>CT05</td></tr>
<tr><td>Objectivo</td><td>Este teste tem como objectivo simular a navegação no portal de vários utilizadores com os diferentes perfis de utilização.

Partindo destes valores e tendo em conta os testes já criados foi distribuído o número de testes da forma descrita a baixo.</td></tr>
<tr><td colspan="2">Input</td></tr>
<tr><td colspan="2">
Distribuição de testes:
<table>
<tr><th>Teste</th><th>%</th></tr>
<tr><td>PRF_01</td><td>50%</td></tr>
<tr><td>PRF_02</td><td>5%</td></tr>
<tr><td>PRF_03</td><td>40%</td></tr>
<tr><td>PRF_04</td><td>5%</td></tr>
</table>
Cenário do teste tem os seguintes parâmetros:
<table>
<tr><th>Campo</th><th>Valor</th></tr>
<tr><td># inicial de utilizadores</td><td>10</td></tr>
<tr><td># final de utilizadores</td><td>200</td></tr>
<tr><td>Aumento</td><td>10</td></tr>
<tr><td>Frequência do aumento (s)</td><td>10</td></tr>
<tr><td>Modo de crescimento</td><td>Contínuo</td></tr>
<tr><td># de iterações realizadas</td><td>1000</td></tr>
</table>
Sequência realizada em cada iteração:

Em cada iteração deste teste um utilizador virtual realiza uma vez o um dos casos de uso da Tabela 7 de acordo com a tabela de distribuição de testes mostrada a cima.
</td></tr>
<tr><td>Output</td><td>Resumo dos valores registados durante o teste para ambas as versões do sistema. O processo deve estar de acordo com os requisitos definidos na Tabela 6.</td></tr>
</table>

5.3. Comparação do espaço ocupado em disco

Com uma quantidade tão grande de informação, outro factor de preocupação é o espaço ocupado pela informação em disco. Como tal, e porque algumas das alterações realizadas tinham em vista a redução deste espaço foi registado para cada tabela de ambas as bases de dados o número de registos destas e o espaço ocupado pelos dados e pelos índices das mesmas. Para que este processo fosse facilitado, foi utilizado o script presente no anexo C deste documento.

De forma idêntica foi também registado o espaço ocupado pelos índices *Lucene* e *Sol*R utilizados pela plataforma para a realização de pesquisas. Este processo, dada a sua simplicidade foi realizado manualmente utilizando as ferramentas do sistema operativo em que os índices se encontram instalados.

Os valores registados tanto para as bases de dados como para os índices são mostrados no capítulo seguinte deste relatório.

Capítulo 6
Resumo e análise dos resultados obtidos

6.1. Testes ao tempo de importação de artigos

Na tabela seguinte estão os valores registados durante os testes de importação de artigos para as bases de dados CisionPoint. Para cada base de dados o teste foi executado 5 vezes. Os resultados aqui apresentados são a média aritmética dos resultados registados em cada uma dessas execuções.

Tabela 16: Resultados dos testes de importação de artigos

	O4k	CisionPoint 24
Duração média (ms)	4.539.281	1.448.920
Desvio Padrão	354.410	9.373
Duração média (mm:ss:ms)	75:39:281	24:08:920

Tendo em conta o número de artigos que foram importados nos testes e dados estes valores em média a importação de cada artigo demora respectivamente para as bases de dados O4K e CisionPoint24, 213.52 e 68.16 milissegundos. Ou seja, a nova versão demorou aproximadamente um terço do tempo (32%) da sua antecessora a importar os mesmos artigos. Dado que a importação de artigos é um processo crítico para o negócio e principalmente porque é mais frequente nas horas em que o sistema regista mais problemas de performance, esta melhoria é de elevada importância para a realização dos objectivos deste estágio e está muito acima dos resultados esperados para este processo.

6.2. Testes à navegação no portal

Nesta secção são mostrados os resultados dos testes de performance realizados no portal CisionPoint tal como foi descrito no capítulo 6 deste documento.

Cada caso de teste foi executado cinco vezes para cada uma das versões do CisionPoint. Os resultados aqui apresentados são a média dos valores registados em cada uma dessas repetições.

Aqui listados estão apenas os valores com relevância para a comparação que se pretende realizar. Na Tabela 17 é feita uma descrição sucinta de cada um destes campos. Os resultados detalhados destes testes podem ser consultados no anexo D deste documento.

