FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Indicadores de Desempenho de Bases de Dados

Eugénio André Leal Ferreira dos Santos

Mestrado Integrado em Engenharia Informática e Computação

Orientador: João Correia Lopes

Co-orientador: Pedro Campos

27 de Janeiro 2017

Indicadores de Desempenho de Bases de Dados

Eugénio André Leal Ferreira dos Santos

Mestrado Integrado em Engenharia Informática e Computação

27 de Janeiro 2017

Resumo

Actualmente, devido à facilidade de acesso à Internet, verifica-se um crescimento acentuado no número de utilizadores que utilizam plataformas de comércio electrónico. Este crescimento acelerado, aliado à exigência de alta disponibilidade associada a este tipo de serviços cria a necessidade de garantia de desempenho nos serviços prestados. As Bases de Dados são um ponto fulcral para o sucesso de qualquer negócio e o seu desempenho é um processo em contínua evolução.

A Farfetch enquadra-se no cenário descrito, uma vez que tem registado um crescimento acentuado e o desempenho do serviço que fornece é essencial para cimentar a sua posição no mercado. Esta dissertação tem por objectivo identificar um conjunto de indicadores, que permitam monitorizar as bases de dados que servem de base ao negócio da empresa e, a partir desses indicadores, desenvolver uma ferramenta que efectue a recolha, armazenamento e apresentação dos indicadores recolhidos de forma a que seja possível melhorar a administração preventiva das bases de dados.

Existem no mercado ferramentas de monitorização de bases de dados, no entanto não estão adaptadas às particularidades de cada negócio, nem apresentam as características proactivas que se pretendem. Como tal é também objectivo deste trabalho que o sistema desenvolvido seja facilmente adaptável e que tenha características que permitam efectuar uma análise proactiva sobre os dados disponibilizados.

A ferramente desenvolvida apresenta então um conjunto de indicadores de performance dos servidores onde estão alojadas as bases de dados, bem como um conjunto de indicadores de performance das próprias bases de dados. Foi também desenvolvida uma funcionalidade de comparação entre dados actuais e dados históricos que possibilita a realização de uma análise proactiva sobre o sistema monitorizado.

Finalmente, a ferramenta é avaliada pelos profissionais encarregados da administração das bases de dados da Farfetch que irão utilizar o sistema. Da avaliação efectuada foi possível concluir que a ferramenta apresenta uma mais valia, em relação aos sistemas actuais, na forma como apresenta os dados para comparação e no facto de os dados serem processados. A principal melhoria a efectuar, de acordo com a avaliação, está relacionada com a usabilidade da ferramenta.

Keywords: Comércio Electrónico, Bases de Dados, Indicadores de Performance.

Abstract

Nowadays, due to the ease of access to the Internet, there has been an increase of users that utilize e-commerce platforms. This accelerated growth together with the demand for high availability associated with these kind of services creates, within the companies, the need to ensure the performance levels of the services they provide. Databases are a main part of the business success and their performance is a process that demands continuous improvement.

Farfetch is a company that fits into the described setting due to the growth recorded on their business, therefore the performance of the services they provide is absolutely essential in order to strengthen their position on the market. This thesis has as its objectives the identification of a set of indicators that can allow for the monitoring of the databases which serve as grounds for the business and from that defined set develop a tool that will collect, store and present the values collected in such a way that will allow the improvement of the preventive administration of databases.

There are some database monitoring tools available on the market, however they don't fit the individual needs of each particular business and they do not possess the proactive characteristics that we aim to develop on this thesis. Therefore, it is also a goal of this thesis to develop a system easily changeable and that presents the data in such a way that allows for a proactive analysis.

The developed tool presents a set of performance indicators for the servers where the databases are stored and also presents a set of performance indicators for each database stored on the server. Another functionality of the tool is the ability to compare data from different time ranges, which allows for a proactive performance analysis.

Finally, the thesis is evaluated by the Database Administration Team currently working at Farfetch that will utilize the developed system. From the conducted evaluation we may conclude that the developed tool represents an improvement, against the current system, namely in the way data is presented and due to the fact that data is processed before being presented to the user. The greatest improvement, detected through the evaluation, is the usability of the tool.

Keywords: E-Commerce, Databases, Performance Indicators.

Agradecimentos

Em primeiro lugar, ao Prof. João Correia Lopes, pela orientação e toda a ajuda prestada.

Ao Eng. Pedro Campos, pela oportunidade, acompanhamento e confiança depositada em mim.

À Farfetch, por me receber, em especial ao Filipe Costa, Carlos Duarte, Paulo Correia e Gonçalo Ventura pelo conhecimento partilhado, pela disponibilidade para esclarecer qualquer dúvida e pela amizade.

Aos meus amigos, pelo incentivo e paciência.

Aos meus Avós, que seguramente estarão orgulhosos.

Aos meus Pais e Irmã, as melhores pessoas que tenho na minha vida. Esta dissertação é para vocês.

Obrigado,
Eugénio André Leal Ferreira dos Santos

Conteúdo

Lista de Figuras

LISTA DE FIGURAS

Abreviaturas e Símbolos

SQL	Structured Query Language
DML	Data Manipulation Language
DDL	Data Definition Language
HTML	Hypertext Markup Language
CRUD	Create Read Update Delete
MB	MegaBytes
SGBD	Sistema de Gestão de Bases de Dados

Capítulo 1

Introdução

Este documento tem por objectivo descrever todas as etapas que fizeram parte do trabalho desenvolvido no âmbito desta tese. Para esse efeito irão ser expostos todos os aspectos relacionados com o tema começando com a definição do problema e enquadramento do mesmo, uma análise e comparação entre diferentes conceitos e uma exposição do trabalho já realizado na àrea incluindo algumas ferramentas existentes. É também descrita a arquitectura definida para o projecto, os diferentes componentes desenvolvidos bem como a interacção entre os mesmos. É efectuada uma descrição em detalhe de todos os componentes implementados, sendo ainda apresentada a ferramenta final com descrição da funcionalidade de cada relatório apresentado. Para finalizar foi efectuada uma avaliação do trabalho desenvolvido a partir da qual são retiradas ilações relativamente ao trabalho futuro a desenvolver bem como à satisfação dos objectivos definidos no início do projecto.

1.1 Contexto

Este trabalho surge do aumento de utilizadores das plataformas de comércio electrónico o que provoca um aumento da utilização dos serviços associados ao negócio entre eles as bases de dados. A manutenção de um alto nível de performance é essencial para o sucesso desta àrea de negócio. É exactamente dessa necessidade que surge a solução proposta neste trabalho, uma vez que as ferramentas disponíveis actualmente têm um paradigma de monitorização reactivo.

Propõe-se então este trabalho a desenvolver uma solução que tenha em conta as necessidades do negócio, neste caso da Farfetch. Para isso será efectuada uma análise que permita identificar quais os indicadores que mais interessam monitorizar.

O objectivo final deste trabalho é desenvolver uma ferramenta que permita monitorizar a performance dos servidores de bases de dados, bem como das bases de dados individuais. A ferramenta permitirá efectuar uma análise proactiva da performance, ou seja irá fornecer as seus utilizadores funcionalidades que permitam investigar, através de comparação com dados históricos,

eventuais problemas de performance.

O projecto foi desenvolvido na Farfetch, empresa inovadora na àrea do comércio electrónico e que tem registado um crescimento considerável no volume de negócios e na sua visibilidade tornando-a na candidata ideal para demonstrar os resultados que se podem alcançar com um sistema deste tipo.

1.2 Motivação

A manutenção da performance dos serviços que suportam plataformas de negócio sempre foi uma necessidade, no entanto com o crescimento do número de utilizadores das plataformas tornou-se um tema ainda mais importante para qualquer empresa. No âmbito deste trabalho iremos apenas abordar a performance das bases de dados associadas aos negócios.

Os desenvolvimentos nesta àrea estão associados a uma monitorização constante das bases de dados, nomeadamente um controlo permanente de certos indicadores de performance associados a alarmes que disparam quando certos limites são ultrapassados. Apesar deste tipo de solução servir para manter as plataformas a funcionar não é a solução ideal e é daí que surge a abordagem sugerida com este tema de dissertação.

Pretende-se desenvolver um sistema de monitorização que tenha por base o mesmo conceito de monitorização de indicadores de performance de bases de dados, no entanto temos por objectivo armazenar os dados recolhidos e através da análise desses dados realizar uma monitorização preventiva, ou seja tentar antever que problemas irão acontecer antes de acontecerem.

1.3 Objectivos

Existem dois tipos de objectivos que este trabalho pretende cumprir, os primeiros são referentes a aspectos de implementação e os segundos estão relacionados com o caso de estudo que está a ser abordado com este projecto.

O primeiro conjunto inclui os seguintes objetivos:

- Identificação das métricas que devem ser tomadas em consideração de forma a monitorizar a performance das bases de dados.
- Desenvolvimento do processo de recolha e armazenamento dessas métricas.
- Apresentação dos resultados de forma clara para que seja possível retirar conclusões.
- Criação de um *Dashboard* para representação visual dos dados recolhidos.

O segundo conjunto de objectivos tem um carácter mais conceptual no sentido em que se pretendem atingir objectivos que validem o conceito apresentado, o segundo conjunto inclui os seguintes objetivos:

- Desenvolver um sistema de monitorização que assente num paradigma de monitorização proactiva.
- O sistema a desenvolver será centralizado no sentido de que não estara instalado em nenhum dos servidores que irá monitorizar, irá operar a partir de um ponto central de onde recolherá os valores das métricas selecionadas.
- Apresentar uma melhoria qualitativa na monitorização da performance das bases de dados da Farfetch, e por consequência, diminuir as quebras de performance.
- O sistema a desenvolver deve facilitar a tarefa dos administradores de bases de dados, nomeadamente na identificação das áreas que estão com quebra de performance para que seja possível corrigir a situação o mais rapidamente possível.

1.4 Estrutura da Dissertação

Para além da introdução, esta dissertação apresenta mais seis capítulos.

No Capítulo 2, é descrito o estado da arte, nomeadamente são abordados os diferentes conceitos e paradigmas que se pretendem utilizar neste trabalho e são apresentados trabalhos relacionados.

No Capítulo 3 é descrita a arquitectura da ferramenta desenvolvida, são abordados em detalhe os diferentes componentes da ferramenta desenvolvida, é descrito também o processo de instalação da ferramenta bem como o fluxo dos dados dentro da mesma. O problema a abordar é definido em detalhe neste capítulo para que seja possível ao leitor perceber mais facilmente o propósito da ferramenta desenvolvida. Um dos pontos mais importantes deste capítulo é a descrição das métricas monitorizadas pois é ponto central deste trabalho.

No Capítulo 4 é descrita em detalhe a implementação de todos os componentes desta ferramenta. É abordada a definição e implementação do modelo de dados, a estrutura de recolha de métricas com diferenciação entre os diferentes tipos de métricas e as diferentes técnicas de recolha, a técnica utilizada para o processamento da *baseline* bem como as diferentes implementações para garantir a performance bem como a manutenção do sistema.

No Capítulo 5 é descrito o resultado final do desenvolvimento, ou seja o modelo de visualização da ferramenta, são descritos individualmente os relatórios desenvolvidos, com uma descrição das suas funcionalidades, é também descrita a interacção entre os diferentes relatórios por forma a ser possível ao leitor perceber as diferentes opções de navegação entre os diferentes relatórios.

No Capítulo 6 é apresentada a metodologia de avaliação utilizada para perceber se os objectivos a que o trabalho se propôs foram alcançados.

Finalmente no Capítulo 7 são apresentadas as possibilidades de trabalho futuro a desenvolver para o projecto e é ainda efectuada uma reflexão relativamente à satisfação dos objetivos a que o trabalho se propôs.

Capítulo 2

Estado da Arte

Neste capítulo pretende-se explicar detalhadamente os conceitos, paradigmas e técnicas que constituem a base do desenvolvimento deste projecto. Iremos abordar o conceito de indicador de performance, a comparação entre paradigma reactivo e pró-activo, serão abordadas também as técnicas de baselining e de análise de performance através de dados históricos. Irão também ser explicadas as ferramentas disponíveis para desenvolvimento do projecto e também as ferramentas existentes que de alguma forma respondem ao problema que este projecto pretende resolver.

De notar que dentro do possível os conceitos serão abordados por duas perspectivas, nomeadamente a definição clássica e a adaptação desses mesmos conceitos à administração de bases de dados.

Pretende-se que no fim deste capítulo seja possível perceber os conceitos e as técnicas que estão na base da solução desenvolvida e que constituem o seu factor diferenciador..

2.1 Indicadores de Performance

Um indicador de performance é um tipo de medida de performance que é usado para avaliar o sucesso de uma entidade ou actividade. Muitas vezes este sucesso é simplesmente a manutenção de certos níveis de valores durante um período de tempo e, outras vezes, o sucesso é medido em forma de evolução dos valores de certos indicadores afectos às actividades. Por consequência, a escolha de indicadores de performance tem por base um conhecimento profundo do que é importante avaliar, de acordo com a actividade a ser avaliada.

Para este projecto têm interesse particular os indicadores de performance de bases de dados que melhor permitam representar as áreas de performance de bases de dados, no âmbito da actividade da empresa. Esta escolha envolve um conhecimento profundo do modelo de negócio da empresa. Por esse motivo, antes da selecção dos indicadores de performance de bases de dados, foi efectuado um estudo do modelo de negócio da Farfetch por forma a que fosse possível identificar que tipo de problemas podem causar mais impacto no negócio desenvolvido pela empresa. A

partir desse estudo foram escolhidos os indicadores de performance que serão monitorizados pela solução a desenvolver.

De seguida serão apresentados três possíveis indicadores de performance de bases de dados e as conclusões que podem ser retiradas dos valores que eles representam.

- **Full scans/sec**

  Número de pesquisas totais a tabelas ou índices por segundo. Quanto menor este valor for, melhor, uma vez que este tipo de pesquisas tende a causar *bottlenecks* e problemas de memória.

- **Buffer cache hit ratio**

  A percentagem de páginas que não precisaram de ser retiradas do disco. Quanto maior for o valor, melhor, uma vez que está a encontrar a informação correcta em memória que tem acessos muito mais rápidos.

- **Number of deadlocks/sec**

  Número de *Deadlocks* por segundo. É importante monitorizar a evolução deste valor, se for detetada uma tendência de aumento isso revela um problema de performance uma vez que existem bloqueios em recursos, o que por sua vez faz com que outras operações não possam ser executadas sobre esses mesmos recursos.

2.2 Paradigmas de Monitorização

No que à monitorização diz respeito podemos reduzir os paradigmas sobre os quais assentam os sistemas de monitorização a duas abordagens: abordagem reactiva e abordagem proactiva.

Pretende-se identificar os conceitos por trás de cada um dos paradigmas, o seu funcionamento e o objectivo que cumprem. Além da definição de cada um dos paradigmas será descrito um caso exemplo para que seja possível perceber mais claramente o funcionamento de cada um dos dois.

2.2.1 Paradigma Reactivo

O paradigma reactivo tem, como o seu nome indica, um funcionamento que assenta no conceito de acção-reacção. Ou seja, a monitorização baseada num paradigma reactivo tem o objectivo de informar os seus utilizadores que existe um problema e que eles têm de tomar uma acção em relação a isso.

Em relação à sua utilização, os sistemas de monitorização com este paradigma são essenciais ao bom funcionamento de um sistema porque irão sempre aparecer problemas imprevistos, sejam novos ou recorrentes que irão precisar de resolução imediata. No entanto do ponto de vista de melhoria de performance este paradigma apresenta falhas. Podemos na realidade afirmar que o grande objectivo de um sistema reactivo é garantir o funcionamento do sistema, não o seu bom funcionamento.

Um sistema clássico de monitorização reactiva apresenta e monitoriza os dados actuais do sistema e caso algum dos valores ultrapasse os limites definidos na configuração do sistema, um alarme é enviado aos utilizadores. Por aqui se percebe a importância destes sistemas, mas ao mesmo tempo as suas lacunas.

Do ponto de vista de alta disponibilidade do sistema a monitorizar, este é o paradigma ideal para a garantir, no entanto no capítulo da melhoria da performance o paradigma proactivo apresenta outras armas que o colocam num patamar superior.

2.2.2 Paradigma Proactivo

O paradigma proactivo tem por base a tomada de acções ou posições de forma a impedir problemas. Existe no entanto uma concepção errada do que deve ser um sistema proactivo: um sistema proactivo não tem por base estar preparado ou ter um plano de acção definido em caso de algo acontecer. Um sistema realmente proactivo tem de permitir que se procurem problemas, que seja possível através de técnicas de análise detectar problemas já existentes e retirá-los do sistema.

De acordo com [Eze14] um sistema proactivo deve ser desenvolvido tendo por base um plano consistente, com base em *baselines*, baseline pode ser definido como um conjunto de dados que representa o comportamento normal do sistema, documentado, repetível e focado em resultados com o objectivo de impedir ou diminuir o *downtime* do sistema.

A monitorização proactiva fornece os meios para que seja possível cumprir os objectivos a que se propõe. No entanto a utilização do sistema contém a outra componente proactiva da monitorização, isto é o sistema deve ser usado agressivamente na procura de problemas, deve ser feita uma procura constante por problemas e não esperar que os mesmos apareçam para que sejam resolvidos.

Na realidade nunca será possível monitorizar sistemas através de um paradigma puramente proactivo, irão sempre existir problemas que acontecem no momento e que em muitos dos casos têm origem em factores externos, logo é necessária uma monitorização reactiva para fazer face a esses problemas.

A perspectiva sobre a qual a proactividade deve ser analisada é a de um contexto de complementariedade, como descrito em [Eze14] os dois paradigmas têm objectivos muito distintos. Poderia ser dito que o grande objectivo de um sistema proactivo é a detecção de erros antes que os mesmos possam causar impacto e tomar acções em relação aos mesmos.

O paradigma de monitorização e planeamento proactivo permite através da criação de *baselines* identificar, por meios de análise e comparação, as tendências em termos de performance, ou seja, é possível perceber, porque existem dados históricos, em que alturas os problemas costumam acontecer. A partir desse ponto é preciso percorrer um processo de análise que permita identificar o problema, resolvê-lo e documentar o que foi feito. Através da repetição constante deste processo é possível aumentar consideravelmente a performance do sistema.

2.3 *Baselining* e Comparação de Dados

Baselining é o processo de criação de uma base de dados históricos que representem o comportamento considerado normal de um sistema. É sobre este processo que são desenvolvidas as técnicas de análise que são a base do sistema proactivo.

Para a criação de uma *baseline* são necessários três requisitos de acordo com [Ape], nomeadamente métricas para monitorizar, base de dados para armazenamento dos dados e um método para recolha de dados.

O que se pretende obter com a criação de uma *baseline* é uma base que sirva como termo de comparação para o comportamento do sistema, para que seja possível perceber se o comportamento actual do sistema é normal e, se não for, em que àreas estão as diferenças.

Existem, no entanto, várias formas de efectuar a comparação entre dados. O primeiro passo consiste na eliminação de variáveis por forma a que a comparação seja credível, de seguida pode ser realizada uma comparação directa entre valores, seja em formato tabular ou gráfico, ou podem ser calculados valores que representem o comportamento da *baseline* e efectuar apenas a comparação com esses valores. O objectivo é encontrar os pontos do comportamento actual em que a variação entre valores é maior, ou seja os pontos em que o comportamento do sistema se altera consideravelmente.

Por forma a eliminar variáveis, o período de análise tem de ser semelhante, ou seja, se os dados são do dia de hoje entre as 14 horas e as 15 horas, a *baseline* tem de conter dados do mesmo período de tempo. Se possível o dia de análise deve ser o mesmo, ou seja, se os dados forem de uma segunda-feira, a *baseline* deve também ser de uma segunda-feira. O objectivo é obter dados de comparação em que o sistema esteja com os mesmos processos e com uma carga semelhante para que os dados sejam fidedignos.

Em relação à comparação entre os dados, podem ser comparados visualmente através da criação de dois gráficos com os valores dos dois períodos e o utilizador tem de realizar a análise e encontrar os pontos de disparidade. Outra opção de análise será o processamento dos dados da *baseline* e a comparação dos valores actuais com as métricas calculadas. Por outras palavras podem ser calculadas por exemplo a média e o desvio normal e realizar a comparação dos valores actuais com esses limites. A ideia é calcular várias métricas e possibilitar ao utilizador a escolha de qual usar na sua análise, sendo apresentadas um conjunto de imagens que ilustram este conceito.

As Figuras 2.1, 2.2 e 2.3 contêm os mesmos dados, sendo a única alteração a medida estatística utilizada para comparação: a primeira imagem compara com a média calculada através dos valores da *baseline*.

Como pode ser analisado na Figura 2.1, a média não se revela uma medida muito boa para termo de comparação para este conjunto de dados uma vez que aproximadamente metade dos valores ultrapassam o limite estabelecido pela mesma. A Figura 2.2 contém o primeiro desvio normal como termo de comparação.

Também o primeiro desvio padrão não se revela a melhor medida para comparação uma vez que ainda contém muitos valores superiores ao seu limite. A terceira medida é o segundo desvio

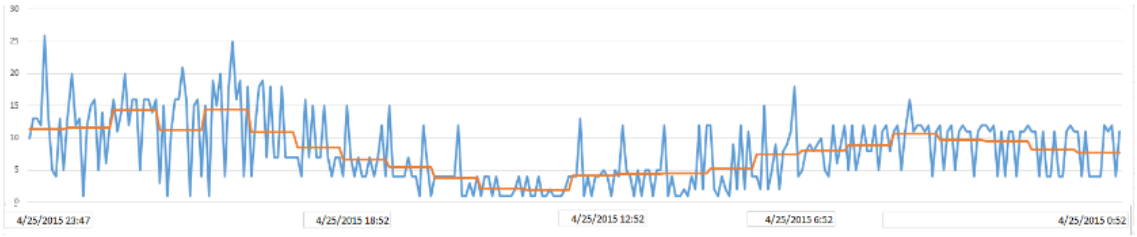


Figura 2.1: Comparação com a Média.

padrão como é possível ver na Figura 2.3.

Com o uso do segundo desvio padrão é possível identificar os pontos com os valores mais elevados, que neste caso revelam os picos de actividade. São esses pontos e a informação relativa à altura do dia em que aconteceram que temos de analisar para perceber o que aconteceu no sistema. De salientar que no exemplo apresentado os dados são processados hora a hora.

2.4 Efeito de Observador

Outro conceito extremamente importante em qualquer sistema de monitorização é o efeito do observador que se refere ao impacto que a monitorização tem sobre o próprio sistema.

Pela definição clássica, o efeito de observador ou *Hawthorne effect* refere-se às alterações de comportamento verificadas quando os sujeitos sobre observação estavam conscientes de que estavam a ser observados, como descrito em [GW00].

Este conceito tem de ser adaptado por forma a ser aplicável ao sistema desenvolvido. Na realidade este efeito está presente em qualquer sistema informático que seja sujeito a observação e em particular a sistemas de monitorização, no sentido em que estamos a observar a performance de um sistema. O próprio acto de observação e monitorização tem impacto sobre o próprio sistema, logo na realidade estamos, não a observar o sistema original, mas um sistema que é também influenciado pelo observador.

Esta é um problema real e no caso do sistema a desenvolver, um sistema de monitorização, é na realidade impossível de eliminar. Obviamente existem acções que podem ser tomadas para minimizar este efeito, nomeadamente a forma como as pesquisas são feitas tem de ser o mais eficiente possível e todo e qualquer processamento de dados não deve ser realizado no ambiente monitorizado. No entanto e mesmo que durante a recolha de certas métricas sejam ignorados

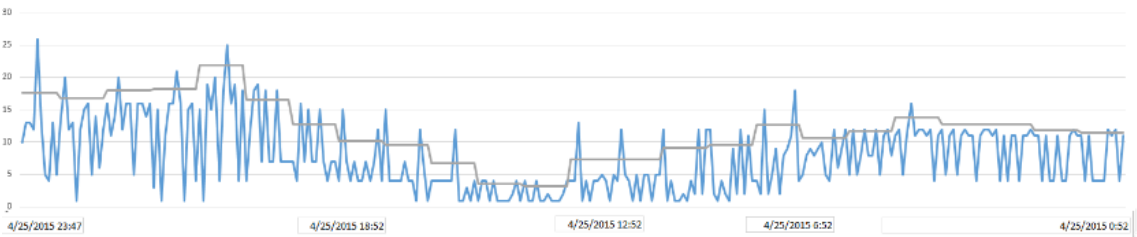


Figura 2.2: Comparação com o Primeiro Desvio Normal

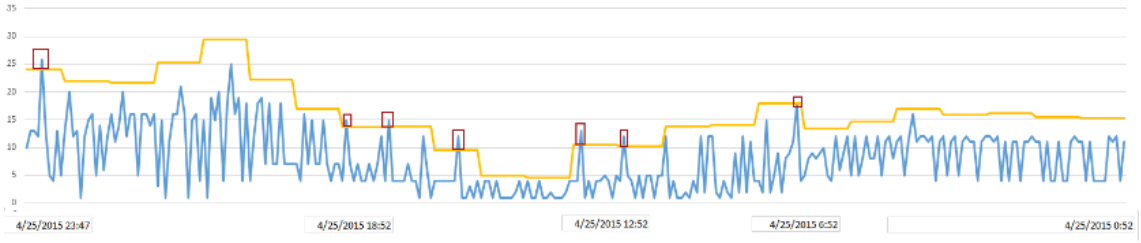


Figura 2.3: Comparação com o Segundo Desvio Normal

resultados com origem directa na monitorização do sistema, não é possível eliminar os impactos indirectos do sistema de monitorização.

Por exemplo, para que seja possível entender de forma clara o conceito, para a recolha de certas métricas o sistema de monitorização efectua pesquisas ao sistema a observar, essas pesquisas têm um custo de execução no sistema observado, logo à partida já causaram impacto. Estes são os dados que facilmente se podem identificar para medir o impacto da monitorização no sistema, no entanto a execução destas pesquisas provocam outros efeitos. A sua execução pode fazer com que outras pesquisas internas ao sistema monitorizado tenham de esperar por recursos, sendo neste caso extremamente dificil de identificar os impactos do sistema de monitorização e acima de tudo que seja mais custoso tentar realizar a identificação do que tomar a decisão consciente de que existe impacto e tentar minimizar o mesmo.

Até agora foram abordados os conceitos sobre os quais foi desenvolvido o projecto a que este trabalho se propôs. Irão agora ser apresentadas as ferramentas existentes no mercado, desde ferramentas para o desenvolvimento do projecto como ferramentas que estão disponíveis e que de alguma forma pretendem responder ao mesmo problema. Outro dos pontos abordados são as ferramentas que actualmente são utilizadas na Farfetch.

2.5 Ferramentas e Tecnologias Existentes

Foi efectuada uma análise das tecnologias existentes que possam ser uma opção para o desenvolvimento do projecto. É descrito o modelo relacional para a criação do modelo de dados e tecnologias disponíveis para a camada de visualização de dados. São também apresentadas e descritas soluções que de alguma forma respondem ao problema que este trabalho pretende abordar.

2.5.1 Base de Dados

Uma base de dados é, de uma forma muito geral, um conjunto organizado de informação. Mais especificamente é um conjunto organizado de informação armazenado num sistema informático que tem por objectivo facilitar o acesso, a manipulação e a consulta de informação pelos seus utilizadores.

Existem, no entanto, vários modelos que a organização dos dados podem assumir mas no âmbito deste trabalho, irá apenas ser abordado o modelo relacional uma vez que o trabalho a

desenvolver terá o seu próprio modelo de dados relacional e as bases de dados sobre as quais se pretende recolher indicadores de performance são bases de dados relacionais.

