Relatório Final

Biblioteca Informatizada

Coordenado por: Prof. Dr. Pedro Brandão

Realizado por: André Tavares Rodrigues, n.º 2004

Turma 3M – Licenciatura Informática

Lisboa, 2015/2016

Resumo

Este projeto tem como objectivo principal demonstrar a eficiência e a facilidade de gestão que um sistema virtualizado tem. Apesar de não ser fácil configurar o sistema, o retorno no investimento pode ser visualizado por quem o utiliza. Para além disso, foi construida uma aplicação para demonstrar a interação entre diversos ambientes de trabalho, assim como uma base de dados que disponibiliza uma forma de armazenamento para a informação relativa aos conteúdos da biblioteca através da respetiva aplicação.

O projeto foi dividido em duas fases: primeira fase foi a de construção do sistema, utilizando três servidores para possibilitar a criação de máquinas virtuais para que utilizadores pudessem interagir com a aplicação; a segunda passou por fazer a aplicação, em ADO.NET na linguagem C#, e garantir a comunicação com a base de dados central.

Disto resultou um sistema que responde às necessidades de cada uma das partes – utilizadores e administradores – tendo os utilizadores uma maneira eficiente de aceder à informação providenciada pela base de dados e acesso à internet; os administradores do sistema têm uma estrutura que responde às necessidades de provisionamento por parte dos utilizadores e facilita o processo de gestão de cada máquina.

Palavras-chave: máquina virtual; *host*; servidor.

Abstract

The following project aims to demonstrate the ease of management and efficiency in a virtualized environment. Although it´s not easy to set up, the benefits outweigh the time and effort. In addition, to demonstrate interaction between the users and the servers, an application was created, which connects to a centralized database in one of the servers.

In order to create the project, the planning was split into two phases: construction of the virtualized environment, which includes three servers to allow the provisioning of virtual machines to the library users and the respective application; the second was the application creation phase, in C# using ADO.NET, which allows communications to the database.

From all of this, a system was created that answers the needs of the users and administrators alike. Users can access the information made available inside the database alongside with having access to the internet, and administrators have a system that easily allows them to provision whatever resources they deem necessary, all of which are easily monitored, allowing better management.

Índice

Índice de Imagens

Introdução

O mundo está cada vez mais virado para a automação de tarefas tentando simplificar o dia-a-dia de cada um. Tendo isto em conta, tentei criar um modelo informático de gestão de uma biblioteca. O modelo de gestão permite a biblioteca ter a sua informação numa base de dados centralizada, facilitando assim a pesquisa por parte das pessoas ao material existente e opera sob um sistema virtualizado, que dispõe aos utilizadores um interface virtual, idêntico ao de um computador.

Para que este cenário funcione, foram criados três servidores num ambiente virtualizado, dos quais um controlador de domínio, um gestor de máquinas virtuais e um servidor de base de dados, que serve como hospedeiro para as respetivas máquinas virtuais criadas.

O programa que permite interagir com a base de dados foi construído na linguagem C#, usando o *framework* ADO.NET. A base de dados foi integralmente construida através do SQL Server.

Os computadores controlados pelos utilizadores são máquinas virtuais, criadas pelo gestor de máquinas virtuais do Windows. Estas máquinas são controladas pelo administrador do sistema.

A elaboração deste projeto passa pela criação dos servidores que permitem manter todo o sistema centralizado, seguido da criação das máquinas virtuais que os utilizadores irão interagir, e pela criação da aplicação e da base de dados.

Cada um dos servidores irá ter um propósito distinto, os quais poderá ver mais adiante.

Estado da Arte

Índice

Introdução

Para a elaboração deste trabalho as tecnologias utilizadas passam pelo uso da Virtualização, do uso da *Cloud Computing*, da linguagem SQL e do ADO.NET. Abaixo descreverei em detalhe cada uma destas tecnologias e algumas vantagens da utilização das mesmas.

Virtualização

Conceito

O conceito surgiu na década de 60, e desde então tem sofrido modificações. A virtualização pode ser vista como uma abstração computacional, "é um termo geral usado para descrever várias tecnologias que dividem os recursos de *hardware* em múltiplos ambientes, aplicando um ou mais conceitos ou tecnologias" - (Paula Agostinho, 2009, p. 1).

Técnicas de virtualização permitem remover limites de recursos e maximizam a utilização dos mesmos (EMC, 2008, p. 5).

Um exemplo disso é a virtualização dos sistemas operativos – a forma mais predominante atualmente. Estes tornaram-se um elemento fulcral nas infraestruturas modernas de TI[1]. Máquinas virtuais são geralmente implementações completas de sistemas operativos *standard*, que correm simultaneamente no mesmo *hardware* (F5, 2007, p. 2).

Virtualização abrange vários segmentos das tecnologias de informação. Servidores, switches, armazenamento, networking, e clientes estão todos encaminhados para a virtualização. Mas este movimento está maioritariamente centralizado em servidores, e a virtualização destes terá maior impacto a nivel das redes de *datacenters* (Broadcom, 2010, p. 2). De acordo com CIO

[1] Tecnologias de Informação

Insight, mais de 50% dos servidores já usam virtualização, e em 2016 esse número deve ascender até os 86%[2].

A revolução no mundo dos servidores

Esta tecnologia revolucionou a maneira como os *data centers* operam. Reduziu os custos (a nível de consumo de energia e de *hardware*) e permitiu um incremento na eficiência computacional – "*more computing power watt of power consumed by the data center*" (Grid, 2010, p. 2).

Os custos reduziram significativamente graças à consolidação de servidores (Fig. 2) – "permite que servidores físicos acomodem uma ou mais aplicações de servidor" afirma a Techopedia. Segundo eles, faz com que seja possível partilhar recursos computacionais de um servidor entre diferentes aplicações e serviços simultaneamente, assim perimitindo o uso total dos recursos de cada servidor (Fig. 1). Tradicionalmente, um servidor físico apenas utiliza entre 15-30% da sua capacidade computacional total. Com a consolidação dos servidores é possível atingir um uso de cerca de 80% das mesmas capacidades computacionais[3]. É usado primordialmente para reduzir o número de servidores (resolve o problema de "*server sprawl*"[4]) necessários numa organização, e, por consequente, reduzir os gastos, tanto energéticos como os de *hardware* necessário para manter o funcionamento dos servidores.

[2] http://www.cioinsight.com/it-strategy/cloud-virtualization/slideshows/useful-virtualization-stats-trends-and-practices.html
[3] https://www.techopedia.com/definition/16016/server-consolidation
[4] O problema de server sprawl: deriva maioritariamente da compra de servidores baratos, *low-end,* e da prática de usar servidores dedicados a aplicações únicas.

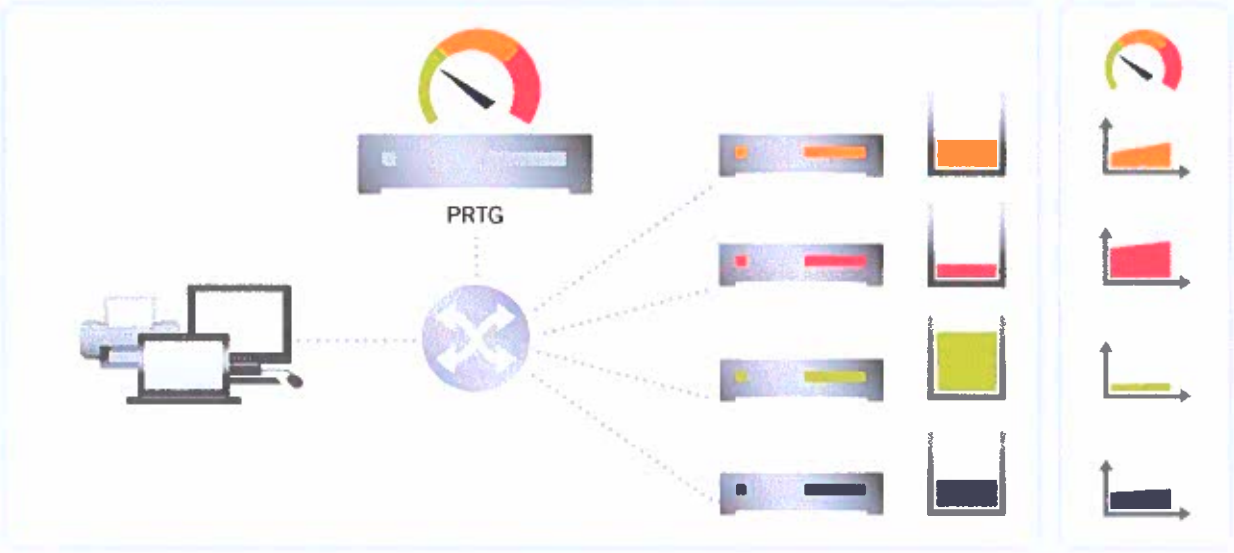


Figura 1 – Ambiente distribuído. É possível utilizar os recursos de cada servidor conforme necessário, obtendo um melhor desempenho. Fonte: (Paessler, 2008)

Reduzir o número de servidores em funcionamento faz mais que apenas poupar o consumo energético; isto cria um efeito dominó no *data center* inteiro.

Consolidados os servidores, há uma redução de *"cooling load"*[5], incremento no tempo de uso do *UPS* e também do gerador, caso haja falhas elétricas (Grid, 2010, p. 3).

Para além de consumirem menos recursos, utilizam, também, menos espaço, devido ao facto de haver menos *hardware* envolvido, o que permite uma melhor gestão em comparação com ambientes com servidores distribuídos – há uma tendência a haver mais servidores centralizados (Paula Agostinho, 2009, p. 3).

[5] Cooling load: energia necessária para manter temperaturas estáveis no *data center* – menos energia gasta em referigeração.

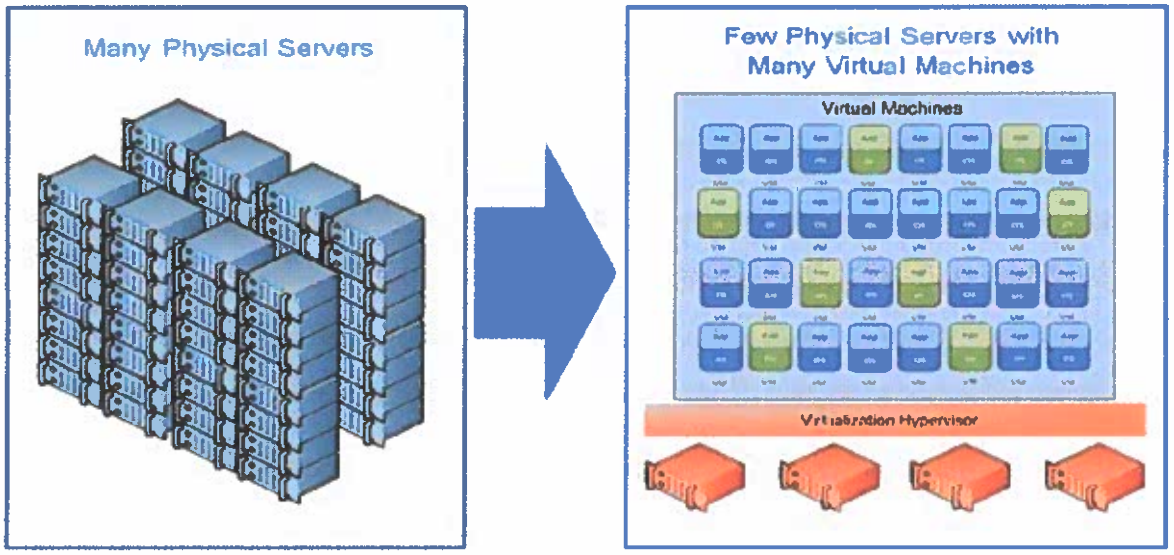


Figura 2 – Consolidação de servidores. Diferença entre usar muitos servidores dedicados face a poucos servidores virtualizados.

Fonte: https://www.boostitco.com/how-virtualization-can-help-your-business/

Noutra vertente, a virtualização traz outros benefícios como é conferido pela (ISACA, 2010): a flexibilidade – é possível criar, relativamente facilmente, ambientes de testes e produção rapidamente, eliminando a necessidade de provisionamento; agilidade e resposta – permite uma adaptação rápida de recursos conforme necessário (é possível utilizar simultaneamente vários recursos físicos, otimizando o uso dos mesmos - geralmente isto acontece quando há um pico de uso dos recursos nos servidores), com provisionamento automático do sistema, isto também permite melhorar os tempos de resposta e reduzir os tempos de *downtime*; simplificação para a equipa de TI - havendo um menor número de servidores existentes, a probabilidade de falhas de equipamento e o custo de manutenção, tornando assim a sua gerência mais fácil.

Cuidados a ter

Embora a virtualização apresentar-se como uma solução muito positiva, há que ter em consideração os seguintes aspetos: a segurança e a sua gestão.

Segundo Neil MacDonald, especialista em segurança da Gartner, hoje em dia, as máquinas virtuais são menos seguras que as máquinas físicas justamente por causa do VMM[6], pois se o sistema operacional hospedeiro tiver alguma vulnerabilidade, todas as máquinas virtuais hospedadas nessa máquina física estão vulneráveis (Fabiano Muchalski, p. 3).

Como consta no *paper* da ISACA (ISACA, 2010, pp. 6-7) - isto denomina-se *hyperjacking*. É "injetado" um *hypervisor* clandestino (também chamado VMM), dentro da infraestrutura com controlo sobre as interações entre o sistema alvo e o *hardware*. A outra questão a ter em atenção relativamente à segurança é a de *"VM jumping"*, este método tira partido das vulnerabilidades nos *hypervisors*, que permite *malware* ou ataques remotos, e assim, compromete a proteção da separação das máquinas virtuais, dando assim acesso às outras máquinas virtuais, ao *host* ou até mesmo ao *hypervisor*.

No que toca a gestão, pode se tornar uma tarefa difícil para a equipa de TI, de onde surge o problema de *VM sprawl*[7]. Sendo mais fácil provisionar e criar ambientes virtuais, do que usar sistemas físicos, usando máquinas virtuais, o número e o tipo de máquinas podem ser excessivos, o que as torna extremamente difícil de controlar[8].

Para colmatar estas dificuldades, o melhor método para mitigar a ameaça de *hyperjacking* é de usar o *hypervisor hardware-rooted* seguro e de confiança. No caso do *"VM jumping"*, sendo o *hypervisor* e as máquinas virtuais *software*, é necessário que haja *updates*, *patches*, para garantir a segurança. Para a gestão, há produtos e serviços que fazem a gestão, isto permite reconhecer e saber a localização das máquinas virtuais, assim como os sistemas e aplicações no *data center*.

---

[6] Virtual Machine Manager: "solução de gestão para o *datacenter* virtualizado, que permite configurar e gerir o host de virtualização e os seus recursos." - https://technet.microsoft.com/pt-br/library/gg610610.aspx

[7] VM Sprawl: "fenómeno que ocorre quando o administrador da rede não consegue efetivamente gerir o número de máquinas virtuais existentes num sistema." - http://whatis.techtarget.com/definition/virtualization-sprawl-virtual-server-sprawl

Hypervisor

A tecnologia

A Global Knowledge (Knowledge, 2011, p. 2) definiu *hypervisor*, também chamado de VMM (Virtual Machine Manager), como sendo a tecnologia que permite que várias instâncias de sistemas operativos, chamados de *guests*, funcionem num único dispositivo, chamado de *host*. Cada máquina virtual "pensa" que está conectada ao seu próprio *hardware*. Os *hypervisors* são componentes de *software* desenhados para gerir os sistemas operativos *guests* nos *hosts*.

A tecnologia do *hypervisor* permite que máquinas virtuais que contenham qualquer *software* sejam modificas, copiadas, restauradas e que seja feito o seu *backup*, ou até mesmo a reversão para um estado anterior com um simples clique do rato. Isto traz imensos benefícios a nível de velocidade e eficiência para todos os aspetos das TI.