Tabela 17: Descrição dos valores registados nos testes de performance do portal

Campo	Descrição
Duração	Tempo que cada caso de teste demorou a ser reproduzido expresso em segundos.
Testes/Segundo	Número médio de testes realizados por unidade de tempo.
Iterações falhadas	Número de iterações do teste que não foram concluídas com sucesso.
Duração de iterações	Duração média de cada iteração do teste em segundos.
Páginas/ Segundo	Número médio de páginas *Web* carregadas por unidade de tempo.
Tempo por página	Duração média de carregamento de cada página do teste em segundos.

Resultados do caso de teste CT01

Resultados do caso de teste que envolve a navegação no portal utilizando a barra lateral deste e a consulta de algumas notícias.

Tabela 18: Resultados do caso de teste 1

	CP24		O4K	
	Média	**Desv. Padrão**	**Média**	**Desv. Padrão**
Duração	00:19:11	00:01:43	00:26:03	00:00:58
Testes/Segundo	0,88	0,08	0,64	0,02
Iterações falhadas	25,60	13,23	26,60	13,98
Duração de iterações	204,00	4,94	336,60	14,64
Páginas/Segundo	30,58	2,74	22,06	0,72
Tempo por página	5,24	0,14	9,20	0,35

Tal como já foi dito atrás, a utilização do menu lateral é um processo exigente ao nível da base de dados por implicar a junção de dados de várias tabelas desta através das chaves primárias destas. Por este motivo foi escolhido este caso de teste para ser realizado em primeiro lugar, para que pudessem ser apuradas as melhorias conseguidas com as alterações realizadas ao nível da reestruturação da base de dados durante a fase inicial deste estágio.

Tal como era esperado, os resultados registados para a nova versão são consideravelmente melhores. A realização de cada iteração demora em média dois terços do tempo na nova versão, melhoria conseguida principalmente com a simplificação do tipo de dados que é comparado na selecção de dados da base de dados, as chaves das tabelas. Esta é uma melhoria não tão importante por acelerar a navegação no portal, mas principalmente por

reduzir o processamento que é provocado ao nível da base de dados por cada utilizador ao navegar no portal.

Resultados do caso de teste CT02

Resultados relativos ao caso de teste em que são consultadas várias notícias e é criado um *Pressbook*.

Tabela 19: Resultados do caso de teste 2

	CP24		O4K	
	Média	**Desv. Padrão**	**Média**	**Desv. Padrão**
Duração	00:33:16	00:00:47	00:35:10	00:01:21
Testes/ Segundo	1,00	0,02	0,95	0,04
Iterações falhadas	195,00	17,33	174,00	14,71
Duração de iterações	206,40	1,02	221,00	4,60
Páginas/ Segundo	15,04	0,34	21,84	0,85
Tempo por página	11,56	0,05	7,81	0,22

Uma parte substancial do tempo utilizado por este teste consiste na criação do ficheiro *PDF* com os artigos seleccionados, o *pressbook*. Este processo não sofreu quaisquer alterações durante este estágio por isso é natural que as melhorias nos resultados não se verifiquem tão evidentemente.

De notar ainda que apesar de o número de testes realizados por segundo ser maior na nova versão o número de páginas por segundo é mais baixo nesta. Este valor, ao contrário do que se possa pensar, não é necessariamente negativo. Deve-se ao facto de, por terem sido feitas optimizações no portal, em cada iteração serem carregadas menos páginas para que seja obtido o mesmo resultado.

Este teste foi o único que violou um dos requisitos estabelecidos por terem sido registados alguns testes com número de iterações superior a 20% para a nova versão da plataforma, ou seja, com número de iterações falhadas superior a 20% do número de iterações total. Este valor e o facto de ser mais elevado para a nova versão do que para a sua antecessora são explicados pelo aumento da velocidade dos acessos à base de dados. Dado que todo o teste fica mais célere, mais pedidos de criação de *PDF* são feitos num mesmo período de tempo fazendo com que o tempo que este demora aumente, devido à limitação dos recursos de processamento do servidor. Deste modo é criado um engarrafamento no sistema provocando a ocorrência de erros de *timeout* no teste. Por este motivo foi decidido ignorar este requisito neste caso de teste em especial.

Resultados do caso de teste CT03

Valores registados durante a execução do caso de teste em que um utilizador visualiza os detalhes de várias noticias sem que tenha realizado *login* no portal.