2.5.1.1 Modelo Relacional

Um modelo de base de dados relacional apresenta a informação organizada em tabelas, com colunas e linhas, em que cada tabela armazena objectos do mesmo tipo de acordo com [Ora]. A integridade dos dados é garantida através de um conjunto de regras, aplicadas de acordo com as necessidades do sistema que suportam. Com um conjunto de regras bem definido a integridade referêncial e a consistência dos dados está garantida.

Uma das características que distingue as bases de dados relacionais é a possibilidade de consultar dados de mais do que uma tabela através do comando *join*. Para que tal seja possível tem de haver uma coluna presente em cada tabela que estabelece a relação entre as duas tabelas. Esta coluna deve ser a chave primária de uma tabela e a chave estrangeira da outra tabela. Se algum valor na tabela em que a coluna é chave primária for apagado, todas as entradas da segunda tabela relacionadas através do valor apagado devem ser eliminadas para garantir a integridade referencial ou então é impedida a eliminação.

A linguagem utilizada para interagir com este tipo de bases de dados é o *SQL* e os seus comandos podem ser divididos em diferentes categorias, tal como referido em [Ora], sendo as principais categorias *Data manipulation Language (DML)* e *Data Definition Language (DDL)*.

Outra das características principais é a utilização de transacções. As transacções são utilizadas para garantir a consistência dos dados apresentados, permitindo concorrência no acesso aos dados. Por exemplo dois utilizadores que estejam a aceder aos mesmos dados, ao mesmo tempo, em que um deles esteja a modificar os dados, o outro utilizador poderia não obter os dados mais actuais com a sua pesquisa.

A Figura 2.4 é um exemplo simples da organização dos dados num modelo relacional. Este modelo é o ideal para o projecto desenvolvido uma vez que serão monitorizadas várias instâncias de servidores de bases de dados ao mesmo tempo e por consequência será necessário relacionar os dados para perceber a que instância se referem os valores recolhidos.

2.5.2 Visualização de Dados

Serão agora analisadas as diferentes opções para o desenvolvimento do modelo de visualização de dados.

2.5.2.1 Reporting Services

Reporting Services é uma tecnologia disponibilizada pela Microsoft para a criação e *deploy* de relatórios sobre os dados do sistema a reportar. A instalação do serviço num computador cria um servidor para o qual é realizado o *deploy* do relatório desenvolvido. O desenvolvimento do relatório é feito com recurso a ferramentas específicas sendo exemplo dessas mesmas o *Report Designer* e o *Report Builder*.

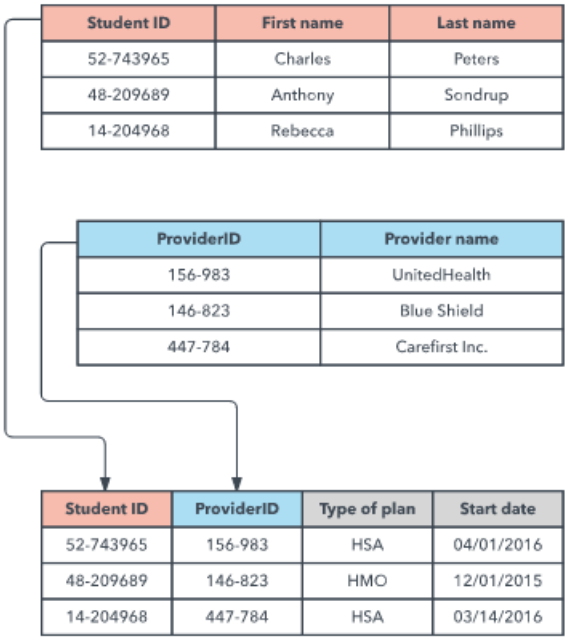


Figura 2.4: Base de Dados - Modelo Relacional

Uma das principais caracteristicas do *Reporting Services* é a sua flexibilidade, nomeadamente permite criar variados tipos de relatórios com diferentes formas de visualização dos dados (tabelas, matrizes, gráficos, ...), permite obter os dados das mais variadas fontes como *SQL Server*, *Analysis Services*, *Oracle*, *Hyperion* entre outras bases de dados, como é descrito em [Mice].

O formato final de consulta do relatório também é bastante flexível sendo possível consultar a versão *HTML*, *PDF*, *Word*, *Excel* entre outras. Outra das vantagens é a granularidade que permite atingir na apresentação dos dados. Após a ligação à fonte dos dados é possível efectuar todos os cálculos e manipulações sobre os dados, não sendo assim apenas uma ferramenta de apresentação de dados. Outra característica muito importante é a possibilidade de injectar código na criação do relatório, o que permite adicionar uma camada de interactividade extra aos relatórios que não seria possível de outra forma.

A Figura 2.5 apresenta um exemplo de um *dashboard* desenvolvido utilizando o *reporting services*.

2.5.2.2 Power View

O *Power View* é outra das tecnologias existentes para a exploração, visualização e apresentação de dados obtidos partir de uma ou várias fontes de dados. Está disponível como parte integrante do *Microsoft Excel 2013* e como *add-in* do *Microsoft Sharepoint Server Enterprise Edition* como descrito em [Micd].

Em termos de funcionalidades o *Power View* permite trabalhar os dados através da criação de pesquisas *ad-hoc*, criação de colunas calculadas, colunas de dados obtidas através de processamento de colunas de dados já existentes, apresentação da informação sobre os mais variados

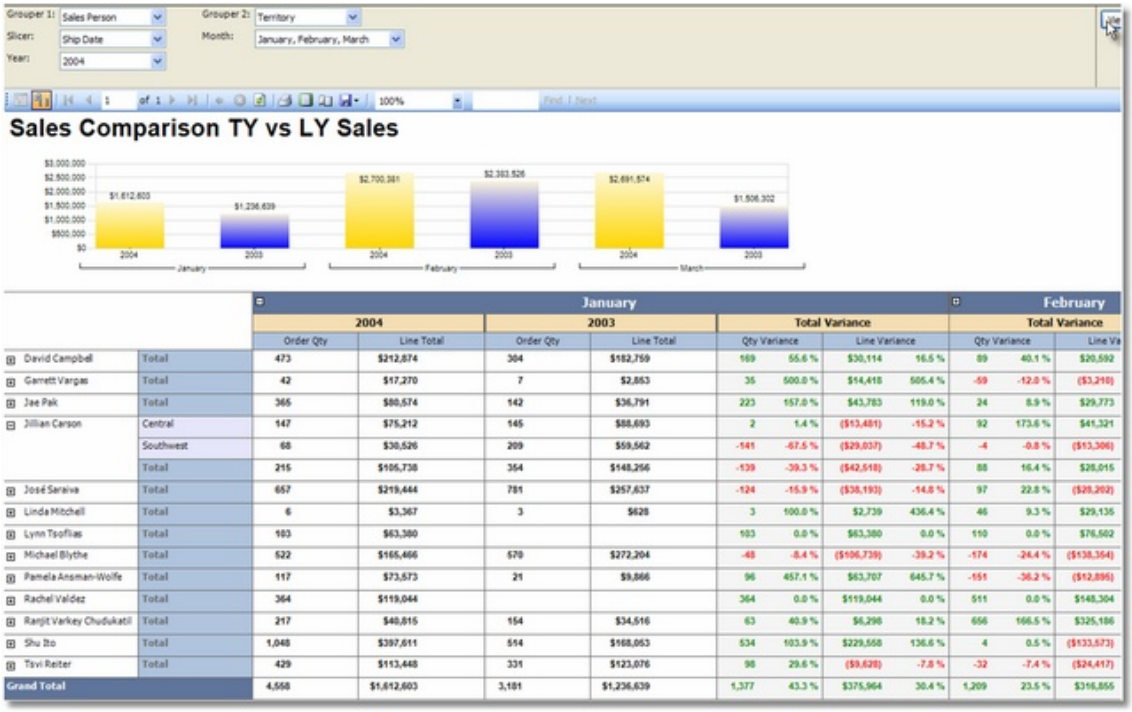


Figura 2.5: *Dashboard - Reporting Services*

formatos, nomeadamente vários formatos de gráficos ou formato tabular. Outra funcionalidade do *Power View* é a possibilidade de filtrar e salientar um conjunto de dados. Tal como descrito em [Micd], o *Power View* usa os metadados do modelo de dados que cria internamente para perceber as relações entre as diferentes tabelas e gráficos presentes no relatório. Através do conhecimento obtido com este processo, é possível filtrar os dados de todos os modelos de visualização a partir de apenas um, o que fornece um grau de interactividade ao relatório muito interessante para o utilizador.

No entanto existem algumas limitações inerentes ao *Power View*, sendo que a principal se prende com a customização dos relatórios em termos de apresentação. Por exemplo não é possível alterar esquemas de cores em gráficos o que cria entraves a certos utilizadores.

A figura 2.6 apresenta um exemplo de um *dashboard* desenvolvido utilizando o *Power View*.

2.5.2.3 Power BI

A solução *Power BI* é a mais recente tendo sido disponibilizada em Outubro de 2016. É constituída por um conjunto de serviços, nomeadamente o *Power BI Service*, *Power BI Desktop* e *Power BI Mobile* como é descrito em [Har16].

Permite estabelecer ligação às mais variadas fontes de dados, inclusive permite combinar dados de diferentes fontes, desenvolver vários tipos de relatórios, paginados ou de apenas uma página, completamente interactivos e com as interfaces mais desenvolvidas das três opções analisadas.

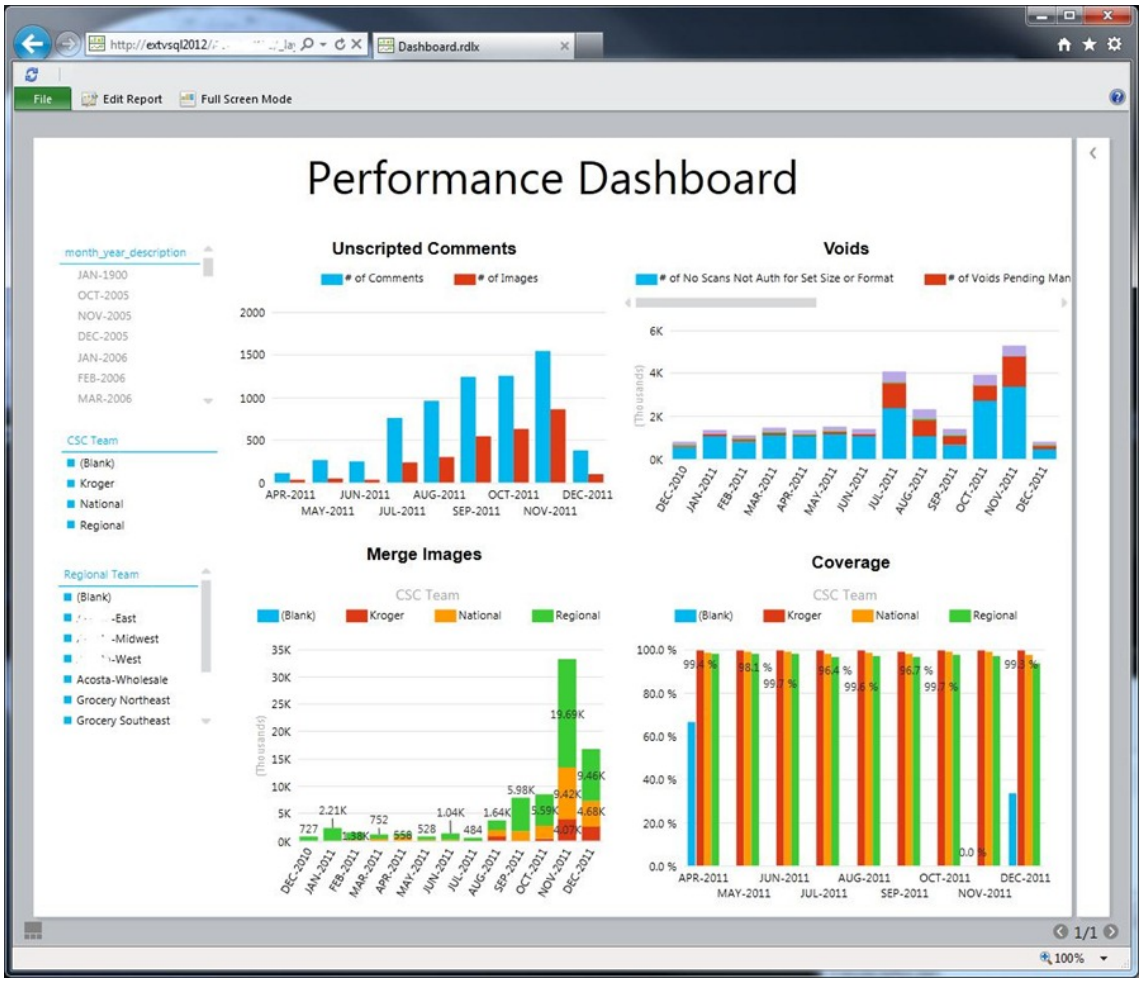


Figura 2.6: *Dashboard - Power View*

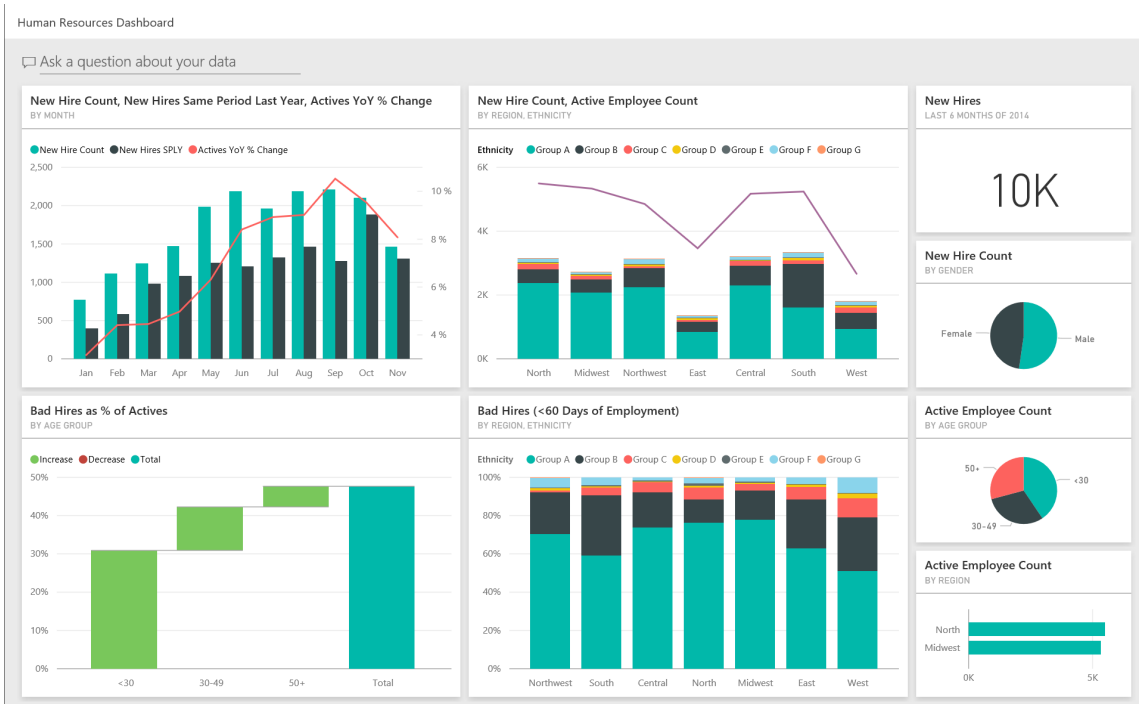


Figura 2.7: *Dashboard - Power BI*

As grandes vantagens do *Power BI* sobre as outras soluções prende-se com a variedade de contéudos que permite desenvolver, nomeadamente disponibiliza uma maior variedade em termos de componentes visuais A utilização da ferramenta é muito mais simples e amiga do utilizador do que as soluções concorrentes.

O facto de estar subdividido em três produtos facilita muito o desenvolvimento de relatórios, primeiro porque não obriga a instalar qualquer tipo de aplicação. Utilizando o *Power BI Service* é possível desenvolver directamente na *Web* e fazer *deploy* automaticamente para os serviços disponibilizados. Se for realizada a instalação do *Power BI Desktop*, a aplicação permite desenvolver localmente e após a realização do *deploy* fica disponível para consulta e edição no serviço *Web*. Todos os relatórios criados e disponibilizados são automaticamente *mobile friendly*, ou seja permitem a consulta através de um dispositivo móvel de forma responsiva e mantêm as características de interactividade do relatório original, de acordo com a descrição apresentada em [Har16], fornecendo assim um conjunto de funcionalidades que as outras soluções não disponibilizam.

Um ponto contra será obviamente o facto de ser uma solução bastante recente, enquanto que em termos de performance não são esperados grandes problemas uma vez que a solução passou por testes, em termos de comunidade de utilizadores, ela ainda é praticamente inexistente o que dificulta a resolução de problemas que possam surgir.

A Figura 2.7 apresenta um exemplo de um *dashboard* desenvolvido utilizando o *Power BI*.

2.5.3 Monitorização

De seguida vão ser descritas algumas soluções de monitorização disponíveis no mercado, enaltecendo as suas características, nomeadamente os seus pontos fortes e os seus pontos fracos.

2.5.3.1 SQL Monitor

O *SQL Monitor* é uma ferramenta de monitorização de performance de servidores *SQL Server* desenvolvida pela RedGate.

A ferramenta permite colectar dados de performance de vários servidores simultaneamente, sendo os dados recolhidos relativos à performance em termos de servidor e base de dados. Tem um carácter maioritariamente reactivo, no sentido em que realiza monitorização em tempo real e disponibiliza uma camada de alarmística que dispara quando os valores monitorizados ultrapassam os limites pré-definidos. Em relação à camada de alarmística, para cada métrica é possível definir um alarme com três níveis de gravidade, o que é positivo no sentido em que permite de alguma forma detectar mais cedo um possível problema. No entanto este tipo de sistema gera demasiada entropia com o disparar constante de alarmes.

De acordo com [DF] o acesso à interface é feito através de um web browser, o que apesar de ser uma vantagem em termos de disponibilidade apresenta problemas em termos de segurança por ter de expôr informação sensível para a rede externa à empresa. Em termos de arquitectura como descrito em [DF] o *SQL Monitor* apresenta uma arquitectura de três níveis, não efectua nenhuma recolha de métricas através da instalação de um agente local nas máquinas monitorizadas de modo a reduzir o efeito de observação e guarda toda a informação numa base de dados *SQL Server*.

No entanto, o grande ponto a favor desta solução está na pré-configuração e na alta customização que oferece aos utilizadores. Após instalação o sistema vem pré-configurado para a recolha de um conjunto de métricas básicas e começa imediatamente a recolha para esse valores, no entanto permite ao utilizador remover essas métricas e adicionar as suas próprias métricas ao processo de recolha, oferecendo assim grande flexibilidade.

O ponto fraco encontra-se no capítulo da proactividade da solução, apesar de oferecer a possibilidade de comparar os valores actuais aos valores de uma *baseline*, os dados que oferece são meramente visuais, ou seja o utilizador tem de analisar, sem qualquer tipo de processamento sobre os dados da *baseline* em que pontos os valores actuais estão fora do padrão considerado normal.

Em termos gerais o *SQL Monitor* é uma solução robusta e bastante completa, no entanto peca no capítulo da proactividade da solução e no facto de a única forma de aceder aos seus dados ser através da interface web. Outros dos pontos contra estão associados ao facto de ser uma solução externa, o que dificulta o trabalho sobre os dados armazenados uma vez que o modelo de dados é completamente desconhecido.

A Figura 2.8 apresenta a interface de monitorização do *SQL Monitor* disponível para os seus utilizadores.

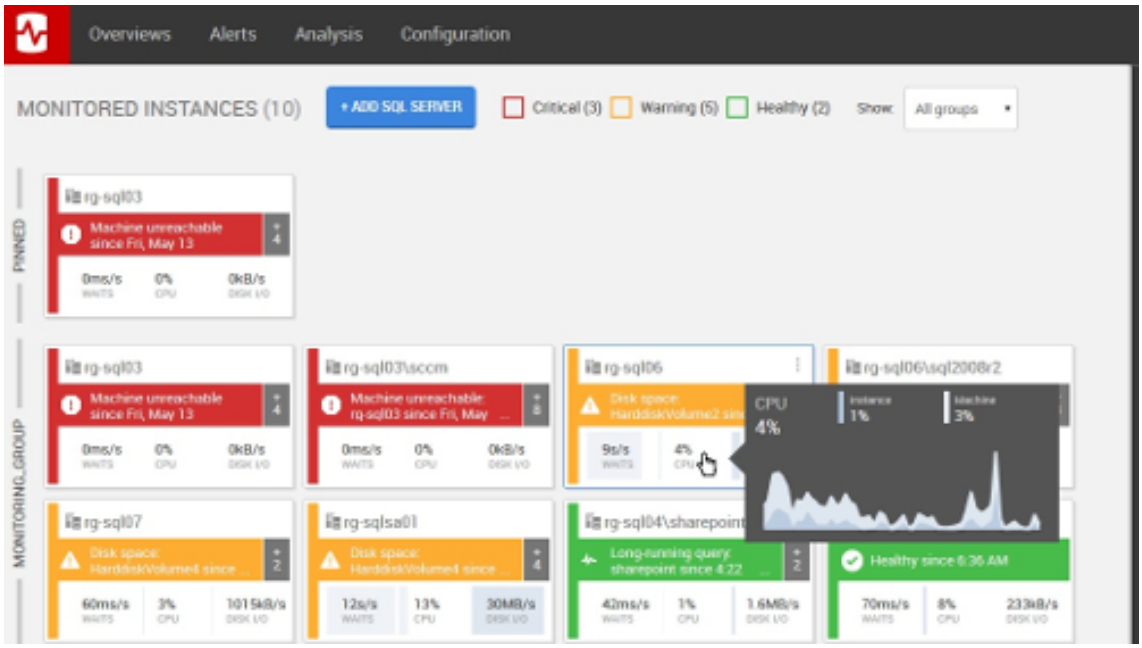


Figura 2.8: *Dashboard - SQL Monitor*

2.5.3.2 SQL Performance Monitor

O *SQL Performance Monitor* é uma ferramenta disponibilizada gratuitamente para a monitorização em tempo real da performance de uma instância de um servidor *SQL Server*.

Em termos de arquitectura e funcionamento da aplicação, tal como descrito em [Haq14], a aplicação não utiliza nenhum agente para recolha de dados, não é necessário realizar instalação ou configuração. É apenas necessário abrir a aplicação, adicionar informação relativa à instância que se pretende monitorizar e a recolha de valores para as métricas começa imediatamente. Outro ponto importante é o facto de armazenar os dados numa base de dados sqlLite.

Os pontos fortes desta solução é a simplicidade de uso, nomeadamente o facto de não ser necessário realizar instalação e a interface é também de fácil compreensão. A forma como a aplicação foi concebida permite facilmente relacionar diferentes tipos de dados o que facilita a tarefa de análise.

Os pontos fracos prendem-se com alguns factores, nomeadamente o facto de apenas monitorizar uma instância de cada vez, o facto de apenas recolher métricas quando o utilizador está com a aplicação aberta é também um ponto contra esta solução no sentido em que não é criado um histórico constante das métricas monitorizadas e se não existir um utilizador a observar constantemente não existe monitorização. Outro aspecto menos positivo, é o facto de ser uma solução maioritariamente reactiva pois não permite armazenar métricas constantemente. Os valores observados são guardados numa base de dados *SqlLite*, o que tendo em conta que as instâncias monitorizadas são *SQL Server* traz um trabalho extra de migração de dados se se pretender trabalhar sobre os dados armazenados.

A Figura 2.9 apresenta a interface de utilização do *SQL Performance Monitor*.

Figura 2.9: *Dashboard - SQL Performance Monitor*

2.6 Base de dados e Sistemas de Monitorização da Farfetch

A presente secção pretende descrever as soluções utilizadas actualmente na Farfetch, nomeadamente tecnologias de bases de dados e ferramentas para a monitorização das mesmas. Para armazenamento de dados a tecnologia utilizada é o *SQL Server* e para monitorização é utilizado o *Nagios* e o *New Relic*. Estas três tecnologias irão ser abordadas agora em detalhe.

2.6.1 Microsoft SQL Server

O SQL Server é um sistema de gestão de bases de dados relacionais cuja função principal é armazenar e retornar dados de acordo com os pedidos efectuados ao sistema. Tem todas as características associadas às bases de dados relacionais, no entanto tem um conjunto de especificidades que embora não estando no âmbito deste projecto importa destacar, a linguagem utilizada para obter dados que é o *T-SQL*, uma variante do *SQL* clássico. É disponibilizada também uma interface para configuração, gestão e administração de todos os componentes associados ao *SQL Server*. Esta ferramenta é o *SQL Server Management Studio* e a Figura 2.10 representa a interface dessa ferramenta.

A importância desta tecnologia no âmbito deste projecto prende-se com o facto de os servidores e as bases de dados a monitorizar utilizarem esta tecnologia, para além disto a ferramenta a desenvolver terá grande parte da sua estrutura também nesta tecnologia.

