Licenciatura em Informática

Um Mundo Virtual

Projeto Global

Autor

David Alexandre Vieira Marques

Nº2029

Orientador

Professor Doutor Pedro Brandão

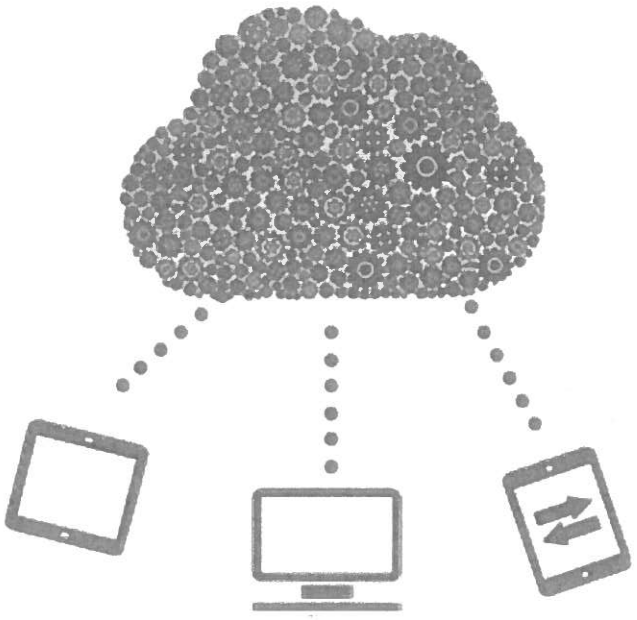
Lisboa

2016/2017

INSTITUTO SUPERIOR DE TECNOLOGIAS AVANÇADAS

Licenciatura em Informática

Um Mundo Virtual



Autor

David Alexandre Vieira Marques

Nº2029

Orientador

Professor Doutor Pedro Brandão

Lisboa

2016/2017

Agradecimentos

Agradeço a todos os professores que me acompanharam ao longo da Licenciatura, e me transmitiram as bases de conhecimento necessárias para o meu sucesso, tanto académico como futuro profissional. Um especial agradecimento ao *Professor Doutor* Pedro Brandão, pela exigência e rigor durante as suas aulas, o que me permitiu evoluir e superar as minhas capacidades de autoaprendizagem.

Agradeço também aos meus colegas de curso, pelo espírito amigável e de entreajuda ao longo dos anos, tornando esta experiência académica mais agradável.

Agradeço acima de tudo à minha família, por todo o apoio e investimento feito na minha formação.

Índice

Índice de Figuras

Resumo

No âmbito do projeto final da licenciatura em Informática do Instituto Superior de Tecnologias Avançadas, foram consolidados neste documento conhecimentos teóricos e práticos de vários autores, em várias vertentes das tecnologias de virtualização, cloud computing, SQL server, .NET, Windows Server 2012 R2, System Center 2012 R2 e Exchange Server 2013.

A componente prática divide-se em 2 partes: laboratório de rede virtual, e desenvolvimento de uma aplicação em C# com interface WPF, e ligação a base de dados através de objetos ADO.NET. Essa aplicação foi instalada numa máquina cliente, que acede remotamente à rede do domínio e à base de dados, por VPN.

Abstract

Within the scope of the final project of the degree in computer science of the *Instituto Superior de Tecnologias Avançadas,* theoretical and practical knowledge from several authors was consolidated in this document regarding virtualization, cloud computing, SQL server, .NET, Windows Server 2012 R2, System Center 2012 R2 and Exchange 2013 technologies.

The practical component is divided in 2 parts: virtual lab, and development of a C# application with WPF interface, which is connected to a database using ADO.NET. That application was installed in a client machine, that can access the domain network and the database remotely, through VPN.

Keywords

virtualization – cloud computing - windows server 2012 r2 – SQL server – .NET – exchange

Introdução

O objetivo deste projeto foi criar e configurar um laboratório de rede, que respondesse a um conjunto de necessidades de determinada empresa fictícia e se aproximasse o mais possível de um cenário real de produção. A empresa era nova, e pretendia montar uma infraestrutura de rede com baixos consumos energéticos e com a mínima ocupação de espaço por *hardware* possível, mas que tivesse um bom desempenho e fornecesse todos os serviços necessários ao cumprimento das suas tarefas.

As necessidades dessa empresa eram as seguintes:

- Gestão de utilizadores e máquinas de forma centralizada;

- Aplicação com ligação a uma base de dados que listasse várias informações de produtos incluindo as suas imagens, e permitisse geri-los (adicionar, editar, eliminar,etc);

- Comunicação entre as várias máquinas da organização;

- Acesso direto à Internet através de uma única NIC;

- Acesso de utilizadores remotos à rede;

- Sistema interno de e-mail;

- Medidas de segurança adequadas a um cenário deste tipo.

Para dar resposta a essas necessidades foi implementada uma infraestrutura de rede com tecnologias de virtualização, em que foi feito o seguinte:

- Criação de um domínio com 2 controladores de domínio. Um principal e um secundário para propósitos de replicação dos dados do Active Directory, entre as 2 infraestruturas de rede físicas da empresa (1 em cada site do AD);

- Configuração de algumas políticas no *Group Policy*, relativas à segurança;

- Criação de um servidor para fornecer serviços de SGBD com o SQL Server 2014;

- Desenvolvimento da aplicação em C#, com interface WPF, incluindo as funcionalidades pedidas e algumas outras adicionais;

- Criação de *switches* virtuais internas para comunicação *offline* entre as máquinas e uma externa para ligação à Internet;

- Criação e configuração de um servidor de *routing*, que fornece serviços de roteamento IPv4, NAT e VPN segura utilizando certificados digitais para autenticação;

- Criação e configuração de um *Network Policy Server*, servidor cuja finalidade é monitorizar e controlar os acessos à rede através de políticas de rede;

- Instalação e configuração do *Microsoft Exchange Server 2013*, para troca de e-mails internamente, com utilização de certificados digitais;

Capítulo I - Estado da arte

1.1 Virtualização

Por volta do ano 2006, a tecnologia de virtualização (já criada há mais de 30 anos), começou a ser implementada em maior escala. Isto ocorreu, pois, a Intel e a AMD adicionaram funcionalidades nos seus processadores para suportar esta tecnologia, e alcançar a virtualização de forma mais simples. A virtualização combina ou divide os recursos computacionais de modo a providenciar diferentes ambientes operacionais, com diferentes metodologias e técnicas tais como particionamento ou agregação de hardware e software, simulação de máquinas total ou parcial, emulação e tempo compartilhado. (Brian et al., 2009)

Um *Virtual Machine Monitor*(*VMM*) ou *Hypervisor* é fundamental para a virtualização. Pode ser definido como uma fina camada de software, que permite a abstração do hardware para o sistema operativo, permitindo a execução concorrente de múltiplos sistemas operativos ou instâncias do mesmo sistema operativo, denominadas como "convidados", num único computador anfitrião(*host*). (Desai, Oza, Sharma, & Patel, 2013)

Existem dois tipos de *VMM*, tipo 1 e tipo 2.

O primeiro, também conhecido como um *hypervisor* nativo, executa diretamente sobre o *hardware*, como se de um sistema operativo se tratasse. A sua responsabilidade é agendar e alocar recursos de sistema para as máquinas virtuais. Alguns exemplos deste tipo de *hypervisor* são o Hyper-V da Microsoft, o ESX da Vmware e o Xen. (Desai et al., 2013)

O segundo, também conhecido como um *hypervisor* alojado, sendo que executa como uma aplicação sobre o sistema operativo *host*, e é tratado pelo sistema como qualquer outro processo. Alguns exemplos deste tipo de *hypervisor* são o Fusion da Vmware e o Virtual Box da Oracle. (Desai et al., 2013)

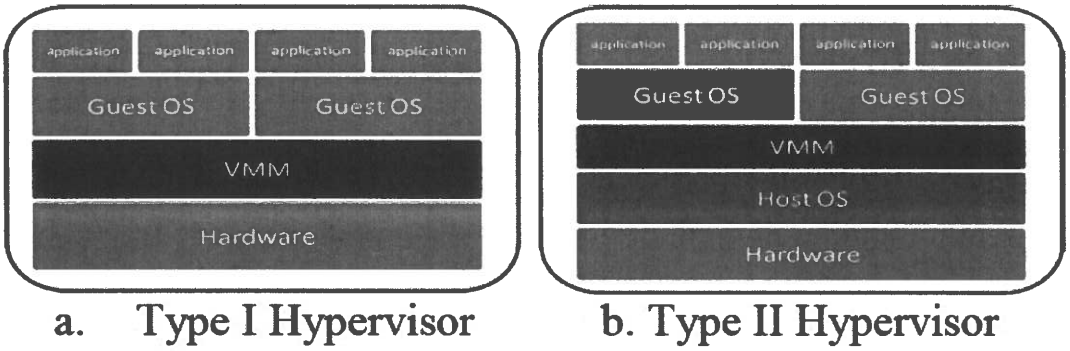


Figura 1 - Tipos de hypervisors (Desai et al., 2013)

Alguns dos benefícios da virtualização:

- Portabilidade de sistemas;
- Permite a consolidação/centralização de servidores;
- Maximiza a utilização de recursos computacionais;
- Reduz o consumo energético e o espaço ocupado por equipamento de arrefecimento;
- Reduz a carga na rede física;
- Permite a gestão de recursos baseada em políticas definidas.

(Barnier, Brown, & Dittmann, 2008)

A diminuição de servidores físicos através da consolidação e virtualização, tem vários impactos positivos num *data center*, um dos maiores sendo a redução de consumos energéticos. (Brandão, P., 2016)

1.1.1 Hyper-V

O Hyper-V foi introduzido pela Microsoft em 2008, sucedendo o Windows Virtual PC como a sua tecnologia de virtualização, e desde então tem sido atualizada a cada versão do Windows Server. É atualmente uma das soluções mais utilizada do mundo e é a tecnologia de virtualização por detrás do Windows Azure. Foi inicialmente lançado no mercado em conjunto com o Windows Server 2008 como um *role* de servidor, mas atualmente já pode ser implementado em vários sistemas operativos cliente, o mais recente sendo o Windows 10. Foi com o Windows Server 2012 que o Hyper-V se tornou um dos melhores *hypervisors* de tipo 1 do mercado. Isto deu-se devido ao grande foco por parte da Microsoft em tecnologias de virtualização, e ao sucesso da sua plataforma de *cloud* pública, o Windows Azure. Esse foco trouxe também algumas novas tecnologias tais como o *Hyper-V Replica*, *Network Virtualization*, *SMB 3.0* e *Live Migration*.

Uma das grandes melhorias trazidas pelo Windows Server 2012 foi relativa à escalabilidade, que era anteriormente um dos pontos mais fracos. A figura seguinte é ilustrativa disso.

(Savill, 2014)

	Resource	Microsoft Hyper-V Server 2008 R2	Microsoft Hyper-V Server 2012	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 2 - Escalabilidade WS2008-WS2012 (Savill, 2014)

Estes novos limites de escalabilidade permitiram que quase qualquer *workload*(mais de 99%) pudesse ser virtualizada com o Hyper-V.(Savill, 2014)

Uma das atualizações chave do Hyper-V com o Windows Server 2012 R2 foi o suporte de criar máquinas virtuais de segunda geração(vhdx).

Têm as seguintes características:

- Baseadas em UEFI - obedecem aos padrões de *UEFI Secure Boot*;
- *Legacy Free* - Ao contrário das *VMs* de primeira geração, foram removidos muitos dos dispositivos emulados e substituídos por *drivers* sintéticos com dispositivos baseados em software;
- *ISCSI Boot* – Permite que as máquinas virtuais iniciem execução diretamente de discos *SCSI* com controladora própria. A controladora IDE não é compatível.
- Implementação mais rápida – Instalações de um sistema operativo *guest* são significativamente mais rápidas quando comparadas com as VMs de geração anterior, devido ao suporte de boot *PXE* e ao melhor desempenho da controladora *ISCSI*. (Tulloch & Team, 2013)

1.1.2 Virtualização de redes

Em vez de fornecer às máquinas virtuais uma ligação direta às placas de rede, estas são tipicamente ligadas a interfaces virtuais. *Switches* virtuais fornecem a ligação entre as interfaces virtuais e as interfaces físicas, e gerem o tráfego entre elas. Ao contrário de *switches* físicas, que ligam os *hosts* a uma rede, as *switches* virtuais residem no *host* e são escritas inteiramente em software. Isto liberta a sua implementação da rigidez do hardware, e possibilita a utilização de funções de *forwarding* sofisticadas e mais eficientes. (Pfaff et al., 2009)

1.1.3 Armazenamento

O armazenamento é um componente crítico em qualquer estratégia de virtualização. As organizações utilizam *storage area networks(SANs)* ou *network-attached storage(NAS)* para armazenar ficheiros. Estes sistemas de armazenamento utilizam tecnologias de virtualização para criar blocos abstratos de armazenamento chamados *logical unit numbers(LUNs)*. Estes, fornecem muitos dos mesmos benefícios de servidores virtuais, tais como a redução da ocupação de espaço num data center, o aprovisionamento de espaço para armazenamento enquanto as aplicações permanecem *on-line*, e a gestão do espaço disponível de forma eficiente e sem *downtime*. (Barnier et al., 2008)

1.1.4 Eficiência dos *data centers*

Os servidores são os maiores consumidores de energia num *data center*. Combinar várias *workloads* no menor número de servidores possível pode aumentar significativamente a eficiência energética de um *data center*. Antes de adicionar novos servidores a um *data center* deve considerar-se a utilização computacional atual, caso contrário poderá ocorrer um indesejável e evitável aumento de consumos energéticos. O mesmo serve para a infraestrutura física do *data center*, ou seja, quantidades desproporcionais de equipamento computacional e de arrefecimento são endémicas à maior parte dos *data centers* existentes e representa um desperdício considerável de energia.

Um problema frequente nos *data centers* é a subutilização de servidores. Grandes poupanças e/ou aumentos de produtividade sem gastos adicionais energéticos podem ser alcançados com a utilização de tecnologias de virtualização. (Talaber, Brey, & Lamers, 2009)

A utilização da virtualização, para além da diminuição de custos, tem um profundo impacto positivo no ambiente, reduzindo as emissões de carbono. Um *data center* virtualizado é um "green" *data center*. (Brandão, P., 2016)

A métrica utilizada para quantificar a eficiência energética geral de um *data center* é a *Power Usage Effectiveness*, ou *PUE*, e foi desenvolvida pela Green Grid. A *PUE* é calculada com a divisão do valor total de energia disponível nas instalações (infraestrutura de iluminação e equipamento de arrefecimento e de IT), pelo valor de energia utilizado pelos equipamentos de *IT*. O resultado ideal de *PUE* é 1.0, mas essencialmente impossível de alcançar. Os resultados mais frequentes em *data centers* estão entre 2.0 e 2.5, onde 2.0 significa que 1 watt de eletricidade é perdido por cada watt utilizado para alimentar equipamentos de *IT*.

A Google construiu alguns dos mais eficientes *data centers* na indústria, tendo alcançado uma *PUE* de 1.12 num dos seus *data centres*.

(Broderick & Jenkins, 2015)

1.1.4.1 Classificação de servidores e consumos energéticos

O típico *data center* é predominantemente ocupado por servidores *volume*. Estes, em combinação com as suas *workloads* têm variados requisitos de desempenho e tempo de operacionalidade. Simplificando, estes servidores serão categorizados em 3 tipos: inovação, produção ou críticos. Estas classificações ajudam a determinar os tipos de arquitetura, processos e infraestrutura a ser implantada, bem como a perceber que oportunidades e esforços seriam envolvidos no processo de consolidação e virtualização.

Os servidores utilizados num ambiente de inovação, são utilizados parar criar novos produtos, alterar os atuais, desenvolver ou melhorar processos que tornam a organização mais competitiva, produtiva, entre outros. Neste ambiente são incluídos servidores tipicamente descritos como: testes, desenvolvimento, garantia de qualidade, etc. Após o desenvolvimento de uma solução no ambiente de inovação, a *workload* é tipicamente movida para um ambiente de produção.