Atualmente, é raro o sistema que não seja afetada pela tecnologia, desde utilizadores comuns, a programadores, todos sentiram, ou sentirão o impacto dela. Esta tecnologia está a mudar o mundo da TI. É esta tecnologia que permite que a virtualização aconteça.

Tipos de *hypervisors*

Há dois tipos de *hypervisors*, o tipo 1, e o tipo 2 (Fig. 3).

Nos *hypervisors* do tipo 1, comumente denominados de "bare-metal *hypervisors*", o *hypervisor* comunica diretamente com o *hardware* da máquina, evitando o uso de um sistema operativo no computador host. Isto traz tremendas vantagens a nível de *performance* para as máquinas virtuais, e traz maior flexibilidade, para além de redizur o *overhead* necessário para correr o *hypervisor*. "Tipicamente estes *hypervisors* são mais eficientes que os do tipo 2, mas em muitas maneiras providenciam o mesmo tipo de funcionalidade porque ambos usam o mesmo tipo de *VM's*"[9]. Um exemplo de *hypervisor* deste tipo é o Hyper-V da Microsoft.

[9] http://searchservervirtualization.techtarget.com/tip/Virtualization-hypervisor-comparison-Type-1-vs-Type-2-hypervisors

Por outro lado, os *hypervisors* do tipo 2 necessitam de ter um sistema operativo instalado no *host*, criando assim uma camada adicional entre a máquina virtual e o *hardware*. Isto reduz a *performance*. É melhor usado como ambientes de teste , ou para uso de um segundo sistema operativo. Em sistemas em que há aplicações de necessidade crítica no *data center*, a melhor opção é o tipo 1 – (Knowledge, 2011, p. 3).

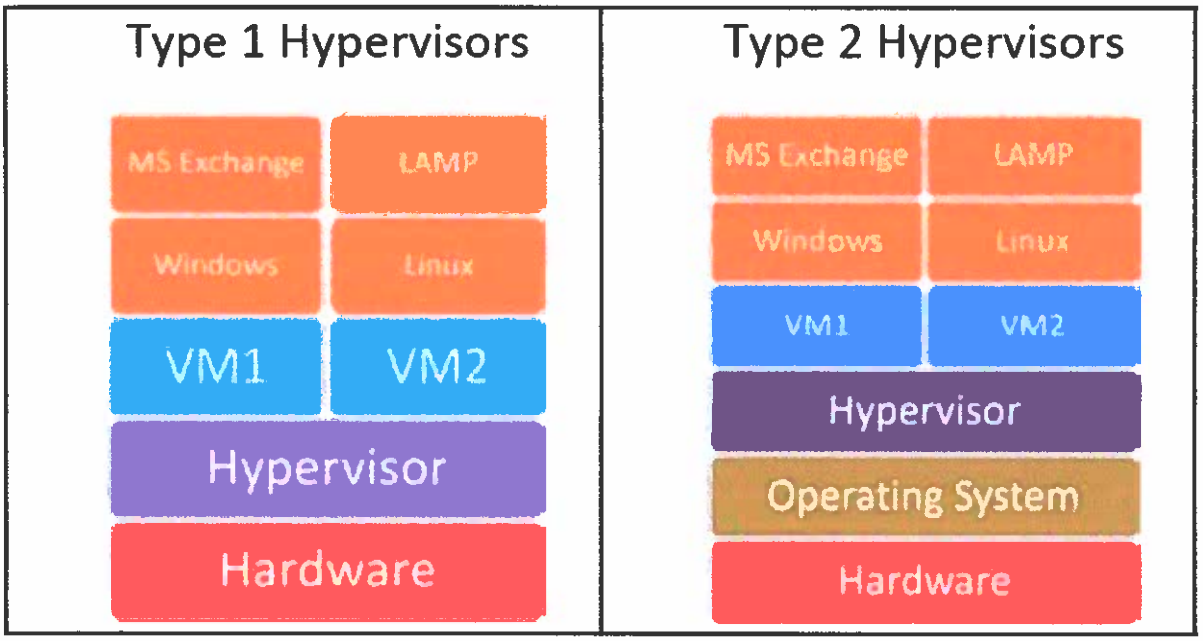


Figura 3 – A diferença entre os dois tipos de *hypervisor*. Fonte: (Knowledge, 2011)

Hyper-V em análise

O Hyper-V é uma *role* que permite criar e gerir um ambiente computacional virtual usando a tecnologia do Windows Server[10]. Foi inicialmente introduzido no Windows Server 2008 (Fig. 4) e desde então tem sido melhorado.

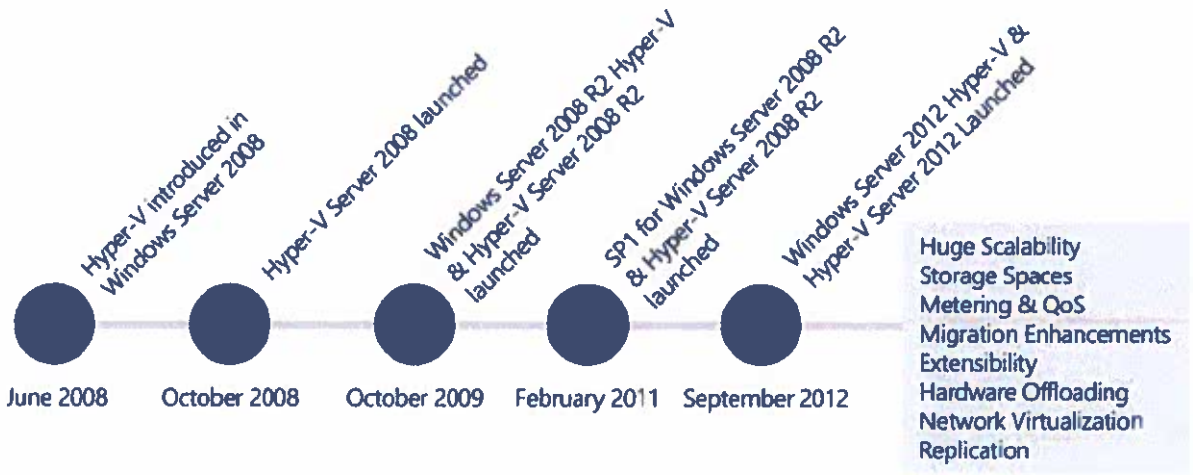


Figura 4 – A história do Hyper-V. Fonte: (Windows, 2013)

Da versão base do Windows Server 2008 para a versão R2, foram introduzidos um número de características que ajudam organizações reduzir custos e simultaneamente aumentam a agilidade e flexiblidade. Alguns exemplos disso são:

. ***Live Migration*** - permite a migração de máquinas virtuais sem interrupções;

. ***Processor Compatibility*** – há maior flexibilidade para o uso do *Live Migration* entre *hosts* com diferentes arquiteturas de processadores;

Após o update do SP1 (Service Pack 1) do Windows, foram introduzidos mais características importantes que permitiam um uso melhor do *hypervisor,* sendo esta de grande importância: . ***Dynamic Memory*** - é possível gerir mais eficientemente a memória e ao mesmo tempo reter um desempenho de carga de trabalho escalabilidade (Windows, 2013, p. 4).

[10] https://technet.microsoft.com/library/hh831531.aspx

Áreas chave

O Windows Server está agrupado, pelo grupo Windows (Windows, 2013, p. 4), em quatro áreas chave, sendo estas:

. **Escalabilidade, Desempenho e Densidade** – há uma necessidade dos consumidores em utilizar máquinas virtuais cada vez maiores e mais potentes, para garantir uma resposta à crescente carga de trabalho. Em adição, à medida que o *hardware* aumenta, há também a necessidade de utilizar esses recursos o melhor possível.

	Resource	Windows Server 2008 R2 Hyper-V	Windows Server 2012 R2 Hyper-V	Improvement Factor
Host	Logical Processors	64	**320**	**5×**
	Physical Memory	1TB	**4TB**	**4×**
	Virtual CPUs per Host	512	**2,048**	**4×**
VM	Virtual CPUs per VM	4	**64**	**16×**
	Memory per VM	64GB	**1TB**	**16×**
	Active VMs per Host	384	**1,024**	**2.7×**
	Guest NUMA	No	**Yes**	-
Cluster	Maximum Nodes	16	**64**	**4×**
	Maximum VMs	1,000	**8,000**	**8×**

Figura 5 – Comparção entre modelos do Windows Server e a sua escalabilidade.

Fonte : (Windows, 2013)

. **Segurança e Manutenção** – *data centers* virtualizados é uma prática comum nos dias de hoje. Organizações de TI e provedores de serviço de infraestruturas (IaaS) são obrigados a garantir a segurança, por vezes, utilizando métodos de cifra, quando necessário.

. **Alta Disponibilidade e Resilência** – com um aumento de confiança dos consumidores na virtualização, a importância em manter as cargas de trabalho ativas aumenta significativamente. Ter funções embutidas na plataforma onde manter essas cargas de trabalho ativas, e também, numa eventualidade de desastre, poder restaurá-las para outra área geográfica é de extrema importância.

. **Infraestrutura Flexível** – nos *data centers* modernos, os consumidores esperam que haja uma resposta imediata e eficiente para corresponder às necessidades dos negócios. Poder mudar cargas de trabalho (*workloads)* flexivelmente pela infraestrutura é de extrema importância.

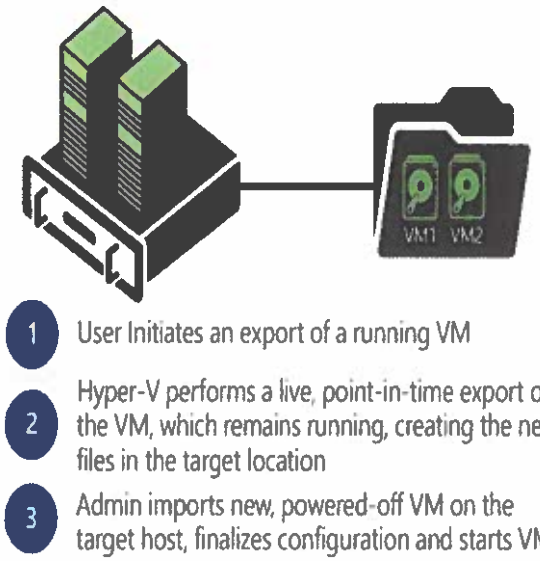


Figura 6 – Importação de *VM's* para o Hyper-V.

Fonte: (Windows, 2013)

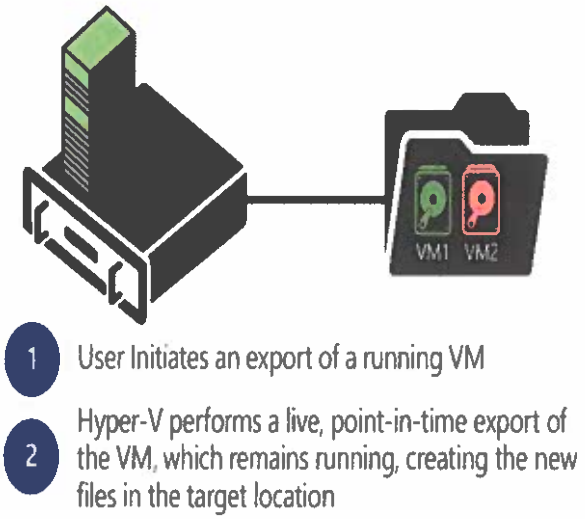


Figura 7 – O uso do *Virtual Machine Live Cloning.*

Fonte: (Windows, 2013)

ADO.NET

Introdução e definição

Acesso, transporte, armazenamento e manipulação de dados são processos fundamentais e subjacentes em qualquer aplicação/sistema (Alfred J. Lendvai, 2007, p. 1). Aplicações modernas requerem serviços de manuseamento de dados em todas as camadas (Microsoft Corporation, 2006, p. 31).

Segundo a Microsoft, o ADO.NET é um conjunto de classes que expõem serviços de acesso a dados e providencia uma série de componentes para criar aplicações distribuidas de partilha de dados[11].

A sua estrutura

Conforme (Alfred J. Lendvai, 2007), o acesso de dados do ADO.NET é baseado num *DataSet* e num *Data provider* que dão acesso ao *Data Source*. É composto por um conjunto chave de objectos que incluem: *DataSet, DataTable, DataColumn, DataRow, DataView, DataRelation* e os *Constraint objects.*

O *DataSet* é como se fosse uma base de dados mini-relacional, que fica guardada em memória, e é um objeto autónomo independente de qualquer provedor de dados. O *Data Provider* fornece os meios de comunicação com qualquer *Data Source*. O *.NET Data provider* fornece um conjunto de objetos que permitem a conexão, a busca de dados, o update dos dados e disconectar do *Data Source*.

O modelo ADO.NET introduz um nível de abstração onde há a separação entre os dados e o processo pelo qual os dados são extraídos, como é possível ver na figura abaixo (Fig. 8). A parte central deste modelo é o *Data Adapter*. Cada *Data Adapter* tem um específico *Data Provider*. É a ponte entre o *DataSet* e o *Data Source* e permite a coordenação entre a memória interna – *DataSet* – e a fonte permanente de acesso aos dados, através do *Data Provider*.

[11] https://msdn.microsoft.com/en-us/library/e80y5yhx%28v=vs.110%29.aspx?f=255&MSPPError=-2147217396

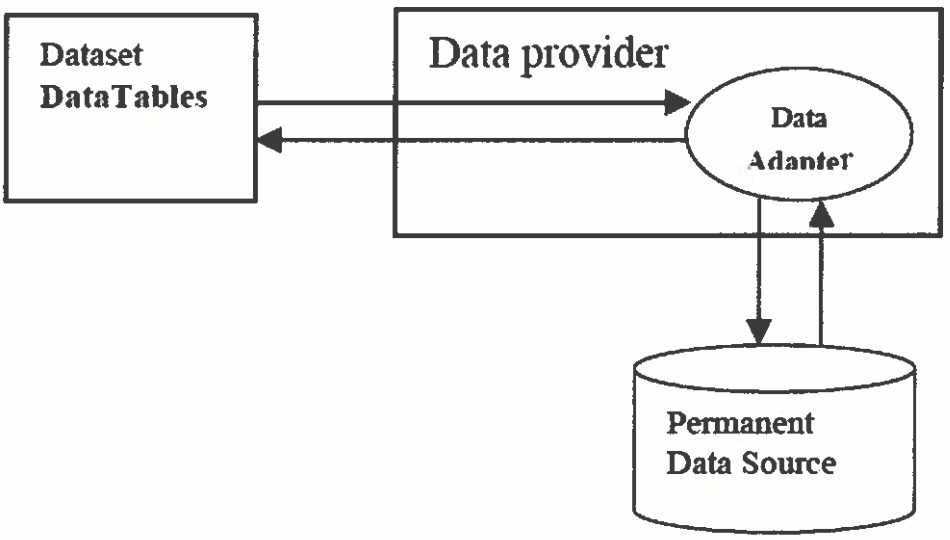


Figura 8 – Abstração ADO.NET. Fonte: (Alfred J. Lendvai, 2007)

O *Data Adapter* usa o *Data Reader* para preencher a tabela utilizando o método Fill. Há quatro comandos que este usa do objeto *Command*, estes são o *Insert*, o *Update*, o *Select* e o *Delete*. O preenchimento do *DataSet* pode ser feito utilizando vários *Data Adapters*.

O *Framework* .NET fornece *Data Providers*, estes dão acesso a vários *providers* como o OLE DB .NET e o SQL Server .NET. O provedor do SQL Server .NET fornece um acesso mais eficiente e direto ao API[12] da base de dados em relação ao outro provedor. Em geral, o provedor do SQL Server excede os outros em termos de *performance*.

[12] Aplication Programing Interface

SQL Server

Introdução

O volume e a complexidade de dados está a crescer exponencialmente, e a habilidade de tirar proveito disso é cada vez mais importante para as organizações.

Organizações de TI têm de balancear o impacto desta explosão de dados contra a necessidade de *uptime* constante, disponibilidade global e um mundo digital onde os consumidores experam mais e melhor.