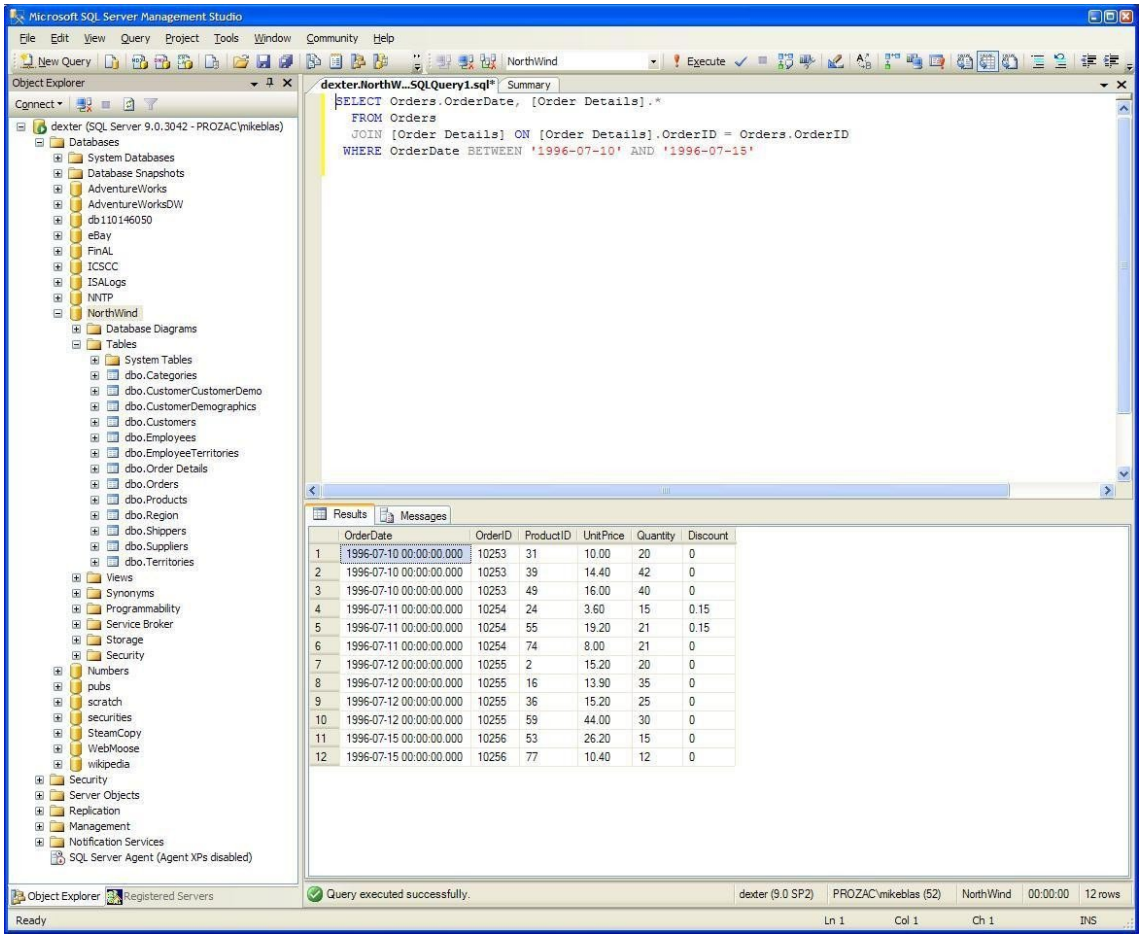


Figura 2.10: *SQL Server Management Studio*

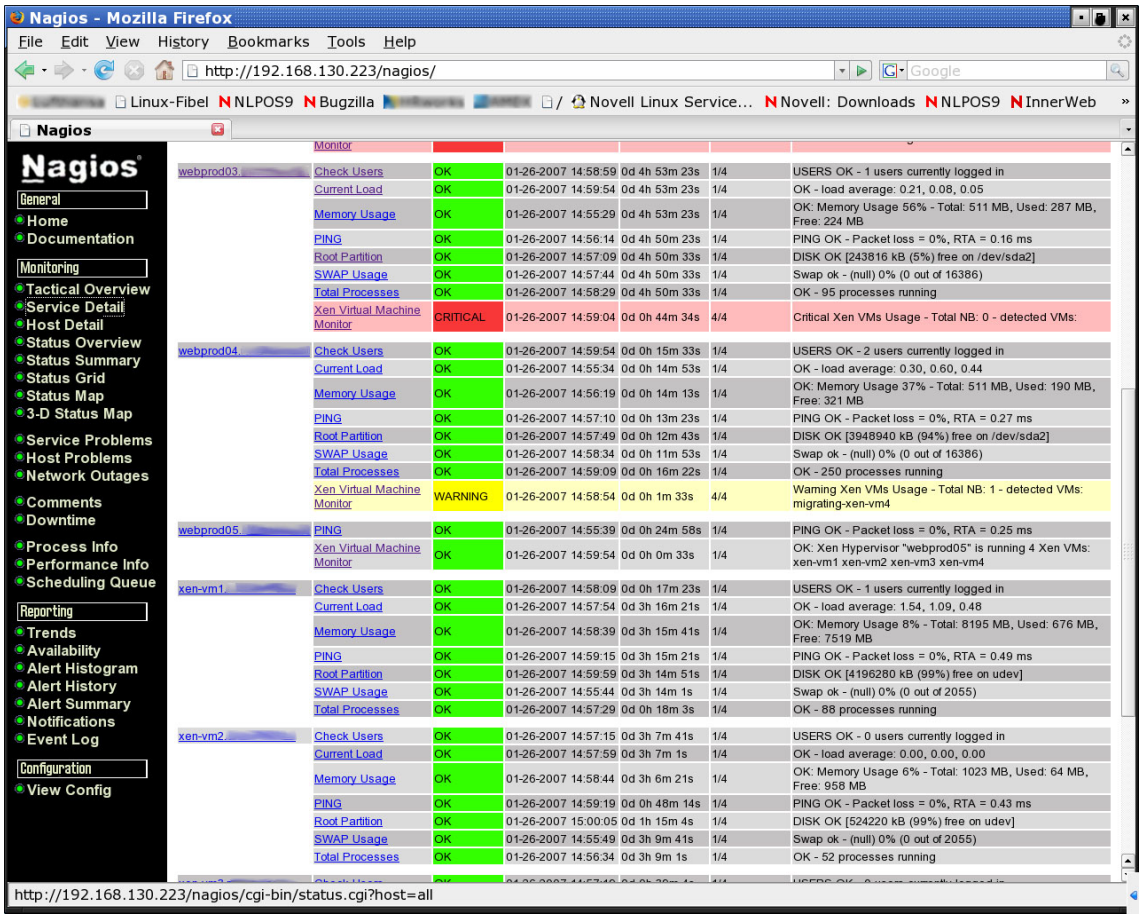


Figura 2.11: *Nagios Interface*

2.6.2 Nagios

O Nagios é um software de monitorização de infraestruturas e é actualmente utilizado na Farfetch.

O Nagios assenta num paradigma reactivo e tem por base um sistema de alarmística. São configurados pesquisas sobre as infraestruturas monitorizadas que são executadas em intervalos de tempo definidos no processo de configuração. Quando os resultados das pesquisas ultrapassam os limites definidos, são enviados alertas e as equipas de suporte respondem a esses alertas. Existem diferentes níveis de alarmística em que cada um tem um valor limite associado para ser possível atacar problemas com mais tempo de antecedência. No entanto é uma ferramenta de monitorização que não permite análise sobre os dados acumulados tendo apenas a função de alertar em caso de problema.

Uma das suas vantagens é o facto de ser altamente configurável, no entanto, o objectivo da ferramenta é completamente distinto do que se pretende desenvolver no âmbito desta dissertação.

Um exemplo de interface do *Nagios* é apresentado na Figura 2.11.

2.6.3 New Relic

O *New Relic* é uma ferramenta de monitorização. A sua tecnologia é disponibilizada no modelo software como serviço e monitoriza aplicações Web e Mobile em tempo real. Os dados são apresentados na plataforma da empresa e permite aos utilizadores a utilização de *plugins* para monitorizar diferentes aspectos das suas aplicações.

O *New Relic* permite monitorizar um conjunto variado de métricas de acordo com a configuração definida pelo utilizador, permite ainda um nível de detalhe bastante elevado sendo possível seguir todos os passos efectuados por um pedido à base de dados, desde a aplicação até à base de dados. Outra funcionalidade interessante é o facto de armazenar os dados recolhidos e ser possível analisar dados históricos, no entanto não fornece nenhum mecanismo de comparação relativamente a métricas de bases de dados que permita comparar performance entre intervalos de tempo distintos.

Existe uma funcionalidade particularmente interessante, que surge da possibilidade de ligar a recolha de informação às mais variadas fontes, em que é disponibilizada a informação do dia e hora em que foi instalada uma nova versão de uma aplicação, alterações à sua base de dados incluídas. Esta informação permite analisar a performance antes da alteração e após a alteração o que é uma grande mais valia para o processo de análise de performance. No entanto, o facto de não disponibilizar modos de comparação mais desenvolvidos torna esta análise um processo moroso e repetitivo.

A Figura 2.12 é ilustrativa da interface da plataforma do New Relic.

2.7 Resumo e Conclusões

Após a leitura deste capítulo espera-se que sejam claros os conceitos que constituem a base do trabalho desenvolvido, desde os paradigmas de monitorização, técnica de processamento da *baseline* até aos possíveis impactos de uma ferramenta como a que foi desenvolvida. Foram ainda descritas ferramentas disponíveis para o desenvolvimento do projecto bem como ferramentas de monitorização disponíveis no mercado.

No próximo capítulo será abordada a arquitectura definida para a ferramenta desenvolvida, bem como a estrutura interna responsável pela recolha das métricas. Serão também apresentadas as métricas recolhidas pela ferramenta.

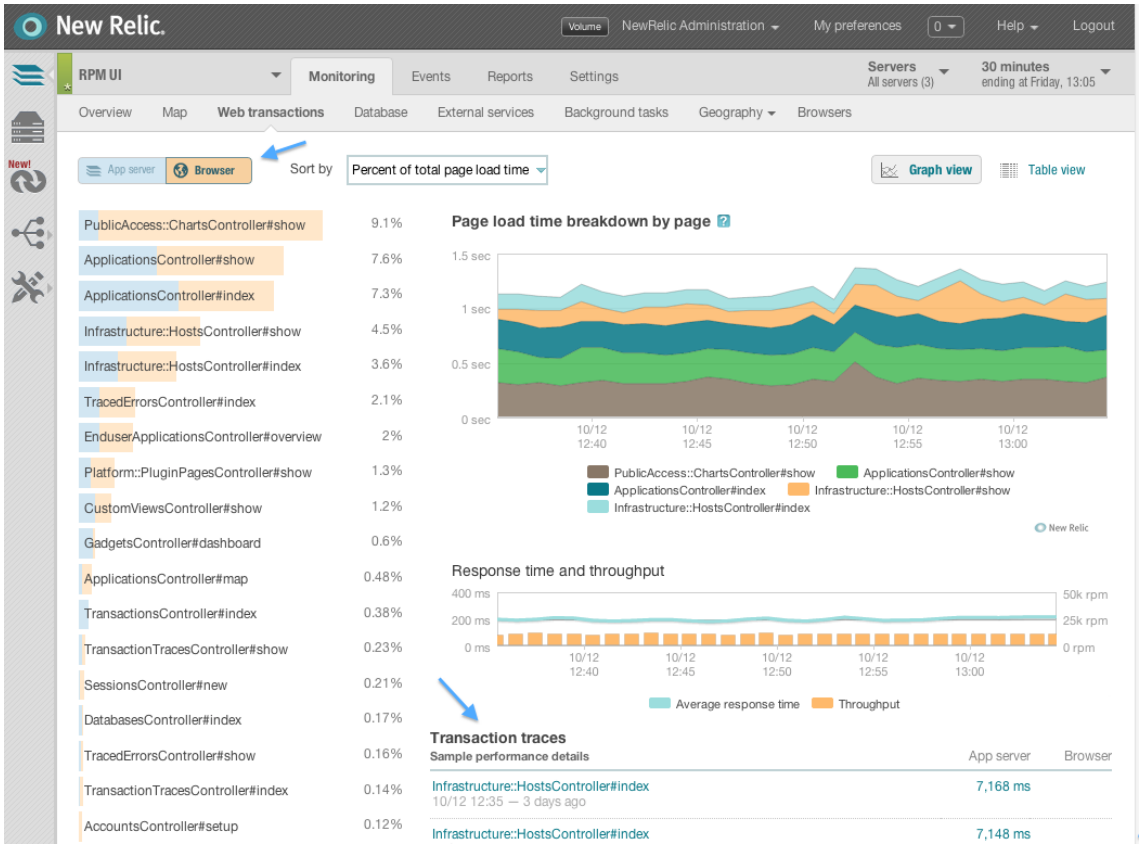


Figura 2.12: *New Relic Interface*

Capítulo 3

Arquitectura

Este capítulo descreve detalhadamente o problema a abordar, a arquitectura da solução implementada, nomeadamente a interacção entre componentes e o fluxo de dados entre esses mesmos componentes. São ainda descritos os indicadores de performance recolhidos e o seu significado.

3.1 Definição do Problema

O problema em análise neste trabalho é a performance de bases de dados, nomeadamente a performance de bases de dados relacionais, neste caso específico das bases da dados da Farfetch mas pode ser aplicado a qualquer base de dados desde que siga o modelo relacional, com as respectivas adaptações ao modelo de negócio que estiver a ser analisado e à tecnologia das respectivas bases de dados.

Pretende-se identificar as métricas que melhor representem a performance associada a uma base de dados relacional, por forma a monitorizar a performance de uma instância *SQL Server* e das bases de dados existentes nela.

No caso especifico serão analisadas as bases da dados da Farfetch que suportam tanto as equipas de desenvolvimento como o negócio da empresa. Iremos analisar bases de dados que suportam uma plataforma de venda electrónica com milhões de utilizadores.

É importante perceber o tipo de negócio para perceber por completo o problema, nomeadamente, existem padrões de actividade associados ao modelo de negócio da empresa que afectam directamente a performance das bases de dados. Esses mesmos padrões de actividade são indicadores das métricas que têm de ser analisadas, por exemplo um aumento do número de utilizadores leva a um aumento do número de vendas, posteriormente as operações sobre a base de dados para tornar persistentes os dados relativos a essas mesmas vendas podem não estar optimizados e em elevado número provocarem um impacto negativo na performance.

O problema que este estudo se propõe a resolver é então a identificação das métricas que melhor representem a performance de uma base de dados associada a um modelo de negócio.

Para a resolução do problema é estudado o que o sistema a desenvolver deveria fazer e de que forma o deveria fazer. O trabalho desenvolvido foi um trabalho de Engenharia de Requisitos que se traduz num conjunto de requisitos funcionais e não funcionais que detalham o sistema a desenvolver para abordar o problema.

São agora apresentados os requisitos que foram identificados, utilizando *User Stories*.

Requisitos funcionais:

- Como utilizador quero poder escolher os servidores a monitorizar.
- Como utilizador quero escolher o servidor cujas métricas serão apresentadas.
- Como utilizador quero poder escolher um intervalo específico de tempo e observar apenas valores relativos a esse intervalo.
- Como utilizador quero poder comparar valores de métricas entre dois intervalos de tempo.
- Como utilizador quero poder escolher uma base de dados específica para análise.
- Como utilizador quero que as métricas me sejam apresentadas num formato de fácil análise.
- Como utilizador quero que me sejam apresentadas métricas relativas ao estado de saúde do servidor.
- Como utilizador quero ter acesso a métricas específicas por base de dados.

Requisitos não funcionais:

- O sitema deve ser centralizado.
- O sitema deve ser capaz de monitorizar em paralelo múltiplos servidores.
- O sistema deve ser instalado de forma automática.
- O sistema deve funcionar de forma automática.
- O sistema deve permitir adicionar novos servidores para monitorização a qualquer momento.
- O sistema deve provocar o mínimo possível de impacto nos servidores monitorizados.
- O sistema deve efectuar o processamento e a apresentação dos relatórios de forma eficaz e rápida, para que a experiência de utilização seja positiva.
- O sistema deve ser o menos intrusivo possível nos servidores monitorizados.

Estando o problema definido, as próximas secções deste capítulo vão descrever a arquitectura pensada para a solução.

3.2 Arquitectura

Esta secção descreve a arquitectura da solução encontrada para o problema descrito anteriormente. Será apresentado um diagrama de componentes *UML* representativo da arquitectura e uma descrição de cada componente da solução desenvolvida.

A figura 3.1 representa a arquitectura pensada para a solução de um ponto de vista global.

São então três os componentes da solução pensada. *Monitoring Server* é o ponto central do sistema desenvolvido, é aqui que são armazenados os dados recolhidos que mais tarde fornecem os dados ao modelo de visualização e a estrutura de recolha de dados está também implementada neste servidor. É também aqui que é realizada toda a lógica de tratamento dos valores recolhidos e processamento de dados. Por uma questão de aproveitamento de recursos no caso do trabalho desenvolvido o *Report Server*, onde está alojado o modelo de visualização dos dados, encontra-se também no *Monitoring Server*, idealmente estaria num servidor separado.

Monitored Servers representa o conjunto de todos os servidores cujas bases de dados irão ser monitorizadas, contêm uma parte da estrutura de recolha de dados, dado que para a recolha de algumas métricas foi necessário efectuar a recolha localmente por questões de performance e impacto nos servidores.

O componente final é o *Web Browser* que será o modo de utilização da ferramenta.

Esta é a arquitectura que foi planeada e implementada, irão agora ser descritos em detalhe as métricas monitorizadas bem como cada componente da arquitectura.

3.3 Métricas Monitorizadas

Esta secção descreve as métricas escolhidas para monitorização, a informação que as mesmas representam e o porquê da sua escolha.

O conjunto de métricas monitorizadas pela solução desenvolvida divide-se em dois grandes grupos: métricas de análise do estado de saúde do servidor e métricas específicas associadas às bases de dados. Apesar do âmbito deste trabalho ser indicadores de performance de bases de dados foi tomada a decisão de monitorizar também o estado de saúde do servidor como um todo, por forma a oferecer ao utilizador uma visão global da instância onde se encontram as bases de dados uma vez que o somatório das performances das bases de dados individuais reflecte-se directamente na performance global da instância. Por este motivo são recolhidas as seguintes métricas de análise do estado da instância:

- *CPU Usage* — esta métrica representa a utilização do processador na máquina, nomeadamente a utilização por parte do SQL Server e a utilização por parte de todos os outros processos que corram na máquina. Valores elevados de utilização de processador por parte do *SQL Server* podem indicar que estão a ser executadas operações que causam elevado impacto, execução de tarefas internas de sistema em conjunto com a carga normal do servidor ou compilação e re-compilação excessiva de planos de execução de pesquisas de acordo com [Mic16b] e [Dav]

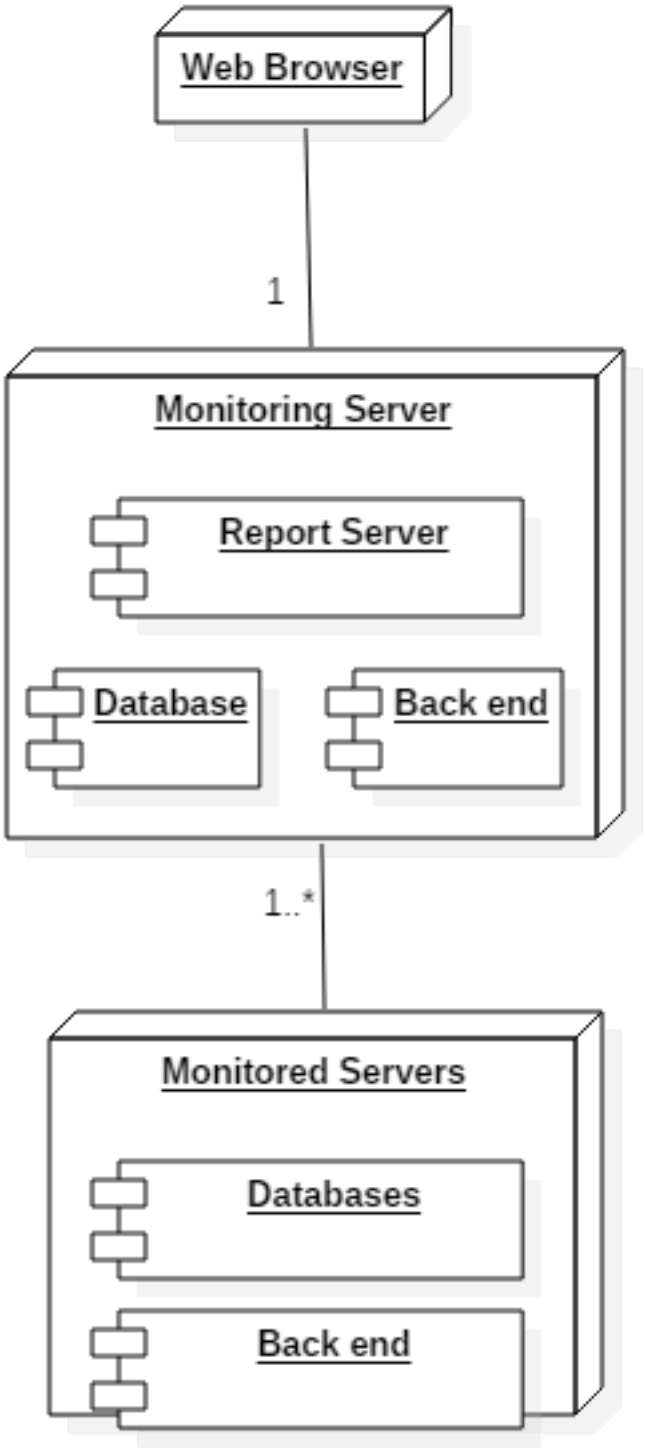


Figura 3.1: Arquitectura da Solução

- *Wait Statistics* — como descrito em [KS14] este conjunto de métricas representam uma parte fundamental da análise de performance de um servidor SQL porque servem como ponto de partida para a análise de performance. Permitem identificar possíveis estrangulamentos e bloqueios entre processos. São apresentados os tipos de espera com maior valor acumulado no período de tempo em análise, sendo aprentados os tipos de espera, o valor acumulado desse tipo de espera bem como o valor médio por identificação. É ainda apresentado o valor percentual da métrica em relação a todas as esperas registadas no intervalo de tempo definido.

- *Active Sessions* — esta métrica retorna o número de sessões activas na instância monitorizada. Como descrito em [DF10] a informação relativa a sessões é obtida através da ligação entre duas vistas de sistema, nomeadamente *sys.dm_exec_connections* e *sys.dm_exec_sessions*. Em conjunto estas vistas fornecem informação sobre o que está a ocorrer na instância, a primeira vista fornece informação centrada na rede enquanto que a segunda tem o seu escopo ao nível do servidor. Um ponto a ter em conta é o facto de uma conexão ao servidor poder gerar várias sessões dentro do mesmo.

- *Active Sessions Cpu* — através da monitorização desta métrica obtemos dados relativos à utilização de tempo de processador por parte das sessões activas no servidor, sendo possível perceber a carga de trabalho que está a ser gerada pelas ligações externas ao servidor. Estes dados estão disponíveis, como descrito em [DF10] na vista de sistema *sys.dm_exec_sessions*.

- *Active Sessions Memory* — esta métrica representa a memória utilizada por todos os pedidos associados a uma sessão. Os valores estão disponíveis na vista de sistema *sys.dm_exec_sessions* e através de um somatório é possível perceber os consumos em termos de memória provocado pelas ligações externas ao servidor. Tal como descrito em [DF10] os valores presentes nesta vista são actualizados quando um pedido associado a uma sessão é terminado, podendo dessa forma não representar exactamente o valor actual mas fornecer uma imagem da actividade realizada no servidor com alguns segundos de atraso.

- *Active Sessions Reads, Writes and Logical Reads* — este conjunto de dados representa o impacto provocado pelas sessões em termos de leituras e escritas. Apresenta dados relativos a *Logical Reads* que representa o número de leituras realizadas sobre a *cache*, *Reads* e *Writes* que representam operações de leitura e escrita realizadas directamente sobre o disco. Os valores permitem perceber os valores de *I/O* presente no sistema, através destes valores podem ser tiradas considerações relativamente a possiveis problemas de performance. Um exemplo será, valores elevados de *Logical I/O*, que podem explicar uma quebra de performance na execução de operações, uma vez que o *SQL Server* pode ser forçado a libertar memória para efectuar as operações. Tendo em conta que os planos de execução são guardados em memória, uma eventual eliminação força o servidor a compilar novos planos de execução, o que provoca impacto nos tempos de execução. Os valores para estas métricas são obtidos através da vista de sistema *sys.dm_exec_sessions*.

- *Server Memory* — esta métrica engloba dois valores distintos: *Total Server Memory* e *Target Server Memory*, que representam valores de memória para o servidor. Mais especificamente e de acordo com a descrição presente em [Pet14a], o valor de *Total Server Memory* representa a memória actualmente atribuída ao *SQL Server* enquanto que o valor de *Target Server Memory* representa a quantidade de memória necessária para que o *SQL Server* tenha a melhor performance. Só faz sentido a análise a estas duas métricas se a mesma for realizada em conjunto, uma vez que é através do rácio entre os dois valores que alguma conclusão pode ser retirada. O valor do rácio entre estas duas métricas deve ser idealmente superior ou muito próximo de 1. Quando não o é pode indicar problemas de memória uma vez que o *SQL Server* não consegue obter a memória que necessita. Convém também verificar o valor da opção *Maximum server memory* pois se o mesmo for baixo pode ser a causa do problema. Os valores para estas métricas são retirados da vista de sistema *sys.dm_os_performance_counters*.

- *Cache Hit Ratio* — esta métrica fornece informação relativa à performance em termos de memória. O seu valor representa o rácio entre o número de páginas encontradas em memória em relação ao número total de pedidos como descrito em [Pet14b]. Se o valor for de 1 todas as páginas são encontradas em memória, quando o valor é inferior significa que alguns pedidos tiveram que obter os dados directamente do disco, o que em si é um processo muito mais demorado e custoso em termos de performance.

- *Memory Grants Pending* — esta métrica, tal como o nome indica, representa o número de processos que estão à espera que lhes seja concedida memória para realizarem o seu trabalho. Tal como descrito em [Pet14c] o valor desta métrica deverá ser constantemente zero, no entanto, se o mesmo não se verificar podem ser tomadas medidas adicionais para perceber o problema. A vista de sistema *sys.dm_query_memory_grants*, de acordo com [Mic16c], oferece informação relativamente às operações que requisitaram memória e estão de momento à espera ou que já receberam a memória desejada, permitindo assim navegar um nível extra na tentativa de perceber o problema.

- *Lazy Writes/sec* — esta métrica envolve o conceito de *Database Checkpoints*. De acordo com [Mic16a], é um ponto no tempo em que as páginas alteradas em memória bem como informação relativa ao *transaction log*, são escritas de memória para disco, criando um ponto de retorno em caso de problemas na instância. *Lazy Writes/sec* é definido segundo [Micg] como o número de vezes por segundo que o SQL Server realoca *dirty pages* de memória para disco. Isto significa que entre dois *Database Checkpoints* o *Lazy Writer* vai libertando páginas em memória para evitar a necessidade constante de *Checkpoints*. Este processo consiste em escrever para disco as páginas que sofreram alterações e marcá-las como livres em memória.

- *Page Life Expectancy* — de acordo com [Pet14b] esta métrica representa o período de tempo, em segundos, que uma página permanece no *buffer* de dados. O valor normal apresentado como referência, 300 segundos, é um valor afastado da realidade atual, já que em

termos de *hardware* hoje em dia os valores médios são bastante mais elevados. No entanto a *Page Life Expectancy* é um valor completamente dependente das caracteristicas do sistema em monitorização, por isso mesmo não é possível definir um valor global de comparação para este valor.

- *Page Lookups/sec* — esta métrica tal como definida em [Micg], representa o número de pedidos por segundo que encontram a informação pretendida em memória. A informação fornecida torna-se relevante em correlação com valores de outras métricas, nomeadamente se este valor for bastante inferior ao *Page Reads/sec* pode indicar que a informação que se encontra em memória não é a mais adequada em relação às pesquisas efectuadas sobre a instância, o que pode indicar um problema na rotação das páginas em memória, daí que analisar em conjunto com *Lazy Writes/sec* possa ajudar a perceber se a rotação das páginas está a ser realizada correctamente.

- *Page Reads/sec* — tal como definida em [Micg], os valores desta métrica representam o número de leituras efectuadas directamente sobre o disco por segundo, relativo a todas as bases de dados presentes no servidor. O valor desta métrica é indicativo da quantidade de actividade existente no servidor e tal, como para a métrica *Page Lookups/sec*, tem de ser analisada em conjunto com outros valores. Tendo em conta que *Physical I/O* tem um custo superior ao *Logical I/O*, esta é uma métrica que deve ser constantemente inferior aos valores apresentados pela métrica *Page Lookups/sec*.

- *Page Writes/sec* — esta métrica representa o número de páginas escritas em disco por segundo, de acordo com [Micg]. Este valor é proveniente da inserção de novos dados no sistema que são persistidos nas bases de dados e é importante acompanhar este valor uma vez que representa uma parte importante das operações de *I/O* que são efectuadas sobre o sistema.

- *Transactions/sec* — esta métrica representa o número de transações efectuadas por segundo no servidor. Como descrito em [Micl], uma transação é um conjunto de operações tratadas como uma única unidade de trabalho e são utilizadas para garantir a integridade dos dados enquanto são efectuadas alterações sobre eles. Esta métrica é um indicador do nível de actividade do servidor.

- *Batch Requests/sec* — como definido em [Mich], esta métrica representa o número de pedidos recebidos por segundo. A importância da monitorização desta métrica prende-se com o facto de ser um indicador da actividade global do servidor, bem como um indicador da velocidade a que o *SQL Server* está a processar os pedidos de utilizadores. Esta métrica é o principal indicador do nível de actividade de um servidor, sendo afectada por várias variáveis, como *I/O* e número de utilizadores entre outras, o que indica que é um excelente ponto de partida para identificação de possíveis problemas de performance num servidor, uma vez que uma flutuação deste valor implica uma alteração em uma das variáveis que o afectam.