(Talaber et al., 2009)

No ambiente de produção, a mudança é muito mais controlada e a escalabilidade é extremamente importante. Neste ambiente, a velocidade de implantação e a flexibilidade são

secundários aos requisitos de nível de serviço. Algumas soluções são vitais à organização, logo são consideradas críticas quando postas em produção.

É importante notar que é no ambiente de inovação que se encontra maior quantidade de servidores subutilizados e que necessitam de menos recursos, logo é normalmente o ponto de partida para aplicar esforços de consolidação e virtualização. Exemplo de um ambiente com 500 servidores:

CATEGORIES	INNOVATION	PRODUCTION	MISSION CRITICAL	TOTAL
Server Count	250	175	75	500
Utilization	3%	6%	10%	5%
Watts (173 Per Server)	43,250	30,725	1,297.5	86,500
Consolidation Ratio	15:1	10:1	5:1	~10:1
Remaining Servers	17	18	15	50
Post-Consolidation Utilzation	50%	50%	50%	50%
Post-Consolidation Watts	3,910	4,140	3,450	11,500
Energy Savings	39,340	26,135	9,525	75,000

Figura 3 - Amostra de servidores num data center (Talaber et al., 2009)

Um estudo feito pela VMware Information Warehouse revelou que mais de 20% dos servidores avaliados pela sua ferramenta CapacityPlanner em 2007, executam abaixo de 0.5% de utilização. Para obter estes dados, foram avaliados mais de 300.000 servidores em milhares de organizações por todo o mundo. Aproximadamente 75% desses executavam abaixo de 5% de utilização. Isto revela uma grande oportunidade de poupança de energia, reduzindo o número de servidores físicos e empregar virtualização.
(Talaber et al., 2009)

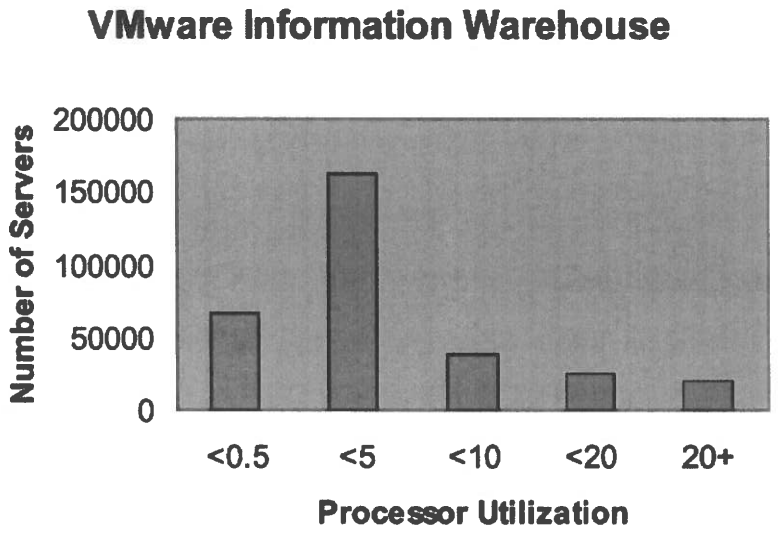


Figura 4 - Utilização de CPU (Talaber et al., 2009)

O consumo energético de um servidor não tem uma relação linear com o trabalho ou produtividade do mesmo. Um servidor inativo pode consumir mais de 40% da energia que um servidor a 100% de utilização. Se o típico servidor executa a 5% de utilização sendo a utilização alvo 50%, o consumo energético poderia ser significativamente reduzido aumentando a utilização desse servidor e desligando servidores inutilizados.

Nas várias categorias de servidores, a utilização média em horário de trabalho, foi observada em aproximadamente 3% para inovação, 6% para produção e 10% para críticos. Melhorar a eficiência energética utilizando tecnologias de virtualização é extremamente importante. Um estudo, observou que a 10% de utilização, um dado servidor consumia 173 watts. Adicionando 10% de carga e levando a máquina aos 20% de utilização, só se registou um aumento de consumo de 16 watts, ou seja, de 173 para 189 watts.

(Talaber et al., 2009)

A linha na figura abaixo representa a quantidade incremental de utilização energética numa máquina para cada 10% de utilização adicionais. A 100% de utilização, o pico de consumo energético atingiu os 276 watts. Verifica-se que à medida que a utilização do *CPU* aumenta, o consumo energético aumenta a um ritmo cada vez mais lento.

(Talaber et al., 2009)

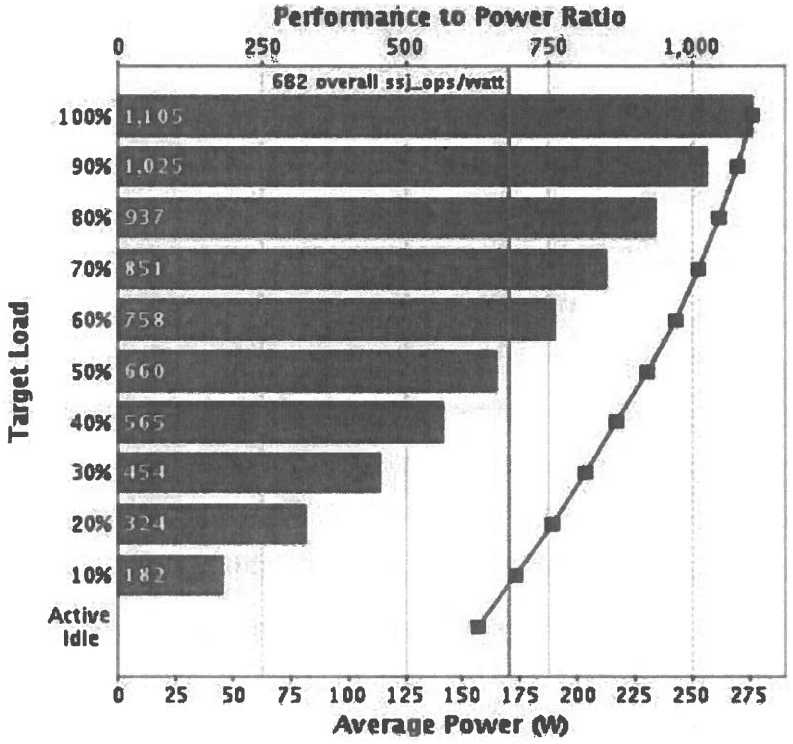


Figura 5 - Rácio de consumo para performance (Talaber et al., 2009)

1.2 Cloud OS

O Cloud OS representa a aproximação visionária da Microsoft ao *IT* e como este pode solucionar as necessidades e enfrentar os desafios atuais das organizações. Esta visão abrange *data centers*, *clouds* privadas, *clouds* públicas e soluções híbridas.

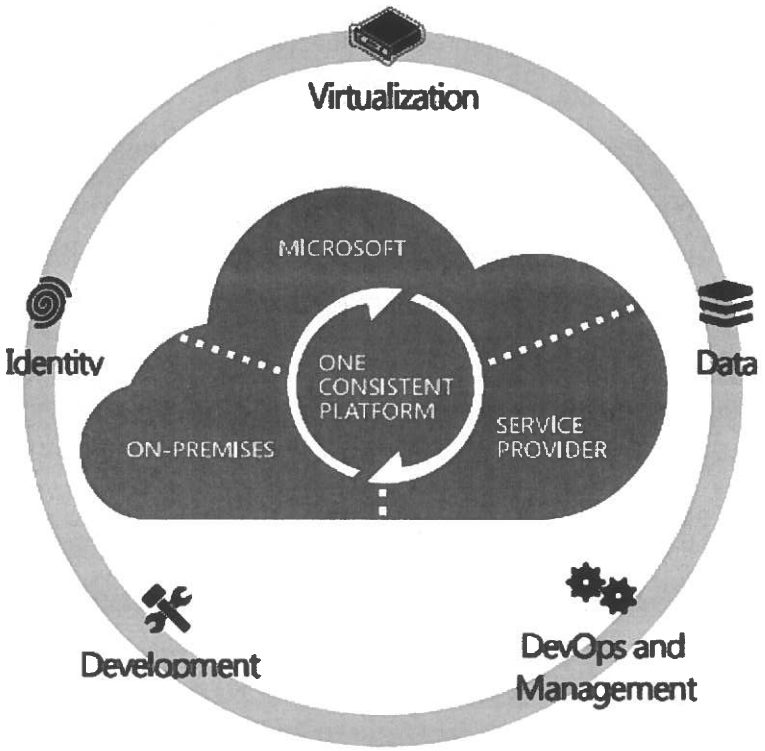


Figura 6 - Cloud OS ("The Cloud OS Becomes Reality," 2012)

As 3 principais plataformas que constituem o Cloud OS são as seguintes:

- Windows Server 2012 R2 – Fornece a fundação para construir soluções empresariais de *data center* e *clouds* híbridas, simples de implementar, economicamente viáveis, focadas no software e centradas no utilizador. É o "coração" do Cloud OS;
- System Center 2012 R2 – Permite uma experiência de gestão unificada pelas *on-premises*, provedores de serviço e ambientes de Windows Azure, de uma forma que seja simples, economicamente viável, focada no *software* e de nível empresarial;
- Windows Azure – Fornece uma plataforma aberta e flexível de cloud para a construção, implementação e gestão de aplicações utilizando qualquer linguagem, ferramenta, ou *framework*, e executa-as numa *cloud* pública segura, alojada numa rede global de *data centers* geridos pela Microsoft. O Azure permite a integração de aplicações de *cloud*

pública, com as existentes num ambiente de *IT on-premise*, permitindo assim capacidades de *cloud* híbrida.

(Chappell, Microsoft, & All, 2013)

1.2.1 Windows Server 2012 R2

O Windows Server 2012 R2 foi o primeiro sistema operativo servidor da Microsoft a permitir integração com a *cloud*. Todos os dados, estejam eles em postos de trabalho físicos ou em máquinas virtuais podem ser guardados diretamente na *cloud*, seja *on-premise* ou fora.

Com o lançamento base do Windows Server 2012, só existiam duas versões, Standard e Datacenter. Com o lançamento do R2, mais duas edições foram lançadas, a Foundation e a Essentials. Todas elas têm as funcionalidades base de servidor, porém, as últimas duas são limitadas em termos de funcionalidade, sendo mais indicadas para organizações mais pequenas. A única diferença entre a Standard e a Datacenter é no limite de instâncias virtuais permitidas, em que no caso do Datacenter são ilimitadas.

(Minasi et al., 2014)

A funcionalidade do Windows Server imediatamente após a sua instalação é muito limitada. Para ter acesso às várias funcionalidades, devem ser instalados os *roles* necessários. Uma *role* de servidor é um conjunto de programas que, quando instalados e adequadamente configurados, permitem executar determinadas funções. O Windows Server 2012 R2 tem um grande número de *roles* e serviços, mas no âmbito deste estudo serão apenas referidos alguns deles.

Active Directory

O AD é uma base de dados centralizada onde podem ser armazenados e geridos todos os objetos (utilizadores, computadores, etc.) da infraestrutura Windows. É uma coleção de serviços que permitem administrar toda a infraestrutura através da rede.

O AD é composto dos seguintes *roles*:

- Active Directory Domain Services (AD DS)
- Active Directory Federation Services (AD FS)
- Active Directory Lightweight Directory Services (AD LDS)
- Active Directory Rights Management Services (AD RMS)

- Active Directory Certificate Services (AD CS)

(Tulloch & Team, 2013)

Cada um deles tem a sua importância e deve ser explorado num ambiente empresarial, mas o serviço vital, ou seja, o que tem de estar instalado para aceder às funcionalidades base do AD é o AD DS. Este serviço está integrado no Windows Server mas não é instalado por defeito. É o serviço responsável pela criação de domínios e promoção dos servidores a controladores de domínio.

(Minasi et al., 2014)

Todos os controladores de domínio têm a sua própria cópia da base de dados do Active Directory, sendo esta atualizada de forma dinâmica pelos outros controladores de domínio. Por esta motivo, é absolutamente essencial que existam pelo menos 2 controladores de domínio, por questões de redundância. O Active Directory também pode ser implementado no Windows Azure, onde os controladores de domínio podem ser virtualizados e armazenados na *cloud*.

(Tulloch & Team, 2013)

DNS

Computadores comunicam entre si através de endereços IP, sejam eles v4 ou v6. No entanto, este tipo de endereço não é fácil de memorizar para o ser humano. Com isso em mente, o DNS (Domain Name System) foi criado. Este sistema é utilizado pelo Windows Server e a sua função principal é a resolução de endereços IP em nomes. Isto facilita imenso a identificação de dispositivos, e é um sistema necessário para o correto funcionamento de muitos serviços, como é o caso do Active Directory e do IIS. No Windows Server 2012 R2 esta role foi atualizada para permitir melhor gestão através do PowerShell.

IIS (Web Server)

O Microsoft Internet Information Services, teve a sua primeira incarnação no Windows NT 3.51. Evoluiu de oferecer um serviço básico como servidor HTTP, para uma plataforma de serviços de aplicações web altamente configurável, e integrada com o sistema operativo. Com o lançamento do IIS 8.0, este tornou-se mais escalável e apropriado para sistemas de *cloud* e virtuais.

(Schaefer, Cochran, Forsyth, Glenderning, & Perkins, 2013)

O IIS é um role que fornece uma plataforma modular e extensível para alojamento de *websites*, serviços e aplicações de forma segura. (Tulloch & Team, 2013)

Uma das mudanças radicais com o IIS 7.0, foi na integração com ASP.NET. Estas melhorias simplificaram drasticamente a implementação do IIS num ambiente de programação ASP.NET e permitiu melhor configurabilidade e implementação de ambos sites e aplicações ASP.NET.

Na instalação por defeito(*default)* do IIS 8.0, apenas podem ser alojadas páginas estáticas, o que não é desejado na maior parte dos casos. Isto pode ser corrigido através da adição de novas funcionalidades ao IIS, como por exemplo a implementação de páginas dinâmicas ASP.NET. Essas funcionalidades podem ser adicionadas através do *wizard* de adição de *roles* e funcionalidades, como mostra a imagem seguinte.

(Schaefer et al., 2013)

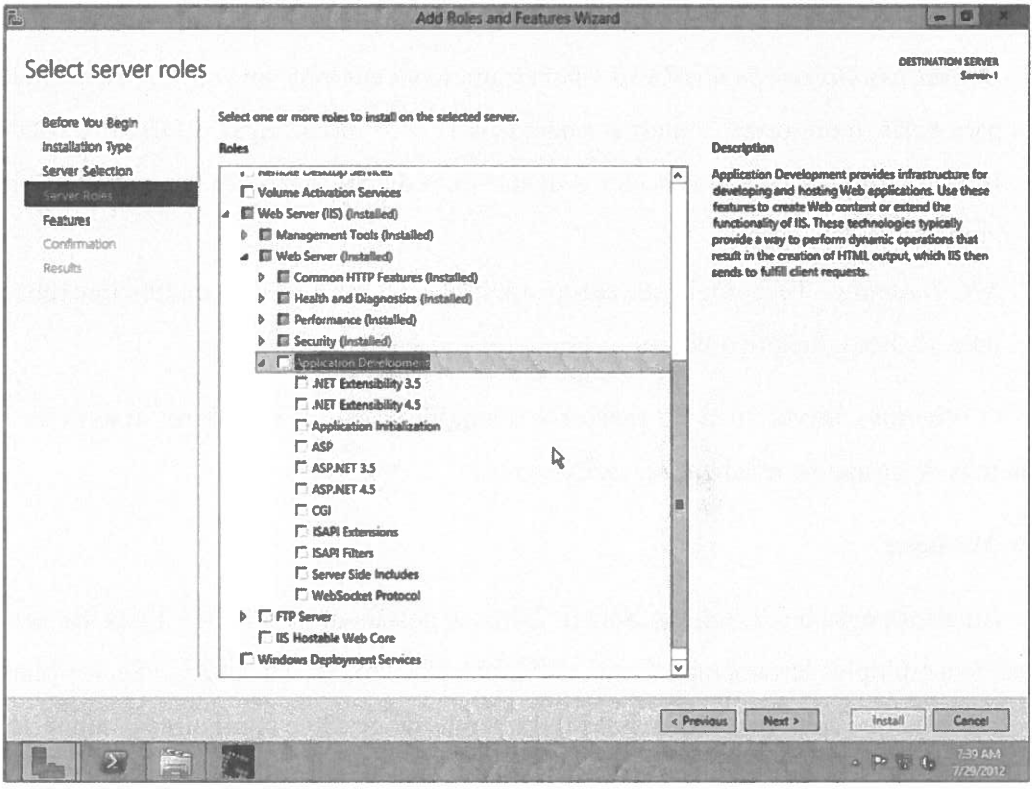


Figura 7 - Add roles and Features Wizard

Novas funcionalidades/mudanças relevantes no IIS 8.0:

- Inicialização de aplicações;
- Restrições dinâmicas de endereços IP;
- Suporte centralizado para certificados SSL;
- *CPU Throttling;*
- Restrições nas tentativas falhadas de início de sessão FTP;
- Suporte para *Server Name Indication;*
- Melhorias no SSL e escalabilidade de configurações;
- Suporte para escalamento multicore em *hardware* NUMA.