Tabela 20: Resultados do caso de teste 3

	CP24		O4K	
	Média	**Desv. Padrão**	**Média**	**Desv. Padrão**
Duração	00:11:55	00:00:54	00:11:32	00:00:46
Testes/ Segundo	2,81	0,23	2,90	0,19
Iterações falhadas	29,80	16,02	22,00	5,73
Duração de iterações	55,28	1,87	55,82	0,99
Páginas/ Segundo	11,24	0,90	11,60	0,72
Tempo por página	6,08	0,44	6,26	0,19

Este caso de teste consiste simplesmente no carregamento da página de visualização dos detalhes de uma notícia. Esta é uma página relativamente simples. Tem um cabeçalho, com alguma informação sobre o artigo, seguido de um ficheiro *PDF* com a imagem do artigo ou de um *clipp* de vídeo ou som. Quando esta página é carregada a informação do cabeçalho é consultada na base de dados e é feito o *download* para o computador do cliente do ficheiro *PDF* ou do *clipp*. Logicamente o tempo gasto neste *download* é muito maior relativamente ao tempo gasto para carregar os restantes componentes da página e por conseguintes gasto a aceder à informação na base de dados. Por esta razão, os resultados deste teste são tão idênticos para ambas as versões, pois a relevância que os acessos à base de dados têm é mínimo comparado com o tempo gasto na obtenção do resto da página.

Resultados do caso de teste CT04

Resultados da execução do caso de teste que simula a realização de uma pesquisa avançada.

Tabela 21: Resultados do caso de teste 4

	CP24		O4K	
	Média	**Desv. Padrão**	**Média**	**Desv. Padrão**
Duração	00:14:32	00:00:14	00:18:43	00:00:10
Testes/ Segundo	1,15	0,02	0,89	0,01
Iterações falhadas	43,60	3,88	113,80	13,06
Duração de iterações	38,72	0,32	51,60	0,42
Páginas/ Segundo	9,17	0,15	7,12	0,06
Tempo por página	0,98	0,04	2,96	0,06

Ao contrário dos casos de teste anteriores, este não tinha como principal foco a utilização da base de dados, mas sim dos índices Lucene e SolR usados pelas pesquisas feitas no portal. Também aqui os resultados se mostraram positivos. Estes valores não se devem à velocidade maior do SolR mas principalmente às alterações que foram feitas na realização das pesquisas

e na obtenção da informação a ser mostrada na lista de resultados destas. O facto de o índice devolver toda a informação para a lista de artigos e de não ser utilizada a base de dados no processo de criação desta lista, mostrou ser uma opção bastante vantajosa para a performance das pesquisas.

Resultados do caso de teste CT05

Resultados do caso de teste que simula a interacção com o portal de utilizadores com perfis de utilização diferentes.

Tabela 22: Resultados do caso de teste 5

	CP24		O4K	
	Média	**Desv. Padrão**	**Média**	**Desv. Padrão**
Duração	00:12:47	00:01:14	00:15:39	00:00:20
Testes/ Segundo	1,31	0,13	1,06	0,02
Iterações falhadas	11,20	3,43	36,40	10,59
Duração de iterações	111,40	2,33	186,00	4,73
Páginas/ Segundo	27,14	2,79	21,78	0,47
Tempo por página	4,13	0,09	7,81	0,24

Os resultados deste último teste são apenas uma confirmação dos resultados dos testes anteriores. Tanto o numero de testes por segundo como o numero de páginas por segundo são superiores na nova versão, em que até o numero de falhas ocorridas é menor.

Facilmente se conclui que com as alterações realizadas durante este estágio foi possível melhorar substancialmente a performance de todo o portal fazendo com que este tenha menos impacto na performance da base de dados e como tal, na performance de todo o sistema.

6.3. Testes ao espaço ocupado pela informação nas bases de dados

Base de dados

Na tabela seguinte estão resumidos os resultados obtidos relativamente ao espaço ocupado em disco pela base de dados. Na coluna Registos está a soma do número de registos que existe em cada uma das tabelas da base de dados. Nas outras 3 colunas estão respectivamente o espaço Reservado no disco para a base de dados, o espaço ocupado pelos dados propriamente ditos e o espaço ocupado pelos índices criados sobre a estrutura de dados.

Tabela 23: Espaço ocupado em disco pelas bases de dados

	Registos	Reservado (Mb)	Dados (Mb)	Índices (Mb)
O4K	5.424.834	4.561,66	3.651,53	897,35
CisionPoint24	3.451.933	3.566,85	3.203,24	347,07
Diferença	1.972.901	994,81	448,29	550,28
Redução (%):	36,37	21,81	12,28	61,32

A tabela completa com os valores do espaço ocupado para cada tabela está no anexo E.