- *SQL Compilations* — a compilação do plano de execução representa grande parte do tempo de execução de uma *query*. De forma a poupar esse tempo, os planos de execução são guardados em memória por forma a que sejam utilizados quando a *query* voltar a ser executada, como é descrito em [Mich]. Esta métrica representa o número de compilações executadas por segundo, ou seja o número de operações que foram executadas e para as quais foi criado e guardado em memória um plano de execução. Um aumento nos valores registados desta métrica pode indicar problemas de memória, uma vez que os planos estão a ser removidos para dar lugar a outros dados.

- *SQL Re-compilations* — esta métrica representa o número de re-compilações por segundo e tem por base o mesmo conceito da métrica *SQL Compilations*, os valores associados a esta métrica devem ser baixos, uma vez que são indicativos da re-compilação de planos já existentes e que por algum motivo deixam de poder ser utilizados pelo *query optimizer*. Esta métrica é recolhida da vista de sistema *sys.dm_os_performance_counters*.

- *Deadlocks* — tal como é definido em [Mica] um deadlock acontece quando duas tarefas distintas se bloqueiam mutuamente. O que acontece é que uma tarefa adquire um bloqueio num recurso enquanto que a segunda tarefa adquire um bloqueio num outro recurso, entretanto a primeira tarefa tenta obter um bloqueio no recurso que esta bloqueado pela segunda tarefa, enquanto que a segunda tarefa tenta obter um bloqueio no recurso que está bloqueado pela primeira. Isto leva a que nenhuma das tarefas consiga concluir, até que uma seja terminada. Apesar do *SQL Server* possuir um mecanismo de detecção e resolução de *Deadlocks*, que consiste em terminar aleatoriamente uma das duas tarefas bloqueadas, *Deadlocking* ainda representa um impacto significativo de performance. O que esta métrica representa é o número de *deadlocks* existentes no momento da medição e é obtida através da vista de sistema *sys.dm_os_performance_counters*.

- *Query Execution Statistics* — este ponto representa um conjunto de métricas por representar uma análise às estatísticas das queries executadas no servidor que são alvo de variadas análises. Tal como descrito em [DF10], os dados são obtidos através da vista de sistema *sys.dm_exec_query_stats* em conjunto com a função de sistema *sys.dm_exec_sql_text* que permite obter o texto da *query* em questão, existem alguns detalhes importantes de implementação na recolha destas métricas (que serão abordados no próximo capítulo) que ajudam a perceber o valor que estas métricas permitem retirar. Foi efetuada uma divisão por três tipos de análise, nomeadamente por consumo de *CPU*, *I/O* e pelo número de execuções.

  De notar que são apenas apresentados resultados, para as três variantes, relativos às 10 operações com valores mais altos do filtro escolhido, estando os resultados ordenados de forma descendente. Os resultados apresentados são referentes às queries executadas no intervalo de tempo definido.

Relativamente às métricas de análise de performance de bases de dados, as métricas escolhidas para recolha e análise são as seguintes:

- *Calling Services* — os valores apresentados por esta métrica permitem saber que serviços estão a executar chamadas à base de dados em análise. Com estes valores é possível saber à partida o tipo de actividade que está a ser executada na base de dados.

- *Requests/sec* — esta métrica apresenta o número de pedidos por segundo que estão a ser efectuados à base de dados, permite perceber a quantidade de actividade que está a ser realizada sobre a base de dados.

- *Average Response Time* — esta métrica representa o tempo médio de resposta a cada pedido realizado à base de dados. Por tempo médio de resposta entenda-se todo o tempo gasto, desde o tempo de execução da *query* a todos os tempos de espera que possam estar associados à execução da mesma.

- *Database Activity* — no painel de *Database Activity* são apresentados valores que permitem perceber que tipo de actividade está a decorrer na base de dados, nomeadamente são apresentados os tipos de pedidos que ocorreram naquele período de tempo, o número de ocorrências por minuto e qual a expressão daquele tipo de pedidos em termos percentuais em relação a todos os pedidos realizados no intervalo de tempo definido. É ainda apresentada a percentagem de tempo consumida pelos pedidos de cada tipo, em relação ao tempo geral gasto na execução de todos os pedidos. Este conjunto de métricas permite ter uma ideia bastante clara do tipo de actividade a ocorrer na base de dados e do peso de cada um desses tipos num contexto geral.

- *Database I/O* — os dados apresentados nesta métrica são relativos à quantidade de operações de escrita e leitura realizada sobre a base de dados no intervalo de tempo determinado.

- *Table sizes analysis* — este painel apresenta informação relativa às maiores tabelas presentes na base de dados, as que contêm mais dados, e apresentam os seguintes dados relativamente às tabelas e aos índicaes criados sobre as mesmas:

  - *Allocated Space MB* — esta métrica apresenta os valores relativos ao espaço físico alocado para a tabela, independentemente de estar totalmente ocupado ou não.
  - *Data Size MB*, esta métrica apresenta o valor associado ao tamanho da tabela em termos de dados.
  - *Average Data Growth* — esta métrica é calculada a partir da variação do tamanho dos dados guardados na tabela desde a primeira medição até à última, devolvendo o valor médio de crescimento da tabela por dia.
  - *Index Size MB* — esta métrica apresenta o valor associado ao tamanho dos índices associados à tabela.
  - *Average Index Growth* — o funcionamento desta métrica é semelhante ao descrito para a métrica *Average Data Growth* mas está aplicado ao tamanho dos índices associados à tabela.

  - *Percentage of Database* — esta métrica devolve a representação, em formato percentual, do tamanho da tabela no contexto da base de dados em análise.

- *Request Analysis* — este ponto engloba um conjunto de métricas relacionadas com os pedidos que são feitos às bases de dados. O *SQL Server* armazena dados relativos às queries executadas ao nível do servidor. Desse modo para ser possível aprofundar o nível de análise foi necessário criar um mecanismo de recolha dos pedidos realizados directamente a cada base de dados. Para a recolha desses dados é utilizada a informação contida na vista de sistema *sys.dm_exec_requests* descrita em [Micj], em conjunto com a função de sistema *sys.dm_exec_sql_text*. A vista *sys.dm_exec_requests* guarda dados relativos a pedidos que estão a ser executados ou em espera para serem executados, quando um pedido acaba a sua execução, a sua informação é removida da vista, daí que seja necessário obter dados de forma constante e processar esses mesmo dados de forma a evitar duplicados. A análise dos pedidos sobre a base de dados permite filtrar a informação por utilização de *CPU*, quantidade de *I/O* e número de execuções. São apresentados os 10 pedidos com os valores mais altos relativamente ao filtro seleccionado.

  Além destas métricas ainda é possível aprofundar a análise mais um nível e analisar a informação de cada execução dos pedidos apresentados. Ou seja, para cada pedido apresentado no painel de *Request Analysis* é possível obter métricas sobre cada execução do mesmo no intervalo de tempo definido.

  São ainda apresentadas métricas globais dos tempos de espera associados às várias execuções do pedido, nomeadamente o *Tipo de espera*, *Tempo de espera* e *Tempo médio de espera*.

Este conjunto de métricas foi alcançado através do equilibrio entre o que reflecte melhor a performance dos servidores, o modelo de negócio da empresa e a tentativa de minimizar ao máximo o efeito de observador sobre os sistemas monitorizados. O processo de recolha e armazenamento das métricas descritas bem como a metodologia de processamento da *baseline* irá ser abordado ao longo dos próximos pontos deste capítulo, no entanto existem vários detalhes de implementação e técnicas utilizadas que apenas serão descritas no Capítulo 4.

3.4 Metodologia de Comparação

A metodologia de comparação refere-se à abordagem escolhida para a comparação entre dados de dois intervalos de tempo.

O tipo de dados de cada métrica e a forma como a própria métrica é calculada influencia directamente o método utilizado para comparação com a *baseline*.

Métricas cujos valores são retirados directamente das vistas de sistema são comparados com valores calculados através dos valores da baseline, nomeadamente a média, o primeiro desvio

padrão e o segundo desvio padrão. O objectivo é fornecer ao utilizador a possibilidade de aprofundar a análise até isolar os valores que representam um comportamento anormal em termos de performance.

Métricas cujo o valor tem um limite ou métricas cujo valor seja calculado a partir de dados extraídos do sistema são comparadas directamente, esta comparação é realizada através da criação de dois gráficos, um com os valores actuais e outro com os valores da *baseline*, tendo neste caso o utilizador de comparar visualmente e tentar encontrar as discrepâncias. Algumas métricas são comparadas directamente, uma vez que são métricas calculadas a partir de dados recolhidos e representam a actividade no período escolhido.

O processamento dos valores dos indicadores foi efectuado em intervalos de cinco minutos, ou seja a média, primeiro desvio normal e segundo desvio normal são calculados em intervalos de cinco minutos para que seja possível um maior nível de detalhe. A abordagem escolhida foi esta porque o cálculo ponto a ponto iria reflectir-se numa sobreposição dos valores de dois intervalos de tempo tornando a análise mais complexa de realizar e o cálculo destes valores para todo o intervalo de tempo definido iria retirar valor à solução uma vez que os valores iriam acabar por ser sempre mais uniformes.

O objectivo com o desenvolvimento desta funcionalidade é fornecer ao utilizador uma ferramenta que permita porcurar activa e agressivamente pelos pontos em que se verificou maior discrepância entre os dados históricos e os dados actuais.

3.5 Recolha de Dados

Pretende-se agora explicar o modelo de recolha de dados do sistema implementado, nomeadamente o modelo de dados e o *deploy* da estrutura de recolha de dados.

O modelo de dados desenvolvido é constituído por 29 tabelas de dados, é um modelo relacional e normalizado por forma a evitar redundância de dados. O modelo foi desenhado e implementado tendo por base o estudo realizado relativamente às métricas que se pretendem monitorizar, à relação entre essas métricas e aos métodos que seriam usados para recolha dos dados relativos às métricas. Todo o modelo está ligado no entanto pode ser dividido em três partes: tabelas centrais, tabelas de dados de monitorização da instância e tabelas de dados de monitorização de bases de dados.

Estruturalmente assenta nas tabelas *Config, DataCenters, Servers, ServerInstances, Instances, Databases, AvailabilityGroups e Files* que contém dados relativos à estrutura de servidores, instâncias e bases de dados a monitorizar. O povoamento destas tabelas é feita a partir da tabela *Config*, que funciona como tabela de configuração onde são inseridos os dados dos servidores a monitorizar.

A Figura 3.2 representa o conjunto de tabelas centrais.

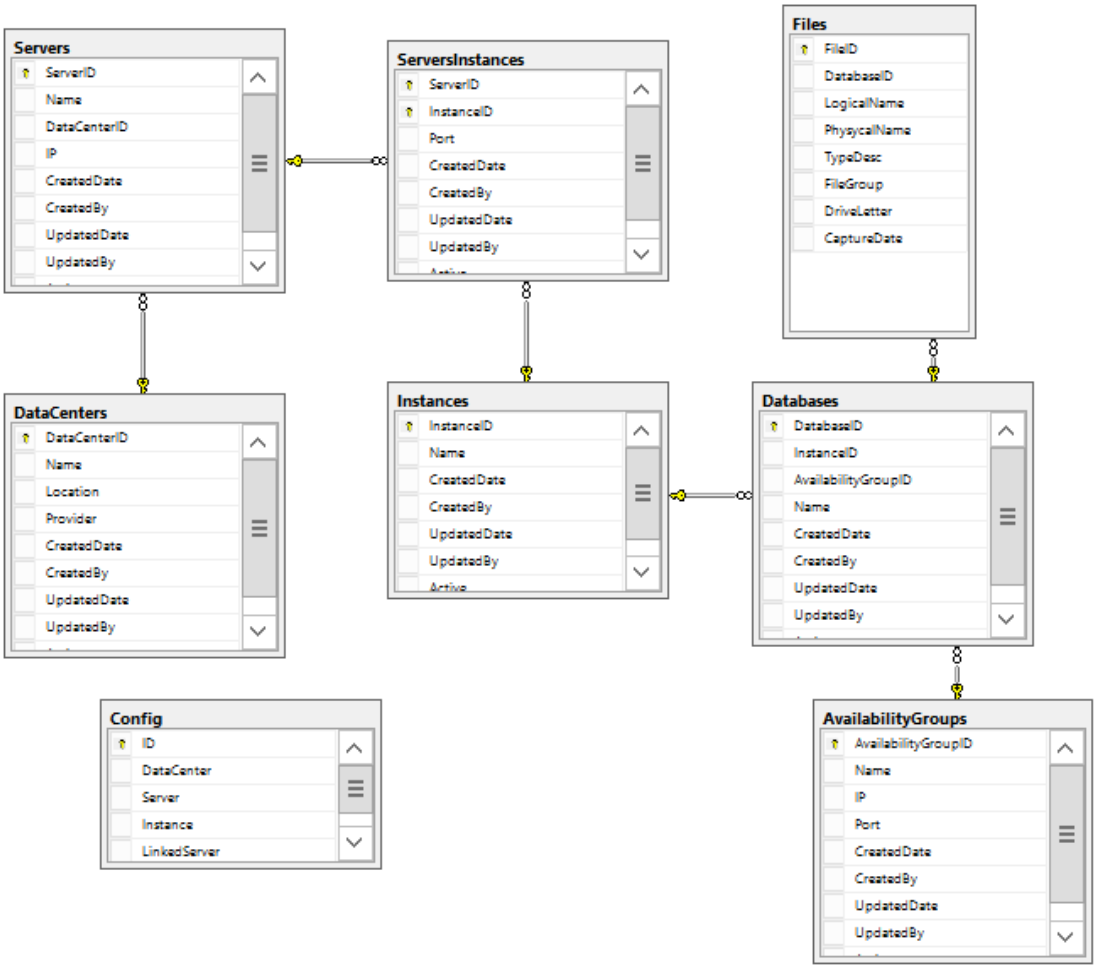


Figura 3.2: Modelo de Dados - Estrutura Central.

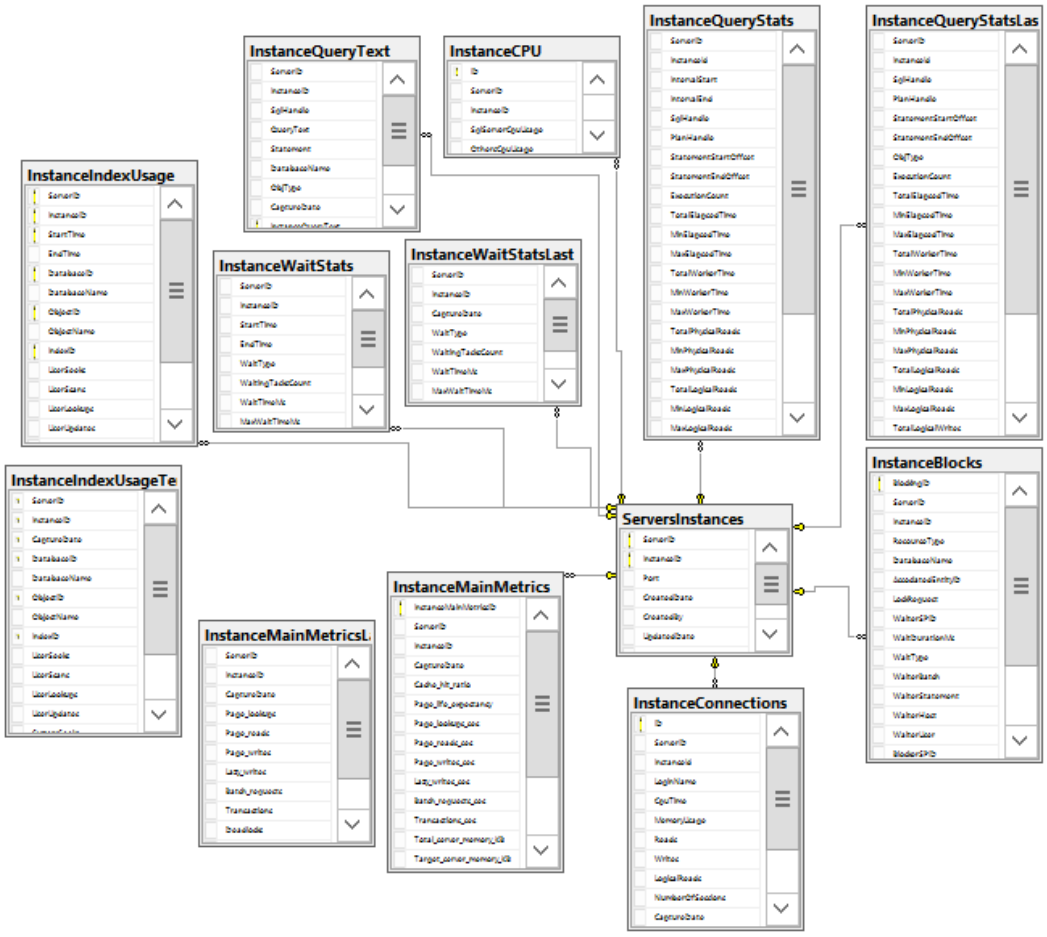


Figura 3.3: Modelo de Dados - Métricas Instância.

Os dados relativos à monitorização do estado de saúde dos servidores encontram-se nas tabelas *InstanceBlocks, InstanceIndexUsage, InstanceQueryText, InstanceConnections, InstanceWaitStats, InstanceQueryStats, InstanceMainMetrics e InstanceCPU*, existindo ainda as tabelas *InstanceQueryStatsLast, InstanceIndexUsageTemp, InstanceWaitStatsLast e InstanceMainMetricsLast* que são tabelas de *staging* usadas como suporte no processo de tratamento de dados antes do armazenamento nas tabelas finais.

A Figura 3.3 representa as tabelas de dados de monitorização da instância.

Dados relativos às bases de dados são armazenados nas tabelas *FilesSize, FilesIO, Tables, RequestInfo, DatabaseSession, RequestWaits, RequestText*, sendo as tabelas *FilesIOLast* e *RequestsStaging* usadas como tabelas de suporte ao tratamento de informação antes do armazenamento no modelo.

A Figura 3.4 representa o conjunto de tabelas onde são armazenados os dados de monitorização de bases de dados.

Após a escolha das métricas a monitorizar e a definição e implementação do modelo de dados, foi necessário delinear o método como os dados serão recolhidos e a forma como a estrutura de

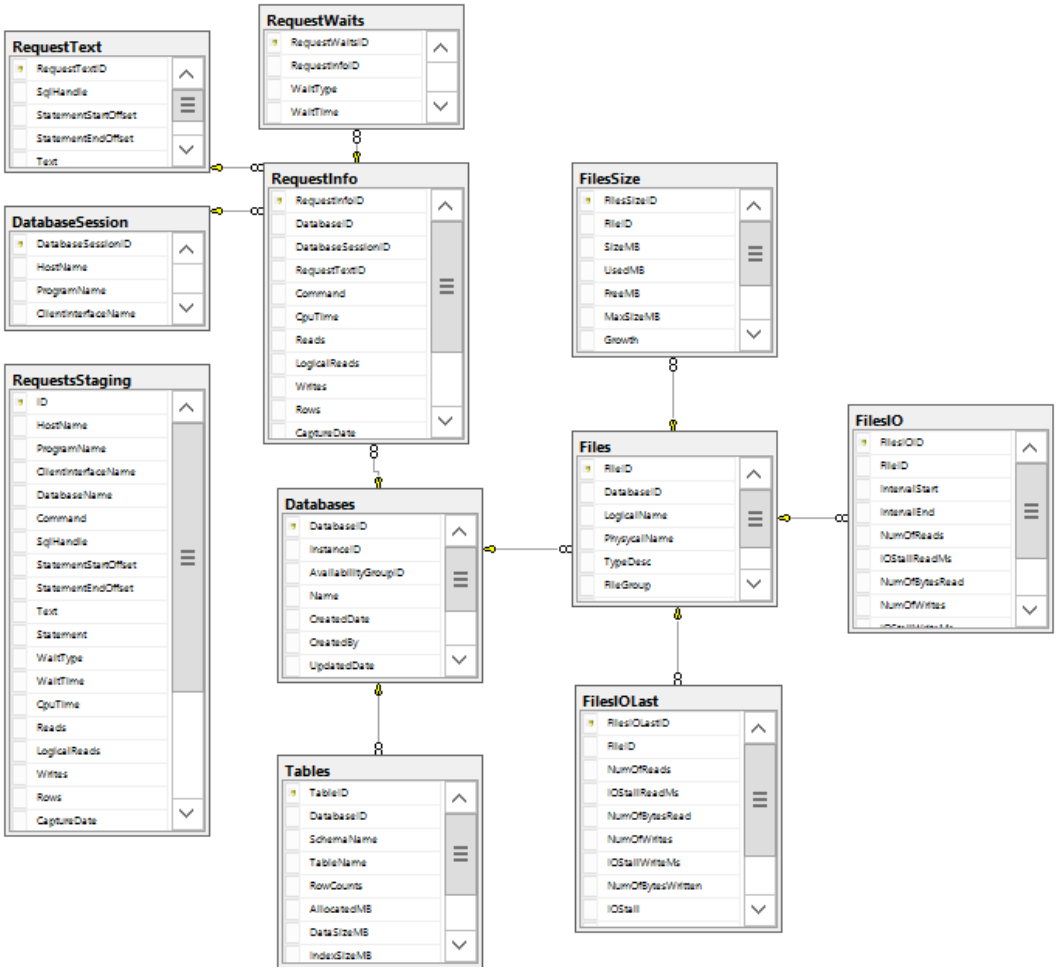


Figura 3.4: Modelo de Dados - Métricas Bases de Dados.

recolha de dados será implementada.

Como definido num dos objectivos do trabalho, a recolha de dados deveria ser centralizada. Assim sendo foi criado um processo que é executado apenas uma vez e que cria toda a estrutura necessária para a recolha de dados. Este processo é constituído por um conjunto de procedimentos e inicialmente percorre uma tabela de configuração que contém a informação dos servidores a monitorizar e cria todos os objectos necessários para a recolha das métricas.

Para cada servidor a monitorizar são executados um conjunto de procedimentos responsáveis por povoar o modelo de dados com a informação relativa ao servidor, criar uma ligação entre o servidor central e o servidor a monitorizar, povoar o modelo de dados com a informação das bases de dados existentes, povoar o modelo de dados com a informação relativa a cada ficheiro de dados e de *log* existentes em cada base de dados, criação de uma pequena estrutura no servidor a monitorizar para a recolha de métricas localmente e finalmente são também criados todos os *SQL Jobs* necessários para a recolha e tratamento dos dados.

Para complementar a estrutura são criados um conjunto de procedimentos adicionais cujo objectivo é a manutenção dos dados e a limpeza de dados antigos. O primeiro é responsável por garantir que a informação relativa às bases de dados de cada servidor está actualizada e o segundo é responsável por eliminar dados superiores a uma data definida.

A figura 3.5 representa as interacções presentes na execução deste processo, através de um diagrama de sequência *UML*.

3.6 Resumo e Conclusões

Este capítulo teve por objectivo dar a entender claramente qual o problema a abordar e a forma como foi abordado no desenvolvimento deste trabalho. Para tal, além da descrição do problema e das funcionalidades a cumprir, foi abordada a solução de um ponto de vista mais conceptual tendo sido descritos os conceitos utilizadas para o desenvolvimento técnico.

Com este objectivo em mente a solução foi descrita do ponto de vista da sua arquitectura e modelo de dados, foram também abordadas as técnicas utilizadas para promover a proactividade da solução e o fluxo dos dados dentro do sistema implementado. Foram também descritas as métricas recolhidas para efeitos de monitorização.

Espera-se que após a leitura deste capítulo seja possível ter uma ideia clara do funcionamento e estrutura da solução implementada, bem como do problema que a mesma se propõe a resolver.

No próximo capítulo será descrito todo o processo de implementação da ferramenta.

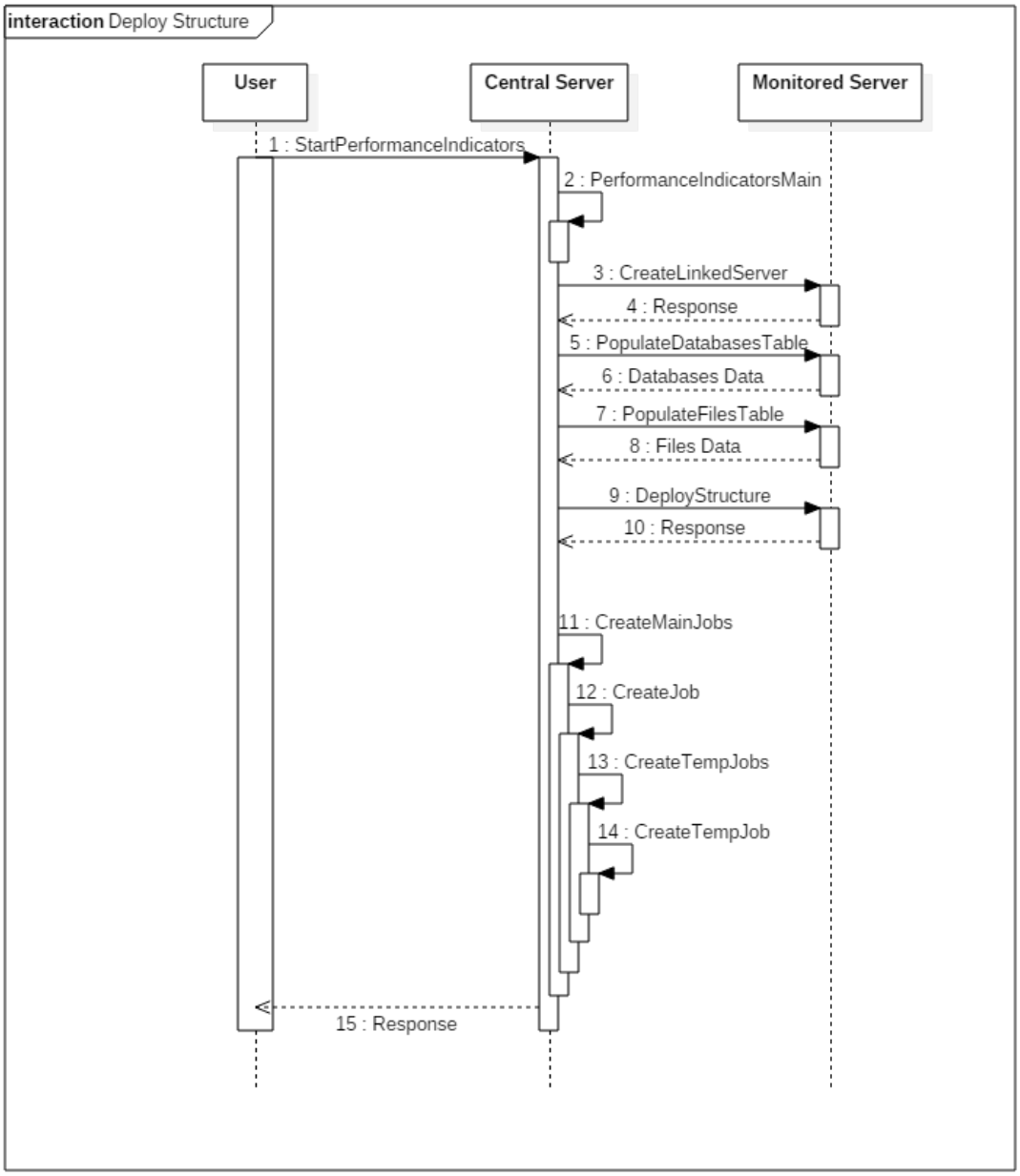


Figura 3.5: Diagrama de Sequência da Recolha de Dados.

Capítulo 4

Implementação

Após a descrição e definição conceptual da solução a implementar irá ser agora descrito o processo de implementação de todos os componentes descritos no capítulo anterior.

Inicialmente serão apresentadas as tecnologias utilizadas para o desenvolvimento da solução, bem como conceitos e técnicas utilizadas durante o desenvolvimento. De seguida será descrita a implementação dos diferentes componentes da ferramenta.