Novas funcionalidades

O Windows Server 2012 R2 introduziu mais de 300 novas funcionalidades. Algumas delas são as seguintes:

- EAP-TTLS – Protocolo usado com padrões 802.1X na autenticação *wired* e acesso *wireless* para fornecer melhor segurança e prevenir acessos não autorizados;
- *IP Address Management(IPAM)* – Esta framework engloba um conjunto de tecnologias para gerir, monitorizar e auditar endereços IP. Ao monitorizar o DHCP e o DNS, o IPAM permite localizar servidores dentro da rede pelos seus endereços IP, e geri-los numa única interface;
- *NIC Teaming* – Tecnologia que agrupa múltiplas placas de rede para funcionarem como uma só. Isto é benéfico por motivos de *failover* e balanço de carga.

O Windows Server 2012 R2 melhorou a administração de servidores através de novas ferramentas de gestão ou atualizações às existentes.

Server Manager

Anteriormente ao Windows Server 2008, a gestão de servidores tinha de ser feita recorrendo a múltiplas ferramentas. Com o WS2008, a Microsoft introduziu o Server Manager, uma ferramenta que centraliza todas essas ferramentas de gestão e configuração numa só.

O Windows Server 2012 R2 expande essas funcionalidades ao possibilitar a gestão de múltiplos servidores (virtuais ou físicos).

(Tulloch & Team, 2013)

WinRM

O WS2012 R2 introduziu um novo protocolo de rede chamado *Windows Remote Management*. Este protocolo veio substituir o *Remote Procedure Call*(RCP), pois este último não era seguro. À semelhança do RPC, o WinRM permite a comunicação entre programas, mesmo que estes estejam em diferentes computadores.

Algumas características do WinRM:

- Não é proprietário, mas é baseado em padrões estabelecidos e não é dependente de uma plataforma específica;
- É uma forma modificada de HTTPS;
- As suas comunicações são encriptadas;
- Requere autenticação.

Alguns dos componentes/funcionalidades do WS2012 R2 que utilizam o WinRM incluem a coleção de *event logs*, a abilidade de utilizar a *snap-in* do Server Manager em servidores remotos, e a linha de comandos shell segura – *Windows Remote Shell*(WinRS).

Group Policy

A Group Policy, é uma infraestrutura hierárquica que permite a um administrador do Active Directory, implementar configurações aplicáveis a utilizadores e computadores, bem como definir políticas de utilizador, segurança e rede ao nível da máquina.

Com o WS2012 R2, a gestão de *Group Policy Objects(GPOs)* tornou-se mais fácil, através da *Group Policy Management Console*. Deixou de ser necessário fazer o *download* e executar ferramentas separadas para monitorizar problemas de replicação ao nível do domínio, e foi atualizada para suportar o Internet Explorer 10.

(Tulloch & Team, 2013)

1.2.2 System Center 2012 R2 – Virtual Machine Manager

O VMM é um componente do System Center e é a solução da Microsoft para a gestão de *data centers* por *software*, com integração na *cloud*. (Ziembicki & Tulloch, 2014)

Permite a gestão e configuração de *hosts* de virtualização, *clusters* virtuais, e recursos infraestruturais usados para criar e implementar máquinas virtuais e serviços para *clouds privadas*. Esses recursos incluem grupos de *hosts*, recursos de rede, recursos de

armazenamento, servidores biblioteca e partilhas. Em conjunto, estes recursos constituem o "*fabric*" do qual as *clouds* privadas são implementadas e geridas através da família de produtos do System Center.(Chappell et al., 2013)

Hosts* e *host clusters

O VMM permite a gestão de múltiplas plataformas de *hypervisor*, incluindo o Hyper-V, Citrix XenServer e VMware ESX.

Grupos de *hosts*

Para facilitar a gestão de um grande número de *hosts* e *clusters* virtuais, o VMM permite organizá-los em grupos lógicos. Estes, podem ser criados com base em diversos critérios, tais como a localização física dos *hosts* ou a forma como os recursos são alocados a eles.

Recursos de rede

Permite a criação de *clouds* privadas com uma larga gama de recursos geridos de rede aprovisionados. O VMM fornece também capacidades de virtualização de rede, incluindo suporte para a criação e gestão de redes virtuais e *gateways*. A virtualização de redes é um conceito paralelo à virtualização de servidores, onde permite a abstração e execução de múltiplas redes virtuais numa única rede física.

Recursos de armazenamento

Permite a descoberta, classificação, aprovisionamento, alocação e designação de ambos armazenamentos locais, onde a sua capacidade é diretamente ligada ao *host* de virtualização, e armazenamento remoto, em que a tarefa da sua gestão é feita num dispositivo de armazenamento externo. O VMM suporta armazenamento em bloco, inclusive *Fibre Channel e ISCSI.*

Servidores biblioteca e partilhas

A biblioteca do VMM contém um catálogo de recursos usados na criação e implementação de máquinas virtuais e serviços, em *hosts* de virtualização. Esses recursos são de 2 tipos:

- Recursos *file-based* – Este tipo inclui *vhd's*, ISO's, *scripts* de powershell, *scripts* de sql server, *drivers*, entre outros.

- Recursos *Non-file-based* – Este tipo inclui modelos(*templates)* de máquinas virtuais, modelos de serviços e perfis usados para estandardizar a criação de máquinas virtuais e respetivos modelos.

A figura seguinte mostra a consola do VMM com um *workspace* selecionado (*VM's and Services*). O *workspace* neste exemplo é usado para a implementação e gestão de máquinas virtuais, redes virtuais, *clouds* e *tenants*. Nesta secção do VMM, podem ser analisados critérios relativos à performance dos *hosts*.

(Chappell et al., 2013)

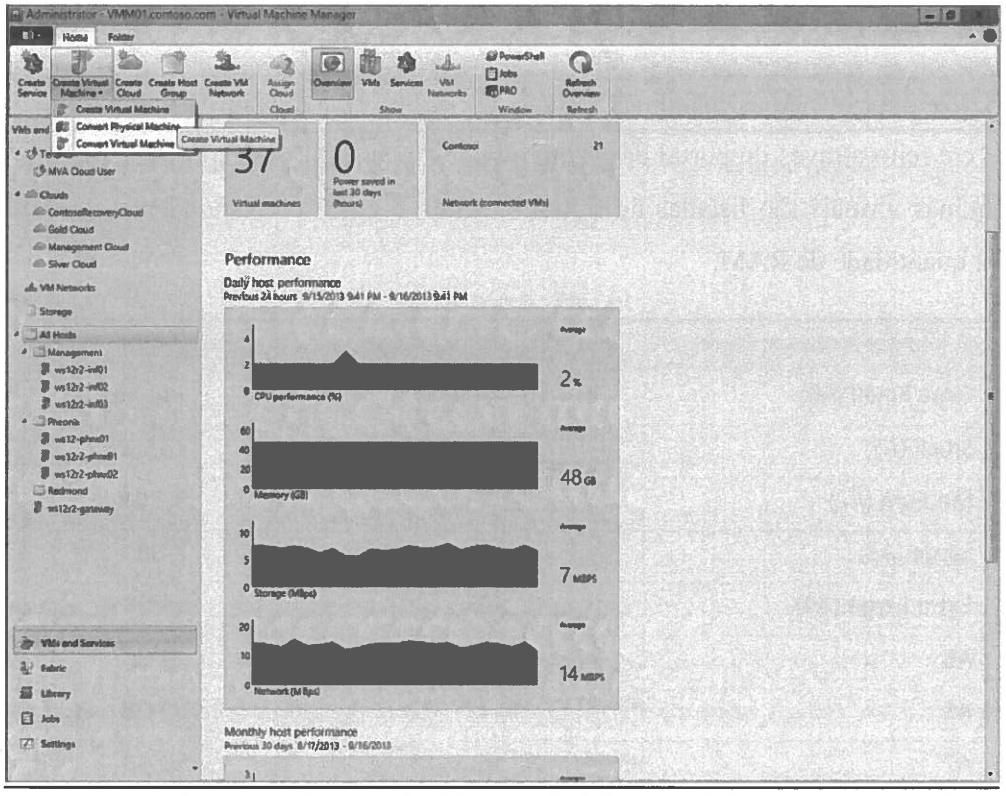


Figura 8 - Performance VMM (Chappell et al., 2013)

1.2.3 Windows Azure

O Azure é a solução de *cloud* pública da Microsoft para *IaaS*. É um dos maiores investimentos na história da Microsoft, tendo em conta os massivos *data centers*, capacidade computacional, de armazenamento e de rede, em adição à investigação e desenvolvimento dos vários serviços englobados. Muitos desses serviços são acompanhados de um *service level agreement(SLA)* e todos eles seguem um modelo em que o consumidor é taxado conforme os recursos que utiliza. Permite que organizações alojem as suas *workloads* e aplicações na *cloud*,

simultaneamente ligadas a recursos *on-premise* com um modelo de *cloud* híbrida. (Ziembicki & Tulloch, 2014)

O m*arketplace* do Azure disponibiliza uma grande quantidade de serviços. Alguns dos quais são:

- Máquinas virtuais;
- *Web sites;*
- Serviços *mobile;*
- Serviços *cloud.*

Máquinas Virtuais

Criar uma ou várias máquinas virtuais no Azure tipicamente não demora mais de 5 minutos e é feito através do portal do Azure ou por Powershell. As especificações oferecidas nas máquinas virtuais são listadas na figura 5, sendo o custo correspondente ao número de cores e à quantidade de RAM.

Compute Instance Name	Virtual Cores	RAM
Extra Small (A0)	Shared	768 MB
Small (A1)	1	1.75 GB
Medium (A2)	2	3.5 GB
Large (A3)	4	7 GB
Extra Large (A4)	8	14 GB
A5	2	14 GB
A6	4	28 GB
A7	8	56 GB

Figura 9 - Azure VM Specs (Ziembicki & Tulloch, 2014)

O Azure suporta uma vasta gama de sistemas operativos nas máquinas virtuais, incluindo o Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, bem como sistemas operativos Linux como o Ubuntu, CentOS e Oracle Linux. Para além disso, o Azure também fornece aplicações para as *VM's* como por exemplo SQL, SharePoint, entre outras.

Web sites

O Azure permite a implementação rápida de aplicações web e integração dos mesmos com várias *frameworks* Microsoft ou de terceiros. Os *web sites* do Azure são elásticos e escaláveis e podem ser expandidos a máquinas virtuais adicionais a pedido ou automaticamente, com base em políticas definidas.

Serviços *mobile*

Os serviços *mobile* do Azure dispõem de funcionalidades para o desenvolvimento de aplicações móveis que permitem estruturar o armazenamento, autenticar utilizadores e enviar notificações.

Serviços *Cloud*

Os serviços de *cloud* do Azure permitem a rápida implementação de aplicações *web* de alta disponibilidade. As aplicações podem ser carregadas(*uploaded)* através deste serviço, e o Azure executa-as de acordo com os detalhes de implementação como o aprovisionamento, balanço de carga(*load balacing)*, e monitorização de saúde.

(Ziembicki & Tulloch, 2014)

1.3 .NET

Antes do lançamento da Microsoft .NET *framework*, os programadores de *software* que desenvolviam aplicações para sistemas operativos Windows, utilizavam o modelo COM (Component Object Model). Este modelo permitia-lhes construir bibliotecas de código que podiam ser partilhadas por diversas linguagens de programação. Apesar desta útil característica, o COM apresentava problemas infraestruturais, um frágil modelo de implementação e era compatível exclusivamente com sistemas operativos Windows.

Para colmatar estes problemas, a Microsoft introduziu a .NET *framework* em 2002. Esta visava oferecer um modelo de programação de aplicações para desktop, *web sites*, serviços e bibliotecas de dados muito mais poderoso, flexível e simples do que o COM.

(Troelsen & Japikse, 2016)

Algumas das funcionalidades principais da .NET:

- Interoperabilidade de código existente – *Software* COM existente pode ser integrado com *software* .NET e vice-versa.

- Suporte para numerosas linguagens de programação – Aplicações .NET podem ser criadas utilizando várias linguagens de programação (C#, Visual Basic, F#,etc).
- Um *runtime engine* comum partilhado por todas as linguagens .NET – Um aspeto deste *engine* é um conjunto bem definido de *types*, que cada linguagem .NET é capaz de interpretar.
- Integração de linguagens – A .NET suporta herança de classes entre várias linguagens. Por exemplo, pode ser definida uma classe em C# e esta ser herdada numa aplicação Visual Basic.
- Uma biblioteca de classes abrangente – Esta biblioteca fornece milhares de *types* predefinidos que permitem compilar bibliotecas de código, aplicações de terminal, aplicações gráficas de desktop e *web sites* de nível empresarial.
- Um modelo de implementação simplificado – Ao contrário do COM as bibliotecas .NET não são registadas no registo do sistema. Para além disso, a .NET permite que múltiplas versões do mesmo .dll existam em harmonia na mesma máquina.

(Troelsen & Japikse, 2016)

1.3.1 ADO.NET

O ADO.NET é a tecnologia de acesso a dados da *framework* .NET. Consiste num conjunto de classes que permitem a conetividade entre aplicações .NET e fontes de dados(tipicamente bases de dados relacionais), a execução de comandos e a gestão de dados. Utiliza uma arquitetura multi-camada que revolve em torno de alguns objetos chave, tais como *Connection, Command* e *DataSet*.

A maior diferença entre a ADO.NET e outra tecnologia de acesso a bases de dados, é a forma como esta lida com o desafio de diferentes *data sources*. No ADO clássico por exemplo, os programadores utilizavam o mesmo conjunto genérico de objetos, independentemente da *data source*. O ADO.NET utiliza um modelo de *data provider*s.

(MacDonald, Freeman, & Szpuszta, 2010)

Data Providers

Um *data provider* é um conjunto de classes do ADO.NET que permite o acesso, a execução de comandos SQL e a recuperação de dados de uma base de dados específica. É de certa forma uma "ponte" de ligação entre a aplicação e uma *data source*.

As principais classes que compõem um *data provider* incluem as seguintes:

- Connection – Estabelece ligação a uma *data source*.
- Command – Executa comandos SQL e procedimentos armazenados.
- DataReader – Fornece acesso rápido *read-only* e *forward-only* aos dados devolvidos de uma consulta a base de dados.
- DataAdapter – Este objeto desempenha 2 tarefas. Primeiro, permite guardar dados extraídos de uma *data source* num *DataSet*, e segundo, permite fazer alterações a uma *data source*, consoante o que foi alterado no *DataSet*.

(MacDonald et al., 2010)

DataSets

O objeto *DataSet* reside em memória e é uma representação dos dados que fornecem um modelo de programação relacional e consistente, independentemente da *data source*. É um conjunto completo de dados, que incluem tabelas, *constraints* e relações entre tabelas.(MacDonald et al., 2010)

A figura seguinte é uma representação deste modelo.