Apesar de aparentemente a redução do espaço ocupado pelos dados ser pequena, uma análise mais cuidada dos resultados evidência o contrário. Esta análise foi feita calculando o espaço ocupado em média para cada registo e verificando se a alteração das chaves do tipo *Guid* para o tipo numérico tinham relevância, ao nível de cada registo, no tamanho das tabelas, visto que esta relevância depende do tamanho total de cada registo e do número de chaves que cada registo tem. Os resultados mostraram que, na maioria das tabelas, o espaço ocupado por cada registo é menor ou igual na base de dados CisionPoint24 que na base de dados O4K.

Por exemplo, em relação à tabela *Clippings*, o espaço ocupado na CisionPoint24 é de menos 512,6 *Mb* que na base de dados original. Isto equivale a uma redução de mais que 55% do espaço ocupado por cada registo. Já no caso da tabela *Articles* o espaço ocupado por cada registo mantém-se praticamente igual. Esta diferença deve-se ao número de chaves que cada uma destas tabelas tem e ao peso que estas têm no tamanho de cada registo. Apesar de a tabela *Articles* ter vários campos que foram alterados de *uniqueidentifier* para *numeric*, como cada registo tem relativamente muita informação (7,39 *Kb* por registo em média) a diminuição do tamanho dos campos chave não afecta tanto o tamanho de cada registo e consequentemente o tamanho da tabela *Articles*.

Como era de esperar, o contributo mais substancial para a redução do espaço utilizado pela base de dados proveio da eliminação das tabelas *ObserverMappingKeys* e *ClippingProfiles* que ocupavam respectivamente 227,8 *Mb* e 187,7 *Mb*. Esta alteração criou a necessidade de adicionar um campo novo, do tipo *nvarchar* ou *Guid*, em várias tabelas. Isto provocou um aumento do espaço destas tabelas. Este aumento é mínimo quando comparado com o espaço da tabela *ObserverMappingKeys*, logo esta alteração é vantajosa para a diminuição do espaço usado pela base de dados. Com a eliminação desta tabela foram reduzidas as relações existentes entre as tabelas e simplificado o acesso a esta informação, é por isso também uma vantagem para a performance da base de dados.

Índices *Lucene* e *SolR*

Na tabela seguinte estão os valores do espaço ocupado em disco pelos índices *Lucene* e *Sol*R.

Tabela 24: Espaço ocupado em disco pelos índices

Índice	Espaço em disco
Lucene	1.355,28
SolR	2.703,41

Dado que o novo índice além de guardar a informação necessária para a realização da pesquisa de artigos, também guarda a informação completa a mostrar no portal como resultado da pesquisa feita, era já esperado que este ocupasse mais espaço que o índice que o precedia. Por este motivo os valores desta tabela não devem ser comparados directamente pois apesar de a nova versão ocupar muito mais espaço, tem também melhorias substanciais ao nível da performance e da complexidade do processo da pesquisa (ver Resultados do caso de teste CT04).

Capítulo7
Conclusões e trabalho futuro

No inicio deste estágio foi apresentado ao estagiário um sistema que possuia vários problemas ao nivel da performance. Estes problemas implicavam a existência de periodos em que o CisionPoint deixava de estar acessivel aos clientes e deviam-se principalmente à sobrecarga a que estava sujeita a base de dados.

Com o objectivo de solucionar estes mesmos problemas, ou pelo menos de minimizar o seu impacto, foram definidas e implementadas algumas medidas que estão compreendidas nos dois seguintes tópicos:

- Optimização da performance da estrutura de dados do sistema;
- Criação de uma estrutura de histórico que permita um acesso privilegiado à informação mais recente;

Em relação ao primeiro tópico iniciaram-se os trabalhos com a alteração do tipo de dados dos identificadores da informação contida na estrutura de dados. Com esta alteração pretendia-se em primeiro lugar optimizar o espaço que esta estrutura ocupa e em segundo lugar acelerar o manuseamento destes identificadores substituindo os existentes por identificadores mais simples. Esta fase foi também muito importante para que o estagiário conhecesse melhor o sistema, principalmente no que toca à estrutura de dados que o suporta.

Ainda tendo em vista a optimização da performance da estrutura de dados foi feita uma reestruturação desta. Foram eliminadas da base de dados tabelas e outras estruturas que se concluíu serem desnecessárias. Assim além de estas deixarem de ocupar espaço em disco, algumas permitiram também que alguns processos de gestão da informação fossem optimizados.