Irão também ser descritas as medidas tomadas para aumentar a performance da solução implementada.

4.1 Ferramentas e Técnicas

Foram utilizadas diferentes tecnologias para o desenvolvimento das diferentes partes da solução, no entanto toda a solução assenta sobre o *SQL Server*. As tecnologias utilizadas foram as seguintes:

- *SQL Server* — toda a solução assenta sobre o *SQL Server* pois as bases de dados monitorizadas estão em *SQL Server* e a monitorização é realizada a partir de vistas de sistema do próprio SQL Server e de capturas de eventos que ocorrem dentro do *SQL Server*. A solução desenvolvida utiliza então o *SQL Server* para alojar o modelo de dados, toda a estrutura de recolha de dados, descrita no Capítulo 3, foi também desenvolvida com esta tecnologia através da utilização de funcionalidades que a mesma disponibiliza. O tratamento e processamento dos dados é também feito com recurso ao *SQL Server*. A escolha desta tecnologia deriva da especificidade do projecto. Apesar de as métricas permitirem medir a performance de qualquer base de dados independentemente da tecnologia em que estão alojadas, existem pormenores de recolha das métricas associadas a cada uma das tecnologias daí que a escolha tenha recaído no *SQL Server* por forma a evitar integração de diferentes tecnologias.

- *Reporting Services* foi a tecnologia escolhida para desenvolvimento do *Dashboard* de apresentação dos dados relativos às métricas monitorizadas. A escolha recaiu sobre esta tecnologia por diferentes motivos, nomeadamente o facto de responder às necessidades identificadas, permitir apresentar os dados da maneira pretendida, ser bastante poderoso na granularidade que permite atingir em relação aos dados tal como referido no Capítulo 3. Outra das razões para a adopção desta tecnologia foi o facto de estar bem testada e documentada, o que aligeira substancialmente a curva de aprendizagem associada ao uso de qualquer tecnologia. O facto de ser uma solução disponível na Farfetch foi também um ponto a favor da utilização desta tecnologia.

- *Report Designer - Microsoft Visual Studio Data Tools* — existem duas soluções para o desenvolvimento de relatórios para *Reporting Services*, nomeadamente o *Report Designer* e o *Report Builder*. Em termos de capacidades de desenvolvimento as duas soluções permitem obter os mesmos resultados, no entanto os ambientes de desenvolvimento são completamente diferentes, daí que a escolha tenha recaído sobre o *Report Designer* uma vez que o desenvolvimento é feito através do *Visual Studio*, um ambiente bastante familiar, em vez do *Report Builder* no qual o desenvolvimento é realizado num ambiente *stand-alone* com uma interface própria. Em termos de funcionalidades de desenvolvimento as duas opções permitem atingir os mesmos objectivos, logo a escolha tem um carácter de preferência pessoal e neste caso foi escolhido o *Report Designer* pela familiaridade já existente com o ambiente de desenvolvimento.

Irão agora ser descritas algumas funcionalidades, técnicas de programação e alguns detalhes de implementação necessários para a compreensão do trabalho desenvolvido. Os tópicos abordados são *Stored Procedure*, *SQL Server Agent*, *Jobs*, *Linked Server*, *SQL* Dinâmico, autenticação e *Dynamic Management Views*.

Um *Stored Procedure* é um conjunto de operações *SQL* criado e armazenado na base de dados. É possível passar parâmetros para um *Stored Procedure*, o que permite que o mesmo *Stored Procedure* seja usado simultaneamente por vários clientes com dados de entrada diferentes. A utilização de *Stored Procedures* promove a redução de tráfego na rede e o aumento da performance. Se um Stored Procedure for alterado, todos os clientes irão ter acesso à versão actualizada.

Existem variadas vantagens associadas à utilização de Stored Procedures, nomeadamente:

- *Stored Procedures* permitem desenvolver programação modular.

- Após a criação e armazenamento do *Stored Procedure* na base de dados, este pode ser executado quantos vezes for desejado no código aplicacional, eliminando assim a duplicação de código.

- Em termos de execução, a performance dos *Stored Procedures* é superior.

- Aquando da primeira execução é gerado e armazenado em memória um plano de execução do *Stored Procedure*, o que significa que, ao contrário de pedidos *ad-hoc* o *Stored Procedure*

não precisa de ser compilado a cada execução o que resulta em tempos de execução bastante inferiores.

- O tráfego de rede é reduzido uma vez que num único *statement* são realizadas várias operações, em vez de enviar através da rede centenas de linhas de código.
- A utilização se *Stored Procedures* permite um controlo superior em termos de permissões atribuídas aos utilizadores, nomeadamente pode ser atribuída permissão de execução, ao utilizador, sobre o *Stored Procedure* sem que o utilizador tenha permissões de leitura e escrita de dados, o que minimiza o perigo de *SQL Injection*.
- Em termos de performance, permite uma maior facilidade de optimização, uma vez que o código executado está centralizado, logo apenas um ponto necessita de ser optimizado.

Existem no entanto pontos menos positivos associados à utilização de Stored Procedures, sendo que o principal é a manutenção associada às operações básicas de *CRUD*. Por exemplo se para cada tabela houver um *Insert*, *Update*, *Delete* e pelo menos um *Select*, significa que para cada tabela irão ser precisos 4 *Stored Procedures*. Escalando para uma base de dados com um tamanho considerável de 400 tabelas, teremos 1600 *Stored Procedures*. Existem opções alternativas, utilização de um *ORM* para gerar automaticamente as operações básicas de *CRUD*, para contornar este problema.

Em relação ao *SQL Server Agent* tal como descrito em [Micf], o *SQL Server Agent* é um serviço que executa tarefas de administração de acordo com horários definidos. Estas tarefas são designadas por *jobs*.

O *SQL Server Agent* é constituído por 4 componentes:

- *Jobs* — são o conjunto de acções específicas que o SQL Server Agent tem de executar. Cada acção é um *step* e um *job* pode conter vários *steps*.
- *Schedules* — específica quando um *job* deve ser executado. Uma *schedule* pode ter vários *jobs* associados.
- *Alerts* — é uma resposta automática à ocorrência de um evento específico. Os alertas podem executar duas acções como resposta, notificar um ou mais operadores ou executar um *job*.
- *Operators* — contêm as informações de contacto do administrador responsável pelo servidor. É esta informação que os alertas usam para notificar no caso da ocorrência de um evento.

Em termos de segurança, apenas os users do grupo *sysadmin* ou os utilizadores a quem sejam atribuídas as permissões *SQLAgentUserRole*, *SQLAgentReaderRole* e *SQLAgentOperatorRole* têm acesso ao *SQL Server Agent*.

Como descrito anteriormente, um *job* é um conjunto de acções determinadas pelo utilizador que têm de ser realizadas pelo *SQL Server Agent*.

Os *jobs* têm horários atribuídos e se não tiverem correm apenas quando o utilizador os iniciar manualmente, podem conter vários passos de execução, ou seja mais de uma acção pode ser realizada dentro de um job.

Existem um conjunto de benefícios associados à utilização de *jobs*: o primeiro é a capacidade de poder planear o trabalho realizado numa instância sem que seja necessária presença humana. A outra grande vantagem consiste num aspecto mais técnico, a utilização de *jobs* permite a paralelização de execuções. Tomando como exemplo a solução desenvolvida neste projecto, os jobs são utilizados para executar o mesmo código em diferentes servidores de forma paralela. Sem os *jobs* o código teria de ser executado de uma só vez, o que iria tornar o processo sequencial levando a um decréscimo da performance do sistema.

Para estabelecer as ligações entre o servidor central e os servidores monitorizados foi utilizado o *Linked Server* que é uma funcionalidade que permite executar comandos contra fontes de dados externas à instância de *SQL Server* onde o comando é executado. O objectivo é permitir aceder e extrair dados de outro servidor, independentemente da tecnologia do servidor de destino, ou seja a partir de uma instância de SQL Server aceder, por exemplo, a dados de uma instância de Oracle.

Tal como descrito em [Micc] existem 3 vantagens principais que o *Linked Server* proporciona:

- Possibilidade de acesso a dados externos à instância de SQL Server.
- Possibilidade de executar operações distribuídas contra vários servidores dentro da empresa.
- Possibilidade de tratar de igual forma as diferentes fontes de dados, independentemente da tecnologia em que os dados estão alojados.

Um *Linked Server* é composto por dois elementos:

- *OLE DB provider* — é o protocolo de comunicação utilizado para estabelecer ligação entre as diferentes fontes de dados.
- Fonte de dados *OLE DB* — é a fonte de dados destino sobre a qual irão ser executados os comandos

A Figura 4.1 representa graficamente as interacções efectuadas na execução de uma pesquisa distribuída através de *Linked Server*.

Para reutilização do mesmo código para todos os servidores monitorizados foi utilizada uma técnica de programação denominada *SQL* dinâmico. Esta técnica é utilizada quando o código desenvolvido não é estático, ou seja, quando o código é passível de mudanças que não são possíveis de prever.

Um exemplo concreto associado ao projecto desenvolvido que permite perceber o conceito é o código para a criação da estrutura de recolha de dados e o código para a recolha das métricas. Estes dois códigos têm vantagem em serem desenvolvidos em *SQL* dinâmico uma vez que as pesquisas alteram-se de acordo com o servidor de destino, o que significa que se o desenvolvimento tivesse sido realizado em *SQL* estático o código iria repetir-se tantas vezes quanto o número de servidores a monitorizar.

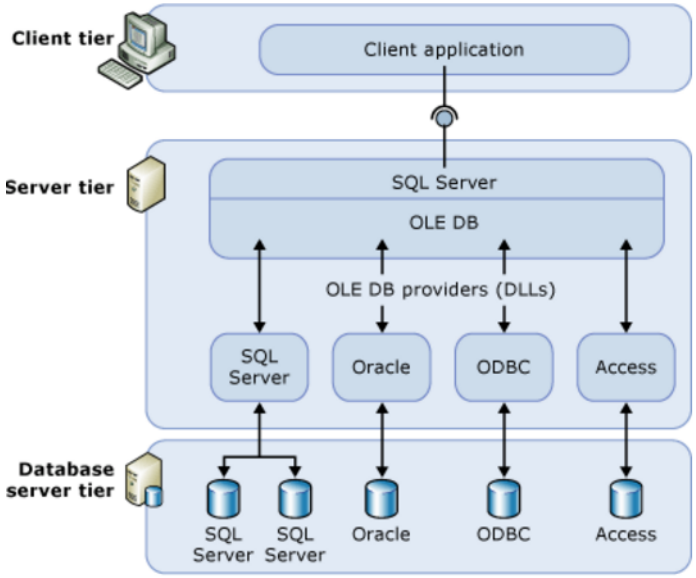


Figura 4.1: *Linked Server*

Outra das vantagens que permitiu foi não ser necessário adicionar código novo quando se pretende adicionar um servidor ao sistema de monitorização, desta forma o sistema é completamente dinâmico, apenas é necessário adicionar o servidor à tabela de configuração, de resto todo o código de recolha está pronto para ser executado independentemente do servidor de destino.

No entanto, existem desvantagens associadas a este tipo de técnica. A primeira está relacionada com performance, uma vez que o código tem de ser processado de cada vez que é executado o que por sua vez torna os tempos de execução mais longos. Existem técnicas para minimizar este impacto, no entanto a performance do *SQL* dinâmico é sempre pior do que a performance do *SQL* estático. A segunda desvantagem é em termos de segurança. A partir do momento que uma aplicação utiliza *SQL* dinâmico está aberta a ataques de *SQL Injection*, que é uma técnica em que um intruso insere dados que levam a aplicação a executar código *SQL* não pretendido. No entato existem um conjunto de cuidados que podem ser tomados por forma a mitigar esta vulnerabilidade segundo [Som03]. Os cuidados a ter em conta quando é utilizado *SQL* dinâmico são um controlo restrito das permissões atribuidas aos utilizadores, para aplicações web nunca demonstrar as mensagens de erro geradas pelo *SQL Server* e por fim utilizar sempre código parametrizado.

No âmbito do trabalho foi preciso criar e configurar uma conta específica para a aplicação, por forma a eliminar vulnerabilidades em termos de segurança, para que a mesma possa executar as suas tarefas. Esta conta foi configurada no *Monitoring Server* onde está armazenada a base de dados da aplicação e foi também configurada nos *Monitored Servers* que estão a ser monitorizados para que o *Monitoring Server* tenha acesso para recolha de dados dos *Monitored Servers*.

Foi criado então uma conta em cada um dos servidores e em cada base de dados que foi necessário aceder foi criado um utilizador para a conta criada anteriormente. As definições da conta são iguais em todos os servidores monitorizados, enquanto que no servidor monitorizador

a conta tem uma configuração diferente uma vez que necessita de mais permissões para poder escrever para a base de dados que suporta a solução desenvolvida.

De salientar que tanto no servidor central como nos servidores monitorizados apenas foram atribuídas as permissões estritamente necessárias para que não existam vulnerabilidade de segurança.

Finalmente importa abordar o conceito de *Dynamic Management Views*, uma vez que grande parte das métricas recolhidas estão presentes nestas estruturas.

As Dynamic Management Views são um subgrupo dos *Dynamic Management Objects* e armazenam informação relativa ao estado do servidor, que pode ser utilizada para monitorizar o estado de saúde da instância, diagnosticar problemas e optimizar a performance.

De acordo com a descrição presente em [Micb] existem dois tipos de Dynamic Management Views, nomeadamente *Server-scoped* e *Database-scoped*. Existem no entanto 26 categorias em que as Dynamic Management Views estão divididas, de acordo com a informação que contêm. Dessas 26, neste projecto foram utilizadas Dynamic Management Views e Dynamic Management Functions presentes nas seguintes categorias:

- *Database Related Dynamic Management Views (Transact-SQL) — sys.dm_db_ partition_stats*.
- *Execution Related Dynamic Management Views and Functions (Transact-SQL) — sys.dm_exec_ query_stats, sys.dm_exec_cached_plans, sys.dm_exec_sql_text, sys.dm_exec_sessions e sys.dm _exec_requests*.
- *Index Related Dynamic Management Views and Functions (Transact-SQL) — sys.dm _db_index_ usage_stats*.
- *I/O Related Dynamic Management Views and Functions (Transact-SQL) — sys.dm_io_virtual _file_stats*.
- *SQL Server Operating System Related Dynamic Management Views (Transact-SQL) — sys.dm_os _wait_stats, sys.dm_os_performance_counters*.

Além das *Dynamic Management Views* e *Dynamic Management Functions* referidas anteriormente, a solução também utiliza um conjunto de *Catalog Views* existentes no *SQL Server* para recolher informação. Estas *Views* contêm metadados sobre as opções de configuração da instância, objectos, tipos de dados e restrições entre outros tipos de informação. Existem 31 categorias em que as *Catalog Views* estão divididas. No âmbito do desenvolvimento desta solução foram utilizadas *Views* pertencentes a 4 destas 31 categorias.

Foi utilizada ainda outra *View*, *sys.dm_os_ring_buffers*, que não está documentada pela *Microsoft* no entato é possível utilizar para extrair informação. Uma explicação mais detalhada do funcionamento desta e de todas as outras *Views* utilizadas será fornecida quando for descrita a implementação da recolha das métricas.

4.2 Modelo de Dados

A implementação do modelo de dados divide-se em duas partes principais: primeiro a criação da base de dados e segundo a criação das tabelas que constituem o modelo de dados.

Para a criação da base de dados foi necessário indicar o nome da mesma, que por sua vez são os nomes dos ficheiros tanto de dados como de *log*, o tamanho inicial para cada um dos ficheiros, o tamanho limite e quanto deveria crescer cada um dos ficheiros em caso de necessidade. Existem ainda configurações extra que não são importantes no âmbito do projecto.

Em relação às tabelas que compõem o modelo de dados, o processo de criação foi evolutivo no sentido em que foram alteradas tabelas, adicionadas novas tabelas de acordo com as necessidades e foram também removidas tabelas que deixaram de ser utilizadas pela solução.

Tal como descrito no Capítulo 3 o modelo é constituído por três partes distintas mas que interagem entre elas, nomeadamente um conjunto de tabelas para armazenamento da informação relativa aos servidores monitorizados, um conjunto de tabelas para armazenamento das métricas do estado do servidor e outro conjunto para armazenamento das métricas das bases de dados. Esses três grupos formam o conjunto das tabelas criadas.

4.3 Recolha de Dados

A estrutura de recolha de dados é o ponto central da solução desenvolvida. Este tópico pretende descrever o seu desenvolvimento de forma detalhada. Irão ser abordados todos os componentes, mas principalmente irão ser detalhadas as principais decisões tomadas que têm influência directa no desenvolvimento da solução.

A estrutura de recolha de dados está dividida em duas partes, a estrutura principal que se encontra no servidor monitorizador e uma pequena estrutura que é implementada em cada servidor monitorizado. Em relação à estrutura principal o objectivo foi desenvolver uma estrutura que fosse independente do número de servidores monitorizados.

A recolha de dados no servidor monitorizador foi implementada através do desenvolvimento de um conjunto de *stored procedures* que executam de forma encadeada e implementam a estrutura. Vão ser agora descritos os procedimentos criados e a forma como se encadeiam.

O processo de *deploy* começa com a execução do *Stored Procedure usp_StartPerformanceIndicators*. O código desenvolvido percorre a tabela de configuração *Config* dentro de um ciclo e para cada linha da tabela extrai a informação, nomeadamente *DataCenter, Server, Instance e LinkedServer*, e passa essa informação como parâmetros para a execução dos *stored procedures PerformanceIndicatorsMain*, *CreateLinkedServer*, *PopulateDatabasesTable* e *PopulateFilesTable*. Para cada entrada da tabela de configuração são efectuadas todas as execuções. A ideia foi desenvolver o código de forma a estar parametrizado e funcionar independentemente do número de servidores.

O objectivo da execução deste conjunto de procedimentos é povoar o modelo de dados com a informação relativa ao servidor a monitorizar, as bases de dados nele contidas e estabelecer uma ligação entre os servidores.

Após a execução dos *stored procedures* abordados até agora, o modelo de dados encontra-se povoado com a estrutura de instâncias e bases de dados que irão ser monitorizadas, sendo agora necessário criar os *jobs* que irão recolher os dados.

As métricas são recolhidas de acordo com a frequência, determinada como a necessária para cada métrica específica. Para esse efeito foi desenvolvido um conjunto de procedimentos que criam *jobs* com determinada frequência de execução. O processo de criação começa com a execução do *stored procedure CreateMainJobs* que para cada frequência de recolha diferente irá executar o *stored procedure CreateJob*, que por sua vez executa o *stored procedure CreateTempJobs*. Este *stored procedure* contém o código para a recolha de todas as métricas e está dividido por frequência de recolha das métricas, ou seja é apenas executada a recolha das métricas associadas à frequência de recolha do *job* criado pelo *stored procedure CreateJob*. Para cada uma das métricas associadas à frequência de recolha, é executado o *stored procedure CreateTempJob* que cria uma tarefa de recolha temporária, que é eliminada após a recolha dos valores no servidor a monitorizar.

Como resultado final teremos um *job* para cada frequência de recolha para cada servidor monitorizado o que facilita a análise do funcionamento do sistema. Outra das razões que levou a esta estratégia de implementação foi a necessidade de ter objectos de recolha diferentes para cada servidor, sendo assim extremamente fácil eliminar um servidor do processo de monitorização ou adicionar um novo. Esta implementação permite também a paralelização da recolha de métricas dos vários servidores, o que era um dos objectivos do projecto.

Será agora abordado o processo de recolha das métricas de forma mais detalhada, pois irão ser apresentadas as diferentes técnicas utilizadas para os diferentes tipos de dados disponíveis para recolha. Irão ser descritos os processos de recolhas para métricas da instância e para métricas de bases de dados.

4.4 Recolha de Métricas de Performance de uma Instância

A recolha de métricas da instância é efectuada com diferentes técnicas e processos dependendo do tipo de dados que são recolhidos e da forma como estão disponibilizados. O objectivo deste ponto é descrever em detalhe todo o processo de recolha nas suas variações, desde as fontes de dados, à forma como os dados estão disponibilizados até às técnicas aplicadas para que os dados tenham o formato pretendido.

A metodologia para a recolha das métricas é influenciada por duas variáveis, o formato dos dados e a forma como se pretende disponibilizar os dados. Para o modelo de visualização e para as métricas da instância, foram implementadas quatro metodologias de recolha de dados diferentes que irão agora ser descritas.

4.4.1 Recolha Directa

Para descrever este método de recolha será utilizado o processo de recolha das métricas associadas a *Active Sessions*. A recolha destas métricas não implica nenhum tratamento de dados nesta fase, basta simplesmente recolher os dados e armazenar directamente no modelo de dados.

No caso das métricas de *Active Sessions* os dados foram recolhidos da *View* de sistema *sys.dm_exec_sessions* que, de acordo com a descrição presente em [Mick], tem uma entrada por cada sessão autenticada no SQL Server e fornece informação sobre essas mesmas sessões. A vista é composta por cinquenta e um campos dos quais são extraídos onze para armazenamento. O ponto a reter destes dados é que a vista apresenta informação do que está a acontecer no momento na instância, daí que seja apenas necessário recolher os dados directamente e armazenar. Existe processamento sobre estes dados, mas só é necessário no contexto da apresentação através do modelo de visualização, já que os dados fonte podem ser recolhidos directamente.

O código seguinte demonstra a recolha deste tipo de dados.

```
insert into Performance_Indicators.dbo.InstanceConnections
([ServerID],[InstanceID],[LoginName],
 [CpuTime],[MemoryUsage],[Reads],
 [Writes],[LogicalReads],[NumberOfSessions],
 [CaptureDate],[HostName],[ProgramName],[ClientInterfaceName])
select @ServerID, @InstanceID, login_name,
       SUM(cpu_time), SUM(memory_usage), SUM(reads),
       SUM(writes), SUM(logical_reads), COUNT(*),
       GETDATE(), host_name, program_name, client_interface_name
from [@LinkedServer].[ADMIN].sys.dm_exec_sessions
where status = 'Running'
group by host_name, program_name, client_interface_name, login_name
```

Alguns pontos a salientar no código apresentado:

- Além dos dados da vista são também recolhidos os dados *@Server* e *@Instance* que são usados para determinar a que instância pertencem os dados recolhidos.
- É utilizada a convenção de quatro nomes para recolher os dados do servidor remoto e esses dados são inseridos directamente na tabela *InstanceConnections*.

4.4.2 Recolha de Dados Cumulativos

Para descrever este método de recolha será utilizado o processo de recolha de métricas associadas a *Query Statistics*. O processo de recolha de *Query Statistics* envolve duas *Dynamic Management Views*, (*sys.dm_exec_query_stats* e *sys.dm_exec_cached_plans*) e uma *Dynamic Management Function*, (*sys.dm_exec_sql_text()*). Em relação à vista de sistema *sys.dm_exec_query_stats*, tal como é descrito em [Mici], apresenta dados cumulativos das estatísticas de execução dos planos das operações que se encontram em memória. Cada linha desta vista apresenta dados para cada *statement* dentro do plano que se encontra em memória, ou seja o mesmo plano pode aparecer várias vezes. No entanto as estatísticas associadas referem-se a *statements* individuais dentro do plano. Esta vista contém sessenta e cinco colunas das quais vinte e duas são utilizadas para recolha de dados. A vista *sys.dm_exec_cached_plans* contém uma entrada para cada plano que se encontra em memória, esta vista é usada para estabelecer ligação entre os dados da vista *sys.dm_exec_query_stats* e os planos que se encontram em memória e é também extraída informação relativamente ao tipo de *query* a que se refere o plano. A *Dynamic Management Function*

sys.dm_exec_sql_text() é invocada com o *token* armazenado na vista *sys.dm_exec_query_stats*, os resultados obtidos são depois trabalhados de forma a que se obtenha apenas o texto do *statement* associado às estatísticas recolhidas.

A dificuldade em criar valor a partir destes dados prende-se com o facto de serem cumulativos, representam o total de todas as execuções desde que o plano entrou para memória, o que torna difícil determinar que pedidos foram executados num determinado intervalo de tempo pois essa informação não é disponibilizada.

A estratégia encontrada para lidar com este tipo de dados foi a criação de uma tabela de suporte que armazena os últimos dados recolhidos. Inicialmente são recolhidos todos os dados presentes nas vistas descritas e ao fim do intervalo de recolha definido é executada uma nova recolha de dados. O processo divide-se agora em várias etapas de refinamento dos dados até que sejam inseridos nas respectivas tabelas do modelo de dados. Inicialmente são inseridos os dados relativos aos textos das operações na tabela *InstanceQueryText*. Esta tabela tem um funcionamento particular porque como os dados de texto têm um peso elevado em termos de armazenamento são armazenados apenas uma vez. De cada vez que existem novos dados, que ainda não estão presentes na tabela são inseridos. O excerto de código seguinte representa a execução do processo descrito.

```
insert into Performance_Indicators.dbo.InstanceQueryText
select @ServerID, @InstanceID, QT.sql_handle,
       MAX(QT.text), MAX(QT.statement), MAX(QT.database_name),
       MAX(QT.objtype), MAX(QT.CaptureDate), QT.statement_start_offset,
       QT.statement_end_offset
from(select #QS.sql_handle, #QS.text, #QS.statement,
            #QS.database_name, #QS.objtype, #QS.CaptureDate,
            #QS.statement_start_offset, #QS.statement_end_offset
     from #QS
     left join Performance_Indicators.dbo.InstanceQueryText as iqt on
     #QS.sql_handle = iqt.SqlHandle
     and #QS.statement_start_offset = iqt.StatementStartOffset
     and #QS.statement_end_offset = iqt.StatementEndOffset
     and iqt.ServerID = @ServerID and iqt.InstanceID = @InstanceID
     where iqt.SqlHandle is null) QT
group by QT.sql_handle, QT.statement_start_offset, QT.statement_end_offset
```

Em relação às estatísticas, são comparados os valores da última recolha com os valores armazenados na tabela de suporte. Esta comparação utiliza o comando *left outer join* o que garante que resultados novos sejam introduzidos. De seguida é realizado um refinamento através de uma restrição em que só são retornados dados em que o número de execuções tenha aumentado. Dessa forma excluímos as operações que não foram executadas no intervalo de tempo definido. Após estes refinamentos temos os dados pretendidos, no entanto ainda é necessário calcular as diferenças entre as estatísticas, para que seja possível obter os valores correspondentes às execuções dentro do intervalo de tempo entre as recolhas.

Finalmente são apagados os dados na tabela de suporte e são introduzidos todos os dados obtidos na última recolha para que seja possível continuar a utilizar este processo.

4.4.3 *Ring Buffer*

Para descrever este método de recolha será utilizado o processo de recolha de métricas de *CPU Usage*. São recolhidos valores para a utilização do processador por parte do SQL Server e por parte de todos os outros processos. Os dados são retirados da vista de sistema *sys.dm_os_ring_buffers*. Esta vista tem um funcionamento particular, no sentido em que armazena os dados numa estrutura circular. No caso específico de utilização de processador são armazenadas 256 entradas de dados, cada entrada tem uma distância temporal de um minuto em relação à anterior, o que significa que ao fim de 256 minutos os dados mais antigos serão substituídos pelos dados mais recentes. Outra particularidade desta vista é que a informação é disponibilizada em formato *XML*, o que implica que seja necessário processar o ficheiro para extrair a informação.

4.4.4 *Performance Counters Dynamic Management View*

Este é um caso específico, no sentido em que os dados disponíveis nesta vista de sistema têm vários tipos, o que implica que tenham de ser recolhidos e processados de forma diferente de acordo com o seu tipo.