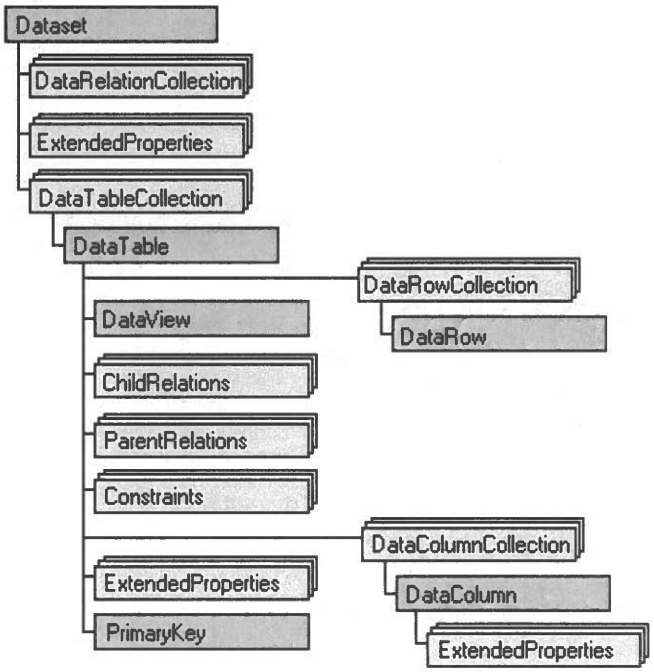


Figura 10 - Modelo DataSet (MacDonald et al., 2010)

No ADO.NET não existem objetos genéricos de *data provider*, mas sim diferentes *data providers* especificamente desenhados para cada tipo de *data source*. Cada um tem as suas próprias implementações para as classes *Connection,Command,DataReader* e *DataAdapter*, otimizadas para um SGBD específico, como o SQL por exemplo.

Data providers incluídos atualmente na framework .NET:

- SQL Server
- OLE DB
- ODBC
- Oracle

(Troelsen & Japikse, 2016)

A figura seguinte é ilustrativa do funcionamento de um *Data Provider*.

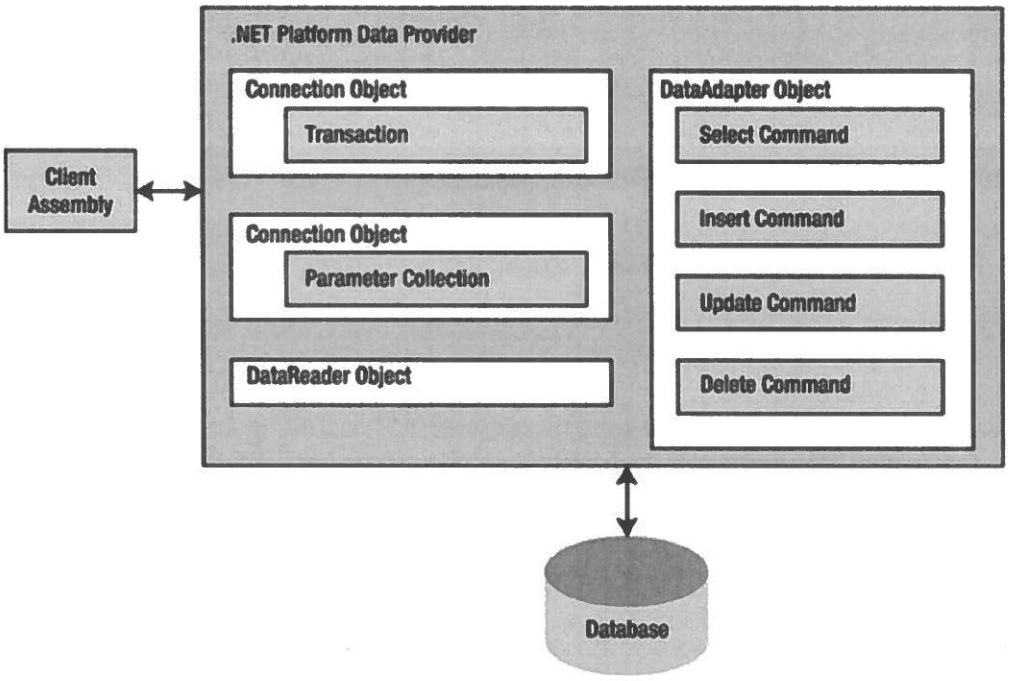


Figura 11 - Data Provider (Troelsen & Japikse, 2016)

1.3.2 ASP.NET

O ASP.NET é uma *framework* da Microsoft, sucessora da tecnologia ASP (Active Server Pages), que permite a criação de páginas *web* dinâmicas, ao contrário de uma simples página HTML que é estática (o seu conteúdo é fixo no ficheiro HTML, ou seja, produz sempre o mesmo resultado). Com o ASP.NET, as páginas *web* podem ser criadas utilizando as linguagens C# ou VB, e executam do lado do servidor. Sendo que são páginas dinâmicas, o seu conteúdo só é definido após o pedido do cliente(*browser*), podendo até aceder a bases de dados para obter determinados dados antes de devolver a página ao cliente. (Pope, 2012)

O ASP.NET evoluiu do ASP com os seguintes benefícios:

- Separação entre a parte lógica da aplicação(código *server-side*) e a apresentação da mesma(HTML);
- Tem um conjunto rico de controlos de servidor que renderizam de forma automática o HTML para cada cliente;
- Gestão de sessões melhorado;
- Um modelo de programação baseado em eventos no lado do servidor;
- Lógica aplicacional que pode ser escrita em qualquer linguagem Microsoft .NET(VB,C#,etc).

(Pope, 2012)

1.4 SQL

A *Structured Query Language* é uma linguagem especificamente desenvolvida para permitir a gestão e manutenção de bases de dados, bem como a utilização dos dados contidos nelas.

Uma base de dados é uma coleção de registos integrados e autodescritivos, enquanto que um registo é a representação de um objeto físico ou concetual. As bases de dados consistem em ambos dados e meta-dados. Os meta-dados descrevem a estrutura dos dados na base de dados.

(Fallis, 2013)

Database Management System(DBMS)

Um sistema de gestão de bases de dados é um conjunto de programas utilizados para definir, administrar e processar bases de dados e as suas aplicações associadas. É a ferramenta que permite a construção da estrutura de uma base de dados e manipular os dados contidos nela.

Modelos de estruturação de bases de dados

Hierárquica:

Bases de dados hierárquicas são compostas de uma arquitetura hierárquica simples, que permite acesso rápido a dados. Sofrem de problemas de redundância e inflexibilidade estrutural, o que torna a manipulação das bases de dados difícil.

Rede:

Bases de dados de rede têm redundância mínima mas uma grande desvantagem é a complexidade estrutural.

(Fallis, 2013)

Relacional:

É o tipo de base de dados mais utilizado atualmente. Este modelo de estrutura oferece flexibilidade estrutural, pois as aplicações escritas para este tipo de bases de dados são mais fáceis de manter, e permite a consulta de combinações de dados. Os dados residem em tabelas, independentes umas das outras. Isto possibilita a manipulação dos dados numa tabela, sem que isto afete de forma alguma as outras tabelas.

(Fallis, 2013)

No núcleo deste modelo estão as relações. Uma relação é um conjunto de colunas(atributos) e linhas, numa estrutura tabular que representa uma entidade, composta de dados relacionados. Os dados de uma relação são armazenados em tuplas, que são conjuntos de dados, cujos valores compõem uma instância de cada atributo definido para essa relação.

Um dos componentes mais importantes das bases de dados relacionais é a forma como as relações de dados estão associadas entre si. Estas associações, ou relacionamentos, interligam as relações de forma relevante, mantendo assim a integridade de dados para que uma

ação tomada numa relação, não afete negativamente a outra relação. Existem 3 tipos de relacionamentos, um-para-um, um-para-muitos e muitos-para-muitos.

(Oppel & Sheldon, 2009)

1.5 SQL Server

O SQL Server é um sistema de gestão de bases de dados desenvolvido pela Microsoft. A sua principal função é armazenar e devolver dados conforme pedido por outras aplicações, locais ou remotas (através da rede local ou na *cloud*).

O SQL Server 2014 fornece a organizações e clientes uma plataforma consistente para infraestrutura, aplicações e dados que se estendem a *data centers*, provedores de serviços de alojamento, e a *clouds* públicas, privadas ou híbridas. Permite a importação de bases de dados diretamente de máquinas virtuais no Azure.

(Mistry & Misner, 2014)

Edições

As principais edições do SQL Server 2014 são as seguintes:

- Enterprise
- Standard
- Business Intelligence

A edição Enterprise é a mais poderosa. Foi desenhada para suportar as mais altas exigências de *data centers* e armazéns de dados de larga escala. Permite implementar *clouds* privadas, ambientes altamente virtualizados e soluções de negócio centralizadas. Inclui todas as características das outras edições.

Algumas características desta edição:

- Não existe limite de núcleos(*cores)* de processamento que podem ser utilizados;
- Capacidade de virtualização ilimitadas;
- Funcionalidade *AlwaysOn*, permitindo alcançar alta disponibilidade;
- Auditoria avançada;
- Funcionalidade de indexação *columnstore*;
- Encriptação de dados transparente;
- Compressão e particionamento.

O SQL Server 2014 satisfaz as crescentes exigências das organizações ao reduzir custos de *hardware* e operacionais, fornecer alta disponibilidade, e o escalamento dos seus negócios através de um ambiente de *cloud* híbrida. A sua integração com o Azure torna simples as tarefas de implementação e gestão de *workloads* por administradores. Oferece soluções híbridas para virtualização de dados, movimento de dados, segurança, alta disponibilidade e escalamento elástico.

(Mistry & Misner, 2014)

Uma das funcionalidades dessa integração é a *cloud disaster recovery*. Permite criar réplicas das instâncias do SQL Server e estende-las a máquinas virtuais do Azure, que executam a sua própria instancia do SQL Server na cloud. Isto pode ajudar na redução de custos relacionados com alta disponibilidade e *disaster-recovery*.

A figura seguinte é ilustrativa de uma solução de base de dados num ambiente híbrido onde parte do ambiente SQL Server executa *on-premise* e a outra na cloud, no Windows Azure.

(Mistry & Misner, 2014)

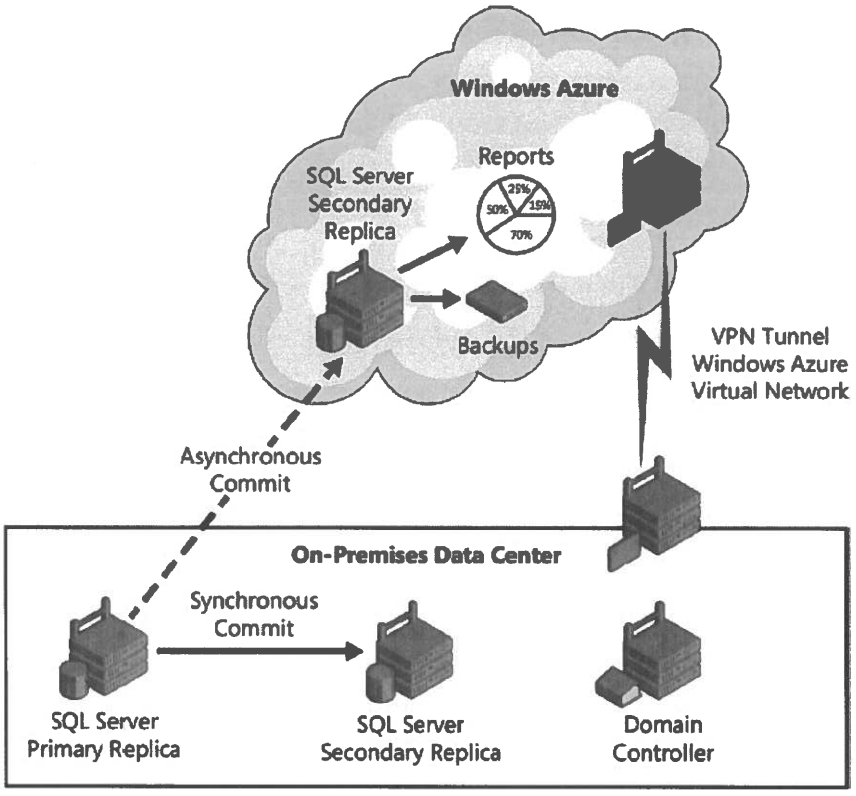


Figura 12 - Azure Disaster-recovery (Mistry & Misner, 2014)

1.5.1 Evolução e o SQL Server 2016

Ao longo dos anos, o desenvolvimento do SQL Server tem sido no sentido de facilitar a integração com fontes de dados externas e simultaneamente simplificar a gestão e administração. Com os recentes desenvolvimentos dos serviços de *cloud* da Microsoft e a mudança de direção para fontes de dados maiores e mais variadas, as capacidades da *cloud* cresceram drasticamente. (Microsoft, 2016)

Desde o SQL Server 2012, a *cloud* tem sido o ponto central de desenvolvimento. O SQL Server 2014 adicionou novas e significativas funcionalidades de integração com a *cloud*, e o SQL Server 2016 melhorou essas funcionalidades. A figura seguinte é ilustrativa da evolução do SQL Server ao longo dos últimos 15 anos.(Microsoft, 2016)

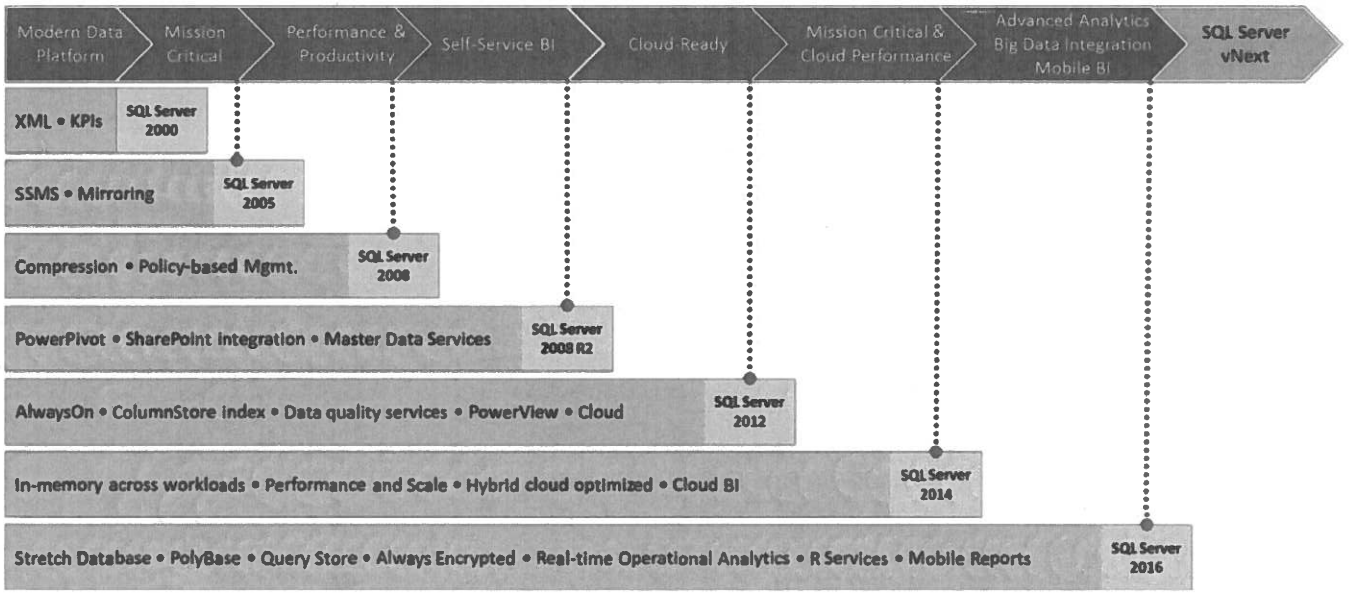


Figura 13 - Evolução do SQL Server (Microsoft, 2016)

Algumas das funcionalidades/melhorias no SQL Server 2016:

- Cópias de segurança*(backups)* híbridas mais rápidas, alta disponibilidade e melhorias de *disaster recovery* com a realização de cópias de segurança das bases de dados *on-premises* para o Azure;
- Melhorada disponibilidade com as réplicas na *cloud*;
- Melhorada consistência e experiência de trabalho com o SQL Server *on-premises*, em máquinas virtuais do Azure, e com o Azure SQL.