Estas alterações fizeram com que todo o sistema deixasse de estar de acordo com a estrutura de dados que o suporta, e foi por isto necessário adaptar o mesmo à nova estrutura de dados. Dada a dimensão do sistema e aproveitando a sua modularidade este processo foi realizado de modo faseado. Em primeiro lugar foi alterado o motor de importação responsável por introduzir informação na estrutura de dados do CisionPoint. Só depois de terminado este processo foram realizadas as alterações necessárias ao portal para que este acedesse devidamente à estrutura de dados. Por fim, para que todo o portal pudesse ficar completamente funcional foi ainda desenvolvida uma nova estrutura de apoio às pesquisas realizadas no portal (um índice de texto utilizando a ferramenta SolR) adaptada às características da nova estrutura de dados e que substitui a estrutura já existente, índice construído no motor de pesquisas Lucene.

Em relação à criação da estrutura de histórico foi estabelecido na fase inicial do estágio ser necessário alterar a estrutura de dados existente para que a informação mais recente fosse acedida mais rapidamente e para que fosse escalável, ou seja, para que o aumento constante do número de notícias não degradasse progressivamente a performance do acesso à informação. Contudo, dadas as mudanças ocorridas nas regras do negócio, esta alteração deixou de ser necessária, pelo menos na estrutura de dados principal do CisionPoint. Por outro lado, dado que o bloco maior de informação, a informação mais antiga, deixou apenas de ser parte da estrutura de dados do portal continuando a existir numa estrutura à parte, continuou também a ter os mesmos problemas de performance. Por esta razão foi decidido implementar a solução de particionamento nesta estrutura. Assim foi desenvolvida uma nova

estrutura de dados constituída pela base de dados que contém toda a informação e por um conjunto de índices pelos quais é particionada a informação constante nessa base de dados e onde são feitas as pesquisas de artigos. Para que esta estrutura pudesse ser utilizada de uma forma limpa e isolada foi criado ainda um *Web Sevice* através do qual seriam chamados os métodos responsáveis pela realização de pesquisas nos índices.

A par da realização das alterações foram sendo feitos testes funcionais, para validar as referidas alterações realizadas. No final de todas as alterações realizadas foram realizados testes de performance para que se pudesse aferir se as alterações feitas foram de encontro às melhorias esperadas para este estágio. Estes últimos consistiram em três conjuntos de testes distintos: testes ao tempo de importação de artigos, testes ao tempo de navegação no portal e testes ao espaço ocupado em disco pela estrutura de dados.

Nos primeiros as melhorias conseguidas foram bastante superiores às esperadas. Dado que o processo de importação de artigos é um processo crucial nas primeiras horas da manhã e que este periodo de tempo é um dos mais problemáticos do sistema, esta melhoria é muito influente na performance não só da importação, mas de todo o sistema.

Nos testes realizados ao portal logicamente as melhorias não foram tão evidentes. Isto porque o acesso à base de dados é um processo menos relevante no carregamento do portal do que na importação de artigos, principalmente porque num são feitas apenas leituras de informação e no outro são realizadas escritas na base de dados. Apesar disto também nestes testes as melhorias observadas foram consideradas bastante positivas e uma mais valia na resolução dos problemas existentes.

Por fim, os resultados do espaço ocupado em disco pela informação têm duas vertentes, uma positiva e outra menos positiva. Por um lado, considerando o espaço ocupado pela nova base de dados e pelo novo índice, o espaço ocupado pela informação é agora superior, isto porque ao colocar mais informação sobre cada artigo no índice foi aumentada logicamente a quantidade de informação replicada. Por outro lado, se apenas considerarmos a base de dados esta é agora mais pequena o que faz com que a realização do seu *backup* e de outros processos de manutenção seja também mais rápida, indo de encontro aos objectivos deste estágio. Além disto as alterações realizadas no modo como são feitas as pesquisas trazem a estas grandes vantagens ao nivel da performance e da utilização da base de dados o que faz com que a utilização de mais espaço pelo índice seja justificada.

Apesar de não ter sido realizado um teste com o objectivo de aferir quantitativamente a disponibilidade que a nova versão do sistema terá, através dos resultados obtidos, podemos facilmente concluir que, dada a redução do tamanho da base de dados conseguida, os processos como a realização de *backups* serão agora muito mais céleres e provocarão menos problemas ao bom funcionamento sistema. O facto de os restantes processos do sistema, como a importação e a navegação no portal, terem menos impacto na performance da base de dados faz também com que tenham menos impacto na performance de processos de manutenção que estejam a ocorrer em simultâneo, contribuindo também para que estes processos sejam realizados mais rapidamente.