A vista é composta por cinco campos *object_name*, categoria à qual os valores da métrica pertencem, *counter_name*, nome da métrica, *instance_name*, nome da instância à qual a métrica se refere, *cntr_value*, valor registado para a métrica, *cntr_type*, tipo de valor registado. Os campos fundamentais são os três últimos, uma vez que indicam se a métrica se refere à instância ou a uma base de dados da instância, o valor registado para a métrica e o tipo de valor, o que nos permite perceber o método de recolha a utilizar para cada métrica desta tabela.

Existem cinco valores possíveis para o campo *cntr_type*, cada valor indica o tipo de dados armazenados o que permite identificar o método necessário para retirar valor dos dados armazenados:

- *Perf_Large_Raw_Base* — este tipo de dados indica que o valor registado tem de ser usado em conjunto com outro valor para ser obtido o valor da métrica. Como o próprio nome indica, este valor irá ser usado como base no cálculo do valor da métrica, ou seja, irá ser realizada a divisão entre o valor obtido, para a mesma métrica, do tipo *Perf_Large_Raw_Fraction* e o valor obtido deste tipo. O resultado do cálculo representa o valor da métrica no momento em que os dados foram recolhidos de acordo com [pss13].

- *Perf_Large_Raw_Fraction* — tal como descrito em [pss13] este tipo de dados representa um valor fracionário como um rácio para o valor correspondente do tipo *Perf_Large_Raw_Base*. O cálculo deste tipo de dados é efectuado com recurso à seguinte fórmula,

$$HitRatio = Perf\_Large\_Raw\_Fraction / Perf\_Large\_Raw\_Base \tag{4.1}$$

- *Perf_Average_Bulk* — a recolha de dados com este tipo tem um funcionamento semelhante ao explicado anteriormente, tal como descrito em [pss13]. É necessário calcular o rácio

entre dois valores, no entanto os dados deste tipo são cumulativos, o que implica capturar duas amostras de dados e realizar o mesmo cálculo mas sobre as diferenças. Assumindo os intervalos de tempo A e B, a fórmula a utilizar seria a seguinte,

$$Result = (B\_P\_Average\_B - A\_P\_Average\_B)/(B\_P\_Large\_R\_B - A\_P\_Large\_R\_B) \tag{4.2}$$

.

- *Perf_Counter_Bulk_Count* — de acordo com o descrito em [pss13], este tipo de dados representa o valor da métrica por segundo. Os dados são armazenados de forma cumulativa, o que implica que para a recolha de métricas deste tipo seja necessário extrair dados em dois intervalos de tempo diferentes, subtrair o valor da primeira recolha ao valor da segunda recolha e dividir o resultado dessa operação pelo intervalo de tempo entre as duas recolhas. O cálculo do valor deste tipo de métricas é efectuado segundo a fórmula,

$$Result/sec = (Second\_Value - First\_Value)/Number\_of\_Seconds) \tag{4.3}$$

.

- *Perf_Counter_Large_Rawcount* — os dados com este tipo apresentam o último valor observado directamente, ou seja não são necessários cálculos adicionais sobre os valores recolhidos já que cada valor representa o valor da métrica no momento de recolha, de acordo com [pss13].

4.5 Recolha de Métricas de Bases de Dados

A recolha de métricas específicas de bases de dados é realizada de duas formas distintas. A primeira é igual à forma de recolha das métricas associadas ao estado da instância, ou seja, os dados são recolhidos a partir do servidor monitorizador, a segunda forma efectua a recolha localmente em cada instância monitorizada e os dados armazenados são depois recolhidos pelo servidor monitorizador e processados.

4.5.1 Recolha Local

A recolha das métricas a partir do servidor monitorizador é realizada através de dois métodos, nomeadamente a recolha de dados cumulativos e recolha directa.

As métricas recolhidas a partir do servidor monitorizador são as seguintes:

- *Database I/O* — a recolha destes dados usa o método explicado anteriormente de recolha de dados cumulativos, ou seja, utiliza uma tabela auxiliar para efectuar comparações entre os valores recolhidos no momento e os valores da última recolha. São realizados os cálculos relativos às diferenças entre os dois valores para obter os dados associados ao

intervalo de tempo entre as recolhas. São apenas armazenados os dados dos ficheiros de dados em que existe actividade, ou seja, para ficheiros em que os dados sejam iguais entre recolhas significa que não foi registada actividade durante o intervalo de tempo. Os dados associados a estas métricas são obtidos das vistas de sistema *sys.dm_io_virtual_file_stats* e *sys.master_files*.

- *Table Sizes* — a recolha dos dados, relativos às métricas de tamanho das tabelas das bases de dados, é efectuada directamente, sendo apenas realizadas algumas operações de agregação para que sejam retornados os dados pretendidos. Os dados são obtidos a partir das vistas de sistema *sys.dm_db_partition_stats*, *sys.all_objects*, *sys.schemas* e *sys.internal_tables*. Existe uma particularidade no código de recolha destes dados, algumas das vistas fornecem apenas dados relativos ao contexto em que a pesquisa for efectuada, ou seja para retornar os dados relativos às tabelas de uma base de dados a pesquisa tem de ser realizada no contexto dessa base de dados, para contornar essa dificuldade foi usado o *stored procedure* de sistema *sp_MSforeachdb* que executa o código passado como argumento em todas as bases de dados presentes numa instância. Outro ponto a destacar no código de recolha destes valores é o facto de os resultados serem filtrados para retornar apenas as tabelas que tenham alguma relevância em termos de tamanho dentro da base de dados. As tabelas analisadas têm de ter um tamanho superior a 100 *MB* e têm de representar pelo menos 0.5% do tamanho da base de dados da qual fazem parte.

4.5.2 Recolha Remota

As métricas recolhidas localmente em cada instância monitorizada têm um processo de recolha completamente diferente dos que foram descritos até agora. O funcionamento do processo divide-se em três blocos principais.

Inicialmente é efectuada a recolha dos dados no servidor monitorizado através de um *stored procedure* que recolhe e armazena os dados numa tabela. Os dados são recolhidos através de três vistas de sistema e de uma função de sistema, nomeadamente *sys.dm_exec_requests*, *sys.dm_exec_sessions*, *sys.databases* e *sys.dm_exec_sql_text*. A intenção é recolher dados sobre os pedidos executados em cada base de dados presente na instância, para isso, o *stored procedure* é executado a cada segundo.

O segundo passo é a recolha dos dados armazenados. Foi criado um *job* no servidor monitorizador que a cada minuto recolhe os dados armazenados nas instâncias, apaga os dados recolhidos e armazena numa tabela de *staging* no servidor monitorizador. A cada execução deste processo de recolha são recolhidos, apagados e armazenados no servidor central todos os dados presentes de momento na tabela da instância monitorizada.

O terceiro passo é o processamento dos dados presentes na tabela de *staging* do servidor monitorizador. Para a execução do processamento foi criado um *stored procedure* que executa a cada cinco minutos e processa os dados dos últimos seis minutos, esta diferença é propositada pois foi

detetado que alguns dados não eram processados por diferenças de segundos e ficavam na tabela de *staging* pendurados.

O processamento destes dados tem por objectivo separar e organizar os dados de forma normalizada. Dessa forma os dados de cada pedido são divididos e armazenados em diferentes tabelas, os dados relativos à sessão que realizou o pedido são armazenados numa tabela própria.

De seguida são processados os dados relativos ao texto dos pedidos efectuados. A terceira parte do processamento diz respeito às estatísticas gerais de execução dos pedidos, estes dados são armazenados numa tabela própria que contém os identificadores para as tabelas onde estão armazenados os textos das operações e a informação das sessões.

Os últimos dados processados são as estatísticas de espera associadas aos pedidos. Estes dados são armazenados em conjunto com um identificador para a tabela que contém as estatísticas dos pedidos.

Existe ainda mais um passo na tarefa do processamento que se refere à limpeza da tabela de *staging*. Após o processamento dos dados para as tabelas, os dados processados são eliminados.

4.6 Processamento da *Baseline*

Quando o utilizador acede a um dos relatórios de comparação existem várias formas de apresentar a comparação entre os dados de dois intervalos de tempo. A forma mais simples de apresentar os dados seria apresentar os resultados para os dois intervalos de tempo mas, dessa forma, todo o esforço estaria do lado do utilizador para retirar resultados da comparação efectuada.

A forma utilizada neste projecto é completamente diferente. Uma vez que o objectivo é desenvolver uma aplicação que facilite a proactividade, a comparação entre dois intervalos de tempo tem de permitir de forma rápida identificar pontos de interesse. No entanto, nem todas as métricas apresentadas permitem que os seus dados sejam processados, sendo efectuada uma comparação directa entre os dados dos dois intervalos de tempo.

A grande maioria das métricas recolhidas permite que seja aplicada a estratégia de processamento delineada. Essa estratégia consiste no processamento dos dados do intervalo que serve de base de comparação e a utilização desse valores processados para detectar em que alturas os dados mais recentes tiveram um comportamento diferente do que seria normal. São então calculados três valores, a média, o primeiro desvio padrão e o segundo desvio padrão em intervalos de tempo de cinco minutos. A ideia por trás desta estratégia é fornecer níveis de análise mais aprofundados, nomeadamente começar a análise com a média, se não for suficiente para facilitar a análise aprofundar ainda mais com uma das outras medidas disponibilizadas.

O funcionamento da estratégia é explicado na secção 2.3 com o conjunto de imagens apresentadas.

O processamento é realizado por um conjunto de *stored procedures* executados quando o utilizador define os parâmetros para a abertura do relatório de comparação.

4.7 Performance

Esta secção explica as implementações realizadas no sentido de garantir uma performance elevada ao sistema desenvolvido. Antes de serem abordadas questões específicas, é de ressalvar que todo o desenvolvimento tem de ser realizado com a performance como uma das considerações na tomada de qualquer decisão, desde o desenho do modelo de dados, ao desenvolvimento do código das pesquisas efectuadas até à criação do sistema de recolha de métricas. No entanto ainda assim são necessárias medidas adicionais para assegurar uma performance alta. No caso deste projecto foi necessário criar um conjunto de índices para aumentar a performance das pesquisas e garantir o nível de performance pretendido para o sistema desenvolvido.

Nesse sentido foram criados dezanove índices para que o modelo de visualização de dados tenha um tempo de resposta elevado e dentro do pretendido.

Uma descrição dos tipos de índices utilizados está presente no anexo B.

4.8 Manutenção e Limpeza de Dados

Para garantir o funcionamento contínuo do sistema de monitorização foram definidos processos para a manutenção dos dados existentes e um processo para limpeza de dados antigos.

O processo de manutenção de dados tem a função de garantir que os dados relativos aos objectos monitorizados se mantêm actuais. Para isso verifica diariamente se foram adicionadas novas bases de dados aos servidores monitorizados ou se alguma foi eliminada. As novas bases de dados serão adicionadas à monitorização e as eliminadas terão os seus dados apagados.

O processo de limpeza de dados elimina todos os dados com uma data superior a um mês. Este processo teve de ser criado para garantir que o volume de dados não crescia exponencialmente em relação ao número de servidores monitorizados.

4.9 Resumo e Conclusões

O objectivo deste capítulo foi descrever todo o trabalho desenvolvido no projecto, a forma como foi desenvolvido e o porquê de ter sido desenvolvido da forma que foi.

Assim é possível perceber claramente o funcionamento do protótipo desenvolvido e as motivações técnicas que direcionaram o desenvolvimento. Foi abordada e descrita toda a fase de desenvolvimento, desde o modelo de dados, a passar pelo processo de recolha até ao desenvolvimento da ferramenta de visualização dos dados recolhidos.

No próximo capítulo é apresentado o modelo de visualização das métricas recolhidas.

Capítulo 5

Modelo de Visualização

5.1 Introdução

Neste capítulo irá ser apresentado o modelo de visualização desenvolvido. Todos os relatórios serão apresentados de forma visual e serão também apresentadas as interações entre os diferentes relatórios por forma a que seja possível perceber o modelo como um todo.

Pretende-se que após a leitura deste capítulo seja possível perceber como são apresentados os dados aos utilizadores e quais são as opções de exploração disponíveis.

5.2 Relatórios Desenvolvidos

Os relatórios desenvolvidos podem ser divididos em dois grupos, os que apresentam métricas associadas ao estado de uma instância do SGBD, e os que apresentam métricas relativas às bases de dados presentes numa instância. Dentro destes dois grupos ainda existe uma outra separação entre relatórios que apresentam métricas recolhidas num intervalo de tempo definido pelo utilizador e relatórios que apresentam uma comparação entre valores de métricas para dois intervalos de tempo definidos pelo utilizador.

Para ilustrar as opções de navegação é apresentado um mapa na Figura 5.1.

O relatório inicial do Modelo de Visualização é o *Server Status*, que apresenta uma vista geral sobre o estado da instância, na qual são apresentadas algumas métricas que permitem ter uma ideia da actividade e da performance da instância, a utilização do processador, o número de sessões activas, número de pedidos à instância, número de compilações e recompilações de planos de execução das pesquisas e as estatísticas das esperas registadas na instância. Todos estes dados são compreendidos para o intervalo de tempo definido como parâmetro no momento de visualização do relatório. A interface deste relatório é apresentada na Figura 5.2.

A partir deste relatório é possível navegar para os seguintes relatórios: *Active Sessions*, *Server Status Comparison*, *Query Analysis*, *Database Status* e *Server Metrics*.

O relatório *Server Status Comparison* permite efectuar uma comparação entre os valores das métricas do relatório *Server Status* para dois intervalos de tempo diferentes. O utilizador define

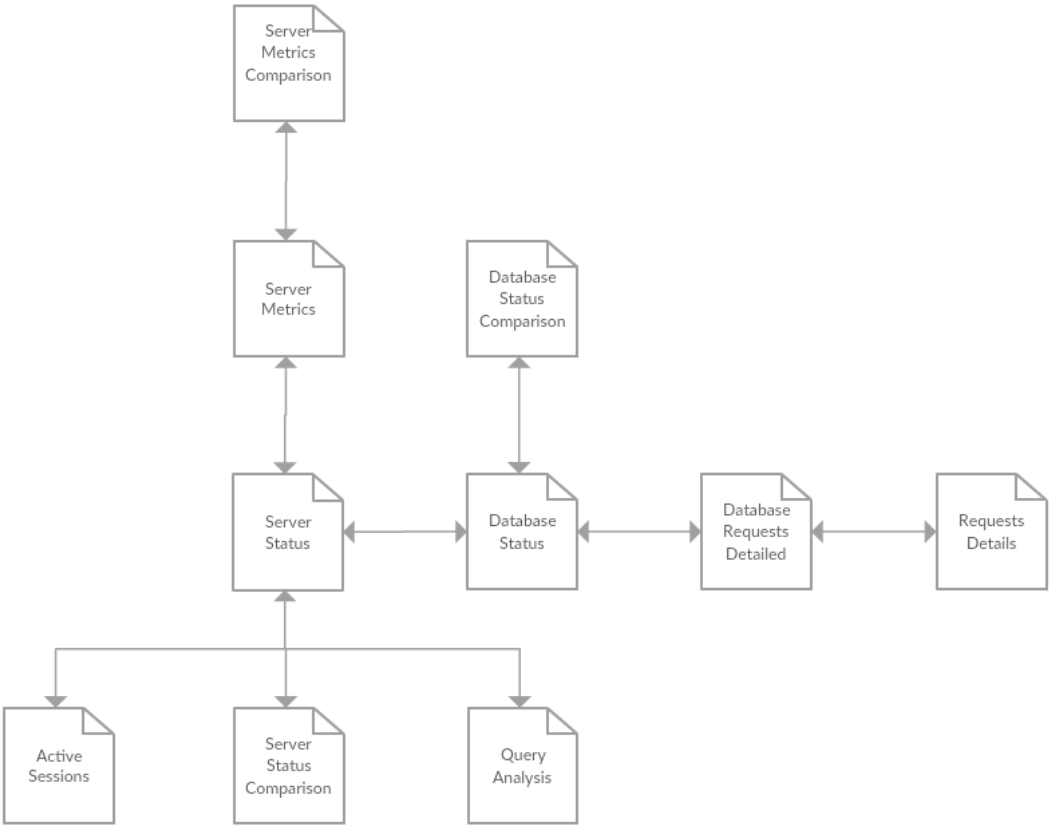


Figura 5.1: Diagrama Representativo das Opções de Navegação entre Relatórios.

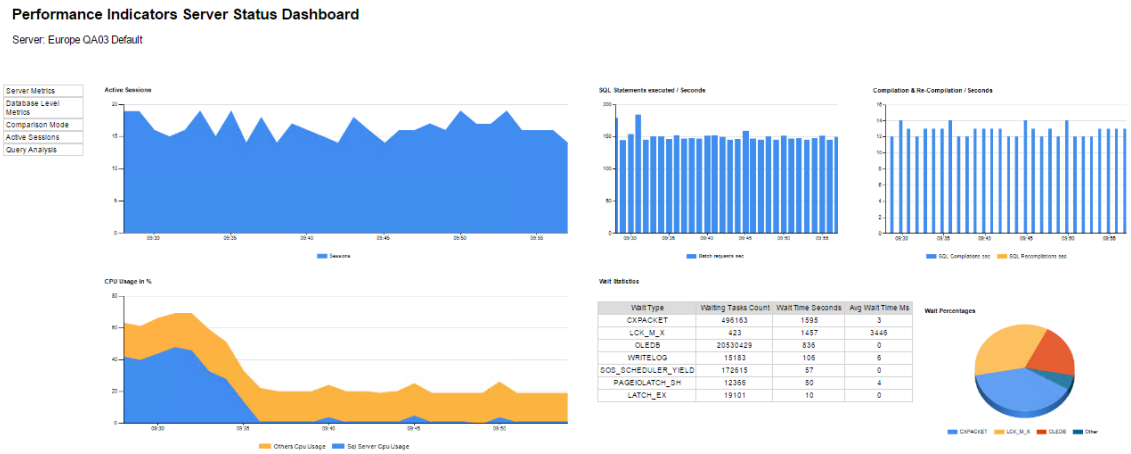


Figura 5.2: Relatório do Estado do Servidor.

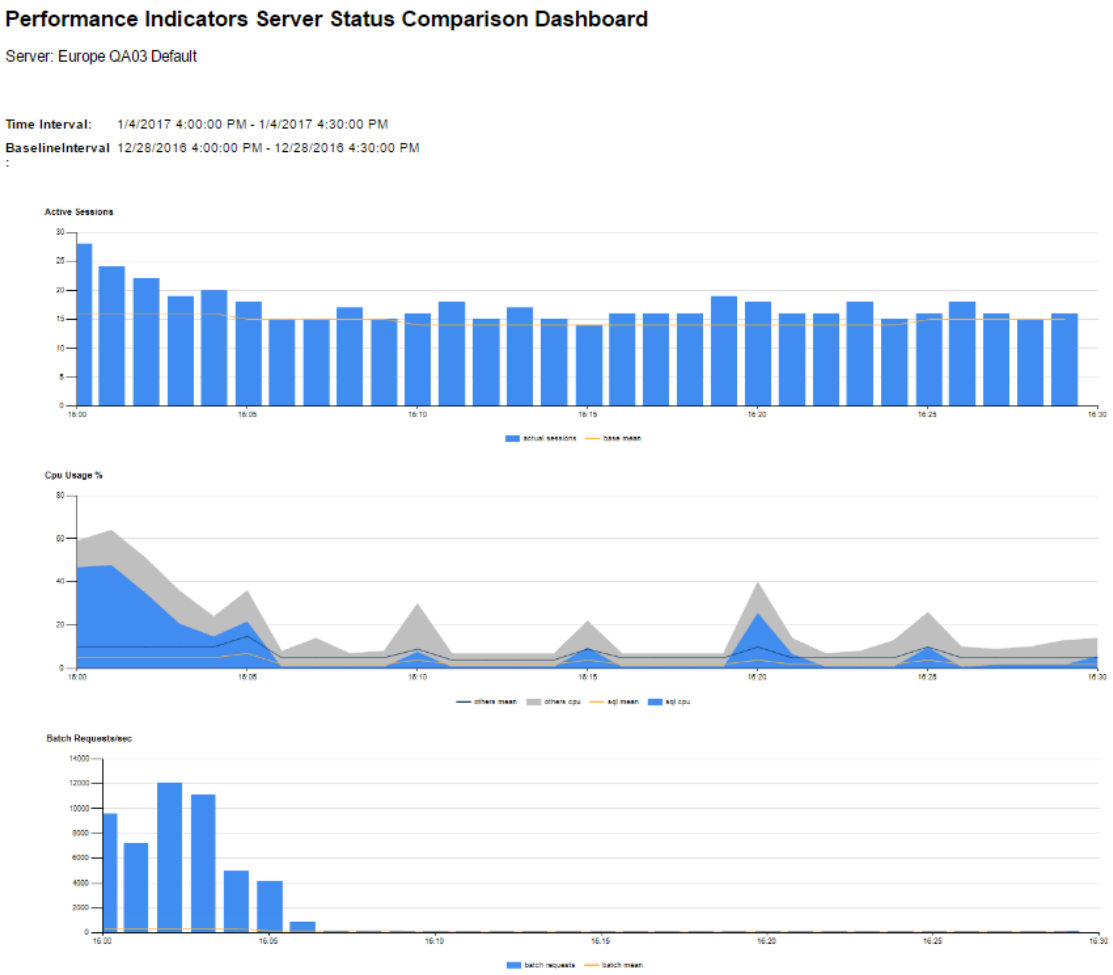


Figura 5.3: Relatório de Comparação do Estado do Servidor.

os intervalos de tempo, os dados obtidos para o segundo intervalo de tempo são processados e apresentados como base para comparação com os valores obtidos para o primeiro intervalo de tempo. A interface deste relatório é apresentada na Figura 5.3.

O relatório *Active Sessions* é outra das opções de navegação a partir do relatório inicial e apresenta um conjunto de métricas associadas às sessões activas na instância para que seja possível ao utilizador perceber o impacto das sessões. São apresentados os consumos das sessões em termos de tempo de processamento, memória e *I/O*. A interface deste relatório é apresentada na Figura 5.4.

Outra das opções de navegação iniciais é o relatório de *Query Analysis* que apresenta as estatísticas das pesquisas executadas sobre a instância no período de tempo definido pelo utilizador. São apresentadas as dez pesquisas com maior impacto para cada tipo de análise, existe um parâmetro que permite ao utilizador escolher qual o tipo de análise que pretende efectuar sendo apresentados apenas os valores de acordo com a escolha. É apresentado o texto das pesquisas e as estatísticas associadas a cada texto. A interface do relatório é apresentada na Figura 5.5.

É ainda possível navegar para o relatório *Server Metrics*. Este relatório apresenta um conjunto

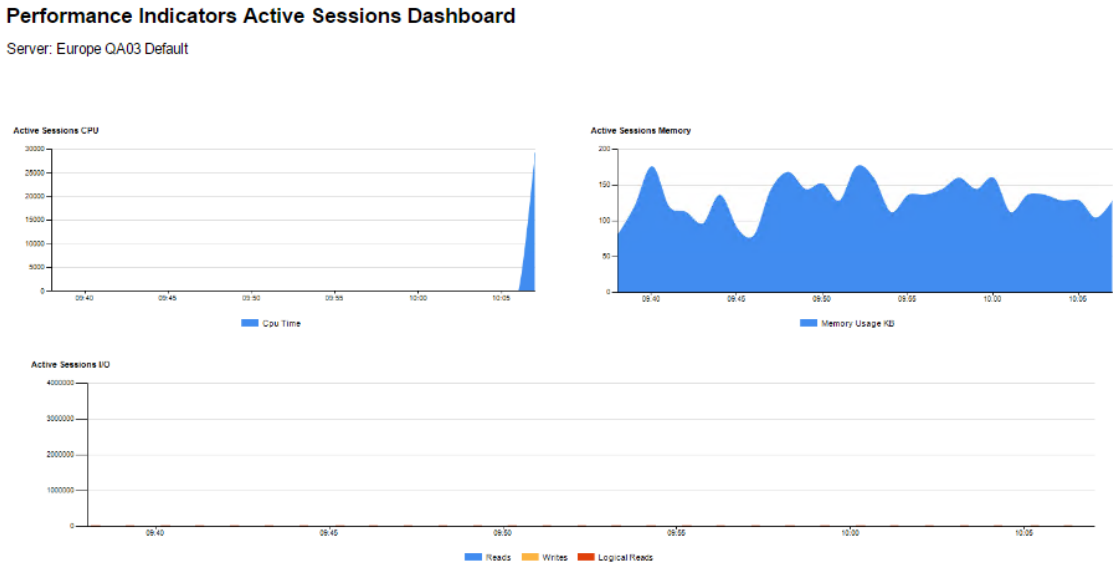


Figura 5.4: Relatório das Sessões Activas na Instância e do seu Impacto.

de indicadores de performance da instância divididos em três grupos: memória, *I/O* e actividade. É possivel ao utilizador observar apenas alguns grupos de métricas ou todas de acordo com o propósito da sua análise. A interface deste relatório é apresentada na Figura 5.6.

A partir deste relatório é possível navegar para um relatório de comparação *Server Metrics Comparison*(ver Figura 5.7), que permite efectuar uma análise comparativa entre os valores das métricas apresentadas no relatório *Server Metrics* em dois intervalos de tempo distintos. Os valores para as métricas que servem de base são processados para que seja possível efectuar uma análise do estado dos valores actuais. É permitido ao utilizador escolher que tipo de processamento pretende utilizar como base de comparação.

Finalmente, o último relatório para o qual é possível navegar a partir do relatório inicial é o *Database Status* (ver Figura 5.8), que apresenta um conjunto de indicadores de performance relativos a uma base de dados definida pelo utilizador. É possível obter informação relativamente a quem efectuou operações sobre a base de dados, que tipo de operações foram realizadas, bem como a quantidade de actividade realizada sobre a base de dados. Outra informação apresentada ao utilizador é o tamanho e a tendência de crescimento das maiores tabelas da base de dados.

A partrir deste relatório é possível navegar para dois outros relatórios: *Database Status Comparison* e *Database Requests Detailed*.

O relatório *Database Status Comparison* (ver Figura 5.9), apresenta os mesmos dados mas para dois intervalos de tempo diferentes para que seja possível ao utilizador efectuar uma comparação da actividade efectuada sobre a base de dados nos dois intervalos de tempo definidos.

O relatório *Database Requests Detailed* (ver figura 5.10), apresenta informação sobre todos os pedidos executados na base de dados e é possível organizar a informação por três valores diferentes: consumo de tempo de processador, impacto em termos de escritas e leituras e número de execuções.