(Microsoft, 2016)

Capítulo II – Contextualização

O conceito de virtualização teve origem no ínicio dos anos 70, após um grande investimento da IBM no desenvolvimento soluções de *time-sharing,* o que se refere à utilização partilhada de recursos computacionais por um grande grupo de utilizadores, com o objetivo de aumentar a eficiência. Este modelo representou um grande avanço na tecnologia computacional, e deu origem a uma redução considerável de preços para usufruir de recursos computacionais. Razões semelhantes têm vindo a impulsionar esta tecnologia nos dias de hoje. A capacidade de um servidor é tão grande que apenas através da virtualização, esses recursos podem ser utilizados de forma eficiente.

Nos dias de hoje, a virtualização é uma tecnologia muito mais abrangente, mas apesar disso, o seu significado nuclear permanece o mesmo: habilitar um ambiente computacional, capaz de executar múltiplos sistemas em simultâneo.

("1.1.1. Brief History of Virtualization," n.d.)

A virtualização evoluiu, e hoje é uma das tecnologias mais importantes no mundo do IT, tendo sido revolucionária para a indústria computacional. A consciencialização crescente para a sua utilização deve-se a vários fatores tais como a eficiência energética, redução de espaço ocupado por equipamentos, maximização de recursos e escalabilidade. Fornece formas de simplificar as operações informáticas, permitindo às organizações responder mais eficientemente às exigências de um mercado em constante mudança.

Neste projeto são demonstradas algumas dessas vantagens, ao consolidar recursos de 6 máquinas virtuais, numa única máquina física.

Capítulo III – Desenvolvimento

3.1 Laboratório Virtual

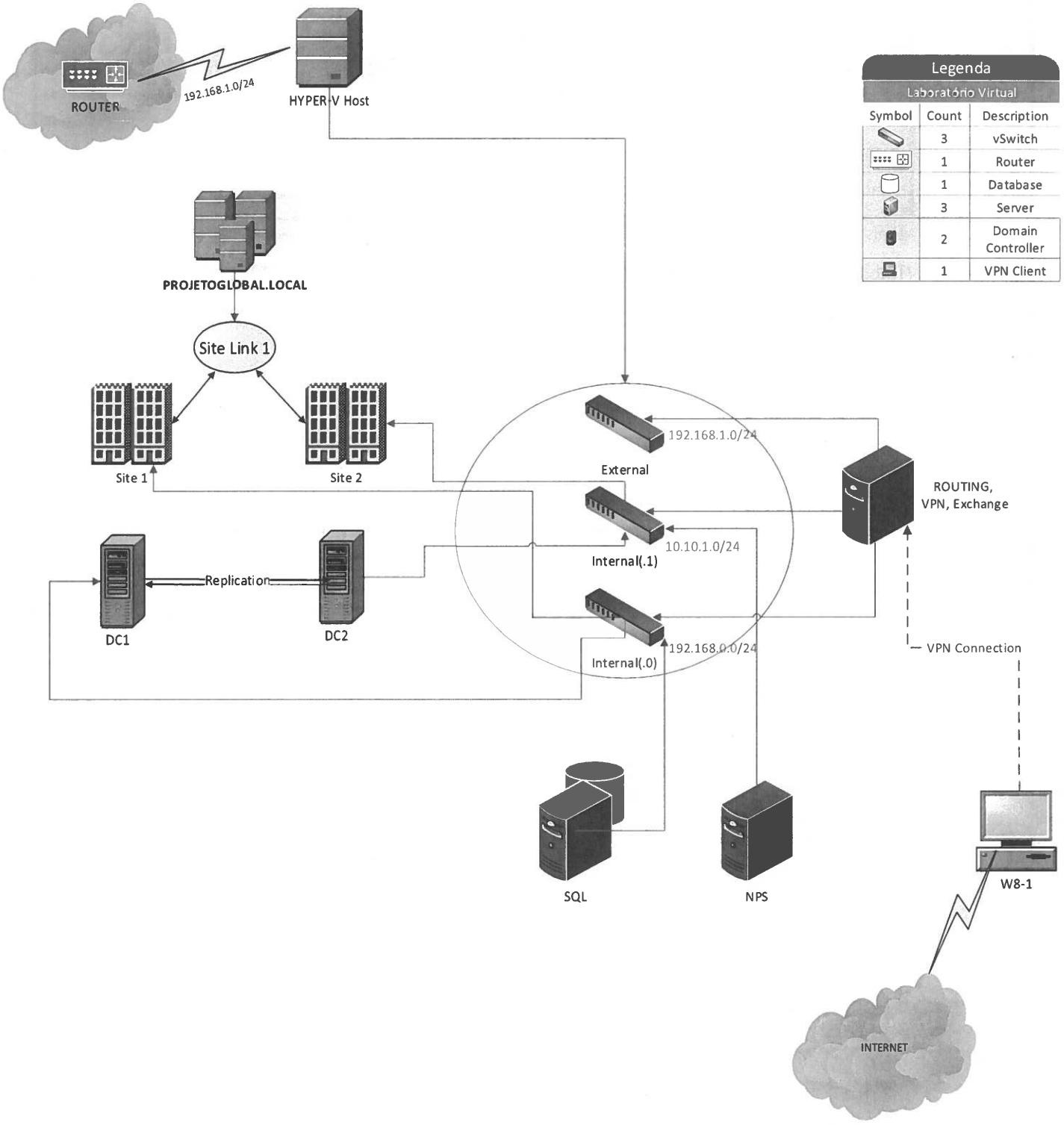


Figura 14 - Diagrama da rede

Máquinas virtuais e endereços

VM	Função	VSwitch e segmento de rede	Endereço IP	Gateway	DNS
DC1	Controlador de domínio principal	Internal(.0) - 192.168.0.0/24	192.168.0.1	192.168.0.3	192.168.0.1
DC2	Controlador de domínio secundário	Internal(.1) - 10.10.1.0/24	10.10.1.2	10.10.1.1	192.168.0.1
ROUTING	Routing, VPN e Exchange	External - 192.168.1.0/24	192.168.1.20	192.168.1.1	8.8.8.8
	Site 1	Internal(.0) - 192.168.0.0/24	192.168.0.3	192.168.0.3	192.168.0.1
	Site 2	Internal(.1) - 10.10.1.0/24	10.10.1.1	10.10.1.1	192.168.0.1
NPS	Controla os acessos remotos à rede	Internal(.1) - 10.10.1.0/24	10.10.1.3	10.10.1.1	192.168.0.1
SQL	Serviços de base de dados	Internal(.0) - 192.168.0.0/24	192.168.0.4	192.168.0.3	192.168.0.1

Tabela 1 - VM's e endereços

3.1.1 Criação do VHDX

Para iniciar o desenvolvimento do laboratório, foi criado um disco virtual vhdx de geração 2, com 220GB, no qual foi instalado o Windows Server 2016 Datacenter para servir como *host* de virtualização. O sistema operativo podia ser inicializado por *nested virtualization* ou fazendo boot diretamente sob o hardware, alterando o bootloader. Neste caso foi feito da forma posterior.

1. No "Disk Management", na barra de ferramentas selecionar a opção "Create VHD", abrindo a janela "Create and Attach Virtual Hard Disk".
2. Após definir a localização para o disco, o seu formato e tipo, clicar em Ok. Isto irá criar o disco virtual e fazer "attach" ao mesmo.
3. Na linha de comandos, executar o seguinte comando para adicionar a nova entrada no bootloader: "bcdboot F: (drive do VHDX)\Windows".

3.1.2 Hyper-V

No Windows Server 2016, foi instalado o role Hyper-V através do Server Manager. Em *Tools*, aceder ao Hyper-V Manager.

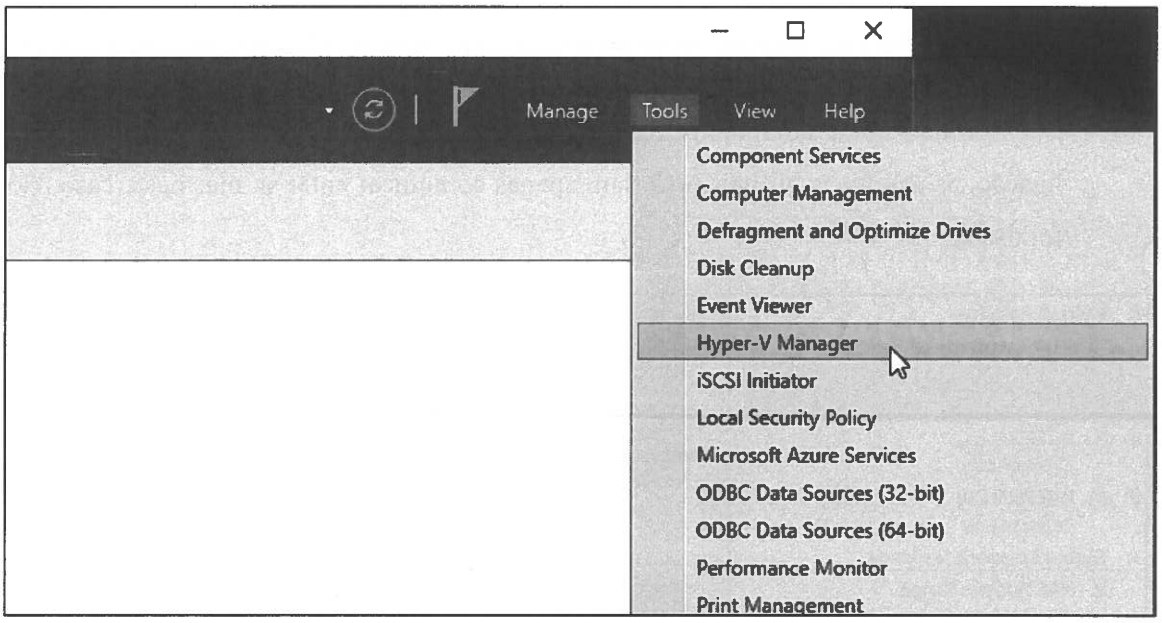


Figura 15 - Acesso ao Hyper-V Manager

3.1.2.1 Switches Virtuais

Para que as máquinas virtuais possam ter conetividade entre si, com o host, e com o exterior, é necessário criar switches virtuais. Para os propósitos deste laboratório, foram criados 3(dois para ligações internas e um para ligações com o exterior).

1. No painel "Actions", aceder ao "Virtual Switch Manager".

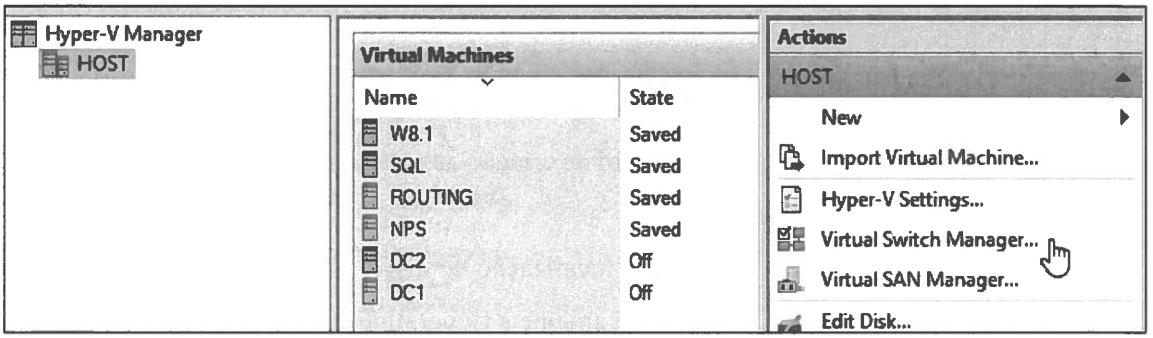


Figura 16 - Hyper-V Manager

2. Criar 3 novas switches virtuais e dar-lhes nomes apropriados. No caso da External switch, esta deve ter o tipo de ligação "External Network", ligada à NIC física. Ativar

também a opção "Allow management operating system to share this network adapter", caso contrário a máquina *Host* deixaria de ter ligação à Internet visto só existir 1 NIC física instalada.

As outras duas vSwitches devem ter o tipo de ligação "Internal Network", para que as máquinas virtuais comuniquem apenas entre si, e com o *Host*, sem acesso direto ao exterior(Internet).

Existe ainda um terceiro tipo de ligação, "Private Network". Com este tipo de ligação, as máquinas virtuais poderiam apenas comunicar entre si, mas neste caso não foi usada.

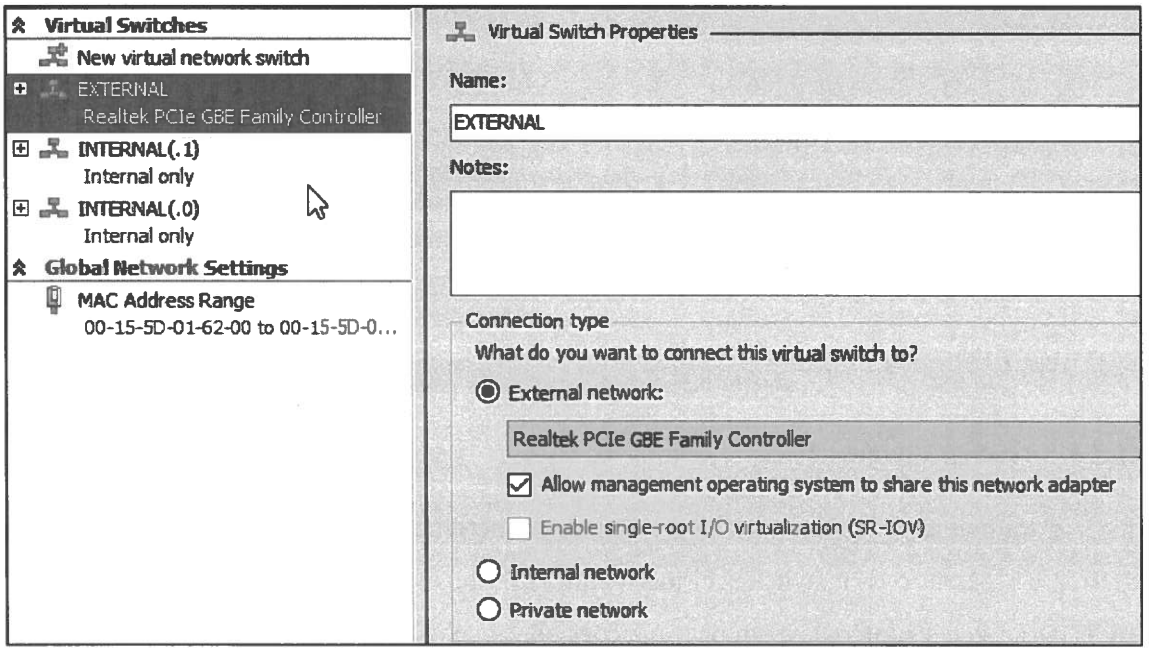


Figura 17 - Create vSwitch

3.1.2.2 Máquinas virtuais

Foram criadas 6 máquinas virtuais, 5 servidores e 1 cliente.

1. No painel "Actions", aceder ao wizard de criação de máquinas virtuais através da opção "New – Virtual Machine".
2. Definir um nome para a VM e a sua localização no disco.
3. Escolher "Generation 2". Comparativamente à Generation 1, esta tem maior capacidade de armazenamento(64TB), e inclui uma série de novas funcionalidades, tais como:
    - "PXE boot" com adaptador de rede *standard*, em vez de ter de instalar um adaptador legacy, e Secure Boot;

- *Boot* a partir de um disco ou DVD, com controlador SCSI;
- "Secure Boot", ajuda a prevenir a execução de *firmware*, sistemas operativas ou drivers UEFI no *boot*.