Mais que nunca, organizações precisam de *performance*, estabilidade, segurança, disponibilidade, e da habilidade de poder utilizar os dados de forma a poderem tirar partido dessa informação para o seu proveito.

A linguagem estruturada

O SQL (Structured Query Language) é uma linguagem standardizada para o acesso e a manipulação das bases de dados. Esta permite que sejam executadas *queries* à base de dados – pedidos de informação; inserir, apagar, atualizar, e buscar informações na base de dados; e criação de tabelas, de stored procedures e novas bases de dados[13].

SQL Server *Performance*

Para satisfazer as necessidades da nova geração de informação, a Microsoft investe fortemente em cinco áreas (Microsoft, 2015, p. 4):

. **Performance:** a memória integrada do SQL Server aumenta o desempenho drasticamente numa variedade de cenários.

[13] http://www.w3schools.com/sql/sql_intro.asp

. **Segurança e Conformidade:** são desenvolvidas novas funcionalidades para proteger os dados, com funções como *"Always Encrypted" e "Row-Level Security"*.

. **Disponibilidade:** o SQL Server é conhecido pela sua solidez e fiabilidade. Melhoria do "AlwaysOn", melhor *load balancing*, e novas funções – flexíveis e eficientes - para a criação de *backups* são criadas.

. **Escalabilidade:** novos avanços em computação, armazenamento e *networking* terão um impacto direto nos *workloads* do SQL Server.

. **Cloud services:** os novos serviços no SQL Server em conjunto com o Microsoft Azure tornam a escalabilidade ainda mais fácil.

Cloud Computing

Definição

Houve muita discussão acerca da definição de cloud computing. O termo parece originar em diagramas de redes de computador que representam a internet como uma *cloud*. A maioria das companhias de TI escreveram papeis acerca do assunto, tentando definir o termo. A discussão foi apaziguada, e uma definição foi reconhecida pelo NIST (National Institute of Standards and Technology).

Esta definição pode ser sumarizada como "*a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction* (NIST, 2011, p. 2)" – traduzido para: um modelo que permite convenientemente, acesso imedeato em rede a um conjunto de recursos computacionais configuráveis (p.e. redes, servidores, armazenamento, aplicações e serviços) que podem ser rápidamente provisionados e despachados com a mínima gestão de esforço ou necessidade de interação com o provedor do serviço.

Características

As características essenciais definidas pelo NIST (NIST, 2011, p. 2) são:

. ***On-demand self-service:*** o consumidor pode unilateralmente provisionar capacidades computacionais, como tempo de servidor ou armazenamento na rede, conforme necessita, automaticamente, sem requerer interação com cada provedor de serviços.

. ***Broad network acess:*** serviços estão disponíveis através da rede e podem ser acedidos em qualquer tipo de plataforma através de mecanismos standard.

. ***Resource pooling:*** as capacidades computacionais do provedor estão divididas para suportarem múltiplos utilizadores, usando um modelo *multi-tenant*, com diferentes recursos – físicos e virtuais – dinamicamente atribuídos e re-atribuídos conforme a necessidade do consumidor.

. ***Rapid elasticity:*** capacidades computacionais podem ser provisionadas e removidas, em alguns casos automaticamente, visando expandir para dentro ou fora, de acordo com a necessidade.

. ***Measured service:*** os sistemas cloud automaticamente controlam e otimizam o uso dos seus recursos, através de mecanismos de medição a um nível de abstração apropriado com o tipo de serviço. O uso de recursos pode ser monitorizado, controlado, e reportado, dando transparência ao provedor e ao consumidor do serviço utilizado.

Modelos de serviço

. **Software as a Service (SaaS):** ao consumidor é conferida a possibilidade de utilizar a infraestrutura *cloud* para uso de aplicações. As aplicações são acessíveis por diversos dispositivos, através de um *"thin client"* como um web browser. O consumidor não gere nem controla os recursos computacionais da infraestrutura.

. **Platform as a Service (PaaS):** é dado ao consumidor a capacidade de implementar na infraestrutura *cloud* aplicações criadas por si próprios, ou compradas, usando linguagens de programação suportadas pelas ferramentas do provedor. O consumidor não tem capacidade de controlo nem de gestão dos recursos computacionais da cloud, isto inclui redes, servidores, sistemas operativos e armazenamento. Somente tem controlo das aplicações que desenvolve/utiliza e, talvez, do ambiente aplicacional do *host.*

. **Infrastructure as a Service (IaaS):** o consumidor tem acesso aos recursos computacionais do *host*, como a rede, armazenamento, processamento, de entre outros, onde tem a capacidade de implementar o seu software, o que pode incluir sistemas operativos e aplicações. O consumidor não gere nem controla a infraestrutura *cloud*, simplesmente controla o sistema operativo, aplicações, rede, e possivelmente alguns componentes da rede.

Modelos de implementação

. **Private Cloud:** a infraestrutura é unicamente gerida para uma organização. Pode ser gerida pela própria organização ou por alguém de fora, e pode existir *on-premise* ou *off-premise*.

. **Public Cloud:** a infraestrutura é destinada ao público em geral, ou uma organização de grande dimensão, e é gerido por uma organização que vende serviços *cloud*.

. **Hybrid Cloud:** a infraestrutura é composta por duas ou mais *clouds* (privada, pública, *community*), onde permanecem entidades únicas mas estão interligadas por tecnologia standardizada e proprietária que permite portabilidade de dados e aplicações. É o exemplo de *cloud bursting* para balanceamento de carga entre clouds.

. **Community Cloud:** a infraestrutura é utilizada por diferentes organizações e é usada com o fim específico de atender as necessidades da comunidade.

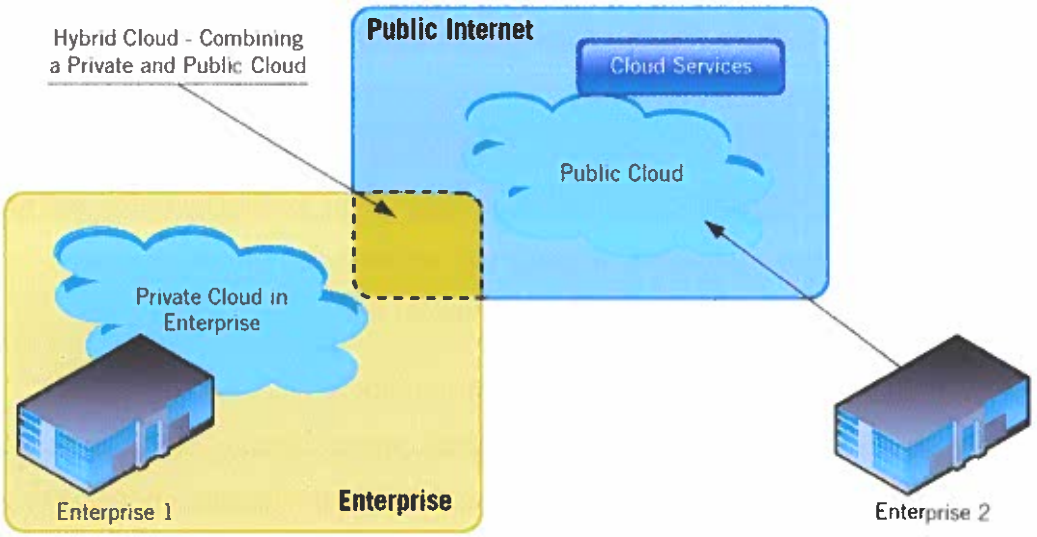


Figura 9 – Os diferentes tipos de *cloud*. Fonte: (Dialogic Corporation, 2010)

Benefícios

Cloud computing representa uma mudança significativa no modo como os recursos de TI são geridos, utilizados e consumidos. Esta mutação trouxe vários benefícios às organizações, promovendo a eficiência e a agilidade[14].

. **Redução de custos:** promove a otimização e utilização dos bens de TI, permitindo fazer mais com menos e atingir uma redução de custos significativos.

. **Flexbilidade/Escalabilidade:** a flexibilidade permite que sejam consumidos recursos extra em alturas de consumo máximo, permitindo a satisfação das necessidades do consumidor. A possibilidade de realocar recursos rapidamente permite a escalabilidade necessária do consumidor.

. **Agilidade:** é possível reduzir o tempo necessário para provisionar e implementar novas aplicações e serviços de meses a minutos.

[14] http://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/cloud-computing/white_paper_c11-617239.html

Conclusão

Tendo uma noção mais profunda das tecnologias que se vão implementar, é possível concluir que estas se apresentam como solução viável na construção da solução para o cenário da biblioteca.

Referências

1. Alfred J. Lendvai, H. S. (2007). *ADO and ADO.NET Object Model Comparisons: A Relational Perspective.*
2. Broadcom. (2010). *Conectivity in the Virtualized Datacenter: How to Ensure Next-Generation Services.*
3. EMC. (2008). *Virtualization: Current Benefits and Future Potential.*
4. F5. (2007). *Virtualization Defined - Eight Different Ways.*
5. Fabiano Muchalski, R. B. (n.d.). Virtualização.
6. Grid, T. G. (2010). *Impact of Virtualization on Data Center Physical Infrastructure.*
7. ISACA. (2010). *Virtualization: Benefits.*
8. Knowledge, G. (2011). *Choosing The Correct Hypervisor: A Critical Decision.*
9. Microsoft. (2015). *Microsoft SQL Server: Achieving mission-critical application performance.*
10. Microsoft Corporation. (2006). *The ADO.NET Entity Framework: Making the Conceptual Level Real.*
11. NIST. (2011). *The NIST Definition of Cloud.*
12. Paula Agostinho, U. L. (2009). *Virtualiazação em SAP.*
13. The Green Grid. (2009). *Using Virtualization to Improve Data Center Efficiency.*
14. Windows. (2013). *Windows Server 2012 R2: Server Virtualization Technical Overview.*

Contextualização

A virtualização tem-se tornado uma prática cada vez mais comum no mundo da tecnologia. Empresas, sejam elas grandes ou pequenas, tem optado por utilizar este tipo de sistema devido às imensas vantagens que traz.

A elaboração deste projecto dá a conhecer a aplicação prática deste sistema num cenário como uma biblioteca mostrando assim as vantagens inerentes a qualquer entidade que use um sistema informático.

Uma biblioteca apresenta-se como um local ideal para realizar o estudo. Pessoas necessitam saber o que a biblioteca tem disponível em relação ao material, para isso foi construida uma maneira que dê acesso fácil e rápido à informação disponibilizada pelos administradores da biblioteca. Os computadores que as pessoas usam, em vez de terem uma estrutura física para cada um, a virtualização proporciona o número necessário de computadores de forma virtual e o utilizador nem se apercebe de que o mesmo é virtual.

Construção do Laboratório

Criação dos servidores

Para começar, criei um servidor com o propósito de servir como base de clonagem para os outros servidores. Este servidor tem como sistema operativo o Windows Server 2012 R2 e será efetuado o "*sysprep*" [15] na máquina com efeito de replicação para os outros servidores.

A instalação do servidor inicial foi efetuada utilizando os passos de criação de uma nova máquina virtual através de uma imagem ISO. Concluida a instalação, executaram-se os passos necessários para efetuar o *sysprep:* usando o *command prompt*, executam-se os seguintes comandos: "cd %WINDIR%\system32\sysprep", seguido de "sysprep /generalize /shutdown /oobe".

A máquina encerra automaticamente, por causa da cláusula "/shutdown" inserida anteriormente na linda de comandos. Encerrada a máquina, cria-se agora um "*snapshot*", uma espécie de *backup*, que permite restaurar imagens de máquinas virtuais.

De seguida foram criados três servidores, utilizando o mecanismo de clonagem nativo no VMWare a partir da *snapshot* criada. São instalados em dois servidores a versão Windows Server 2012 R2 Datacenter, em que um dos servidores aloja o *System Center Virtual Machine Manager* (doravante denominado por SCVMM), e o outro tem o propósito de ser usado como máquina *host* para as máquinas virtuais, e contém também uma base de dados relacional – o SQL Server 2012 – e foi denominado de "SQL Datacenter", doravante chamado por SQL DC. O terceiro servidor será o controlador de domínio (DC), e usa a versão Standard do Windows Server 2012 R2.

[15] "O comando *sysprep* é uma ferramenta de preparação do sistema. Permite que seja usada uma instalação do Windows para geração de imagens." - https://technet.microsoft.com/pt-br/library/cc721940(v=ws.10).aspx

Após a conclusão, foi adicionada uma placa de rede em cada um dos servidores com efeito de poderem comunicar dentro da rede interna que se criou mais à frente, dando assim a possibilidade de gerir o domínio.

As placas adicionais têm o propósito de habilitar comunicação dentro da rede do domínio. Esta placa foi configurada em modo Host-only. A placa criada por defeito está configurada em modo NAT, para que possa haver comunicação com o exterior da rede (atualizações e afins).

Pasta partilhada: Para questões de praticalidade, foi configurada uma pasta partilhada. Esta fornece acesso a ficheiros, independentemente da máquina. A pasta será acedida utilizando as opções "Shared Folder" do VMWare.

Conversão do servidor em controlador de domínio

A conversão do servidor em controlador de domínio foi feita com o comando *PowerShell*, executado em modo administrador, "Install-ADDSForest –DomainName "lab.local". Para instalar as *Management Tools*[16] executa-se o comando "Add-WindowsFeature AD-Domain-Services –IncludeManagementTools". *

[16] Ferramentas de gestão do *Active Directory*.

Configuração das placas de rede

Para que seja possível haver comunicação dentro do domínio, é necessário criar uma rede. Neste caso, sendo um ambiente local, é possível criar redes "internas". Para tal efeito, foi criada a rede MGMNET.

Para a rede de gestão – MGMTNET – foi usada a *IP Pool* 100.1.0.0 .

Os protocolos IPv6 nas placas de rede para comunicação interna foram desativadas.

A imagem abaixo (Fig. 1) dá-nos um panorama mais perceptível do diagrama de rede.

Para além da rede MGMTNET, foi adicionada outra placa de rede à máquina "SQL DC", com o propósito de servir de *Virtual Switch* para as máquinas virtuais, com o nome de "VMNET".

Nome da MV	Função	Tipo Rede Vmware	Rede	Segmento de Rede	Mascara Subnet	IP	Gateway	DNS
DC	Domain Controller	NAT Host-Only	NAT MGMTNET	(auto) 100.1.1.0	(auto) /16 (255.255.0.0)	(auto) 100.1.1.2	(auto) 100.1.1.100	(auto) 127.0.0.1
SQL DC	Storage	NAT Host-Only Host-Only	NAT MGMTNET VMNET	(auto) 100.1.1.0 (auto)	(auto) /16 (255.255.0.0) (auto)	(auto) 100.1.1.3 (auto)	(auto) 100.1.1.100 (auto)	(auto) 100.1.1.2 (auto)
SCVMM	Virtual Machine Manager	NAT Host-Only	NAT MGMTNET	(auto) 100.1.1.0	(auto) /16 (255.255.0.0)	(auto) 100.1.1.4	(auto) 100.1.1.100	(auto) 100.1.1.2

Figura 1 – Quandro síntese das configurações das placas de rede.

Rede MGMNET

Para garantir que há comunicação entre todos os nós da rede, efetuou-se um *ping* para cada máquina conectada. No caso das máquinas com versão *Datacenter* é necessário permitir que o pedido *Echo* seja feito, por defeito esta opção não está activada. As regras que controlam tal pedido encontram-se nas regras do *Firewall*, na secção *"Inbound Rules"* e são as seguintes: "File and Printer Sharing – ICMPv4-In" e "File and Printer Sharing – ICMPv6-In", cada uma para o seu respetivo protocolo. Como temos o protocolo IPv6 desativado, é apenas necessário activar a regra para o IPv4.

Adição das máquinas ao domínio criado

As máquinas SQL Datacenter e SCVMM foram adicionadas ao domínio existente: "lab.local", utilizando a janela do "Server Manager", na secção "Local Server", nas respetivas máquinas, e inserindo o nome do domínio na caixa "domain" seguido das credenciais necessárias.