Performance Indicators Query Statistics Analysis Dashboard

Server: Europe QA03 Default

Query Analysis based on CPU statistics

```
UPDATE IB
SET StockTotal = T1.StockTotal
FROM [BoGeoItemBase IB
INNER JOIN (
      SELECT   IB.ItemId,
               IB.StoreId,
               CASE
                     WHEN SUM(ISNULL(BoStocks.Total,0)) < 0 THEN 0
                     ELSE SUM(ISNULL(BoStocks.Total,0))
                     END AS StockTotal
      FROM     BOGeoItemBase IB (nolock)
               INNER JOIN BoStockPoints (nolock)
               INNER JOIN BoStocks (NOLOCK) ON BoStockPoints.PointID = BoStocks.LocalID
      ON       IB.ItemID = BoStocks.CheveArtigo
               AND IB.StoreId = BoStockPoints.SiteID
      GROUP BY IB.ItemId, IB.StoreId ) T1
      ON   IB.ItemId = T1.ItemId
      AND IB.StoreId = T1.StoreId
WHERE IB.StockTotal <> T1.StockTotal
```

Executions	Total Cpu Time	Avg Cpu/Exec	Min Cpu Time	Max Cpu Time	Avg Elapsed Time/Exec	Min Elapsed Time	Max Elapsed Time
4	58300	14575	12556	17778	4464	3356	13843

```
INSERT INTO ADMIN.dbo.RequestsStaging(
      HostName,
      ProgramName,
      ClientInterfaceName,
      LoginName,
      DatabaseName,
      Command,
      SqlHandle,
      StatementStartOffset,
      StatementEndOffset,
      Text,
      Statement,
      WaitType,
      WaitTime,
      CpuTime,
      Reads,
      LogicalReads,
      Writes,
      Rows,
      TotalElapsedTime,
      CaptureDate,
      StartTime,
      Status
      )
      SELECT des.host_name,
            des.program_name,
            des.client_interface_name,
            des.login_name,
            d.name,
            der.command,
            der.sql_handle,
            der.statement_start_offset,
            der.statement_end_offset,
            st.text,
            SUBSTRING(st.text, (der.statement_start_offset/2)+1,
                  ((CASE der.statement_end_offset
                    WHEN -1 THEN DATALENGTH(st.text)
                   ELSE der.statement_end_offset
                   END - der.statement_start_offset)/2) + 1),
                  der.wait_type,
                  der.wait_time,
                  der.cpu_time,
                  der.reads,
                  der.logical_reads,
                  der.writes,
                  der.row_count,
                  der.total_elapsed_time,
                  (GETDATE()),
                  der.start_time,
                  der.status
      FROM sys.dm_exec_requests der
      INNER JOIN sys.dm_exec_sessions des ON der.session_id = des.session_id
      INNER JOIN sys.databases d ON der.database_id = d.database_id
      CROSS APPLY sys.dm_exec_sql_text(der.sql_handle) st
      WHERE (d.database_id>4 AND d.user_access=0 AND d.state=0) OR d.name='tempdb'
```

Executions	Total Cpu Time	Avg Cpu/Exec	Min Cpu Time	Max Cpu Time	Avg Elapsed Time/Exec	Min Elapsed Time	Max Elapsed Time
874	14948	17	13	51	20	13	1820

Figura 5.5: Relatório das Operações Executadas sobre a Instância.

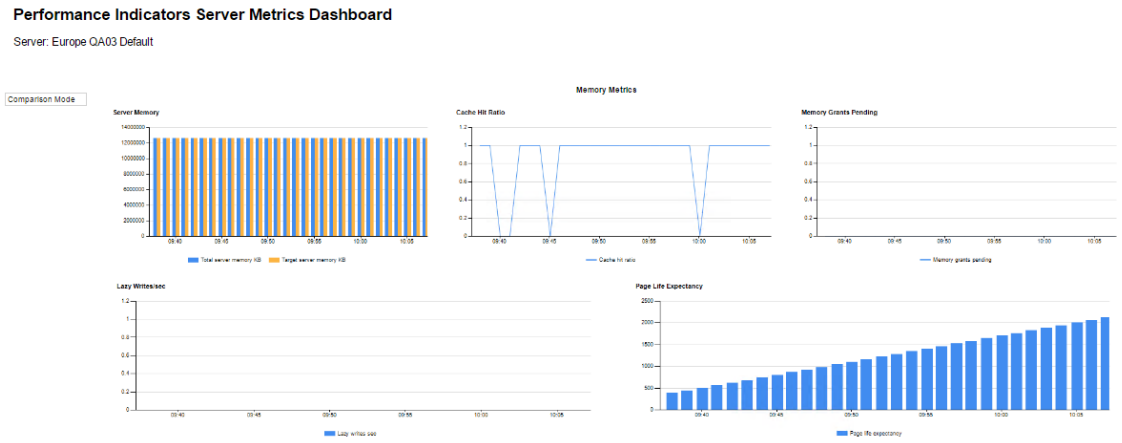


Figura 5.6: Relatório com Métricas da Instância.

A ordenação é definida pelo utilizador através de um parâmetro, dependendo da escolha tomada, serão apresentados os 10 pedidos com os maiores valores para a configuração escolhida. É apresentado o texto associado a cada um dos pedidos e um conjunto de estatísticas da execução desses pedidos.

A partir deste relatório é possível navegar para o relatório *Request Details* (ver Figura 5.11), que apresenta uma visão mais aprofundada sobre os pedidos apresentados no relatório *Database Requests Detailed*, ou seja para cada pedido apresentado é possível navegar para este relatório em que são apresentados os dados estatísticos para cada execução individual do pedido, enquanto que no relatório anterior são apresentados os valores agregados. Além dos valores individuais por execução são apresentados valores acumulados para as esperas associadas à execução.

5.3 Resumo e Conclusões

Este capítulo demonstra o modelo de visualização implementado, tendo sido abordados cada relatório individualmente, bem como as interacções entre os diferentes relatórios.

Espera-se que após a leitura seja possível identificar com clareza a função de cada relatório e quais as possibilidades de aprofundamento da análise disponíveis em cada relatório.

No próximo capítulo irá ser descrito o método utilizado para avaliar o trabalho desenvolvido.

Figura 5.7: Relatório de Comparação de Métricas da Instância.

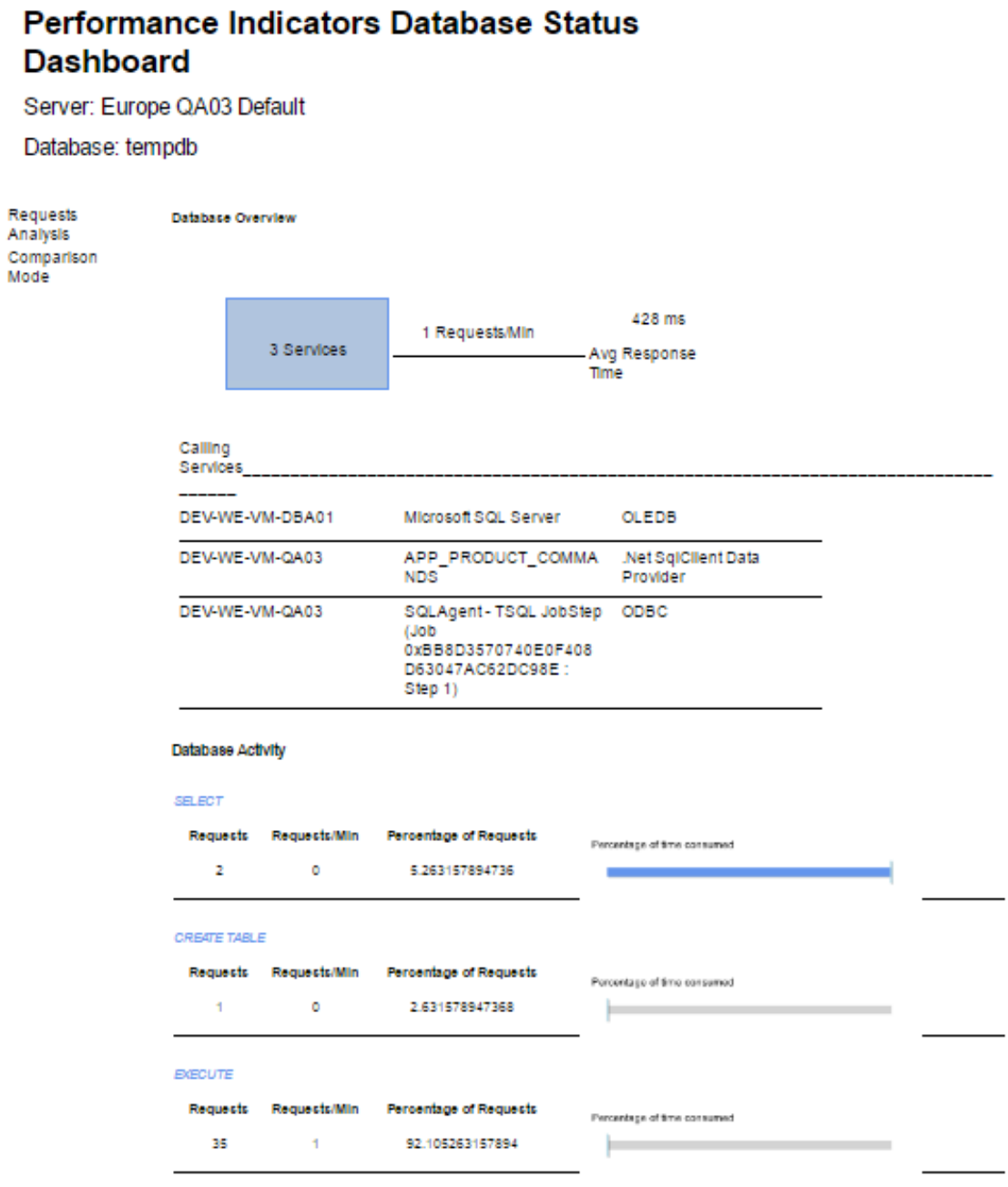
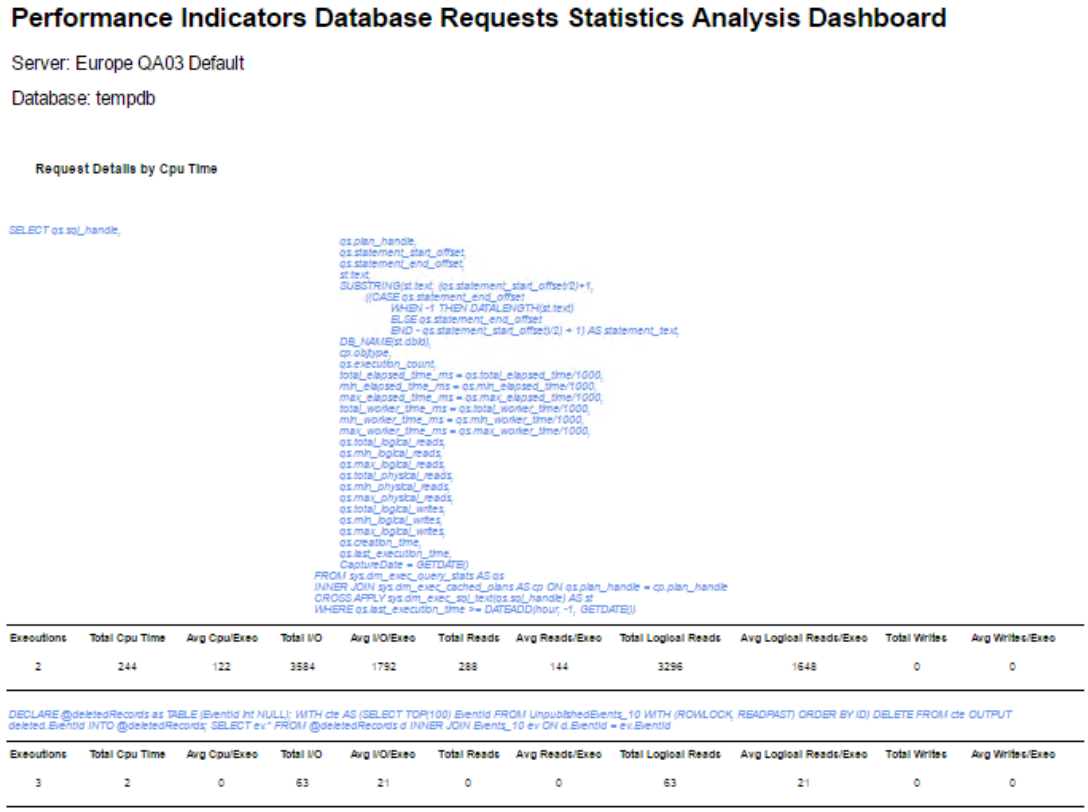


Figura 5.8: Relatório do Estado de uma Base de Dados.

Figura 5.9: Relatório de Comparação do Estado de uma Base de Dados.



Performance Indicators Database Requests Statistics Analysis Dashboard

Server: Europe QA03 Default

Database: tempdb

Request Details by Cpu Time

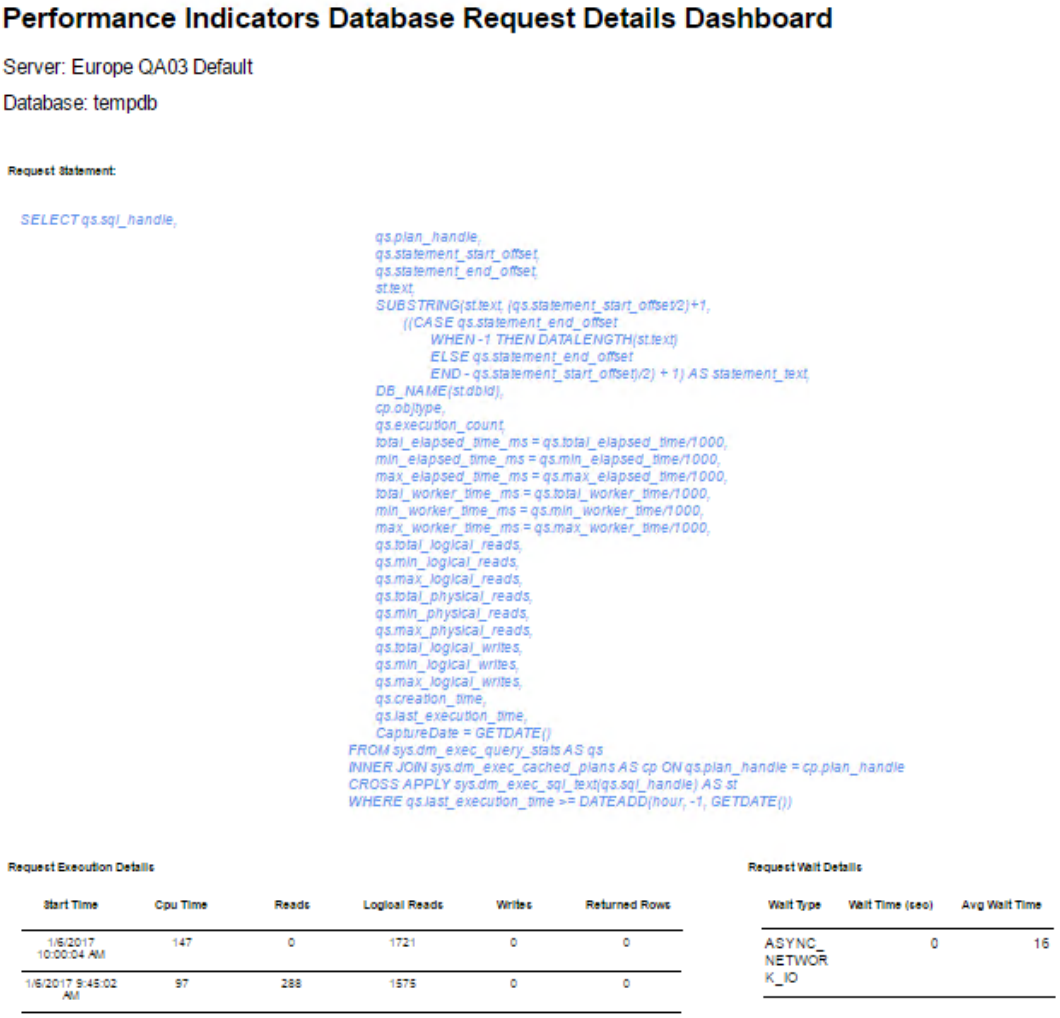
```
SELECT qs.sql_handle,
        qs.plan_handle,
        qs.statement_start_offset,
        qs.statement_end_offset,
        st.text,
        SUBSTRING(st.text, (qs.statement_start_offset/2)+1,
            ((CASE qs.statement_end_offset
                WHEN -1 THEN DATALENGTH(st.text)
                ELSE qs.statement_end_offset
                END - qs.statement_start_offset)/2) + 1) AS statement_text,
        DB_NAME(st.dbid),
        cp.objtype,
        qs.execution_count,
        total_elapsed_time_ms = qs.total_elapsed_time/1000,
        min_elapsed_time_ms = qs.min_elapsed_time/1000,
        max_elapsed_time_ms = qs.max_elapsed_time/1000,
        total_worker_time_ms = qs.total_worker_time/1000,
        min_worker_time_ms = qs.min_worker_time/1000,
        max_worker_time_ms = qs.max_worker_time/1000,
        qs.total_logical_reads,
        qs.min_logical_reads,
        qs.max_logical_reads,
        qs.total_physical_reads,
        qs.min_physical_reads,
        qs.max_physical_reads,
        qs.total_logical_writes,
        qs.min_logical_writes,
        qs.max_logical_writes,
        qs.creation_time,
        qs.last_execution_time,
        CaptureDate = GETDATE()
FROM sys.dm_exec_query_stats AS qs
INNER JOIN sys.dm_exec_cached_plans AS cp ON qs.plan_handle = cp.plan_handle
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) AS st
WHERE qs.last_execution_time >= DATEADD(hour, -1, GETDATE())
```

Executions	Total Cpu Time	Avg Cpu/Exec	Total I/O	Avg I/O/Exec	Total Reads	Avg Reads/Exec	Total Logical Reads	Avg Logical Reads/Exec	Total Writes	Avg Writes/Exec
2	244	122	3584	1792	288	144	3296	1648	0	0

```
DECLARE @deletedRecords as TABLE (EventId int NULL); WITH cte AS (SELECT TOP(100) EventId FROM UnpublishedEvents_10 WITH (ROWLOCK, READPAST) ORDER BY ID) DELETE FROM cte OUTPUT
deleted.EventId INTO @deletedRecords; SELECT ev.* FROM @deletedRecords d INNER JOIN Events_10 ev ON d.EventId = ev.EventId
```

Executions	Total Cpu Time	Avg Cpu/Exec	Total I/O	Avg I/O/Exec	Total Reads	Avg Reads/Exec	Total Logical Reads	Avg Logical Reads/Exec	Total Writes	Avg Writes/Exec
3	2	0	63	21	0	0	63	21	0	0

Figura 5.10: Relatório das Operações Executadas sobre uma Base de Dados.



Performance Indicators Database Request Details Dashboard

Server: Europe QA03 Default

Database: tempdb

Request Statement:

```
SELECT qs.sql_handle,
       qs.plan_handle,
       qs.statement_start_offset,
       qs.statement_end_offset,
       st.text,
       SUBSTRING(st.text, (qs.statement_start_offset/2)+1,
           ((CASE qs.statement_end_offset
               WHEN -1 THEN DATALENGTH(st.text)
               ELSE qs.statement_end_offset
               END - qs.statement_start_offset)/2) + 1) AS statement_text,
       DB_NAME(st.dbid),
       cp.objtype,
       qs.execution_count,
       total_elapsed_time_ms = qs.total_elapsed_time/1000,
       min_elapsed_time_ms = qs.min_elapsed_time/1000,
       max_elapsed_time_ms = qs.max_elapsed_time/1000,
       total_worker_time_ms = qs.total_worker_time/1000,
       min_worker_time_ms = qs.min_worker_time/1000,
       max_worker_time_ms = qs.max_worker_time/1000,
       qs.total_logical_reads,
       qs.min_logical_reads,
       qs.max_logical_reads,
       qs.total_physical_reads,
       qs.min_physical_reads,
       qs.max_physical_reads,
       qs.total_logical_writes,
       qs.min_logical_writes,
       qs.max_logical_writes,
       qs.creation_time,
       qs.last_execution_time,
       CaptureDate = GETDATE()
FROM sys.dm_exec_query_stats AS qs
INNER JOIN sys.dm_exec_cached_plans AS cp ON qs.plan_handle = cp.plan_handle
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) AS st
WHERE qs.last_execution_time >= DATEADD(hour, -1, GETDATE())
```

Request Execution Details

Start Time	Cpu Time	Reads	Logical Reads	Writes	Returned Rows
1/6/2017 10:00:04 AM	147	0	1721	0	0
1/6/2017 9:45:02 AM	97	288	1575	0	0

Request Wait Details

Wait Type	Wait Time (sec)	Avg Wait Time
ASYNC_NETWORK_IO	0	16

Figura 5.11: Relatório das Estatísticas de cada Execução de uma Operação.

Capítulo 6

Avaliação

Este capítulo descreve a metodologia a utilizar no processo de avaliação da ferramenta. Será descrita a metodologia utilizada para a avaliação, os items desenvolvidos para realizar a avaliação, a interpretação e análise dos resultados obtidos e por fim serão descritas as conclusões a retirar.

A avaliação será realizada através de um questionário à equipa de administração de bases de dados actualmente em funções na Farfetch.

6.1 Metodologia

O questionário que servirá de base da avaliação irá usar uma escala de *Likert* e será desenvolvido de acordo com a metodologia associada ao desenvolvimento deste tipo de relatórios. Uma escala de *Likert* é uma escala que oferece várias opções de resposta para que quem responde ao questionário tenha oportunidade de expressar objectivamente a sua opinião sobre um tópico.

Como descrito em [NB14], uma das vantagens destes questionários é o facto de permitirem recolher um grande volume de dados em pouco tempo.O grande objectivo do desenvolvimento de um questionário deste tipo é a obtenção de dados válidos e objectivos, para isso será utilizada a metodologia descrita em [NB14].

De acordo com a metodologia existem cinco etapas fundamentais para assegurar o desenvolvimento de um bom questionário:

1. Definição e compreensão das áreas que se pretendem avaliar.
2. Criação das perguntas de forma a que permitam obter dados concretos sobre o que se pretende avaliar.
3. Definição da escala associada a cada pergunta.
4. Escolha e definição do modelo de análise das respostas obtidas.
5. Refinamento das perguntas através da partilha.

Todos estes tópicos estão ligados no sentido em que uns dependem dos outros e, como o desenvolvimento do questionário é um processo contínuo, as alterações realizadas sobre um dos pontos têm influência directa sobre os outros pontos. Tendo em conta estas considerações serão agora descritas as particularidades associadas a cada etapa do desenvolvimento e que resultados se pretendem obter.

Independentemente do que se pretenda avaliar, o ponto de partida para o desenvolvimento de um questionário é ter uma ideia perfeitamente clara do que se pretende avaliar. No caso específico do trabalho desenvolvido esta percepção passa por definir que diferentes aspectos do projecto desenvolvido se pretende avaliar, e dentro dessas àreas quais são os principais aspectos a avaliar.

Esta etapa está intrinsicamente relacionada com a segunda, no sentido em que finda a primeira devem estar claramente identificados os aspectos a avaliar, é a partir desta definição que se vão desenvolver as perguntas necessárias para obter dados objectivos sobre os aspectos identificados. Esta relação próxima confere uma grande vantagem aos questionários desenvolvidos de acordo com a metodologia, no sentido em que as perguntas não são arbitrárias uma vez que têm por base as definições feitas na etapa anterior. Desta forma é possível associar directamente os dados obtidos de uma pergunta a um dos pontos que se pretendem avaliar.

As perguntas desenvolvidas na segunda etapa, por forma a garantir a qualidade dos resultados, devem seguir o seguinte conjunto de regras aquando da sua criação:

- Cada pergunta deve medir apenas um aspecto de um tópico que se pretenda avaliar.
- As perguntas devem ser diretas e fáceis de entender de maneira a que não seja ambígua.
- Não devem ser definidas perguntas com gramaticalidade complexa e não devem ser utilizadas conjunções como *e*, *ou* e *mas* uma vez que indicam a presença de duas ideias na mesma pergunta, o que mais tarde irá dificultar a tarefa de análise dos resultados uma vez que será difícil identificar a qual das perguntas a resposta foi dada.

Além das regras de escrita existem ainda outras considerações a ter em conta de acordo como o descrito em [NB14], nomeadamente o número de perguntas necessárias para avaliar um aspecto e se serão aplicados diferentes níveis de complexidade aos grupos de perguntas. Esta é uma das considerações mais importantes a ter no processo de desenvolvimento do questionário, uma vez que o nível de confiança e precisão dos resultados obtidos dependem directamente deste aspecto.

Idealmente seis a oito perguntas serão capazes de avaliar de forma concreta e precisa cada um dos aspectos identificados na primeira etapa. Outras considerações importantes a ter em conta são, nomeadamente, se as perguntas estão formuladas de uma forma positiva ou negativa. Este ponto é particularmente importante pois pode afectar a unidimensionalidade dos dados obtidos. A outra preocupação prende-se com a possibilidade das perguntas serem efectuadas na língua nativa dos alvos do questionário, elimina erros dos dados gerados pela má interpretação da pergunta. No final desta etapa, tanto os tópicos que se pretendem avaliar como as perguntas que irão avaliar cada tópico, devem estar claramente definidas.

A terceira etapa prende-se com a definição da escala de opções de resposta que será utilizada no questionário. Esta etapa deve ser abordada com o devido cuidado, uma vez que é através da escala definida que as respostas dadas serão categorizadas e classificadas para futura análise.

Uma escala de *Likert* é constituída por um conjunto de hipóteses de resposta que formam um contínuo, ou seja, variam desde uma opção claramente negativa até uma opção claramente positiva. Segundo [NB14], a grande maioria das escalas de *Likert* devem ser compostas por quatro ou seis opções de resposta, no entanto entre estas duas opções sempre que possível deve ser utilizada uma escala de seis opções, porque permite uma precisão de análise bastante superior. As escalas de quatro opções devem ser utilizadas quando os alvos dos questionários estão pouco motivados para responder, uma vez que o esforço necessário para responder é muito menor quando existem apenas quatro opções. No entanto, independentemente do tamanho da escala definida, existe um conjunto de características que a escala deve ter por forma a ajudar a garantir a qualidade dos resultados, nomeadamente, a escala deve avançar desde uma posição negativa até uma posição positiva em relação à pergunta, tal como uma medição realizada por um sistema métrico em que a escala avança desde valores mais pequenos até valores maiores. Outra característica importante é que a escala não tenha nenhum valor neutro principalmente por duas razões: primeiro porque respostas neutras causam problemas em termos de análise estatística, uma vez que é difícil de encaixar um valor neutro numa análise estatística, e segundo porque só devem ser realizadas perguntas às quais os alvos do questionário saibam responder o que torna uma opção de resposta neutra inútil. No final desta etapa devem estar definidos os aspectos a avaliar, as perguntas a colocar e a escala de respostas a aplicar.

A quarta etapa prende-se com a definição do modelo de análise para as respostas obtidas. O questionário a desenvolver tem a particularidade de não avaliar quem está a responder mas avaliar o projecto desenvolvido através da opinião de especialistas. Tendo em conta esta particularidade pretende-se analisar separadamente os dados respectivos a cada aspecto a avaliar por forma a ser possível apresentar dados concretos sobre cada um dos aspectos. No entanto, será também efectuada uma análise global por forma a aferir a avaliação do projecto desenvolvido como um todo. Para tal serão recolhidas as respostas e será efectuada a análise sobre cada aspecto individual e posteriormente com recurso a estes dados será realizada a análise global do projecto.

A etapa final prende-se com o refinamento das perguntas escolhidas e com a recolha das respostas. O refinamento das perguntas será realizado através da recolha de *feedback* sobre as mesmas e da utilização desse *feedback* para melhoramento das questões. O *feedback* será obtido de intervenientes que não venham a fazer parte do grupo de pessoas que irão responder ao questionário, por forma a não adulterar o processo, uma vez que com conhecimento prévio o interesse em responder ao questionário iria diminuir e por consequência a qualidade dos dados também. A parte final desta etapa consiste em recolher as respostas ao questionário. O questionário será respondido por um grupo restrito de pessoas, neste caso particular a equipa de administração de bases de dados actualmente em funções na Farfetch.

Importa salientar que a equipa teve um período para explorar a ferramenta desenvolvida, e que o questionário foi colocado a quatro pessoas.

6.2 Questionário

Nesta secção será descrito o questionário criado de acordo com a metodologia descrita anteriormente. Serão definidas as àreas que se pretendem avaliar, a quantidade de perguntas colocadas, bem como a escala de respostas definida para o questionário.