4. Definir a memória RAM para o arranque da VM e ativar a opção "Use Dynamic Memory for this virtual machine". Habilitando essa opção, a memória RAM disponibilizada para a máquina virtual não é fixa, podendo aumentar e diminuir consoante a necessidade dos recursos.

A memória dinâmica é aconselhável nas máquinas que executem aplicações em que pode ser necessária temporariamente memória adicional. No caso de aplicações que precisem de utilizar determinada quantidade de memória continuamente, ou que necessitem de controlo total sob a alocação de memória à VM em que são executadas, deve ser utilizada memória estática, como é no caso do Microsoft Exchange.

("Does Exchange play well with Hyper-V Dynamic Memory? (So many questions. So little time. Part 21.) | Full of I.T.," 2012)

5. Definir a switch virtual apropriada à máquina virtual a ser criada(neste caso todas as *VM'S* exceto o Routing, irão utilizar apenas switches internas).
6. Criar um novo disco virtual, definir um nome, localização e tamanho(36GB por máquina, tendo em conta o limite de 220GB).

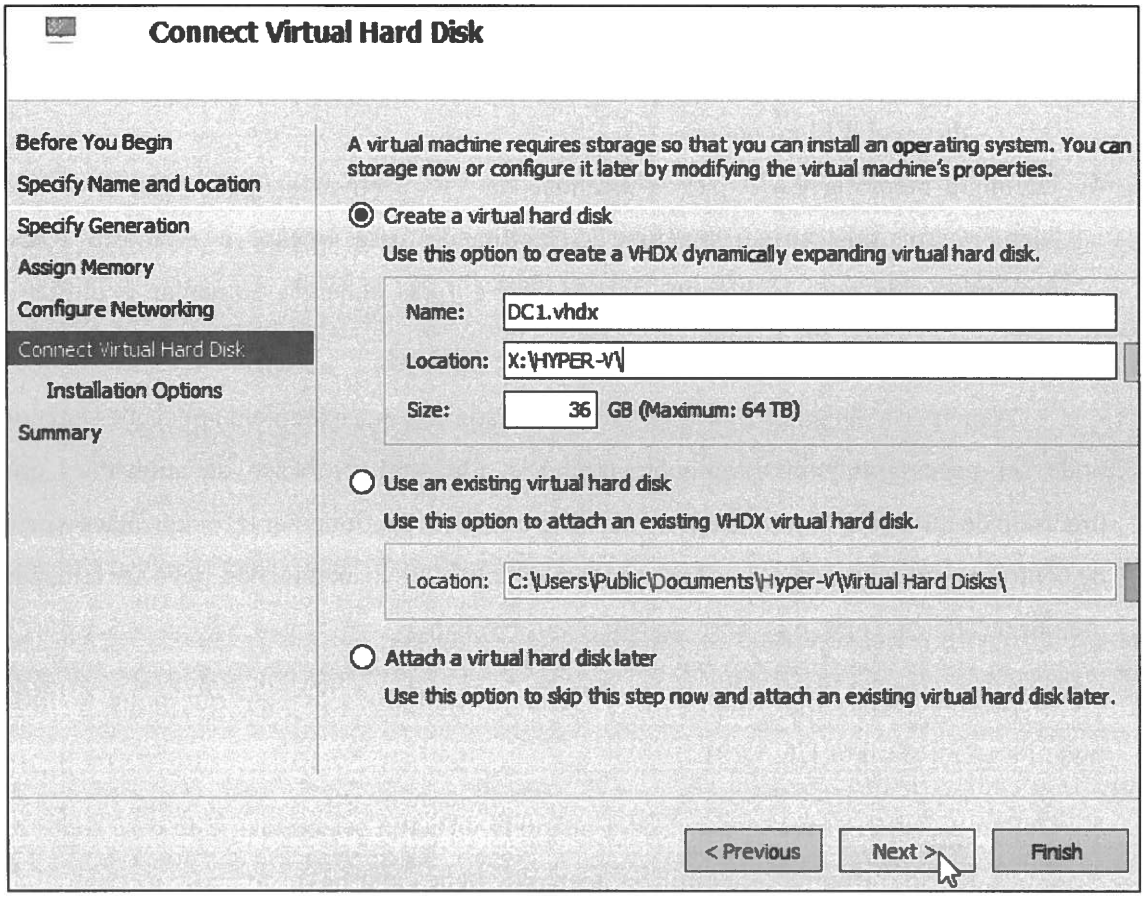


Figura 18 - Criação de máquina virtual

7. Nesta etapa, pode ser definida a media de instalação para o sistema operativo. Selecionar a opção "Install an operating system from a bootable image file" e o ISO do sistema operativo pretendido. Nos servidores virtuais foi instalado o Windows Server 2012 R2, enquanto que na máquina cliente foi instalado o Windows 8.1.
8. Foram repetidos os passos anteriores até todas as máquinas virtuais estarem criadas, e no Hyper-V.
9. Antes de iniciar as máquinas, aceder às *Settings* de cada uma no Hyper-V, e nos "Integration Services", ativar a opção "Guest services". Este serviço permite a transferência de ficheiros entre o *Host* e as *VM's*, através do Hyper-V Virtual Machine Bus(VMBus), ou seja, sem necessitar de ligações de rede virtuais. Isto pode ser realizado através de comandos em Powershell.
10. No Hyper-V, iniciar cada máquina virtual e instalar os respetivos sistemas operativos como normalmente seria feito(no caso do WS2012 R2 instalar a versão com GUI).

Após iniciar sessão com conta de Administrador (Administrator – P@ssw0rd), alterar o nome da máquina nas propriedades de sistema e configurar endereços IP estáticos, e o DNS com o endereço IP do DC1 (fazer isto em todas as máquinas virtuais servidor).

3.1.3 DC1

Um role é um conjunto de programas que quando instalados e apropriadamente configurados, permitem a uma máquina, desempenhar determinadas funções para múltiplos utilizadores ou computadores na rede. ("Roles, Role Services, and Features," n.d.)

Por defeito, o único role instalado no Windows Server 2012 R2 é o "File and Storage Services". Este, oferece a funcionalidade básica de gestão de armazenamento.

Esta máquina virtual foi configurada para servir de controlador de domínio principal, com serviços de DNS e Global Catalog, e como entidade certificadora.

3.1.3.1 Active Directory: Domain Services

O AD:DS é um *role* fundamental para trabalhar com redes de domínio em Windows Server. Inclui uma série de serviços que fornecem armazenamento seguro, estruturado e hierárquico de objetos na rede tais como utilizadores, computadores, impressoras, etc. ("Active Directory Domain Services Overview," 2013)

Para estabelecer um domínio, é necessário efetuar a instalação desta *role* no "Server Manager – Manage – Add Roles and Features". Após a sua instalação, clicar em "Promote this server to a domain controller" para proceder à sua configuração como controlador de domínio.

1. Selecionar a opção "Add a new forest" e definir um *Root domain name*(PROJETOGLOBAL.LOCAL).
2. Deixar o *Forest functional level* e o *Domain functional level* com os valores padrão (WS2012 R2). Este nível funcional define quais os sistemas operativos que podem ser adicionados como controladores de domínio ao nível do domínio ou da floresta, para além das funcionalidades disponíveis (versões anteriores têm menos funcionalidades). Nesta etapa também tem de ser definida uma *password* para o *DSRM*. O *Directory Services Restore Mode* é uma opção de arranque em *safe mode* para controladores de domínio que permite reparar ou recuperar uma base de dados do Active Directory.
3. Deixar as etapas seguintes com os valores padrão e iniciar a promoção clicando *Install*.

A partir deste momento todas as máquinas virtuais deverão ser adicionadas ao domínio "PROJETOGLOBAL.LOCAL".

3.1.3.1.1 Users and Computers

Este *snap-in* permite gerir vários tipos de objetos tais como utilizadores, grupos, computadores, domínios e unidades organizacionais.

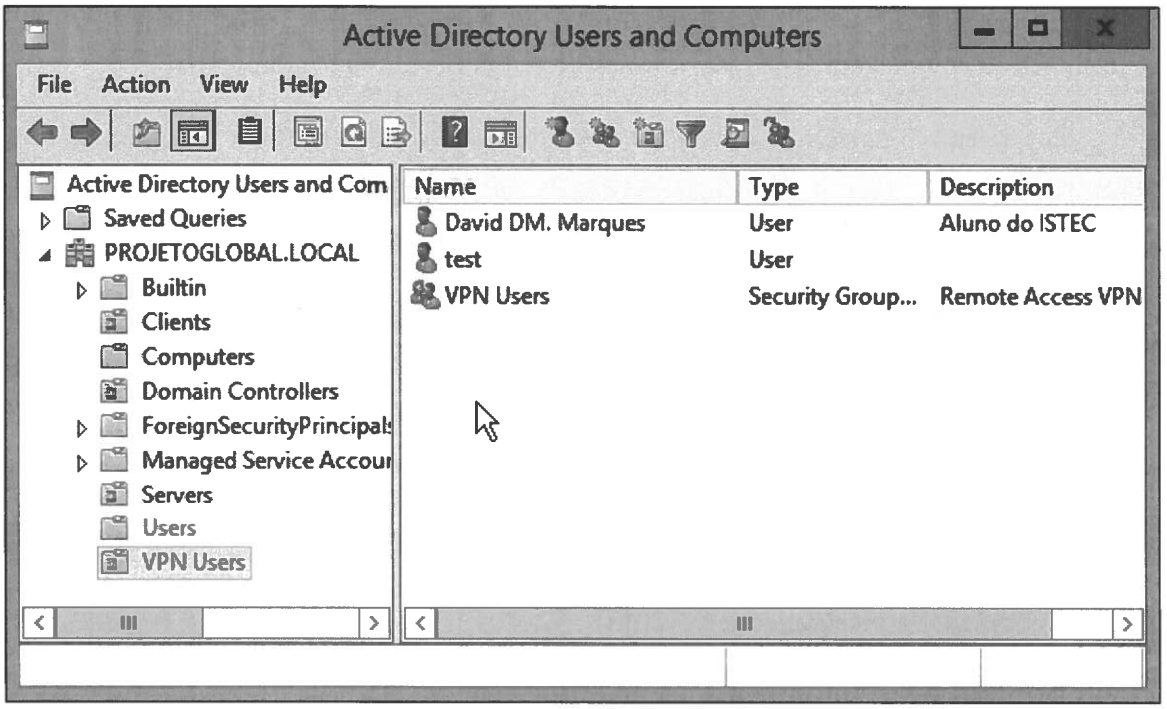


Figura 19 - Users and Computers

Neste caso foram aqui criadas 3 unidades organizacionais (*containers* de objetos) para uma melhor organização.

1. *Right-click* no domínio – *New* – *Organizational Unit*

A primeira contém objetos de servidor, a segunda de cliente, e a terceira utilizadores destinados a acesso por *VPN*. Foram adicionados também 4 grupos de segurança, "VPN Users", "Domain Users", "Domain Servers" e "Domain Computers".

O primeiro foi usado para controlo de acessos remotos, que terá permissões para efetuar a ligação por *VPN*.

O segundo foi usado para conter novos utilizadores no domínio, com permissões de requisição para um certificado de cliente;

O terceiro e o quarto foram usados para conter máquinas servidor e cliente do domínio, respetivamente.

2. Adicionar um novo utilizador dentro da unidade organizacional "VPN Users" (David Marques) e aceder às suas propriedades.
3. Configurações por aba:

General – Adicionar descrição e e-mail;

Member Of - Adicionar o grupo "VPN Users";

Profile – Existem 3 tipos de perfis de utilizador.

- Local: Perfil automaticamente criado na primeira vez que um utilizador faz *login* num computador, e é armazenado localmente.

- Roaming: Criado por um administrador e armazenado num servidor na rede. Este perfil permanece disponível ao utilizador em qualquer máquina na rede, e é atualizado automaticamente.

- Mandatory: Tal como um perfil roaming, são armazenados e obtidos através da rede, mas não são atualizados com alterações realizadas pelo utilizador. Apenas o administrador pode fazer alterações.

É de notar que as configurações feitas ao nível da group policy, têm precedência sobre configurações de utilizador.

("Introduction to Configuration and Management," n.d.)

Aqui foi definido um *roaming* profile e uma *home folder*, ambos armazenados na máquina virtual "Routing".

No *profile path*, é boa prática usar uma variável (%username%), pois devolve automaticamente o nome de utilizador, para evitar que ocorram enganos nos nomes.

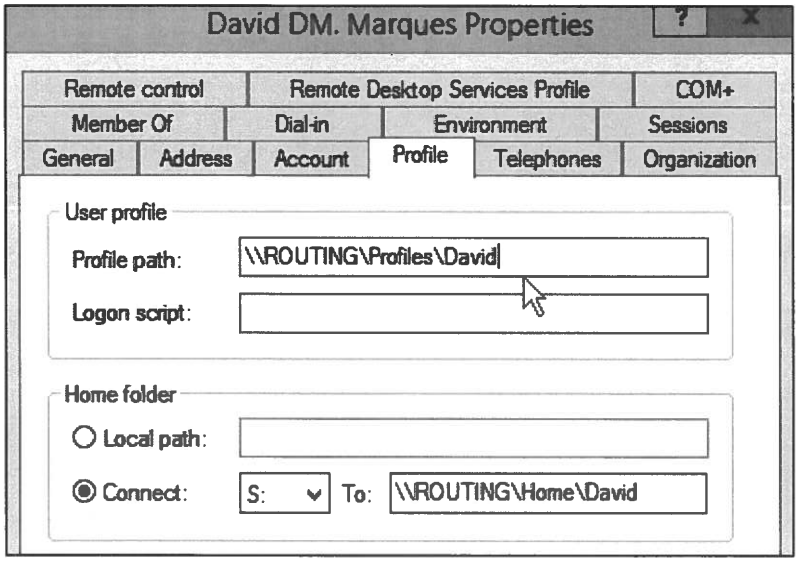


Figura 20 - User profile

Dial-in – Selecionar a opção "Control access through NPS Network Policy. Desta forma, o acesso à rede por parte do utilizador, é controlado pelo servidor NPS.

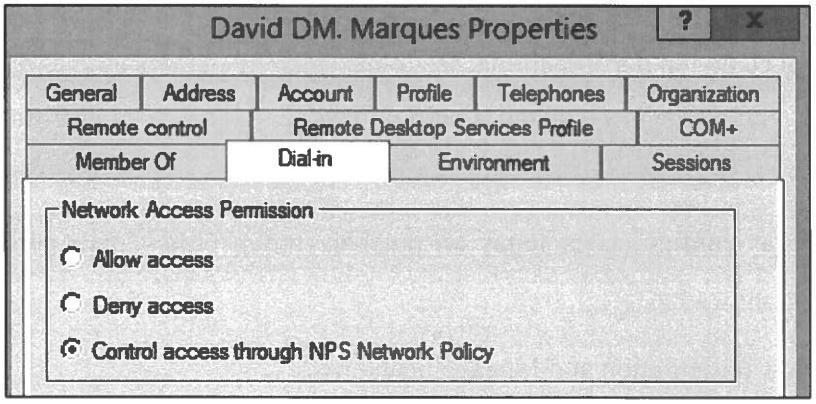


Figura 21 - User Dial-in

3.1.3.1.2 Sites and Services

Este *snap-in* permite configurar e gerir a replicação de dados entre *sites* que façam parte de uma floresta no AD:DS.

Um *site* no AD:DS, representa a estrutura física, ou topologia de uma rede. O AD:DS utiliza a informação dessa topologia, armazenada na diretoria como *sites, subnets*, e objetos de *site link*, para criar a mais eficiente topologia de replicação. Essa topologia consiste no conjunto de objetos de ligação que permitem a replicação *inbound* de um *DC* de origem, para um *DC* de

destino. O *Knowledge Consistency Checker(KCC)*, cria esses objetos de ligação automaticamente em ambos os *DC's*. ("Understanding Sites, Subnets, and Site Links," 2012)

Quando se instala os AD:DS, um *site* é criado automaticamente, com o nome "Default-First-Site-Name". O nome desse *site* deve ser alterado para um nome intuitivo, caso se pretenda utilizar múltiplos *sites*. Para os propósitos deste projeto, foram utilizados 2 *sites*, de forma a simular 2 estruturas físicas ou topologias de rede diferentes.