Após conclusão com êxito, é possível ver o nome do domínio na janela. Para garantir que as máquinas estão de facto inseridas no domínio, muda-se de utilizador e acede-se ao domínio usando uma conta existente dentro do mesmo. Neste momento apenas há a conta de administrador, sendo assim necessário usar essa conta.

Contas de serviço

Foram criadas contas de serviço no domínio, e de seguida adicionadas a unidades organizacionas. Para efectuar isto, na janela "Active Directory Users and Computers", escolhendo o domínio "lab.local", criaram-se duas unidades organizacionais: "System Center Accounts" – contas com permissão no *System Center*; e "SQL Server Service Accounts" para a conta do *SQL Server*.

Estas contas têm o propósito de fornecer segurança a serviços em execução no contexto do domínio.

System Center Accounts

Nesta unidade organizacional criaram-se as contas: scvmm_acc, scvmm_svc e scvmm_adm. Por conveniência usou-se a mesma palavra passe que nas contas anteriores e ativou-se a opção de nunca expirar a palavra passe.

Tendo efetuado o passo anterior, põem-se agora os utilizadores num grupo que será agora criado "SCVMMAdmins". Será um grupo do tipo "Security" com o escopo de grupo "Global". Como estamos num cenário de laboratório local, o escopo de grupo não faz diferença.

SQL Server Service Accounts

No SQL Server, apenas se criou uma conta: sql_svc com as mesmas condições acima descritas.

System Center Virtual Machine Manager

Pré-requesitos

Antes de instalar o System Center Virtual Machine Manager, é imperetrível que se instale o Windows Assessment and Deployment Kit (ADK) e o SQL Server.

Instalação

Uma das contas que consta no grupo anteriormente criado será necessária para concluir o processo de instalação, pelo que se inicia a sessão com a conta "scvmm_adm" com as respetivas credenciais no servidor SCVMM.

Durante a instalação foi usado o sistema de Distributed Key Management (DKM). Para configurar o DKM adicionou-se ao controlador de domínio, na ferramenta "ADSI Edit", permissões ao grupo "SCVMMAdmins" num contentor. O contentor foi criado dentro do domínio.

Conta Run As (SCVMM)

Foi associada a conta previamente criada, "scvmm_acc". Esta conta é criada a partir da janela do Virtual Machine Manager tem o propósito de executar tarefas no mesmo.

Esta conta permite que o administrador guarde com segurança as suas credenciais no VMM que podem ser reútilizadas para uso de diversas tarefas administrativas[17].

[17] http://www.virtualizationadmin.com/blogs/lowe/news/adding-a-run-as-account-or-profile-to-scvmm-2012-239.html

* **Nota:** Foi desativada a opção de criação automática de nomes de redes lógicas, já que serão criadas mais à frente manualmente.

Preparação do servidor para alojar as máquinas virtuais clientes

O servidor SQL Storage foi usado como host. Para preparar o servidor executaram-se os seguintes passos:

. Nas definições do servidor, na janela do VMWare, ativaram-se as opções "Virtualize Intel VT-x/EPT or AMD-V/RVI" e "Virtualize CPU performance counters";

. Adicionam-se as seguintes linhas no ficheiro VMX:

hypervisor.cpuid.v0= "FALSE"

mce.enable = "TRUE"

vhu.enable = "TRUE"

Após isso, inicia-se o servidor e instala-se a role Hyper-V com o seguinte comando *PowerShell*:

Install-WindowsFeature –Name Hyper-V –IncludeManagementTools

Seguidamente, no Hyper-V Manager cria-se um switch virtual "vSwitch", que vai estar associado à placa de rede VMNET utilizando a opção "External Network". Isto garante que haja comunicação entre as máquinas virtuais e o host, assim como com a rede exterior ao host.

Adição do host ao VMM

Na máquina host (SQL Datacenter):

Neste servidor encontram-se configuradas e armazenadas as máquinas virtuais. Para que tal aconteça é necessário preparar o servidor:

O primeiro passo passa pela ativação do iSCSI Initiator nas "Tools" da janela do "Server Manager". Segue depois a adição do grupo "SVCMMAdmins" ao grupo de utilizadores locais. E por fim instala-se a feature "Multipath I/O" apenas após terminada a adição ao VMM, a fim de evitar configurações manuais dos drivers MPIO ou DSM específico do fabricante.

No SCVMM:

Na máquina do SCVMM adiciona-se o servidor host através da "Fabric" – "Add Resources" – "Hyper-V Hosts and Clusters".

Durante a adição do servidor, definiu-se o host group, foi ativada a opção "Allow unencrypted BITS file transfers" e definiram-se as opções para os recursos do host.

*** Nota: Servidor SQL DC**

Foi adicionado um disco rígido virtual através das configurações no VMWare, e de seguida foi inicializado e alocado.

Configuração da Fabric

Para configuração da *Fabric* foram necessários executar determinadas tarefas:

. Criação de uma ligação SAN (Storage Area Network) por iSCSI a um array de armazenamento (para completar este passo foi necessário dar permissões ao grupo SCVMMAdmins na máquina SQL DC);

. Preparar uma LUN (Logical Unit);

. Criação de uma ligação entre o SCVMM e o servidor host utilizando SMB3;

. Criação de uma pasta de partilha *(file share)* no servidor host;

. Ligação do host à LUN.

Com isto, assegura-se a conectividade de rede entre o host e o sistema de armazenamento.

Alocação do armazenamento ao host

Para alocar o armazenamento ao host, é necessário alocar a *storage pool* iSCSI e a LUN ao host, assim como a partilha SMB3 do *File Server*.

Posteriormente, desativa-se a possibilidade do host utilizar armazenamento local para as máquinas virtuais: a opção "Available for placement" no disco C:.

Configurações adicionais

Network Fabric

Para configurar a *network fabric* começou-se com a criação de uma rede lógica ("VM Production Network") com uma gama de endereços (Subnet 100.1.0.0/16), seguido da definição de um IP Pool (100.1.0.1 - 100.1.10.254) e dos IP de *default gateway* e do servidor DNS (DC).

Por fim, criou-se uma "VM Network" e associou-se à rede lógica anteriormente criada.

Rede do host

Para garantir comunicação entre o host e o SCVMM, foi necessário configurar as redes associadas ao hosts, para que as máquinas virtuais a pudessem utilizar.

Nas propriedades de hardware, na placa de rede virtual (VMNET) do host, ativou-se a opção de associação à rede lógica assim como a opção "Available for placement". A segunda opção assegura que o tráfego só passa na placa de rede (VMNET).

Ultimando, na placa de rede (MGMTNET) ativou-se a opção "Used by management" permitindo assim o envio de comandos de gestão pela rede usando este *interface*.

Servidor Library do VMM

O servidor *library* é basicamente uma pasta partilhada, e é o local de armazenamento para os processos de automatização para a configuração de *templates* das máquinas virtuais.

Começou por se associar a *Library* à rede lógica de produção "VM Production Network" e atribuiu-se permissões de patilha e segurança – através das propriedades da pasta de *sharing* e *permissions* – ao host à pasta "Virtual Machine Manager Library Files".

Perfil Hardware

Este perfil define as características técnicas das máquinas virtuais, ou seja, os recuros que poderão usar, o *hardware*. Para a criação do perfil usaram-se os seguintes parâmetros:

. Compatibilidade – Hyper-V;

. Memory dinâmica com *startup* 512 MB, e no máximo 1536 MB;

. Placa de rede – Connected to VM Network, Static IP, MAC: Static.

Perfil de sistema operativo

O perfil de sistemas operativos define previamente o *software* que a máquina irá usar. Criou-se um guest OS profile, com as especificações:

. Sistema Operativo: Windows 8.1;

. Domínio: "lab.local";

Criação do VM Template

O VM Template – *Virtual Machine Template* – permite a criação de máquinas virtuais e as suas respetivas configurações sendo tudo guardado sob a forma de *template* (modelo), evitando assim que seja necessário repetir os passos de criação sempre que se quiser criar uma máquina nova.

Para que seja possível criar o *template* é primeiro necessário criar uma máquina virtual genérica, e só depois é possível criar o *template*. A máquina virtual será transformada num *template*.

Máquina virtual genérica

As máquinas virtuais que foram e serão criadas, seguiram os perfis especificados nesta máquina genérica. É possível alterar esses valores, mas, com isto, fica definido um padrão que será seguid aquando a criação de máquinas virtuais.

O processo de criação da máquina virtual foi feita através da janela do VMM, escolhendo o comando "Create Virtual Machine", e foi feito a partir de um disco virtual vazio, selecionando também o perfil de hardware anteriormente criado. Esta máquina ficou alojada no servidor host (SQL DC) numa pasta específica: "E:\VM's".

Na *Library* do servidor SCVMM foi adicionada uma pasta em "C:\ProgramData\Virtual Machine Manager Library Files" com o nome "ISO's". Esta pasta contém o sistema operativo que será instalado nas máquinas cliente, o Windows 8.1.

Nas propriedades da máquina virtual genérica, na secção de *Hardware*, foi adicionado o ISO do sistema operativo no *DVD drive*, para que o mesmo possa ser instalado.

Após a criação estar concluida, iniciou-se a máquina virtual para dar seguimento à instalação do sistema operativo. Para isso acedeu-se à máquina virtual usando o comando "Connect via Console", e procedeu-se com a instalação normalmente.

Template

O processo de criação do *template* implica que a máquina virtual será apagada, sendo ela convertida posteriormente. Na *library* do SCVMM criou-se uma pasta para armazenar o template que será criado.

Durante a criação do template definiu-se o local (acima mencionado) para armazenamento aquando a criação de uma máquina virtual.

Tendo feito tudo isto, as máquinas virtuais ficaram prontas a serem criadas. Estas seguem o modelo padrão do "template" e podem ser facilmente criadas e monitorizadas pelo VMM do Windows.

Aplicação

A aplicação contém um interface amigável para o utilizador, para que possa interagir de forma fácil com a informação proveniente da base de dados. A conexão é feita por IP. A "connection string" utiliza o IP da máquina host (SQL DC), onde está instalado o servidor SQL.

Para possibilitar a conexão por IP foi necessário:

. Habilitar a opção de permitir conexões por TCP/IP na máquina SQL, e garantir a utilização da porta estática 1433.

. Na *Firewall*, criar novas regras "Inbound" a possiblitar o uso da porta estática 1433, a porta UDP 1434, o serviço MSSQLSERVER e a aplicação SQLSERVER.EXE.

Nas páginas seguinte estão imagens de cada janela, assim como o código de cada janela (*code behind*), de forma comentada, para que se tenha melhor percepção do funcionamento da aplicação.

Na pasta da aplicação foram colocados ficheiros (iTextSharp) para que seja possível exportar a informação para formato PDF.

* **Nota:** a aplicação foi desenvolida para utilizar o horário e data em formato português e sendo os sistemas operativos Windows em inglês, foi necessário alterar essa definição em cada máquina que utilize a aplicação.

Janela inicial:

Código:

```
public class FRM_Splash : System.Windows.Forms.Form
        {
                DBLib db;
                #region " Windows Form Designer generated code "

                public FRM_Splash() : base()
                {
                        Load += Splash_Load;

                        //This call is required by the Windows Form Designer.
                        InitializeComponent();
                        //Add any initialization after the InitializeComponent() call

                }

                //Form overrides dispose to clean up the component list.
                protected override void Dispose(bool disposing)
                {
                        if (disposing) {
                                if ((components != null)) {
                                        components.Dispose();
                                }
                        }
                        base.Dispose(disposing);
                }
```