Logicamente o objectivo desta nova versão do sistema é o de substituir a estrutura existente no ambiente de produção que serve neste momento os clientes. Contudo para que isto seja possivel mais algumas tarefas têm ainda que ser levadas a cabo. Por não fazem parte do âmbito deste estágio ou por implicam necessariamente a utilização do ambiente de produção, não foram ainda implementadas.

Em primeiro lugar é necessário popular ambas as bases de dados, tanto a do CisionPoint como a da estrutura de histórico, com todos os artigos necessários. Este processo implica,

além da importação de todos os artigos mais antigos existentes no CisionPoint original, o início da importação diária e constante de artigos novos para que a base de dados passe a estar sincronizada com o sistema antigo.

Em segundo lugar é necessário criar uma nova versão da aplicação CP Desktop que utilize a nova estrutura de dados e o novo Web Service para obter os dados de que necessita. Só assim será possivel continuar a disponibilizar ao utilizador todas as noticias existentes na estrutura de histórico e que já não constem da base de dados do CisionPoint.

Depois de sincronizada a informação existente na estrutura de dados utilizada actualmente com a nova estrutura de dados poderá então ser instalado o portal no servidor web de produção para que fique disponivel para os clientes.

Num possível projecto futuro, caso as regras de negócio passem assim a exigir novamente, poderá voltar a ser reunida a estrutura de histórico na estrutura de dados do CisionPoint e assim fazer com que o portal apesar de possibilitar a pesquisa de artigos com algum tempo tire partido das vantagens do particionamento implementado na estrutura de histórico.

Referências

Manning, Christopher, Raghavan, Prabhakar and Schütze, Hinrich . *Introduction to Information Retrieval.* s.l. : Cambridge University Press, 2008.

Bellevue Linux Users Group. The Linux Information Project. *Scalable Definition.* [Online] 2004. http://www.linfo.org/scalable.html.

Cision. Cision Portugal. [Online] http://pt2.cision.com/.

Gomes, Christian, et al. Cálculo da Disponibilidade. *Guia do Servidor Conectiva Linux.* [Online] 2001. http://www.dimap.ufrn.br/~aguiar/Manuais/Servidor/x11391.html.

Microsoft Corporation. *Using Clustered Indexes.* [Online] 2011. http://msdn.microsoft.com/en-us/library/aa933131(v=sql.80).aspx.

—. Data Integrity. [Online] 2011. http://msdn.microsoft.com/en-us/library/ms184276.aspx.

—. decimal and numeric (Transact-SQL). [Online] 2010. http://msdn.microsoft.com/en-us/library/ms187746.aspx.

Mrdenny. *IT Knowledge Exchange.* [Online] 2008. http://itknowledgeexchange.techtarget.com/itanswers/how-to-create-a-maintenance-plan-in-sql-server-management-studio-express/.

Oxford University Press. Oxford Dictionaries. [Online] http://oxforddictionaries.com/.

Schumacher , Robin . Improving Database Performance with Partitioning. [Online] 2010. http://dev.mysql.com/tech-resources/articles/performance-partitioning.html.

The Apache Software Foundation. A High Performance C# Search Engine. [Online] 2011. http://incubator.apache.org/lucene.net/.

Walters, Robert, Fritchey, Grant and Taglienti, Carmen. *Beginning SQL Server 2008 Administration.* s.l. : Apress, 2009.

Anexos

1. **Anexo A - Tabelas eliminadas da BD CisionPoint**
   Documento com a listagem de todas as tabelas, campos de tabelas ou procedimentos que foram eliminadas no processo de optimização da base de dados CisionPoint.

2. **Anexo B - Principais módulos da base de dados afectados pela eliminação das tabelas**
   Conjunto de diagramas de tabelas que ilustram as alterações feitas na organização da base de dados.

3. **Anexo C – Script para registar tamanho da base de dados**

   Script utilizado para registar o espaço ocupado em disco pelas tabelas e índices de uma dada base de dados.

4. **Anexo D – Resultados dos testes de performance do portal CisionPoint**
   Tabelas com os valores detalhados registados nos testes realizados ao portal CisionPoint.

5. **Anexo E - Tamanho das tabelas do CisionPoint**
   Registo da quantidade de registos e espaço ocupado pela base de dados do CisionPoint.