O *container* "Servers" dentro de um *site* neste contexto, refere-se exclusivamente a controladores de domínio, e não a qualquer tipo de servidor.

Os passos seguintes foram executados depois da máquina virtual DC2 ter sido promovida a controlador de domínio secundário, na mesma floresta.

1. *Right-click* em "Sites – New Site", dar um nome ao *Site* e selecionar o objeto *Site Link*, a que este novo *Site* irá pertencer. Por defeito é criado um objeto de *Site Link* que pode ser selecionado aqui (o seu nome pode ser posteriormente alterado).

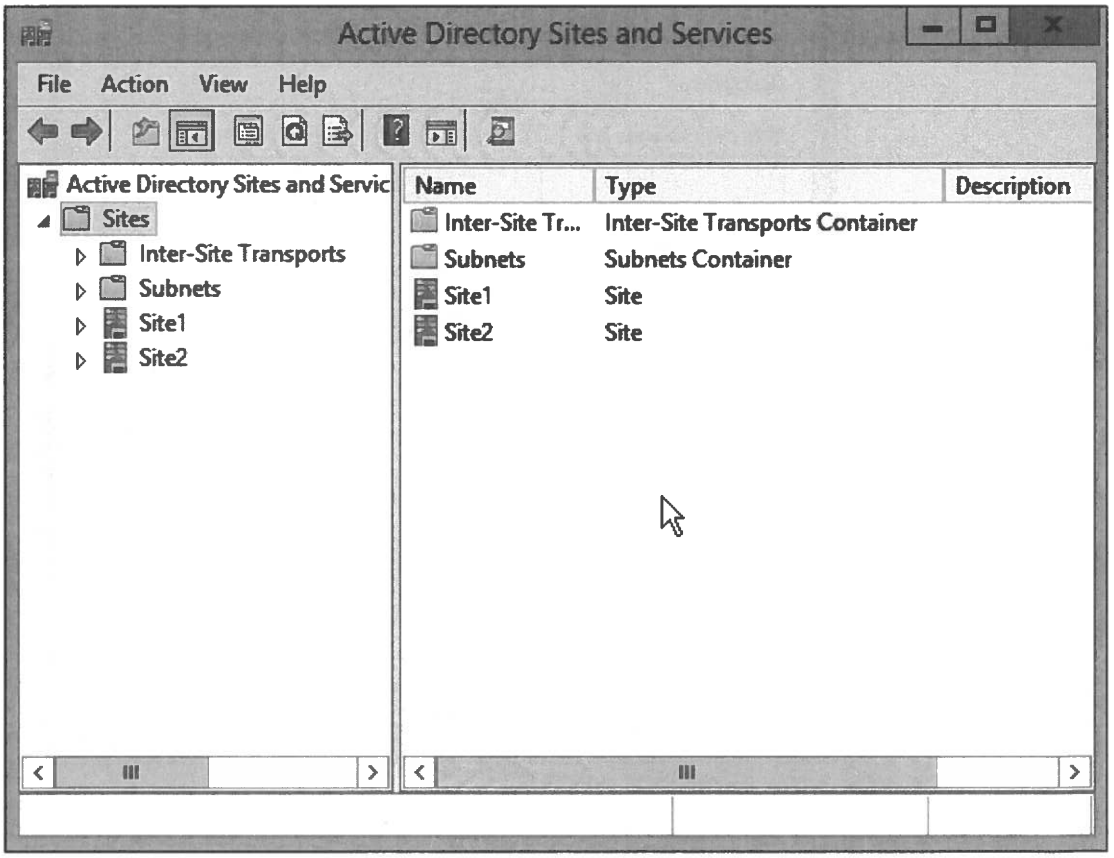


Figura 22 - Sites and Services

2. Num cenário com múltiplos *Sites* como é o caso, tem de ser associado a cada um deles uma *subnet* diferente. *Right-click* em "Subnets – New Subnet", especificar o prefixo de rede pretendido, e escolher o objeto de *Site* a que corresponde essa *subnet*. Este procedimento deve ser realizado para cada *Site* existente.
3. Nas propriedades da *Site Link,* em "Inter-Site Transports – IP", verificar que ambos os *Sites* se encontram associados ao mesmo objeto de ligação. Nesta janela pode também ser definido o "custo" desta ligação e o intervalo de tempo entre cada tentativa de replicação de dados entre os *Sites* pertencentes a essa ligação. O "custo" neste caso é usado para definir qual o "caminho" prioritário para replicação entre *sites* e deve ser calculado com base em vários fatores, tais como a largura de banda da ligação, disponibilidade, latência e custo monetário.

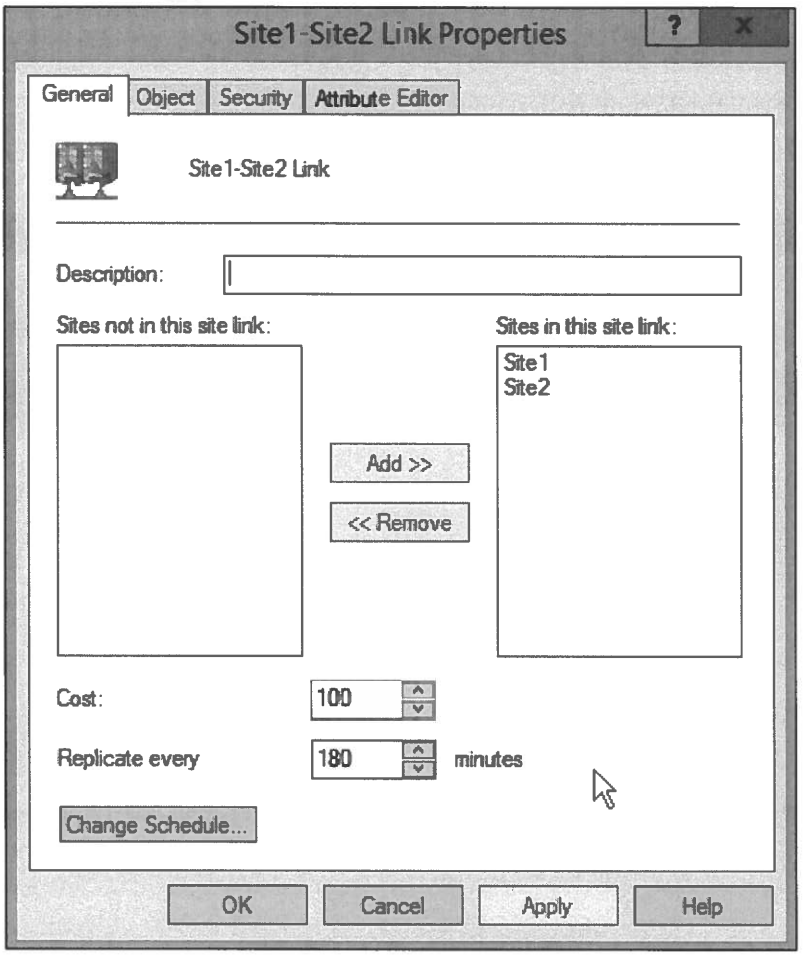


Figura 23 - Site link Properties

Em "Change Schedule", pode ser configurada em maior pormenor a periodicidade das tarefas de replicação.

4. No objeto *Site 1,* arrastar o DC2 para o objeto *Site 2*.

Dentro de cada servidor, existe um objeto "NTDS Settings". Este, representa o controlador de domínio no sistema de replicação. Estes objetos armazenam objetos de ligação, que tornam a replicação entre dois ou mais controladores de domínio possível.

Podem ser usados para verificar o funcionamento da replicação, ou para ativar ou desativar o *global catalog* num servidor. ("Overview of Active Directory Sites and Services," 2012)

3.1.3.2 Active Directory: Certificate Services

Os serviços desta role permitem emitir e gerir certificados digitais usados em sistemas de segurança com tecnologias de chave pública. Esses certificados podem ser usados para encriptar documentos, assinar mensagens, autenticar computadores, utilizadores ou contas de dispositivo numa rede.

Asseguram:

- Confidencialidade através de encriptação;

- Integridade graças a assinaturas digitais;

- Autenticação, ao associar chaves de certificado a um computador, utilizador ou dispositivo.

("Active Directory Certificate Services Overview," 2013)

Neste projeto foram emitidos 4 certificados. Um para o controlador de domínio, e três com o propósito de autenticação segura de utilizadores remotos na rede, através do NPS.

1. No Server Manager, instalar a role Active Directory: Certificate Services, incluindo as *management tools*.
2. Nos *roles* de serviço adicionais, deixar selecionado apenas a opção "Certificate Authority" e inicar a instalação.
3. Após a instalação clicar em "Configure AD:CS on the destination server".
4. Verificar que as credenciais estão corretas (têm de ser de um administrador do domínio).
5. Selecionar o *role service* "Certificate Authority".

6. Selecionar a opção "Enterprise CA".

   Existem 2 tipos de instalações de uma autoridade certificadora, *Enterprise* e *Standalone*. Na primeira, a autoridade certificadora tem de pertencer a um domínio e permanecer *online* para poder emitir certificados ou políticas. No segundo tipo de autoridade certificadora, esta pode ser configurada fora de um domínio e *offline*. Esta opção é a mais segura, pois não é vulnerável a ataques externos.

7. Selecionar a opção "Root CA".

   Existem 2 tipos de *CA's*. *Root* e *Subordinate*. O primeiro, é o tipo de autoridade certificadora de maior confiança na infraestrutura de chave pública de uma organização e é a partir desta autoridade que originam os certificados, logo deverá ter políticas de segurança mais rigorosas. Tipicamente num cenário empresarial, a *Root CA* não é usada para emitir certificados para utilizadores, mas sim para autoridades certificadoras intermédias que desempenham essas funções, chamadas *Subordinate CA's*.
   ("Types of Certification Authorities," n.d.)

8. Selecionar a opção para criar uma nova chave privada e deixar as configurações criptográficas nos seus valores padrão.
9. Dar um nome à *CA* e especificar a sua validade.
10. Deixar os outros passos com valores padrão e finalizar a configuração.
11. Na linha de comandos digitar "MMC" para abrir a consola de gestão de *snap-ins*.
12. "File – Add/Remove Snap-in", adicionar os *snap-ins* "Certificates" para "Computer account – Local computer e "Certificate Authority" para "Local Computer".
13. Aceder à primeira *snap-in*, onde podem ser consultados todos os certificados instalados na máquina.

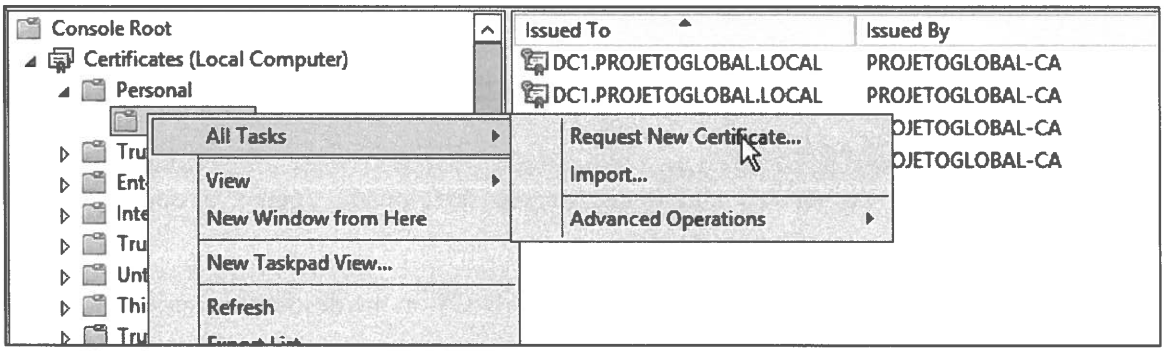


Figura 24 - Request New Certificate

14. Expandindo *Certificates* e fazendo *right-click* em "Personal – All tasks – Request New Certificate", é pedida a emissão de um novo certificado.
15. Efetuar o pedido de um certificado do tipo *Domain Controller*.

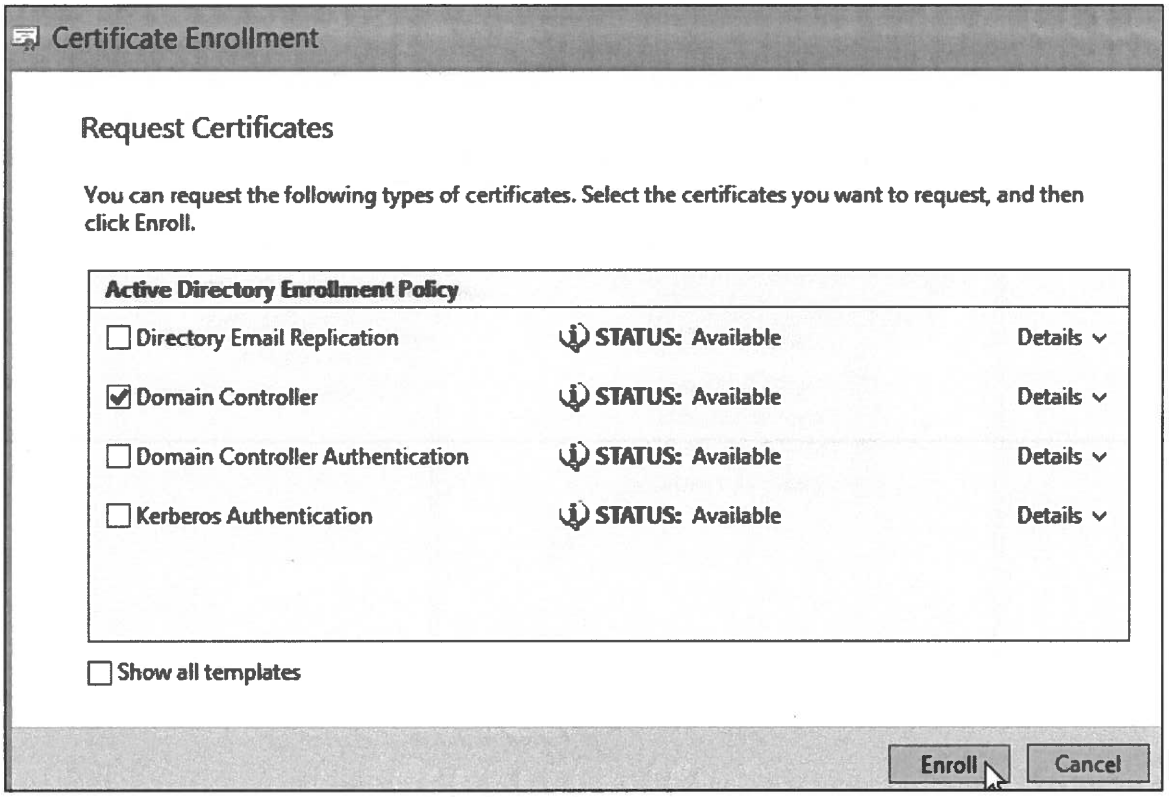


Figura 25 - Certificate Enrollment

Dentro do *container Certificates*, verifica-se que se encontra emitido o novo certificado para o controlador de domínio.