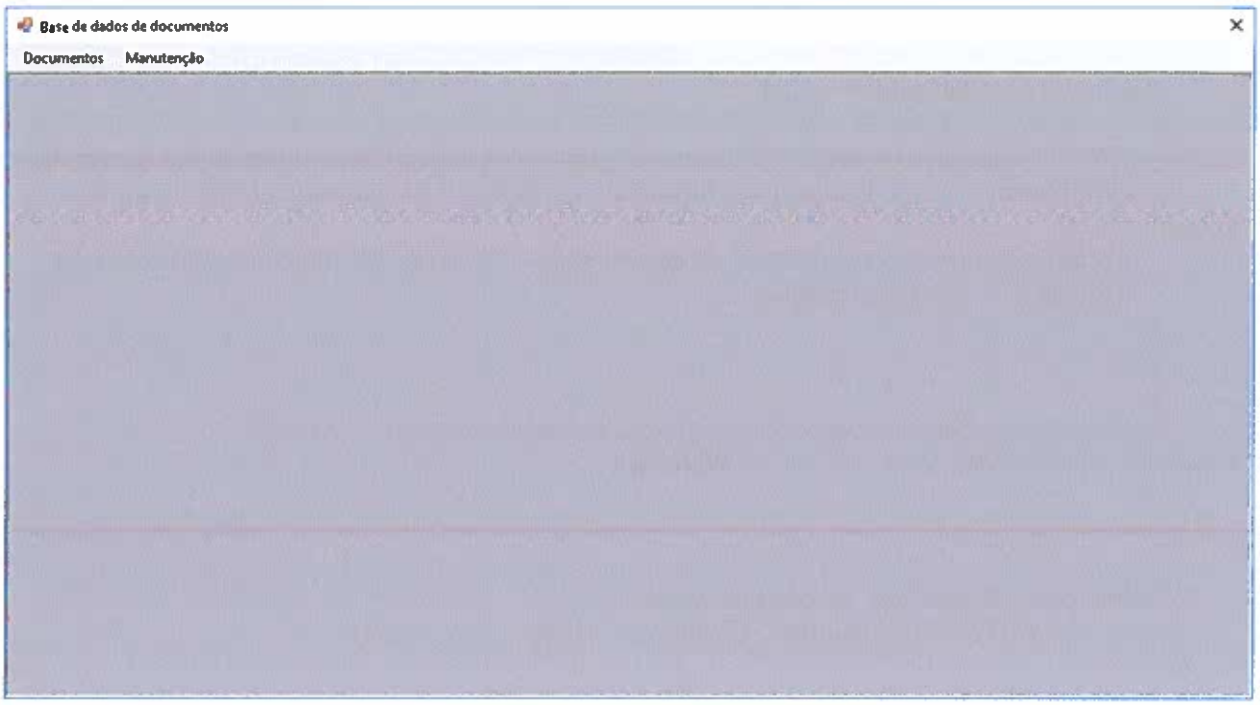
```
        //Required by the Windows Form Designer

        private System.ComponentModel.IContainer components;
        //NOTE: The following procedure is required by the Windows Form Designer
        //It can be modified using the Windows Form Designer.
        //Do not modify it using the code editor.
        internal System.Windows.Forms.Label Label1;
        internal System.Windows.Forms.Label Label3;
        internal System.Windows.Forms.Label Label4;
        private System.Windows.Forms.TextBox withEventsField_usr;

        internal System.Windows.Forms.PictureBox PictureBox1;
    private TextBox usr;
    private TextBox ps;
    private Button SUBMETER;
        private System.Windows.Forms.TextBox withEventsField_ps;

        [System.Diagnostics.DebuggerStepThrough()]
        private void InitializeComponent()
        {
        this.Label1 = new System.Windows.Forms.Label();
        this.Label3 = new System.Windows.Forms.Label();
        this.Label4 = new System.Windows.Forms.Label();
        this.PictureBox1 = new System.Windows.Forms.PictureBox();
        this.usr = new System.Windows.Forms.TextBox();
        this.ps = new System.Windows.Forms.TextBox();
        this.SUBMETER = new System.Windows.Forms.Button();
        ((System.ComponentModel.ISupportInitialize)(this.PictureBox1)).BeginInit();
        this.SuspendLayout();
        //
        // Label1
        //
        this.Label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 26.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.Label1.Location = new System.Drawing.Point(232, 16);
        this.Label1.Name = "Label1";
        this.Label1.Size = new System.Drawing.Size(376, 100);
        this.Label1.TabIndex = 100;
        this.Label1.Text = "Base de dados Documental";
        this.Label1.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
        //
        // Label3
        //
        this.Label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.Label3.Location = new System.Drawing.Point(336, 155);
        this.Label3.Name = "Label3";
        this.Label3.Size = new System.Drawing.Size(112, 16);
        this.Label3.TabIndex = 5;
        this.Label3.Text = "Username:";
        //
```

```
            // Label4
            //
            this.Label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.Label4.Location = new System.Drawing.Point(336, 187);
            this.Label4.Name = "Label4";
            this.Label4.Size = new System.Drawing.Size(112, 24);
            this.Label4.TabIndex = 6;
            this.Label4.Text = "Password:";
            //
            // PictureBox1
            //
            this.PictureBox1.Image = global::DBDoc.Properties.Resources.download;
            this.PictureBox1.Location = new System.Drawing.Point(12, 33);
            this.PictureBox1.Margin = new System.Windows.Forms.Padding(0);
            this.PictureBox1.Name = "PictureBox1";
            this.PictureBox1.Size = new System.Drawing.Size(260, 192);
            this.PictureBox1.TabIndex = 0;
            this.PictureBox1.TabStop = false;
            //
            // usr
            //
            this.usr.Location = new System.Drawing.Point(408, 155);
            this.usr.Name = "usr";
            this.usr.Size = new System.Drawing.Size(100, 20);
            this.usr.TabIndex = 101;
            //this.usr.KeyDown += new System.EventHandler(this.usr_KeyDown);
            //
            // ps
            //
            this.ps.Location = new System.Drawing.Point(408, 184);
            this.ps.Name = "ps";
            this.ps.PasswordChar = '*';
            this.ps.Size = new System.Drawing.Size(100, 20);
            this.ps.TabIndex = 102;
            //
            // SUBMETER
            //
            this.SUBMETER.Location = new System.Drawing.Point(408, 215);
            this.SUBMETER.Name = "SUBMETER";
            this.SUBMETER.Size = new System.Drawing.Size(75, 23);
            this.SUBMETER.TabIndex = 103;
            this.SUBMETER.Text = "Submeter";
            this.SUBMETER.UseVisualStyleBackColor = true;
            this.SUBMETER.Click += new System.EventHandler(this.SUBMETER_Click);
            //
            // FRM_Splash
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.BackColor = System.Drawing.Color.White;
            this.ClientSize = new System.Drawing.Size(626, 274);
```

```
            this.Controls.Add(this.SUBMETER);
            this.Controls.Add(this.ps);
            this.Controls.Add(this.usr);
            this.Controls.Add(this.PictureBox1);
            this.Controls.Add(this.Label1);
            this.Controls.Add(this.Label4);
            this.Controls.Add(this.Label3);
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "FRM_Splash";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "DBDoc";
            ((System.ComponentModel.ISupportInitialize)(this.PictureBox1)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

                    }

                    #endregion

        // Evento para o botão submeter.
                    private void SUBMETER_Click(System.Object sender, System.EventArgs e)
                    {
                            ligar();    // Invoca o método ligar.
                    }

        // Estabelece a conexão com a base de dados.
                    private void ligar()
                    {

                            string query = null;
                            DataSet ds = new DataSet();
                            MDI_DBDoc f = null;

            // Verifica se há ligação à base de dados.
            // Se sim, então, verifica-se na tabela de utilizadores se o login e password existem.
            // Se existirem - o dataset tem registos - abre o formulário principal da aplicação.
            if (this.db.isConnected)
            {
                                    query = "select * from utilizador where username = '" + usr.Text + "' and
password = '" + ps.Text + "'";   // Query de verificação de login e password.

                                    ds = this.db.Read(query);

                                    if (ds.Tables[0].Rows.Count > 0)
                {
                                            f = new
MDI_DBDoc(Convert.ToInt32(ds.Tables[0].Rows[0]["is_admin"]));   // Cria o formulário principal da
aplicação, enviado a variável para ver se o utilizador é administrador ou não.
                    this.Hide();    // Esconde o formulário.
                                            f.ShowDialog(); // Abre o formulário principal.
```

```
                try
                {
                    this.Show();
                }
                catch (Exception e){}
                                }
            else
            {
                MessageBox.Show("Utilizador ou password inválidos.", "Erro",MessageBoxButtons.OK,
MessageBoxIcon.Error);
                                        usr.Focus();   // Coloca o cursos no campo do utilizador.
                                        return;
                                }
                        }
                }

    // Função que é carregada automaticamente quando o formulário é aberto, criando a ligação à base de
dados e coloca o cursor no campo para o utilizador preencher o login.
    private void Splash_Load(System.Object sender, System.EventArgs e)
                {
                        this.db = new DBLib();
                        usr.Focus();
                }

    // Função que é executada quando é carregada uma tecla no campo "Username".
    private void usr_KeyDown(object sender, System.Windows.Forms.KeyEventArgs e)
                {
                        if (e.KeyCode.Equals(Keys.Enter))  // Se carregou no enter.
        ps.Focus(); // Então coloca o cursor para preencher a password.
    }

    // Função que é executada quando é carregada uma tecla no campo "Password".
    private void ps_KeyDown(object sender, System.Windows.Forms.KeyEventArgs e)
                {
                        if (e.KeyValue.Equals(Keys.Enter))
                                ligar();   // Invoca o método ligar.
                }

        }
}
```

Janela Principal:

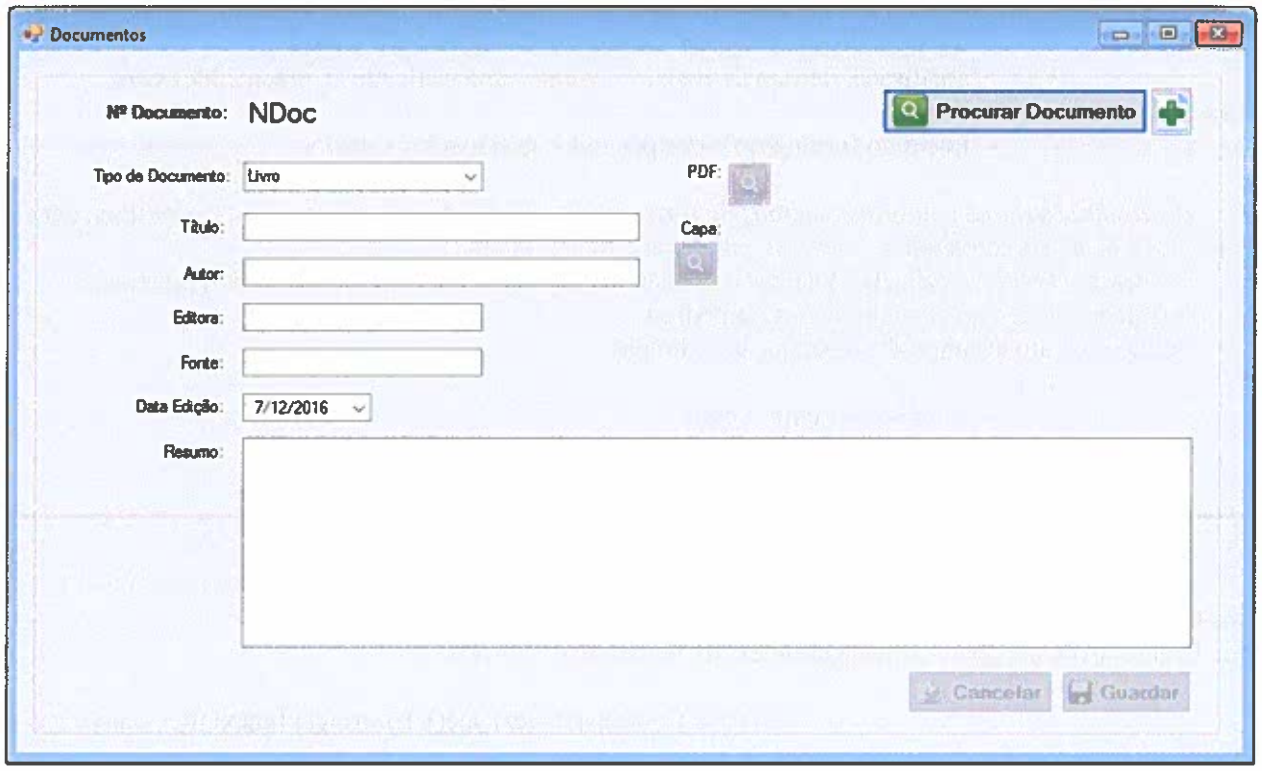


Código:

```
// Esta classe corresponde ao formulário principal da aplicação.
// https://msdn.microsoft.com/en-us/library/xyhh2e7e(v=vs.110).aspx

// Trata-se de um formulário MDI (Multi-Document Interface).
// Este tipo de formulário cria o conceito de um formulário "pai" onde sobre ele podem existir vários formulários "filho" (através da propriedata MDIParent desses mesmos filhos).
// Permite também, neste caso, ter um menu onde podemos aceder a todas as funionalidades da aplicação.
// No fundo, o MDI form trata-se do interface base da aplicação onde todas as funcionalidades(sub-formulários) assentam.

namespace DBDoc
{
        public partial class MDI_DBDoc
        {
                int isAdmin;    // Variável de controlo que verifica se o utilizador é administrador, tendo acesso a mais funcionalidades que um utilizador normal.

    // Início da classe.
    public MDI_DBDoc(int adm)
                {
                        this.isAdmin = adm;
                        InitializeComponent();
                }
```

```
        // Função para o botão "Documentos" na barra de opções.
        private void documentosToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if ((this.ActiveMdiChild == null))
            {
                FRM_Documento d = new FRM_Documento(); // Cria o formulário de gestão dos documentos.
                d.MdiParent = this; // Define que o formulário pai do FRM_Documento. O MDI Form fica
agregado a este.
                d.StartPosition = FormStartPosition.CenterScreen;  // Coloca o formuláio no centro do ecrã.
                d.Show();  // Abre o formulário.
            }
            else
            {
                MessageBox.Show("Tem de fechar o formulário ativo primeiro.", "Alerta",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }

        // Evento para o botão "sair" na barra de opções.
        private void sairToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.Exit();    // Fecha a aplicação.
        }

        // Evento para o botão "Tipo de documento". na barra de opções.
        private void tiposDeDocumentoToolStripMenuItem_Click_1(object sender, EventArgs e)
        {
            // Verifica a permissão de admin, cuja variável foi defeinida aquando da criação do formulário a
partir do formulário de splash verificando o campo da tabela de utilizadores isAdmin.
            if (this.isAdmin == 0)
            {
                MessageBox.Show("Não tem permissões para aceder a esta funcionalidade", "Alerta",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }

            // (Duplicado de cima)
            if ((this.ActiveMdiChild == null))
            {
                FRM_TipoDoc d = new FRM_TipoDoc();
                d.MdiParent = this;
                d.StartPosition = FormStartPosition.CenterScreen;
                d.Show();
            }
            else
            {
                MessageBox.Show("Tem de fechar o formulário ativo primeiro.", "Alerta",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        } } }
```

Janela de documentos:



Código:

```
public partial class FRM_Documento
        {
                DBLib db;  // Controlador de acesso à base de dados

                bool changeCapa;
                byte[] pdfContents;
                int selDoc;

    // Evento de carregamento do formulário. É executado automaticamente quando o formulário é carregado (iniciado).
    // Aqui são inicializadas as variáveis (DBLib), carregados os valores das comboboxs e desativados os campos do formulário.
    private void FRM_Documento_Load(object sender, EventArgs e)
                {
                        this.db = new DBLib();
                        this.loadCombo(cmbTipoDoc, "select id_tipodoc, ds_tipodoc from tipodoc");  // Carrega a combobox.

                        btnCancelar.Text = "Cancelar";  // Botão com o texto cancelar.
                        this.ativaCampos(false, false);
```

```
    }

    // Esta função permite ativar e desativar os campos do formulário consoante estamos em modo de:
                //    - "inserção de novo registo": campos todos ativados e vazios
                //    - "atualização/leitura de registo": campos todos ativados e preenchidos com o
respetivo valor
                //    - "inicio do formulário": campos todos desativados e vazios

    // Percorre todos os controlos (campos do formulário), colocando os controlos do tipo textbox, data e
combobox no estado pretendido - descrito em cima, e limpo ou não.
    // Coloca o botão de cancelar, guardar e imagem de capa com o status (ativo ou não) pretendido.
    // Coloca o focus no primeiro campo (tipodoc).
    public void ativaCampos(bool status, bool limpa)
                {
                        ComboBox cmb = null;
                        DateTimePicker dt = null;

                        for (int I = this.Controls.Count - 1; I >= 0; I += -1)
      {
                                if ((this.Controls[I].GetType().ToString().IndexOf("TextBox") > -1 |
this.Controls[I].GetType().ToString().IndexOf("DateTimePicker") > -1 |
this.Controls[I].GetType().ToString().IndexOf("ComboBox") > -1))
          {
                                        if (this.Controls[I].GetType().ToString().IndexOf("ComboBox")
> -1)
            {
                                                cmb = (ComboBox)this.Controls[I];
                                                cmb.Enabled = status;
                                                if (limpa)
              {
                                                        cmb.SelectedIndex = -1;
                                                }
                                        }
            else if (this.Controls[I].GetType().ToString().IndexOf("DateTimePicker") > -1)
            {
                                                dt = (DateTimePicker)this.Controls[I];
                                                dt.Enabled = status;

                                                if (limpa)
                                                        dt.Text = DateTime.Now.Date.ToString();

                                                dt.Text = DateTime.Now.Date.ToString();
                                        }
            else
            {
                                                this.Controls[I].Enabled = status;

                                                if (limpa)
                                                        this.Controls[I].Text = "";

                                        }
```

```
                    }
                }

                btnCancelar.Enabled = status;
                btnGuardar.Enabled = status;
                btnLoadCapa.Enabled = status;
        btnExportar.Enabled = status;

                cmbTipoDoc.Focus();
                this.changeCapa = false;
            }

    // Carrega a respetiva combobox com a query de leitura (select) específica.
    public void loadCombo(ComboBox combo, string query)
            {
                DataSet ds = null;

        ds = db.Read(query);    // Executa o comando select e retorna o dataset.

        // Se o dataSet tem registos definine a origem de dados da combobox como sendo a respetiva
tabela do dataset (ds.Tables[0]).
        // Define-se o DisplayMember (coluna/campo da tabela cujos valores/opções aparecem na
combobox ao utilizador) e o ValueMember (coluna/campo da tabela associado ao DisplayMember).

        // ex: query = select id_tipodoc, ds_tipodoc from tipodoc
        //     A combobox surge com as opções de ds_tipodoc visíveis ao utilizador(ex: revista, livro,
jornal, pdf)
        //     sendo que "internamente" essas opções assumem o valor correspondente por exemplo de
1,2,3,4...

        if ((ds.Tables[0].Rows.Count > 0))
        {
                    combo.DataSource = ds.Tables[0];
                    combo.ValueMember = ds.Tables[0].Columns[0].ColumnName;
                    combo.DisplayMember = ds.Tables[0].Columns[1].ColumnName;
                }

            }

    // Procura o documento na base de dados baseado no ID escolhido pelo utilizador.
            private void procurarDoc(int id)
            {
                DataSet ds = null;
                ArrayList whereValues = new ArrayList();
                SqlCommand cmd = null;

                whereValues.Add(new SQLValue("@id", id));   // Prepara a variável
whereValues do comando select (o que vem a seguir ao where).
        ds = this.db.Read("select * from documento where id_documento = @id", whereValues);     // No
dataset, lê a linha correspondente ao ID pedido.
```

```
        // Preenche cada campo do formulário com o respetivo campo do dataSet.
        lblNDoc.Text = ds.Tables[0].Rows[0]["id_documento"].ToString();
        cmbTipoDoc.SelectedValue = ds.Tables[0].Rows[0]["id_tipodoc"].ToString();
        txtTitulo.Text = ds.Tables[0].Rows[0]["titulo"].ToString();
                        txtAutor.Text = ds.Tables[0].Rows[0]["autor"].ToString();
        dtEdicao.Text = ds.Tables[0].Rows[0]["dt_edicao"].ToString();
        txtEditora.Text = ds.Tables[0].Rows[0]["editora"].ToString();
        txtFonte.Text = ds.Tables[0].Rows[0]["fonte"].ToString();
        txtResumo.Text = ds.Tables[0].Rows[0]["resumo"].ToString();

        // Se o campo de resumo estiver vazio, não permite a exportação do documento - o botão fica
invisível.
                        if (txtResumo.Text == "")
                                btnExportar.Visible = false;
                        else
                                btnExportar.Visible = true;

                        // Leitura da da capa do livro.
                        cmd = new SqlCommand("select capa from documento where id_documento = '"
+ id.ToString() + "'", this.db.MyConn);   // Seleciona a capa através do ID do documento, e abre uma
conexão à base de dados.
        byte[] imageData = null;

                        try
        {
                                imageData = (byte[])cmd.ExecuteScalar();   // Lê o conjunto de bytes
guardados no campo da capa do livro no registo da base de dados do respetivo documento.
        }
        catch (Exception ex) {}

        // Se o valor imageData não for nulo, carrega a imagem a partir da base de dados.
                        if ((imageData != null))
        {
          // Coloca os bytes lidos correspondente à imagem num memorystream, preenchendo o mesmo.
          // Esse stream é usado para definir a imagem do controlo "picturebox" (imgCapa).
          using (MemoryStream ms = new MemoryStream(imageData, 0, imageData.Length))
          {
            ms.Write(imageData, 0, imageData.Length);
            imgCapa.Image = System.Drawing.Image.FromStream(ms, true);
            this.changeCapa = false;
          }
                        }

                        this.selDoc = (int)ds.Tables[0].Rows[0]["id_documento"];   // Refere-se ao
documento atualmente selecionado.
    }


    // Operações executadas quando o evento click no botão de "ADICIONAR" documento é acionado.
    private void btnAdd_Click(object sender, EventArgs e)
                {
```

```
                this.ativaCampos(true, true);   // Ativa os campos do formulário.
        btnProcurar.Enabled = false;    // Retira a opção de procurar documento.
                btnCancelar.Text = "Cancelar";  // Botão com o texto "Cancelar".
                this.selDoc = 0;    // Documento atualmente selecionado é "0", ou seja, nenhum
está selecionado.
        cmbTipoDoc.Focus(); // Coloca o focus na combobox cmbTipoDoc.
    }

    // Inicializador da classe FRM_Documento.
    public FRM_Documento()
            {
                Load += FRM_Documento_Load;
                InitializeComponent();
            }

    // Evento do botão cancelar.
    // Se o seldoc = 0 então estamos em modo "inserir novo documento". O "cancelar" cancela a
operação que o utilizador está a efetuar.
    // Senão, fica ativo o modo de "atualização de registo" e aqui o cancelar vai remover o documento
atual.
    private void btnCancelar_Click(object sender, EventArgs e)
    {
      if (this.selDoc == 0)
      {
        this.ativaCampos(false, true);
        btnAdd.Enabled = true;
        btnProcurar.Enabled = true;
      }
      else
      {
        if ((MessageBox.Show("Tem a certeza?", "", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes))
        {
          int ret = 0;

          ret = this.db.Delete("DELETE FROM documento WHERE id_documento = " +
this.selDoc.ToString(), null, true);

          // Se surgir um erro, não efetua a operação.
          if (ret < 0)
          {
            return;
          }

          MessageBox.Show("Dados removidos com sucesso", "", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);   // Caixa de texto

          // Colocamos o formulário em modo de "inserção" com os campos todos ativos, mas todos
vazios.
          this.ativaCampos(false, true);
          btnAdd.Enabled = true;
```

```
                btnProcurar.Enabled = true;
                btnCancelar.Text = "Cancelar";
                this.selDoc = 0;
            }
        }
    }

    // Evento do botão guardar. Guarda os valores na base de dados após o click do botão.
    private void btnGuardar_Click(object sender, EventArgs e)
    {
        byte[] data = null;
        ArrayList insValues = new ArrayList();
        ArrayList upValues = new ArrayList();

        int res = 0;
        string more_campos = null;
        string more_values = null;
        string str = null;
        ComboBox cmb;
        DateTime dt;

        // Percorre todos os campos do formulário do GroupBox1 e verifica se algum campo está por
preencher. Se sim, mostra uma mensagem de erro e coloca o "focus" nesse controlo.
        for (int I = GroupBox1.Controls.Count - 1; I >= 0; I += -1)
        {
            if (GroupBox1.Controls[I].Name.IndexOf("txtEditora") == -1 &&
GroupBox1.Controls[I].Name.IndexOf("txtFonte") == -1 &&
(GroupBox1.Controls[I].GetType().ToString().IndexOf("TextBox") > -1 ||
GroupBox1.Controls[I].GetType().ToString().IndexOf("DateTimePicker") > -1 ||
GroupBox1.Controls[I].GetType().ToString().IndexOf("ComboBox") > -1))
            {
                if (GroupBox1.Controls[I].GetType().ToString().IndexOf("ComboBox") > -1)
                {
                    cmb = (ComboBox)GroupBox1.Controls[I];
                    str = cmb.SelectedIndex.ToString();

                    if (str == "-1")
                        str = "";
                }
                else
                    str = GroupBox1.Controls[I].Text;


                if ((string.IsNullOrEmpty(str)))
                {
                    MessageBox.Show("Por favor preencha o campo " + GroupBox1.Controls[I].Name, "",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    GroupBox1.Controls[I].Focus();
                    return;
                }
            }
```

```
        }

        // Preparação das variáveis insValues com os campos e respetivos valores para o comando SQL.
        insValues.Add(new SQLValue("@idtipodoc", cmbTipoDoc.SelectedValue));
        insValues.Add(new SQLValue("@titulo", txtTitulo.Text));
        insValues.Add(new SQLValue("@autor", txtAutor.Text));
        insValues.Add(new SQLValue("@editora", txtEditora.Text));
        insValues.Add(new SQLValue("@fonte", txtFonte.Text));

        //Transforma a data em formato dd-MM-yyyy (da aplicação) para yyyy-MM-dd (da base de
dados).
        dt = DateTime.ParseExact(dtEdicao.Text.ToString(), "dd-MM-yyyy",
CultureInfo.InvariantCulture);
        insValues.Add(new SQLValue("@dt_edicao", dt.ToString("yyyy-MM-dd")));

        insValues.Add(new SQLValue("@resumo", txtResumo.Text));
        more_campos = "";
        more_values = "";

        // Se existe uma capa associada ao novo documento então prepara o respetivo parâmetro com o
conteúdo do ficheiro da capa para inserir/atualizar na base de dados.
        if (this.changeCapa)
        {
            MemoryStream ms = new MemoryStream();
            imgCapa.Image.Save(ms, imgCapa.Image.RawFormat);
            data = ms.GetBuffer();

            if (this.selDoc > 0)
                more_campos += ", capa = @capa";
            else
            {
                more_campos += ", capa";
                more_values += ", @capa";
            }

            insValues.Add(new SQLValue("@capa", data));
        }

        // Se estamos em modo "atualização/leitura de registo", faz o update, caso contrário faz o insert.
        if (this.selDoc > 0)
        {
            res = this.db.Update("update documento set id_tipodoc = @idtipodoc, titulo = @titulo, autor =
@autor, editora = @editora, fonte = @fonte, dt_edicao = @dt_edicao, resumo = @resumo"
                + more_campos + " where id_documento = " + this.selDoc.ToString(), insValues, null, true);

            if ((res != -1))
                MessageBox.Show("Dados gravados com sucesso");

            btnProcurar.Enabled = true;
        }
        else
```

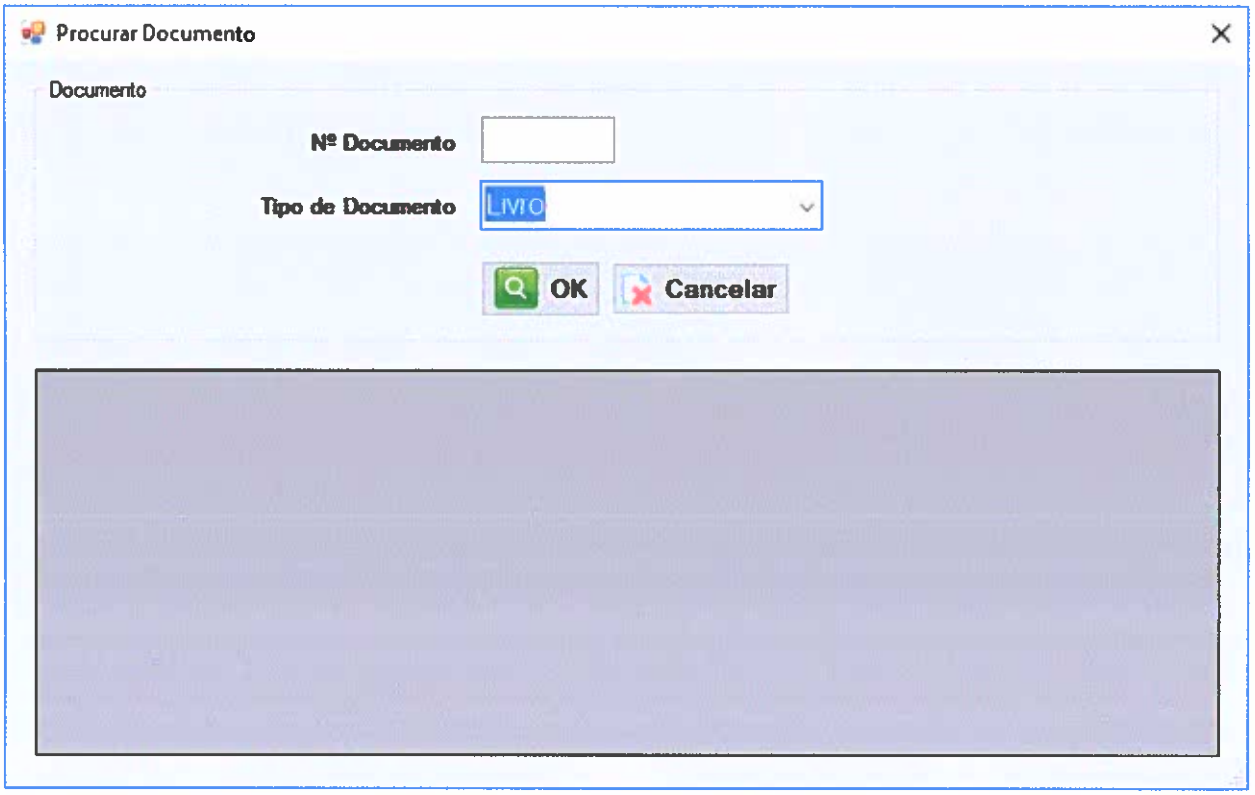
```
        {
            res = this.db.Insert("insert into documento (id_tipodoc, titulo, autor, editora, fonte, dt_edicao,
resumo" +
                more_campos + ") values (@idtipodoc, @titulo, @autor, @editora, @fonte, @dt_edicao,
@resumo" + more_values + ")", insValues, true); // Insere na tabela os valores nas colunas
correspondentes.

            if ((res > 0))
            {
                MessageBox.Show("Dados gravados com sucesso", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }

            btnProcurar.Enabled = true;
            this.selDoc = res;
            lblNDoc.Text = this.selDoc.ToString();
        }

    }

    // Evento para o botão Procurar.
    // Abre um formulário FRM_procurarDoc para procurar um documento na base de dados
(ShowDialog).
    // Se retornar "OK", carrega o formulário com os dados do documento selecionado (ProcurarDoc).
    // Pode não retornar "OK" se o utilizador fechar o formulário sem clicar "OK", e operação não é
feita.
    // Ativa os campos, sem os limpar, pois estão preenchidos, com os dados do documento selecionado.
    private void btnProcurar_Click(System.Object sender, System.EventArgs e)
    {
        FRM_ProcurarDoc d = new FRM_ProcurarDoc();
        int res = 0;

        res = Convert.ToInt32(d.ShowDialog());
        if (res == Convert.ToInt32(DialogResult.OK))
        {
            this.procurarDoc(d.DocSel);
            this.ativaCampos(true, false);
            btnCancelar.Text = "Remover";
        }
    }

    // Permite fazer o upload da capa.
    // Atribui à variável uma caixa de diálogo para poder abrir o ficheiro escolhendo o último diretório
aberto.
    // Filtra as extensões dos ficheiros.
    // Verifica se existe um ficheiro com o mesmo nome e verifica se o caminho do ficheiro existe.
    // Se o formulário voltar com o resultado "OK", mostra a imagem (a capa do livro).
    private void btnLoadCapa_Click(object sender, EventArgs e)
    {
        OpenFileDialog openFileDialog1 = new OpenFileDialog();
```

```
            openFileDialog1.RestoreDirectory = true;
            openFileDialog1.Filter = "PNG files (*.png)|*.png|JPG (*.jpg)|*.jpg";
            openFileDialog1.CheckFileExists = true;
            openFileDialog1.CheckPathExists = true;
            DialogResult dr = openFileDialog1.ShowDialog();

            if (dr == DialogResult.OK)
            {
                imgCapa.Image = System.Drawing.Image.FromFile(openFileDialog1.FileName);
                this.changeCapa = true;
            }
        }

        // Evento para o botão Exportar. Permite a exportação dos dados para um ficheiro em formato PDF.
        // Primeiro escolhe do local para gravar o ficheiro abrindo o último diretório aberto antes de fechar.
        // Grava o ficheiro com a extensão PDF.
        // Mostra uma mensagem de confirmação de overwrite do ficheiro caso ele já exista.
        private void btnExportar_Click(object sender, EventArgs e)
        {
            SaveFileDialog saveFileDialog1 = new SaveFileDialog();
            saveFileDialog1.RestoreDirectory = true;
            saveFileDialog1.Filter = "PDF files (*.pdf)|*.pdf";
            saveFileDialog1.OverwritePrompt = true;

            DialogResult dr = saveFileDialog1.ShowDialog();

            if (!string.IsNullOrEmpty(saveFileDialog1.FileName))
            {
                // Para a exportação do ficheiro, usando a library iTextSHarp, usa-se código em HTML.
                // É transformado para HTML, pois o DLL que o transforma em PDF assim o exige.
                // Cria-se um string reader com o conteúdo dos campos.
                String html;
                String foto;

                html = "<html><head>< style >img {margin - left: 20 %;}</style></ head >< body>";
                html += "<b>Título</b>: " + txtTitulo.Text + "<br /><br />";
                html += "<b>Autor</b>: " + txtAutor.Text + "<br /><br />";
                html += "<b>Editora</b>: " + txtEditora.Text + "<br /><br />";
                html += "<b>Data</b>: " + dtEdicao.Text + "<br /><br />";
                html += "<b>Resumo</b>: " + txtResumo.Text + "<br /><br />";


                // Se a PictureBox tiver uma imagem então vamos à base de dado obter o conteúdo e guardar
para um ficheiro temporário para gerar o PDF.
                // Como o PDF é gerado a partir de HTML, as imagens são carrergadas a partir da tag HTML
<img>, cujo source (src) é um ficheiro existente no disco.
                // Daí, haver necessidade de se criar um ficheiro com a imagem para ser carregado pelo HTML
a partir do qual será gerado o PDF.
                if (imgCapa.Image != null)
                {
```

```
            foto = Path.GetDirectoryName(Application.ExecutablePath) + "/foto_" +
DateTime.Now.Ticks;

            SqlCommand cmd = new SqlCommand("select capa from documento where id_documento =
'" + selDoc.ToString() + "'", this.db.MyConn);
            byte[] imageData = null;

            try
            {
               imageData = (byte[])cmd.ExecuteScalar();
            }
            catch (Exception ex){}

            if ((imageData != null))
            {
               MemoryStream ms = new MemoryStream(imageData, 0, imageData.Length);
               ms.Write(imageData, 0, imageData.Length);
               System.Drawing.Image img = System.Drawing.Image.FromStream(ms);
               img.Save(foto, System.Drawing.Imaging.ImageFormat.Jpeg); // Grava a imagem em
formato JPEG.
            }


            html += "<img src=\"" + foto + "\" width=\"300px\" height=\"400px\" />";
         }
         else

         html += "";
         html += "</body></html>";



         StringReader reader = new StringReader(html);

         Document document = new Document(PageSize.A4, 30, 30, 30, 30); // Inicializa-se o
documento.

         try
         {
            PdfWriter writer = PdfWriter.GetInstance(document, new
FileStream(saveFileDialog1.FileName, FileMode.Create)); // Inicialização do documento PDF.
            document.Open();

            // Faz o parsing do texto em HTML (reader) e coloca-o em "document" e escreve-o através
da variável "writer".
            XMLWorkerHelper.GetInstance().ParseXHtml(writer, document, reader);
            MessageBox.Show("Conteudo exportado para PDF com sucesso!", "",
MessageBoxButtons.OK, MessageBoxIcon.Information);
         }
         catch (Exception ex)
         {
```

```
                MessageBox.Show("Erro ao exportar para PDF. Erro: " + ex.Message, "",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

            // Fecha a janela do documento.
            try
            {
                document.Close();
                reader.Close();
                reader.Dispose();
            }
            catch (Exception ex) {}
        }
    }
  }
}
```