6.2.1 Àreas a Avaliar

As áreas que se pretendem avaliar com o questionário desenvolvido são as seguintes:

- Usabilidade e Performance — pretende-se perceber a opinião dos utilizadores relativamente à forma como é possível interagir com os relatórios apresentados, navegar entre os diferentes relatórios, bem como a performance dos mesmos em termos do tempo de demora para a apresentação dos dados. Para a avaliação desta àrea do projecto foram definidas oito perguntas, em todas as perguntas foi utilizada a escala de resposta definida.

- Interface — o objectivo deste ponto prende-se com a avaliação da interface dos relatórios apresentados. Foram definidas cinco perguntas para a avaliação desta àrea do trabalho e foi utilizada a escala de respostas definida.

- Qualidade das métricas de servidor — pretende-se obter a opinião dos utilizadores sobre a qualidade das métricas apresentadas e a importância das mesmas no contexto de monitorização da performance de uma instância de um servidor de SQL Server. Para a avaliação desta àrea foram definidas sete perguntas com a escala de respostas criada e foi colocada também uma pergunta de resposta aberta.

- Qualidade das métricas de bases de dados — este ponto pretende avaliar a qualidade e a pertinância das métricas de performance das bases de dados apresentadas. Foram definidas sete perguntas para avaliar esta àrea. Das seta perguntas seis utilizam a escala de resposta definida e uma é de resposta aberta.

- Modo de comparação dos dados históricos — pretende-se com este ponto perceber a opinião dos utilizadores relativamente ao modo desenvolvido para comparação de valores de métricas entre dois intervalos de tempo distintos. Para a avaliação da estratégia de comparação criada foram definidas oito perguntas, seta das quais utilizam a escala de resposta definida e uma é de resposta aberta.

6.2.2 Escala de Resposta

Tendo em conta, que o questionário será colocado a profissionais perfeitamente confortáveis com o tema, foi definida uma escala com seis valores de resposta. Desta forma é possível atingir maior granularidade nos dados recolhidos. A escala de resposta definida para o questionário foi a seguinte:

1	Discordo completamente.
2	Discordo.
3	Discordo parcialmente.
4	Concordo parcialmente.
5	Concordo.
6	Concordo completamente.

Estando definidas as àreas a avaliar, as questões a colocar e a escala de resposta, foi colocado o questionário à equipa. Na próxima secção serão apresentados e interpretados os resultados obtidos.

6.3 Interpretação dos Resultados

Para a interpretação dos resultados foi definida a estratégia: de analisar os resultados para cada àrea específica de avaliação. Após essa análise será também efectuada uma análise global para ser possível entender o grau de satisfação com a ferramenta como um todo.

A Figura 6.1 representa a dispersão das respostas às perguntas colocadas relativamente à usabilidade e performance da ferramenta.

Os resultados obtidos são bastante divididos. Após a análise verificou-se que em cinco das oito questões as respostas foram positivas, sendo que quatro delas foram muito positivas. Das restantes três, duas tiveram respostas negativas e uma muito negativa. Podemos concluir que de uma forma geral a usabilidade e performance da ferramenta é boa, no entanto tem alguns problemas que foram possíveis de identificar através das respostas.

No caso específico em que os intervalos de tempo seleccionados são muito grandes os tempos de resposta da ferramenta são bastante longos o que indica um problema de performance. Outro problema detectado prende-se com o facto da ferramenta precisar que sejam introduzidos os parâmetros de pesquisa de cada vez que o utilizador navega para outro relatório. Este problema afecta a experiência de usabilidade negativamente uma vez que é um processo bastante repetitivo.

Relativamente à interface, as respostas obtidas podem ser observadas na Figura 6.2.

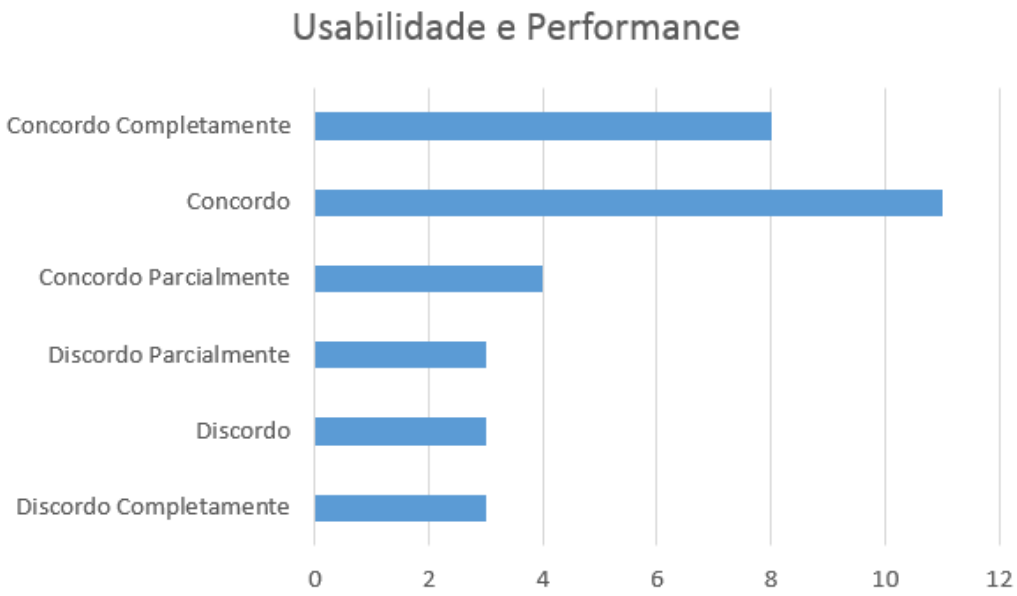


Figura 6.1: Respostas de Usabilidade e Performance.

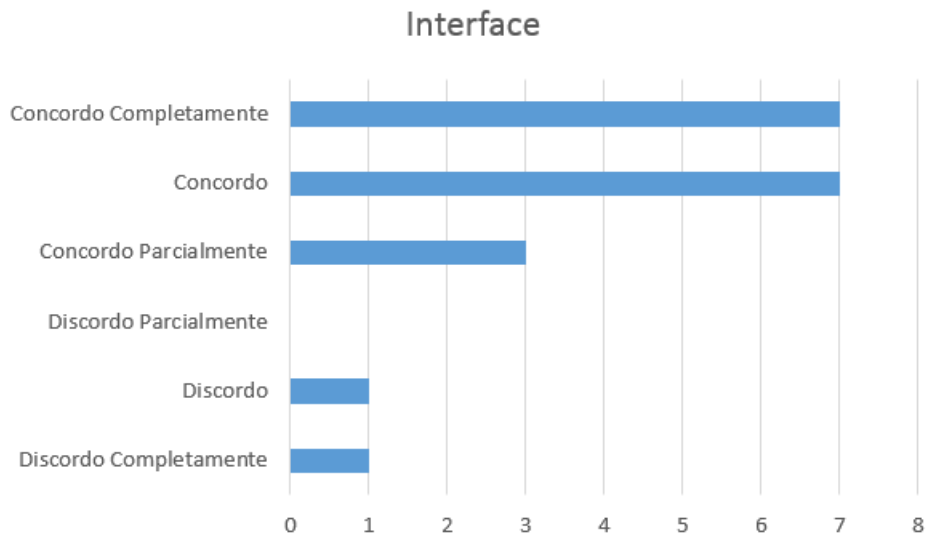


Figura 6.2: Respostas de Interface.

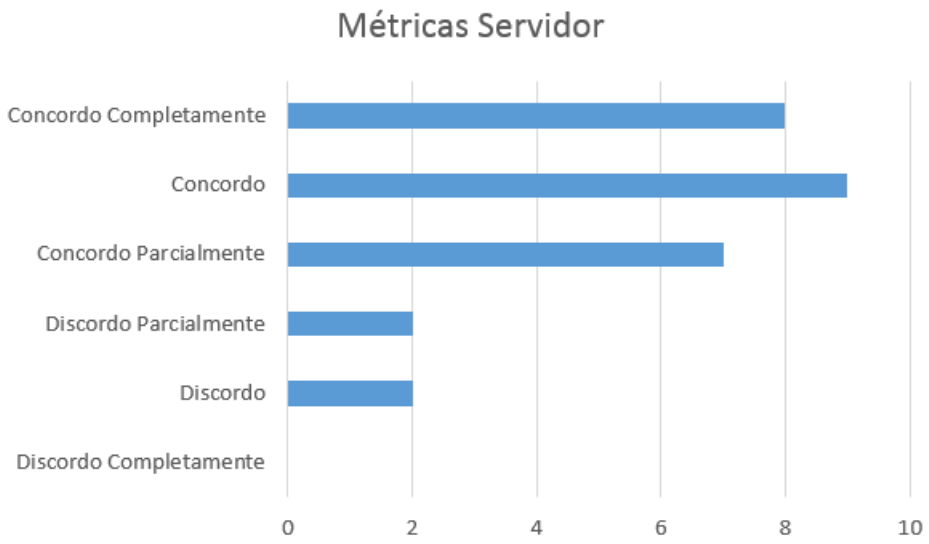


Figura 6.3: Respostas sobre Métricas do Servidor.

Os resultados obtidos para a avaliação da interface da ferramenta são bastante positivos. Praticamente todas as questões tiveram bons resultados, o que indica que os utilizadores estão satisfeitos com a interface apresentada. As àreas em que a avaliação foi mais positiva foram o facto de todos os relatórios terem o mesmo tipo de interface e também o tamanho dos objectos onde as métricas são representadas. O único aspecto a melhorar, de acordo com os resultados, será a eliminação de relatórios com múltiplas páginas, uma vez que dificultam as tarefas de análise.

Após a avaliação das àreas mais gerais, vamos agora analisar as àreas de maior interesse, nomedamente as métricas escolhidas e a metodologia de análise de dados comparativos.

Em relação às métricas escolhidas para a monitorização do estado do servidor, a Figura 6.3 apresenta a dispersão das respostas obtidas.

Após a análise dos resultados é possível extrair algumas conclusões: é consensual que os dados apresentados permitem ter uma ideia da performance do servidor, no entanto o conjunto de métricas de memória, *I/O* e Actividade não é suficiente para perceber se existem problemas de performance no servidor. Segundo os resultados, as métricas apresentadas em cada grupo apresentam informação relevante.

Relativamente à análise das operações que ocorrem no servidor, os dados apresentados permitem detectar a maioria das operações que impactam a performance do servidor apesar de serem apresentados apenas dez resultados por filtro. O único ponto negativo foi o relatório das sessões activas no servidor. De acordo com os resultados os dados apresentados não permitem aprofundar a análise a partir da informação que apresentam.

Esta secção continha uma pergunta de resposta aberta, a análise das respostas obtidas permitiu identificar um conjunto de possíveis melhorias a implementar. A apresentação do top dez de operações com maior custo por utilizador foi uma das sugestões, outra foi a apresentação da informação relativa aos índices que estão em falta no servidor.

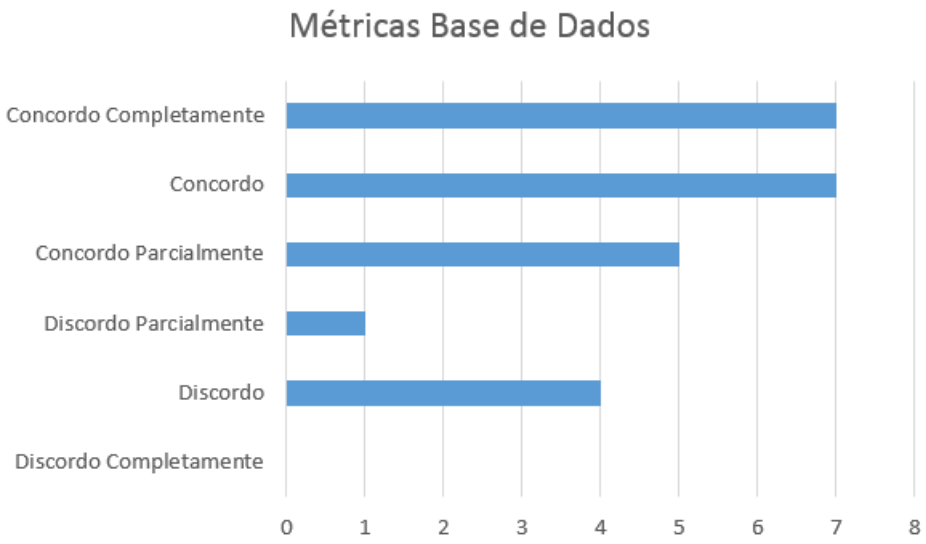


Figura 6.4: Respostas sobre Métricas de Base de Dados.

Relativamente às métricas de base de dados os resultados obtidos estão presentes na Figura 6.4.

É opinião geral que os dados apresentados, no relatório de performance de base de dados, disponibilizam informação suficiente para o utilizador perceber o que está a ser executado na base de dados. Ficou também claro que as relações entre as diferentes métricas permitem perceber a performance da base de dados. Um dos pontos mais importantes para os utilizadores é o facto de ser apresentado o conjunto de utilizadores que executam pedidos nas bases de dados, dado que essa informação ajuda bastante no processo de investigação de problemas de performance. O único ponto negativo é a informação apresentada sobre as tabelas presentes na instância. De acordo com as respostas os dados apresentados não são muito interessantes, no entanto foi feita a sugestão de apresentar dados das tendências de crescimento das tabelas nos últimos meses.

A apresentação de dados relativos ao consumo de processador por base de dados foi outra das sugestões para a melhoria deste relatório.

Por fim, as respostas recolhidas relativamente à metodologia de comparação de dados implementada teve a dispersão de respostas apresentada na Figura 6.5.

A principal conclusão a tirar das respostas obtidas é o facto de todos os utilizadores terem concordado que o processamento dos dados ajuda na tarefa de análise de problemas de performance, a grande maioria também indica que prefere analisar os dados comparativamente a uma *baseline* do que quando é realizada uma comparação directa entre os valores dos dois intervalos. De acordo com as respostas obtidas os utilizadores gostariam de aceder a relatórios de comparação das operações executadas nas bases de dados.

Da análise de resultados podemos concluir que de uma forma geral os utilizadores estão satisfeitos com o protótipo apresentado, no entanto é de enaltecer o facto de a metodologia de comparação de dados ter sido bem recebida. O ponto mais fraco, de acordo com as respostas obtidas, é

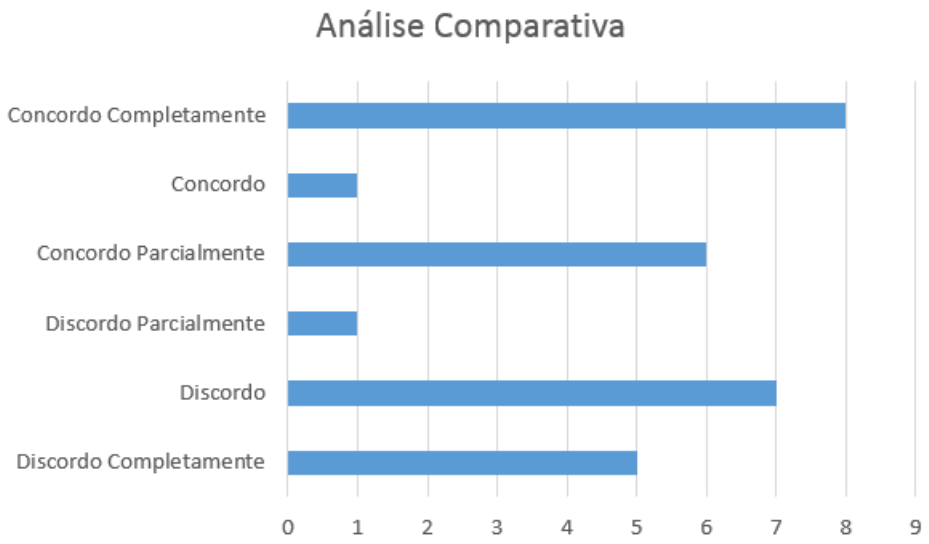


Figura 6.5: Respostas da Análise Comparativa.

sem dúvida a usabilidade da ferramenta.

6.4 Resumo e Conclusões

Este capítulo descreveu a metodologia de avaliação utilizada. Foram definidas as àreas que se pretendiam avaliar e foi desenvolvido um questionário para cumprir esse propósito. Espera-se que após a leitura deste capítulo seja possível perceber a opinião dos utilizadores relativamente à ferramenta e quais as àreas que necessitam de ser melhoradas.

No próximo capítulo são apresentadas as conclusões gerais sobre o trabalho desenvolvido e se o mesmo cumpriu os objectivos a que se propôs, será também apresentado o trabalho futuro a desenvolver sobre a ferramenta.

Capítulo 7

Conclusões e Trabalho Futuro

Neste capítulo vão ser revistos os objectivos da dissertação e os contributos feitos. Irão ser também apresentadas sugestões de trabalho futuro.

7.1 Resumo do Trabalho Realizado

A performance de um serviço é um tópico em constante evolução e que requer que as empresas se adaptem a novas realidades. Particularmente a monitorização da performance das bases de dados é um processo contínuo e passível de melhorias. Para fazer frente a esta necessidade foi desenvolvida a ferramenta descrita neste trabalho, um protótipo desenvolvido em cima do *SQL Server* que permite aos seus utilizadores realizar uma análise de performance dos servidores e das bases de dados neles alojadas através de um modelo de visualização de dados.

Foi também implementada uma metodologia de análise de dados históricos, que permite ao utilizador adoptar uma postura proactiva na análise da performance, uma vez que permite efectuar a comparação entre dados históricos e dados actuais.

A ferramenta desenvolvida recolheu dados reais dos servidores de desenvolvimento utilizados na Farfetch e foi testada pela equipa de administração de bases de dados em funções na empresa. Os resultados desta experiência foram favoráveis, uma vez que a avaliação efectuada revelou uma preferência dos utilizadores pela forma como os dados são apresentados no trabalho desenvolvido. O facto da ferramenta permitir uma análise de dados históricos foi também um ponto importante, em contraste com o que é actualmente utilizado.

7.2 Trabalho Futuro

O trabalho desenvolvido pode ser melhorado em vários aspectos. A ferramenta desenvolvida é apenas um protótipo e pode ser melhorada no capítulo das métricas monitorizadas. Devem ser adicionadas mais métricas e deve ser permitido ao utilizador definir as suas próprias métricas. Esta

alteração implica também uma re-estruturação da lógica de recolha das métricas. Para permitir este tipo de customização, o código de recolha de métricas deve estar armazenada em tabelas de configuração para ser possível ao utilizador adicionar as métricas que quiser. O modelo de visualização de dados é outro ponto passível de grandes melhorias. Deverá ser desenvolvida uma interface própria para eliminar os problemas detetados na navegação entre relatórios. No entanto, a grande mais valia estaria na possibilidade de atribuir ao utilizador a possibilidade de gerir todos os aspectos da ferramenta sem ter contacto com a implementação.

Outra das melhorias a considerar é o melhoramento do processamento dos dados históricos. Devem ser disponibilizadas mais opções de análise ao utilizador, nomeadamente opções diferentes dependendo do tipo de dados recolhidos.

Anexo A

Modelo de Dados

Este anexo demonstra a implementação de uma tabela do modelo de dados. O exemplo escolhido será a tabela *DataCenters*.

Tabela ***DataCenters***:

- Colunas:
  - *[DataCenterID] [int] IDENTITY(1,1) NOT NULL*, identificador único da entrada na tabela.
  - *[Name] [nvarchar](155) NOT NULL*, nome do Data Center.
  - *[Location] [nvarchar](155) NULL*, localização do Data Center.
  - *[Provider] [nvarchar](155) NULL*, provider do Data Center.
  - *[CreatedDate] [datetime] NULL*, data de inserção da entrada na tabela.
  - *[CreatedBy] [nvarchar](155) NULL*, utilizador que inseriu os dados.
  - *[UpdatedDate] [datetime] NULL*, data de actualização da entrada na tabela.
  - *[UpdatedBy] [nvarchar](155) NULL*, utilizador que efectuou a actualização da entrada na tabela.
  - *[Active] [bit] NULL*, campo de controlo sobre a utilização desta entrada pelo sistema de monitorização.
- Restrições:
  - *[PK_DataCenters] PRIMARY KEY CLUSTERED [DataCenterID] ASC*, chave primária da tabela, neste caso específico como foi criada com a indicação de *CLUSTERED* os dados são reorganizados fisicamente pelos parâmetros da chave primária, neste caso o *DataCenterID* de forma ascendente.
  - *[CreatedDate_DataCenter] Default Constraint getdate()*, restrição que em caso de a instrução de inserção de uma entrada na tabela não contenha um valor para a data de entrada insere a data actual.

- *[UpdatedDate_DataCenter] Default Constraint getdate()*, restrição que caso não seja passado um valor a este campo no momento em que é realizado um update na tabela insere a data actual.

A tabela *DataCenters* armazena informação relativa aos Data Centers onde os servidores estão alojados.

Anexo B

Índices

Durante o desenvolvimento do projecto foi necessário implementar um conjunto de índices para garantir a performance do modelo de visualização de dados. Foram criados dois tipos de índices: *Clustered* e *Nonclustered*. Este anexo demonstra a criação de um índice de cada tipo e salienta as diferenças.

- IX_FilesIO_FileID_IntervalStart_IntervalEnd
  - Tabela: FilesIO.
  - Chave do índice: FileID, IntervalStart e IntervalEnd.
  - Colunas incluídas: NumOfReads e NumOfWrites.
  - Tipo: NONCLUSTERED.
- CIX_InstanceConnections
  - Tabela: InstanceConnections.
  - Chave do índice: ServerID e InstanceID.
  - Colunas incluídas:
  - Tipo: CLUSTERED.

A principal diferença entre os dois tipos de índice é que um índice do tipo *Clustered* organiza os dados fisicamente pela chave usada na criação do índice, enquanto que um indíce *Nonclustered* fornece apenas uma estrutura para acelarar a pesquisa, no entanto não afeta a estrutura dos dados.

Referências

[Ape] ApexSQL. How to detect sql server performance issues using baselines – part 2 – collecting metrics and reporting. Disponível em http://solutioncenter.apexsql.com/how-to-detect-sql-server-performance-issues-using-baselines-part-2-colle Último acesso: 2016.

[Dav] Pinal Dave. Troubleshooting high cpu issues in sql server (part 1). Disponível em http://logicalread.solarwinds.com/troubleshoothigh-cpu-sql-server-pd01/#.WBy4ttWLSpo. Último acesso: 2016.

[DF] Tony Davis e Grant Fritchey. Sql server performance troubleshooting with sql monitor 5. Último acesso: 2016.

[DF10] Louis Davidson e Tim Ford. *Performance Tuning with SQL Server Dynamic Management Views*. Simple Talk Publishing, 2010.

[Eze14] Chuck Ezell. Changing your approach, from reactive to proactive. Disponível em https://www.datavail.com/resources/changing-your-approach-from-reactive-to-proactive/, 2014.

[GW00] Tom Bendix Gustav Wickström. The hawthorne effect — what did the original hawthorne studies actually show? *Scandinavian Journal of Work, Environment and Health*, 26(4):363–367, 2000.

[Haq14] Sarjen Haque. Real-time sql server performance monitor - extreme visibility to sql server database engine. Disponível em http://sqltouch.blogspot.pt/2014/03/free-real-time-sql-server-performance.html, Março 2014.

[Har16] Michele Hart. Power bi - basic concepts for power bi service. Disponível em https://powerbi.microsoft.com/en-us/documentation/powerbi-service-basic-concepts/, 2016.

[KS14] Jonathan Kehayias e Erin Stellato. Sql server performance tuning using wait statistics: A beginners guide. Technical report, Simple Talk, Abril 2014.

[Mica] Microsoft. Deadlocking. Disponível em https://technet.microsoft.com/en-us/library/ms177433(v=sql.105).aspx. Último acesso: 2016.

[Micb] Microsoft. Dynamic management views and functions (transact-sql). Disponível em https://msdn.microsoft.com/en-us/library/ms188754.aspx. Último acesso: 2016.

[Micc] Microsoft. Linked servers (database engine). Disponível em https://msdn.microsoft.com/en-us/library/ms188279.aspx. Último acesso: 2016.

[Micd] Microsoft. Power view: Explore, visualize, and present your data. Disponível em https://support.office.com/en-us/article/Power-View-Explore-visualize-and-present-your-data-98268d31-97e2-42aa-a52b-a CorrelationId=9556f877-ddd3-46c3-a1f3-3d89b337af1b&ui=en-US&rs=en-US&ad=US&ocmsassetID=HA102835634. Último acesso: 2016.

[Mice] Microsoft. Reporting services features. Disponível em https://technet.microsoft.com/en-us/library/ms159273(v=sql.105).aspx. Último acesso: 2016.

[Micf] Microsoft. Sql server agent. Disponível em https://msdn.microsoft.com/en-us/library/ms189237.aspx.

[Micg] Microsoft. Sql server, buffer manager object. Disponível em https://technet.microsoft.com/en-us/library/ms189628(v=sql.130).aspx. Último acesso: 2016.

[Mich] Microsoft. Sql server, sql statistics object. Disponível em https://technet.microsoft.com/en-us/library/ms190911.aspx. Último acesso: 2016.

[Mici] Microsoft. sys.dm_exec_query_stats (transact-sql). Disponível em https://msdn.microsoft.com/en-us/library/ms189741.aspx. Última revisão Agosto de 2016.

[Micj] Microsoft. sys.dm_exec_requests (transact-sql). Disponível em https://msdn.microsoft.com/en-us/library/ms177648.aspx. Último acesso: 2016.

[Mick] Microsoft. sys.dm_exec_sessions (transact-sql). Disponível em https://msdn.microsoft.com/en-us/library/ms176013.aspx. Última revisão Agosto de 2016.

[Micl] Microsoft. Transactions (database engine). Disponível em https://technet.microsoft.com/en-us/library/ms190612(v=sql.105).aspx. Último acesso: 2016.

[Mic16a] Microsoft. Database checkpoints (sql server). Disponível em https://technet.microsoft.com/en-us/library/ms189573.aspx, Setembro 2016.

[Mic16b] Microsoft. Monitor cpu usage. Disponível em https://msdn.microsoft.com/en-us/library/ms178072.aspx, 2016.

[Mic16c] Microsoft. sys.dm_exec_query_memory_grants (transact-sql). Disponível em https://technet.microsoft.com/en-us/library/ms365393.aspx, Fevereiro 2016.

[NB14] Tomoko Nemoto e David Beglar. Developing likert-scale questionnaires. In N. Sonda & A. Krause (Eds.), editor, *JALT2013, Conference Proceedings. (Tokyo, 25, 26, 27 e 28 Outubro de 2013)*, pages 1–8. Tokyo:JALT, 2014.

[Ora] Oracle. A relational database overview. Disponível em https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html. Último acesso: 2016.

[Pet14a] Milena Petrovic. Sql server memory performance metrics – part 2 – available bytes, total server, and target server memory. Disponível em http://www.sqlshack.com/sql-server-memory-performance-metricspart-2-available-bytes-total-server Fevereiro 2014.

[Pet14b] Milena Petrovic. Sql server memory performance metrics – part 4 – buffer cache hit ratio and page life expectancy. Disponível em http://www.sqlshack.com/sql-server-memory-performance-metrics-part-4-buffer-cache-hit-ratio-page Março 2014.

[Pet14c] Milena Petrovic. Sql server memory performance metrics – part 5 – understanding lazy writes, free list stalls/sec, and memory grants pending. Disponível em http://www.sqlshack.com/sql-server-memory-performance-metrics-part-5-understanding-lazy-writes-f Março 2014.

[pss13] psssql. Interpreting the counter values from sys.dm_os_performance_counters. Disponível em https://msdn.microsoft.com/en-us/library/ms189741.aspx, Setembro 2013.

[Som03] Erland Sommarskog. The curse and blessings of dynamic sql. Disponível em http://www.sommarskog.se/dynamic_sql.html#SQL_injection, Dezembro 2003. Última revisão Abril de 2015.