Janela de procura de documentos:



Código:

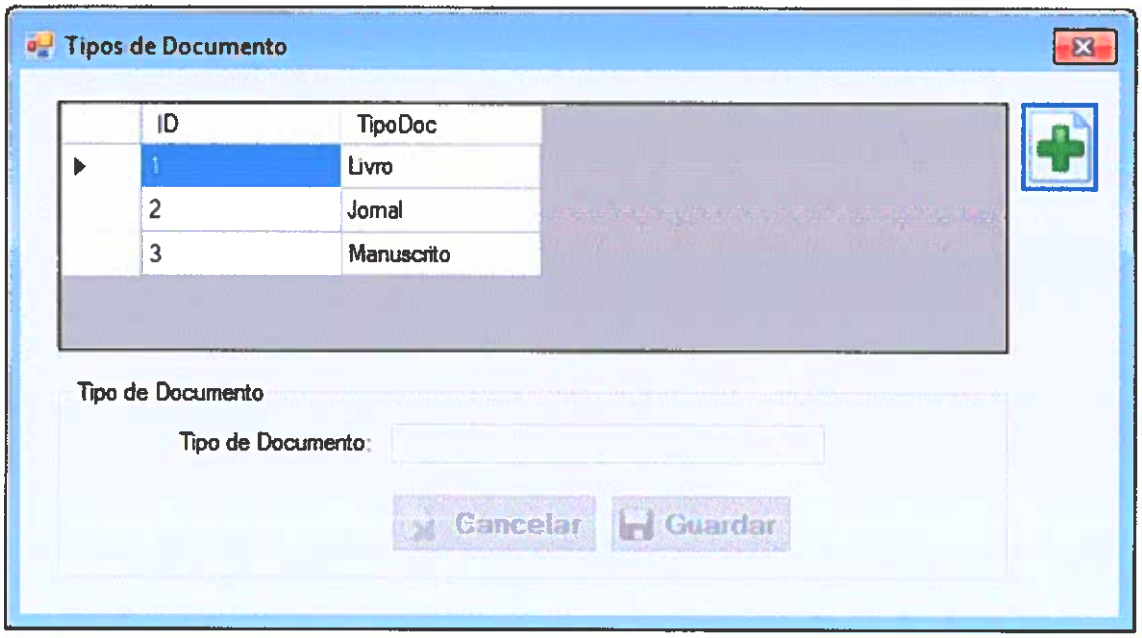
```
public partial class FRM_ProcurarDoc
        {
                DBLib db;
                bool ativo;
                public int DocSel;

    // Carrega o formulário.
                private void FRM_ProcurarDoc_Load(System.Object sender, System.EventArgs e)
                {
                        DataSet ds = null;
                        ativo = false;  // Para não executar o evento SelectionChanged.

        this.db = new DBLib();
                        ds = this.db.Read("select id_tipodoc, ds_tipodoc from tipodoc", null);  // Lê na
base de dados os tipos de documentos existentes, escolhendo as colunas "id_tipocdoc e ds_tipodoc".

        // Preenche a combobox com os valores existentes no dataSet.
        if (ds.Tables[0].Rows.Count > 0)
        {
                                cmbTipoDoc.DataSource = ds.Tables[0];
                                cmbTipoDoc.ValueMember = ds.Tables[0].Columns[0].ColumnName;
```

```
                cmbTipoDoc.DisplayMember = ds.Tables[0].Columns[1].ColumnName;
            }

            this.Show();   // Obriga a mostrar o formulário atual.
            cmbTipoDoc.Focus();
            ativo = true;  // Volta a ativar o evento SelectionChanged.
        }

    // Função que permite procurar os registos a partir dos filtros preenchidos pelo utilizador.
    private void procurar(int c = 0)
        {
            string where = "";
            DataSet ds = null;

            if (!string.IsNullOrEmpty(txtNDoc.Text))
    {
                where += " and id_documento like '%" + txtNDoc.Text + "%'";
            }

            if (c > 0)
    {
                where += " and d.id_tipodoc = " + c.ToString();
            }

            if (!string.IsNullOrEmpty(txtTitulo.Text))
    {
                where += " and titulo like '%" + txtTitulo.Text + "%'";
            }

            if (!string.IsNullOrEmpty(txtAutor.Text))
    {
                where += " and autor like '%" + txtAutor.Text + "%'";
            }

    // Percorre as colunas da base de dados à procura da pesquisa do utilizador.
            try {
                ds = this.db.Read("select id_documento, t.ds_tipodoc, titulo, autor,
editora, dt_edicao from documento d, tipodoc t where d.id_tipodoc = t.id_tipodoc" + @where, null);
                if (ds.Tables[0].Rows.Count > 0)
      {
                    DataGridView1.DataSource = ds.Tables[0];
                    DataGridView1.Columns[0].HeaderText = "ID";
                    DataGridView1.Columns[1].HeaderText = "TipoDoc";
                    DataGridView1.Columns[2].HeaderText = "Título";
                    DataGridView1.Columns[3].HeaderText = "Autor";
                    DataGridView1.Columns[4].HeaderText = "Editora";
                    DataGridView1.Columns[5].HeaderText = "Data edição";
                    DataGridView1.Refresh();
                }
      else
      {
```

```
                                        MessageBox.Show("Documento não encontrado");
                                    }
                                }
                catch (Exception ex)
                {
                    MessageBox.Show("Erro ao procurar o documento!", "", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                                }
                        }

        // Método para a mudança de valor na combobox.
                        private void cmbTipoDoc_SelectedIndexChanged(System.Object sender,
System.EventArgs e)
        {
            // Se foi selecionado algum tipo de documento e o evento está ativo.
            if (cmbTipoDoc.SelectedIndex > -1 && ativo)
            {
                try
                {
                    this.procurar(Convert.ToInt32(cmbTipoDoc.SelectedValue));
                }
                catch (Exception ex) {}
            }
        }

        // Evento sobre um click numa célula da DataGrid.
        private void DataGridView1_CellClick(object sender,
System.Windows.Forms.DataGridViewCellEventArgs e)
        {
            this.DocSel = Convert.ToInt32(DataGridView1.Rows[e.RowIndex].Cells[0].Value);
            this.DialogResult = DialogResult.OK;
            this.Close();
        }

        // Evento para o botão Cancelar.
                        private void btnCancelar_Click(System.Object sender, System.EventArgs e)
        {
            this.DialogResult = DialogResult.Cancel;
            this.Close();
        }

        // Inicializador da classe FRM_ProcurarDoc.
        public FRM_ProcurarDoc()
                        {
                                Load += FRM_ProcurarDoc_Load;
                                InitializeComponent();
                        }

        // Evento para o botão Ok. Invoca o método procurar.
        private void txtOk_Click(object sender, EventArgs e)
        {
```

```
            this.procurar();
        }
            }
    }
```

Janela de tipos de documento:



Código:

```
public partial class FRM_TipoDoc
        {
    // Inicialização de variáveis.
                int selTipoDoc;
                DBLib db;

    // Inicializa o formulário.
    public FRM_TipoDoc()
                {
                        Load += FRM_TipoDoc_Load;
                        InitializeComponent();
                }

    // Função executada automaticamente quando é carregado o formulário.
    private void FRM_TipoDoc_Load(object sender, EventArgs e)
                {
                        this.db = new DBLib();
                        this.loadDataGrid();    // Invoca o método para preencher a DataGrid.
                        this.ativaCampos(false, true);  // Coloca o formulário em modo de espera para o
utilizador escolher se quer editar ou inserir um novo registo, desativando os campos do formulário de
inserção/edição de dados.
    }

    // Efetua o carregamento da DataGrid com os registos existentes na tabela de Tipos de Documento.
    public void loadDataGrid()
```

```
            {
                DataSet ds = null;

                ds = db.Read("select id_tipodoc, ds_tipodoc from tipodoc"); // Lê na base de
dados os campos "id_tipocdoc" e "ds_tipodoc".

        // Se o dataSet tiver registos então define a origem de dados (dataSource) como sendo a tabela do
dataSet. Definimos a designação das colunas e fazemos o refresh para popular a DataGrid.
        if ((ds.Tables[0].Rows.Count > 0))
        {
                    dtGridTipoDoc.DataSource = ds.Tables[0];
                    dtGridTipoDoc.Columns[0].HeaderText = "ID";
                    dtGridTipoDoc.Columns[1].HeaderText = "TipoDoc";
                    dtGridTipoDoc.Refresh();
                }
        // Se não houver registos então a datagrid é desativada.
        else
        {
                    dtGridTipoDoc.Enabled = false;
                }

            }

    // Função que controla a ativação dos campos do formulário de acordo com o modo de inserção ou
update.
    public void ativaCampos(bool status, bool limpa)
            {
                txtTipoDoc.Enabled = status;
                btnCancelar.Enabled = status;
                btnGuardar.Enabled = status;

                if (limpa)
                    txtTipoDoc.Text = "";

                if (status == true)
                    txtTipoDoc.Focus();

            }

    // Evento para o botão de adicionar tipo de documento.
            private void btnAddTipoDoc_Click(object sender, EventArgs e)
            {
                this.selTipoDoc = 0;    // Coloca-se o formulário em modo de "inserção".
        dtGridTipoDoc.Enabled = false;  // Desativa a DataGrid.
                btnCancelar.Text = "Cancelar";
                btnAddTipoDoc.Enabled = false;  // Desativa o botão para adicionar um novo
tipo de documento.

                this.ativaCampos(true, true);   // Ativa os campos do formulário e limpa o seu
conteúdo para o utilizador preencher os respetivos valores.
        this.btnGuardar.Enabled = true; // Ativa o botão guardar.
```

```
            }

        // Evento para o botão "Cancelar".
        private void btnCancelar_Click(object sender, EventArgs e)
        {
            // Se estamos no modo de inserção, cancela e desativa os campos de introdução de dados e esvazia
o seu conteudo.
            if (this.selTipoDoc == 0)
            {
                dtGridTipoDoc.Enabled = true;  // Ativa a DataGrid.
                btnAddTipoDoc.Enabled = true;  // Ativa o botão para adicionar um novo tipo de documento.
                this.ativaCampos(false, true);
            }
            else
            {
                // Se estamos no modo de atualização de um registo, antes de cancelar, pede ao utilizador que
confirme a operação, pois poderá ter cancelado por engano.
                if ((MessageBox.Show("Tem a certeza?", "", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes))
                {
                    int ret = 0;

                    ret = this.db.Delete("DELETE FROM TipoDoc WHERE id_tipodoc = " +
this.selTipoDoc.ToString(), null, true);

                    // Se ocorreu um erro, sai.
                    if (ret < 0)
                        return;

                    MessageBox.Show("Dados removidos com sucesso", "", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                    // Coloca o formulário no estado inicial, com a DataGrid ativa e com os campos de
introdução de dados do formulário desativados e vazios.
                    dtGridTipoDoc.Enabled = true;
                    this.loadDataGrid();
                    this.ativaCampos(false, true);
                }
            }
        }

        // Valida os campos se, neste caso, apenas o tipo de documento, estão preenchidos.
        public Boolean validaCampos()
                    {
                            if (string.IsNullOrEmpty(txtTipoDoc.Text))
        {
            MessageBox.Show("Por favor preencha o tipo de documento", "", MessageBoxButtons.OK,
MessageBoxIcon.Information );  // Caixa de texto com botão ok com o ícone de interrogação.
            return false;
                            }
                            return true;
```

```
            }

    // Grava o novo tipo de documento na base de dados.
            private void gravar()
            {
                int ret = 0;

    // Se há campos por preencher então não grava os campos.
    if (!this.validaCampos())
    {
                        return;
                    }

    // Se estamos em modo de inserção, executa o comando "insert", caso contrário, executa o
comando "update".
    if (this.selTipoDoc == 0)
    {
                        ArrayList insValues = new ArrayList();
                        insValues.Add(new SQLValue("@tipodoc", txtTipoDoc.Text));  //
Prepara os parâmetros do comando "insert" de um novo registo na base de dados.

        ret = this.db.Insert("INSERT INTO TipoDoc (ds_tipodoc) values (@tipodoc)", insValues, true);
// Chama a função que insere o novo tipo de documento.

        if (ret < 0)
          return;

        MessageBox.Show("Dados inseridos com sucesso", "", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
        btnAddTipoDoc.Enabled = true;  // Ativa o botão para adicionar um novo tipo de documento.
                    }
    else
    {
                        ArrayList upValues = new ArrayList();
                        upValues.Add(new SQLValue("@tipodoc", txtTipoDoc.Text));   //
Prepara os parâmetros do comando "update" do registo que está a ser editado.

        ret = this.db.Update("UPDATE TipoDoc SET ds_tipodoc = @tipodoc WHERE id_tipodoc = "
+ this.selTipoDoc.ToString(), upValues, null, true);   // Chama a função que atualiza o registo.

        if (ret < 0)
                            return;

        MessageBox.Show("Dados atualizados com sucesso", "", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
    }

                    dtGridTipoDoc.Enabled = true;
                    this.loadDataGrid();   // Invoca o método loadDataGrid para preencher a
DataGrid com os dados inseridos/atualizados.
                    this.ativaCampos(false, true);
```

```
        }

        // Evento que ocorre após um click numa célula da DataGrid.
        private void dtGridTipoDoc_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            this.selTipoDoc = (int)dtGridTipoDoc.Rows[e.RowIndex].Cells[0].Value;  // Adquire o valor da 1° coluna da linha (registo da tabela) em que clicou. Essa coluna corresponde ao id do tipo de documento.
            btnCancelar.Text = "Remover";

            txtTipoDoc.Text = dtGridTipoDoc.Rows[e.RowIndex].Cells[1].Value.ToString(); // Preenche o campo do formulário do tipo de documento (para edição) com a coluna correspondente à descrição do tipo de documento.

            this.ativaCampos(true, false);  // Ativa o modo de edição dos campos.
            this.btnGuardar.Enabled = true; // Ativa o botão para guardar o tipo de documento.
        }

        // Evento para o botão Guardar.
        private void btnGuardar_Click(object sender, EventArgs e)
        {
            this.gravar();  // Invoca o método gravar.
        }
        }
```

DBLib.CS

```
// A classe DBLib é o controlador de comunicação com a base de dados. Permite centralizar todas as operações na base de dados, que na aplicação (e em grande parte das aplicações)
// tratam-se das operações de leitura, inserção, remoção e atualização de registos na base de dados.
// Esta classe não é obrigatória numa aplicação, no entanto, como é facilmente percetível, permite poupar muito código, concentrando todas as fases de cada uma das operações numa única função.
// Assim apenas é executado uma linha de código, excetuando a preparação da variáveis setvalues e wherevalues, que contêm os campos e valores dos respetivos comandos.

namespace DBDoc
{

        public class DBLib
        {
    //Private Const connStr = "Data Source=100.1.1.3;Initial Catalog=DBDoc;User ID=login;Password=login;Persist Security Info=True;"
    public String connStr = "Data Source=localhost;Initial Catalog=DBDoc;Integrated Security=True;";
// Atribui à string connStr o método de conexão à base de dados.

    // Para acesso à base de dados é necessário um controlador (variavel do tipo SQLConnection).
    // É criado também um controlador de transações (SqlTransaction). Uma transação é composta por um conjunto de operações de escrita(apenas de escrita) na base de dados.
    // Podem existir apenas uma operação ou várias. Esse controlo de transações não é obrigatório, mas é extremamente aconcelhável.
    // Esse controlo permite garantir uma das propriedades mais importante de uma base de dados: a consistência de uma base de dados.
    // O controlo de transações é composto por 3 fases:
    //    . Inicio (com a definição do this.MyConn.BeginTransaction();
    //    . Operações de escrita (insert, delete, update);
    //    . Final:
    //        Nesta fase há sempre uma verificação se algum erro ocorreu(esse controlo é feito pelo programador).
    //        Se houve algum erro, então fazemos o "rollback", ou seja, cancelamos todas as operações dessa transação. A base de dados fica no estado antes do inicio da transação.
    //        Se não houve erro então faz o "commit", ou seja, efetua as operações dessa transação, ficando a base de dados no estado modificada pelas operações de escrita que foram entretanto efetuadas.

    // O dataAdapter possibilita a comunicação entre a base de dados e o dataset. É usado na aplicação nas operações de leitura ("select") à base de dados. Permite preencher o dataset com os valores do "select".
    // O dataSet, como o próprio nome indica, é um conjunto de dados guardados numa variavel "dataset" que representa exatamente a base de dados (que neste caso é SQL, mas pode ser de outro tipo).
    // Por outras palavras, o dataAdapter é um adaptador que transpõe os dados que podem ser de várias origens (neste caso é apenas de uma origem, mas de um determinado tipo: SQL Server),
    // adaptando os dados para uma representação que é independente da origem e do tipo de origem de dados: o dataSet.
    // A ligação à base de dados externa à aplicação é garatida pelo SqlConnection, o SqlTransation efetua as operações de escrita.
    // O dataSet é composto por dataTables (tabelas) que por sua vez, cada uma, é composta por linhas e colunas (campos da tabela). As linhas equivalem aos registos da tabela.
```

```
// Assim, podemos dizer que o dataSet é semelhante a uma folha de excel.

public SqlConnection MyConn;
        public SqlTransaction MyTrans;
        private SqlDataAdapter MyDA = new SqlDataAdapter();
        public bool isConnected;

// Inicia a conexão à base de dados.
public DBLib()
        {
            try {
                this.MyConn = new SqlConnection(connStr);  // Fica guardado na
variável MyConn o conteúdo da string connStr
                this.MyConn.Open();    // Abre a conexão à base de dados com
utilizando a string anterior
                this.MyDA = new SqlDataAdapter();  // Estabelece a comunicação entre
a base de dados e o dataset
                this.isConnected = true;   // Atribui o valor verdadeiro à variável
            }
    // Se a ligação não for bem sucedida "limpa" as variáveis, e mosta a mensagem "Erro de Ligação
ao Servidor"
    catch (SqlException ex)
    {
      MessageBox.Show("Erro de Ligação ao Servidor", "Erro", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                this.MyConn = null;
                this.MyTrans = null;
                this.MyDA = null;
                this.isConnected = false;
            }
        }

// Prepara a query SELECT * FROM table WHERE whereValues.
// Os whereValues são enviados para o nosso "cmd" através de parâmetros.
// Não é obrigatório usarmos parâmetros (nem sempre usa-se essa estratégia na aplicação).
// No entanto, a utilização de parâmetros é mais seguro, evitando:
//     . A ocorrência de erros nas operações e na base de dados, devido à introdução de carateres
"estranhos" (plicas e aspas, por exemplo) no preenchimento dos formulários da aplicação;
//     . Algo improvável, mas possível, ataques de SQLInjection através de comandos SQL inseridos
por parte do utilizador no preenchimento dos campos dos formulários.

// Executa o "cmd" atavés do SqlAdapter.
// Preenche o dataSet com os valores lidos da base de dados.
// Retorna o dataSet com os dados para serem processados pela aplicação.
public DataSet Read(string query, ArrayList whereValues = null, string table = "")
        {
            try
    {
                DataSet ds = new DataSet();
                SqlCommand cmd = new SqlCommand();
                cmd.Connection = this.MyConn;
```

```
            SQLValue val;

            cmd.CommandText = query;

                                    if (whereValues != null)
            {
                                            for (int I = 0; I <= whereValues.Count - 1; I++)
              {
                 val = (SQLValue)whereValues[I];
                                                    cmd.Parameters.AddWithValue(val.Field, val.Value);
                                            }
                                    }

                                    ds.Reset();
                                    this.MyDA.SelectCommand = cmd;

                                    if (string.IsNullOrEmpty(table))
            {
                                            this.MyDA.Fill(ds);
                                    }
            else
            {
                                            this.MyDA.Fill(ds, table);
                                    }

            return ds;
                            }
                    catch (Exception ex)
                    {
                                    MessageBox.Show("Erro na leitura dos dados do servivor: " +
ex.Message, "[SELECT]", MessageBoxButtons.OK, MessageBoxIcon.Error);
                                    return null;
                            }
                    }


            // Esta função Insert permite, igualmente ao read, poupar muito código, concentrnado todas as fases
de uma operação de inserção de um registo na base de dados, numa única função.
            // Assim apenas é executado uma linha de código (excetuando a preparação da variável
whereValues).
            // As fases que compõem o insert são semelhantes às da leitura, com a exceção da criação de uma
transação para garantir a consistência da base de dados após a operação de escrita (com erros ou sem
erros).

            // Se tudo correr bem após a execução do comando de insert(cmd.ExecuteScalar()), o identificador
do novo registo é retornado nessa função.
            // Esse id trata-se, por exemplo, o id de um novo livro ou de um novo tipo de livro que foi inserido.

            public int Insert(string query, ArrayList insValues, bool usetrans = true)
                    {
                            SqlCommand cmd = new SqlCommand();
```

```
        SQLValue val;

                        try
        {
                                if (usetrans)
                                        this.MyTrans = this.MyConn.BeginTransaction();

                                cmd.Connection = this.MyConn;
                                cmd.Transaction = this.MyTrans;

          for (int I = 0; I <= insValues.Count - 1; I++)
          {
             val = (SQLValue)insValues[I];
                                        if (val.Type == "image")
             {
                                                SqlParameter p = new
SqlParameter(val.Field.ToString(), SqlDbType.VarBinary);
                                                p.Value = val.Value;
                                                cmd.Parameters.Add(p);
                                        }
             else
             {
                                                cmd.Parameters.AddWithValue(val.Field, val.Value);
                                        }
                                }

                                cmd.CommandText = query;
                                cmd.ExecuteNonQuery();

          string query2 = "Select @@Identity";
          cmd.CommandText = query2;
                                int newId = Convert.ToInt32(cmd.ExecuteScalar());

                                if (usetrans)
                                        this.MyTrans.Commit();


                                return (newId);
                        }
        catch (Exception ex)
        {
                                if (usetrans)
                                        this.MyTrans.Rollback();

          MessageBox.Show("Erro na inserção dos dados no servivor: " + ex.Message, "",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                                return -1;
                        }

                        if (usetrans)
                                this.MyTrans.Dispose();
```

```
        }

    // Função que agrupa numa só função, todas as fases de um comando "update" de um (ou mais) registo(s) numa determinada tabela da base de dados.
    // É semelhante às anteriores no que concerne ao "workflow".
    // Neste caso quanto aos parâmetros, são enviados em duas varáveis.
    // Isto porque o comando Update tem a seguinte sintaxe: update table set @setValues1 = x, @setValues2 = y values where @whereValues1 = t, @whereValues2 = z.
    // Portanto são enviadas a variável setValues e whereValues com os respetivos campos e valores.
    // No final da execução do comando (cmd.ExecuteNonQuery()) é retornado o número de registos afetados pelo comando.
    // A função retorna esse valor caso o comando seja corretamente executado, ou "-1" se houve algum erro.
    public int Update(string query, ArrayList setValues, ArrayList whereValues = null, bool usetrans = true)
        {
            SqlCommand cmd = new SqlCommand();
      SQLValue val;

      try
      {
                if (usetrans)
                    this.MyTrans = this.MyConn.BeginTransaction();

                cmd.Connection = this.MyConn;
                cmd.Transaction = this.MyTrans;

                if (setValues != null)
        {
                    for (int I = 0; I <= setValues.Count - 1; I++)
          {
            val = (SQLValue)setValues[I];
                        cmd.Parameters.AddWithValue(val.Field, val.Value);
                    }
                }

                if (whereValues != null)
        {
                    for (int I = 0; I <= whereValues.Count - 1; I++)
          {
            val = (SQLValue)whereValues[I];
                        cmd.Parameters.AddWithValue(val.Field, val.Value);
                    }
                }

                cmd.CommandText = query;
                int ret = cmd.ExecuteNonQuery();

                if (usetrans)
                    this.MyTrans.Commit();
```

```
                    return ret;

                }
            catch (Exception ex)
            {
                    if (usetrans)
                        this.MyTrans.Rollback();

                MessageBox.Show("Erro na actualização dos dados do servivor: " + ex.Message, "",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return -1;
                }

                if (usetrans) {
                    this.MyTrans.Dispose();
                }
                return 0;
            }


        // Método para apagar registos da base de dados.
        // É semelhante às funções anteriores, retornando "1" se correr tudo bem, ou "-1" caso contrário.
        // Sintaxe: delete from table where @wherevalue1 = x and @wherevalue2 = y.
        public int Delete(string query, ArrayList whereValues = null, bool usetrans = true)
                {
            SQLValue val;

            try
            {
                    if (usetrans)
                        this.MyTrans = this.MyConn.BeginTransaction();

                    SqlCommand cmd = new SqlCommand();

                    cmd.Connection = this.MyConn;
                    cmd.Transaction = this.MyTrans;

                    if (whereValues != null)
                {
                        for (int I = 0; I <= whereValues.Count - 1; I++)
                    {
                        val = (SQLValue)whereValues[I];
                            cmd.Parameters.AddWithValue(val.Field, val.Value);
                        }
                    }

                    cmd.CommandText = query;
                    cmd.ExecuteNonQuery();

                    if (usetrans)
```

```
                    this.MyTrans.Commit();

                    return 1;
                }
            catch (Exception ex)
            {
                    if (usetrans)
                        this.MyTrans.Rollback();

                MessageBox.Show("Erro na remoção dos dados no servivor: " + ex.Message, "",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return -1;
                }

                if (usetrans)
                    this.MyTrans.Dispose();
        }

    }
}
```

Program.CS

```
// Trata-se da classe principal do programa, ou seja, é a primeira classe a ser executada
quando iniciamos a aplicação.

namespace DBDoc
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);

            Application.Run(new FRM_Splash()); // Chama o formulário de login.
        }
    }
}
```

SQLValue.CS

```
// Esta classe preenche cada parâmetro relativo aos comandos da base de dados (select,
insert, update e delete), com o respetivo campo e valor.

// Essas funções estão centralizadas na class DBLib.
//         Em cada uma dessas funções, o respetivo comando é preenchido com os respetivos
parâmetros.
//      Cada parâmetro tem um campo e o respetivo valor.

//      Ex:
//            UPDATE TipoDoc SET ds_tipodoc = @tipodoc WHERE id_tipodoc = @tipodoc
//          upValues -> ds_tipodoc = @tipodoc
//          whereValues -> id_tipodoc = @tipodoc

//    Os parâmetros da variável upValues, dentro da função Update do DBlib são
preenchidos através da seguinte função:
//    cmd.Parameters.AddWithValue(val.Field, val.Value);

// Cada parametro enviado para dentro da função(neste caso a fuunção "Update", mas para
as restantes funções o funcionalmente é igual),
// trata-se de um SQLValue.Cada SQLValue é composto então pelo nome do campo e o
respetivo valor.

//      Ex: SQLValue("@tipodoc", txtTipoDoc.Text);
```

```
// De referir que os parametros dos comandos de base de dados enviados para as respetivas
funções da DBLib são enviados numa coleção de dados do tipo SQLValue, ou seja, uma
ArrayList de SQLValues

namespace DBDoc
{
    public class SQLValue
       {
              public string Field;
                     // Não pode ser string. Tem de ser object, por causa dos valores
"double". Sendo string, perde-se o "," ou o ".". Ex: 3,45 fica 345.
              public object Value;
              public string Type;
              public SQLValue(string field, object value, string type = "")
              {
                     this.Field = field;
                     this.Value = value;
                     this.Type = type;
              }


              public object cmbValue {
                     get { return Value; }
                     set { this.Value = value; }
              }
              public string cmbField {
                     get { return Field; }
                     set { this.Field = value; }
              }
       }
}
```

Tabela documentos:

id_documento	autor	titulo	editora	fonte	pdf	capa	dt_edicao	resumo	id_tipodoc
1	Edmundo Mon...	Engenharia de...	FCA		NULL	<Binary data>	2011-02-01	Desde que em ...	1
2	Vários	Administração ...	FCA		<Binary data>	<Binary data>	2011-03-01	A gestão, ou ad...	1
3	Miguel Mira da ...	IT Governance,...	FCA		NULL	<Binary data>	2008-09-01	Numa altura e...	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Tabela tipodoc:

id_tipodoc	ds_tipodoc
1	Livro
2	Jornal
3	Manuscrito
4	Artigo
NULL	NULL

Tabela de utilizadores:

id_utilizador	username	password	is_admin
1	Admin	P@ssw0rd	1
2	User1	biblio	0
NULL	NULL	NULL	NULL

Conclusão

Ficou demonstrado neste projeto a complexidade em construir um sistema virtualizado. Porém, ao ter tudo centralizado, tem-se uma maneira eficiente de gestão e manutenção, em conjunto com a monitorização dos recursos de cada máquina.

Ao ter os servidores agregados numa única máquina física a gestão dos servidores e das máquinas virtuais torna-se mais fácil, assim como a gestão dos recursos físicos. Para além disto permite poupar recursos energéticos e espaço físico.

A capacidade de provisionamento é evidente no sistema de gestão de máquinas virtuais, podendo criar facilmente máquinas virtuais num espaço de tempo relativamente curto, na quantidade que for necessário.

As máquinas virtuais disponíveis aos utilizadores da rede têm a possibilidade de consultar informação na base de dados através da aplicação e têm também acesso à internet devido à sua configuração.

O *framework* ADO.NET mostrou ser uma tecnologia apropriada a ser usada neste cenário, assim como a linguagem C#. A aplicação permite acesso à base de dados, e disponibiliza os dados de forma amigável para o utilizador através do seu interface. A adição de uma camada de segurança na forma de *login* permite que apenas o administrador tenha a possibilidade de alterar os dados inseridos.

Concluindo, após feita toda a pesquisa, e ter o laboratório criado, é possível afirmar que um sistema virtualizado juntamente com uma aplicação ligada a uma base de dados apresentam uma solução viável para um local como uma biblioteca.

Bibliografia

Ambiente Virtualizado:

- PDF's fornecidos pelo professor Pedro Crispim.

Aplicação e SQL Server:

- Projeto elaborado na cadeira de Programação III
- Aulas de Programação IV
- *http://www.mikesdotnetting.com/article/54/getting-the-identity-of-the-most-recently-added-record*
- *https://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.executenonquery(v=vs.110).aspx*
- *https://msdn.microsoft.com/en-us/library/xyhh2e7e(v=vs.110).aspx*
- *http://stackoverflow.com* – diversos tópicos para resolução de problemas como por exemplo:
    - *http://stackoverflow.com/questions/2822843/itextsharp-html-to-pdf*
    - *http://stackoverflow.com/questions/1887388/how-to-add-value-to-combobox-item*
    - *http://stackoverflow.com/questions/36208209/saving-and-retrieving-an-image-from-database-in-c-sharp*