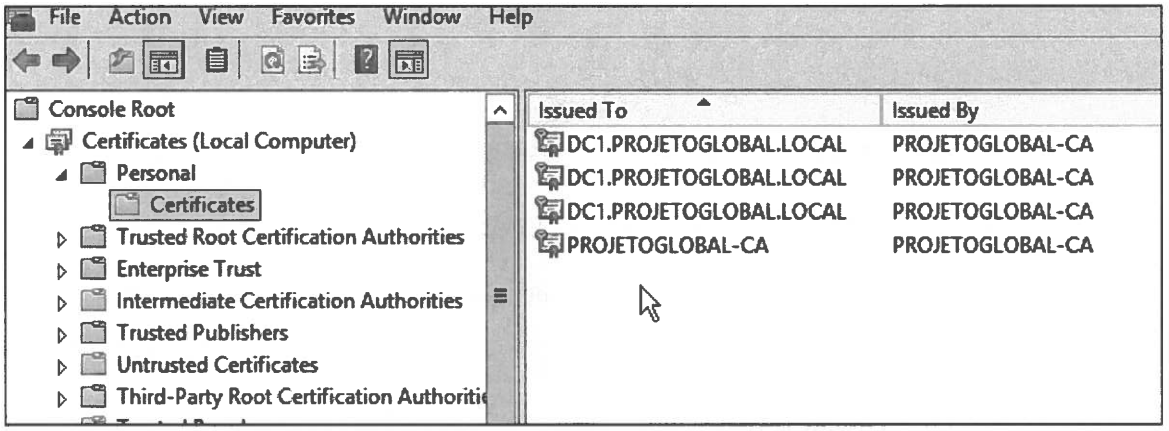


Figura 26 - Issued Certificates

16. Aceder à segunda *snap-in*, onde podem ser consultados e geridos os certificados emitidos, pendentes, falhados, ou revogados pela autoridade certificadora, bem como os modelos de certificado existentes.

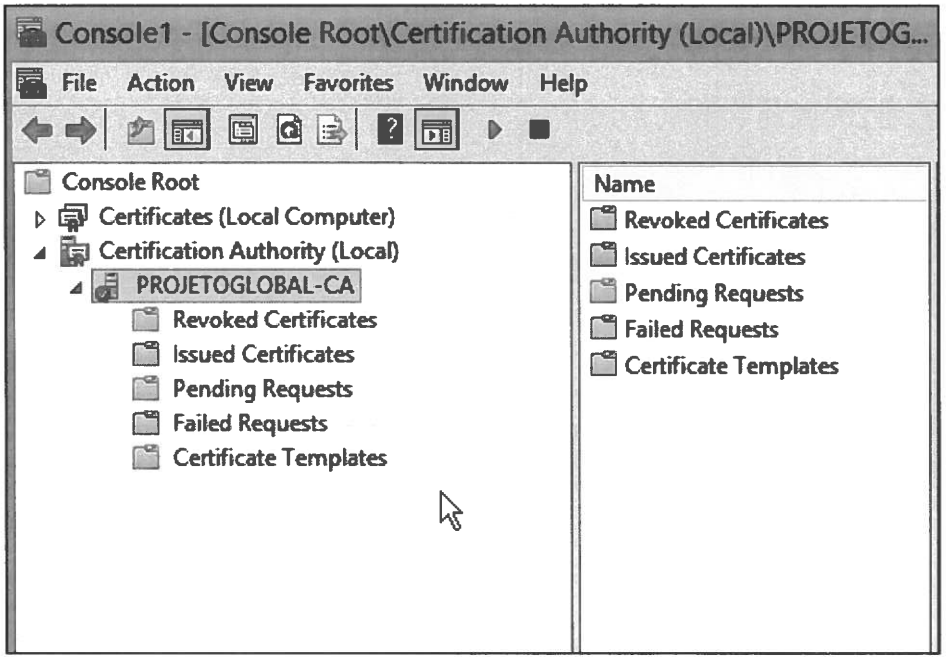


Figura 27 - Certification Authority

No *container* "Certificate Templates", estão listados os modelos de certificado que podem ser emitidos no momento por aquela *CA*. Estes modelos são utilizados por razões de segurança para que determinados utilizadores ou computadores na rede, tenham permissões de acordo com o propósito do certificado emitido para eles. ("Create a New Certificate Template," n.d.)

Neste cenário foram criados 3 certificados:

Domain Client Authentication - Autenticação de máquinas cliente;

Domain Server Authentication – Autenticação de máquina servidor;

Domain User - Autenticação de utilizadores na rede.

17. Para isso, foram duplicados 3 modelos de certificado já existentes. *Right-click* em "Certificate Templates – Manage", para abrir a janela de gestão de modelos.
18. *Right-click* no modelo "Workstation Authentication – Duplicate Template".

19. Na janela de propriedades do modelo existem muitas configurações que podem ser feitas relativas ao modelo de certificado, mas neste cenário apenas algumas foram realizadas.

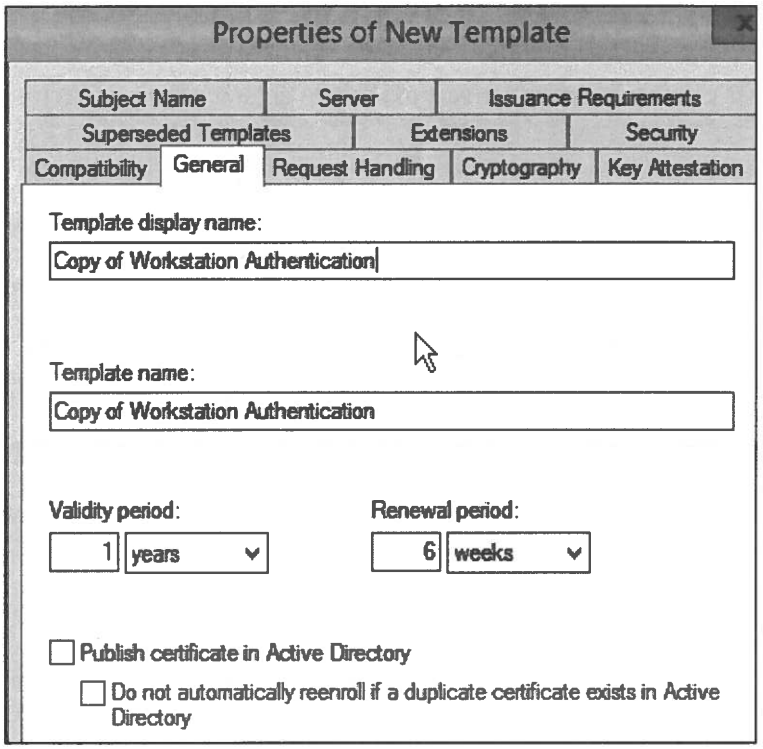


Figura 28 - Duplicate Template

Configurações comuns aos 3 modelos:

Compatibility – Foram alteradas as definições de compatibilidade para o nível do Windows Server 2012 R2 na *CA*, e para o nível do Windows 8 / Windows Server 2012 em *Certificate Recipient*;

General - Foi dado um nome ao modelo, e definida a sua validade.

Security – Foi adicionado um dos grupos de segurança criados anteriormente no AD:UC, correspondente ao modelo em questão, atribuindo permissões de *Enroll.*

Configurações do duplicado de "Computer":

Issuance Requirements - Ativar a opção "CA certificate manager approval". Esta opção faz com que a emissão de um certificado com este modelo, dependa da aprovação do administrador. Sendo este um modelo que irá ser usado por um servidor responsável por

autenticar clientes remotos na rede, por razões de segurança não é indicado que este seja emitido sem verificação.

Configurações do duplicado de "User":

General - Habilitar a opção "Publish certificate in Active Directory", para que quando um utilizador obtiver este certificado, ele seja associado ao seu objeto no Active Directory.

Subject Name – Habilitar as opções "Include e-mail name in subject name", "e-mail name" e adicionar esses dados no perfil do utilizador no AD. Estas opções permitem que este certificado seja utilizado para assinar e-mails.

20. Para que novos modelos possam ser emitidos pela autoridade certificadora, têm primeiro de ser ativados. No *snap-in* da *CA*, *right-click* em "Certificate Templates – New – Certificate Template to Issue", e selecionar os 3 modelos criados anteriormente.

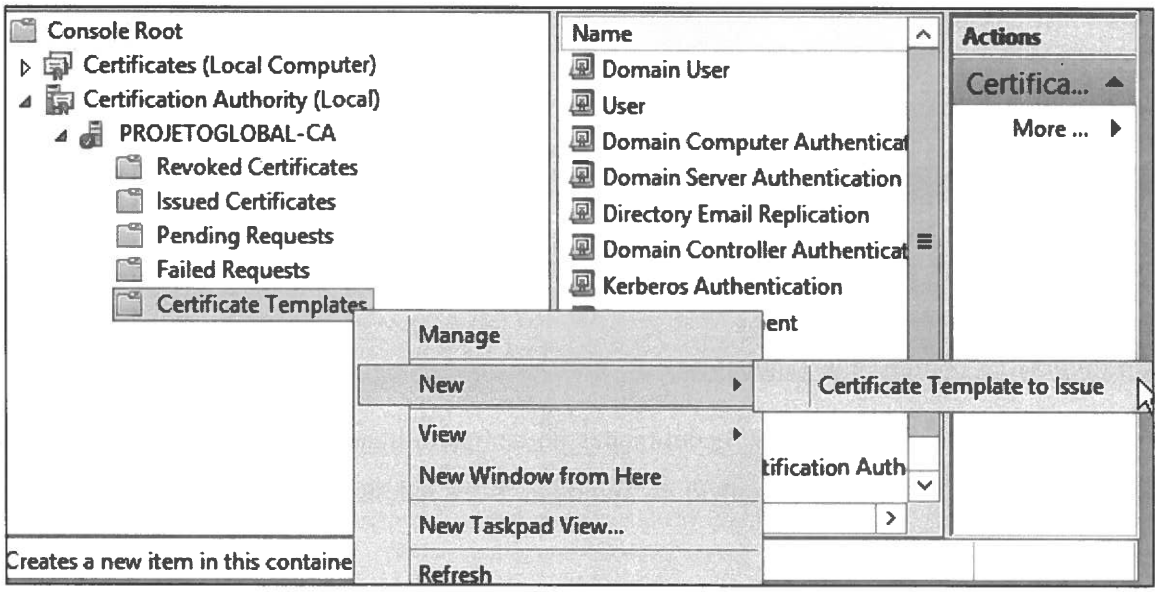


Figura 29 - Issue New Certificate Templates

3.1.3.3 Group Policy

O Group Policy é um componente muito poderoso, que deve ser explorado ao máximo num cenário empresarial. Para os propósitos deste laboratório foram apenas definidas algumas políticas e configurações.

1. No Server Manager, aceder a "Tools - Group Policy Management".

2. No separador do domínio, *right-click* em "Default Domain Policy – Edit", para editar este *Group Policy Object*, que foi criado automaticamente no momento da criação do domínio. Este *GPO* é aplicado a todo o domínio.

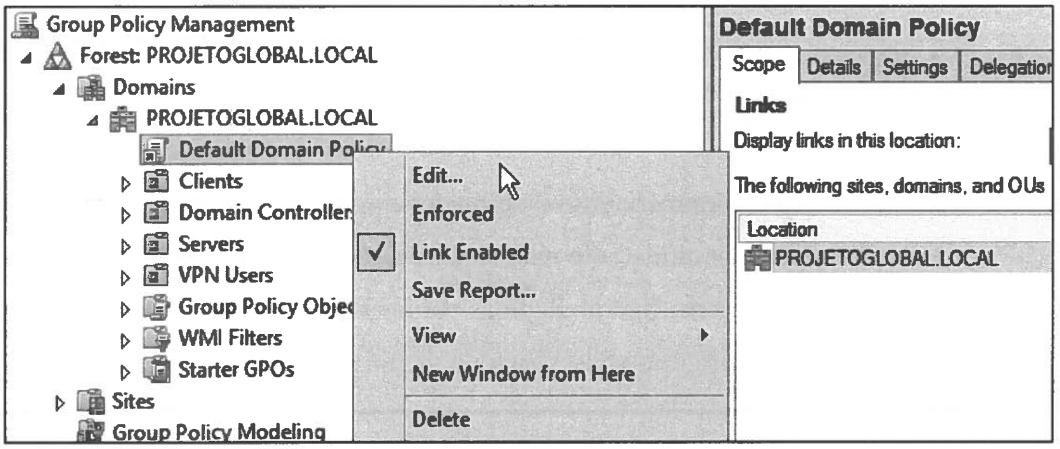


Figura 30 - Edit GPO

3. Uma das configurações básicas, mas que deve ser feita em qualquer domínio, é a nível da segurança de contas. Essas políticas podem ser definidas em "Computer Configuration – Windows Settings – Security Settings – Account Policies".

Password Policy:

- Enforce password history – "24 passwords remembered" (o número de *passwords* únicas que têm de ser definidas a uma conta até que uma possa voltar a ser reutilizada);
- Maximum password age – "50 days" (tempo máximo que uma *password* pode ser utilizada até ter obrigatoriamente de ser alterada);
- Minimum password age – "1 day" (tempo mínimo que uma *password* tem de ser utilizada até que seja permitido alterá-la);
- Minimum password length – "7 characters" (número mínimo de caracteres que uma *password* pode ter);
- Password must meet complexity requirements - "Enabled" (a *password* definida tem de cumprir com uma série de requisitos);
- Store passwords using reversible encryption – "Disabled" (armazenamento de *passwords* com a utilização de encriptação reversível. Não é aconselhável na maior parte dos casos, pois são armazenadas em plaintext).

Account Lockout Policy:

- Account lockout duration – "30 minutes" (tempo que uma conta permanece bloqueada;
- Account lockout threshold – "5 invalid logon attempts" (número de tentativas falhadas que causam que uma conta seja bloqueada);
- Resrt account lockout after – "30 minutes" (tempo para o *reset* do contador de tentativas de *logon* falhadas).

4. Na etapa anterior, foram criados e habilitados novos modelos de certificado. Para que um certificado seja emitido automaticamente a um utilizador quando este é criado, é necessário habilitar e configurar algumas políticas a nível da group policy.
   Em "Computer Configuration – Windows Settings – Security Settings – Public Key Policies":

- "Certificate Services Client – Certificate Enrollment Policy" – Habilitar, e em "Properties", ativar as opções "Enable for automatic enrollment and renewal", e "require strong validation during enrollment";

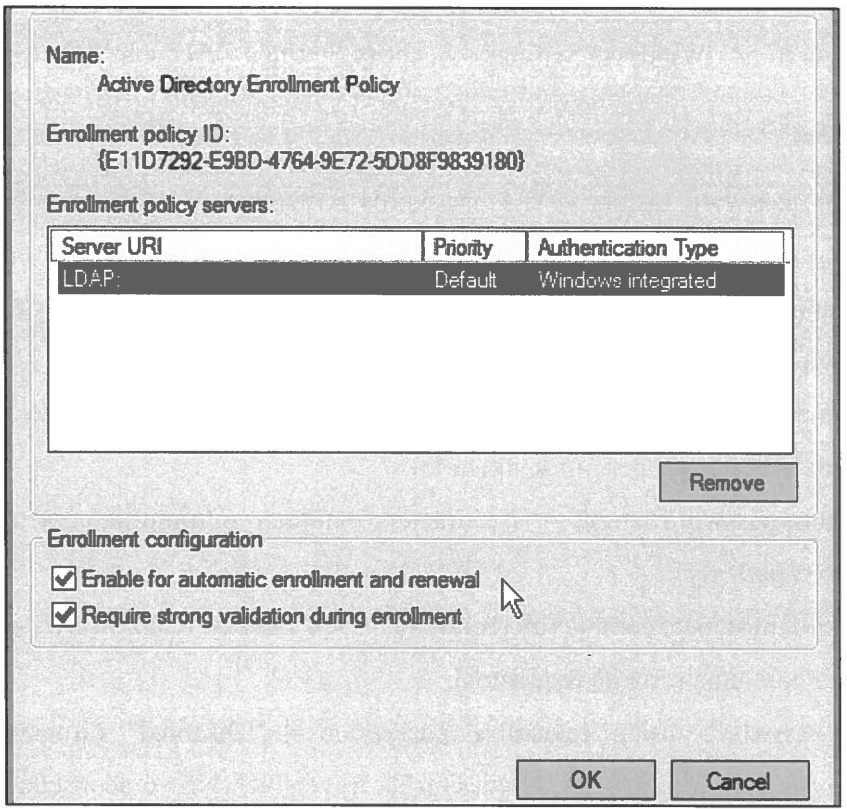


Figura 31 - Certificate Enrollment Policy

- "Certificate Services Client – Auto-Enrollment" – Habilitar, e ativar as opções "renew expired certificates, update pending certificates, and remove revoked certificates", e "update certificates that use certificate templates".

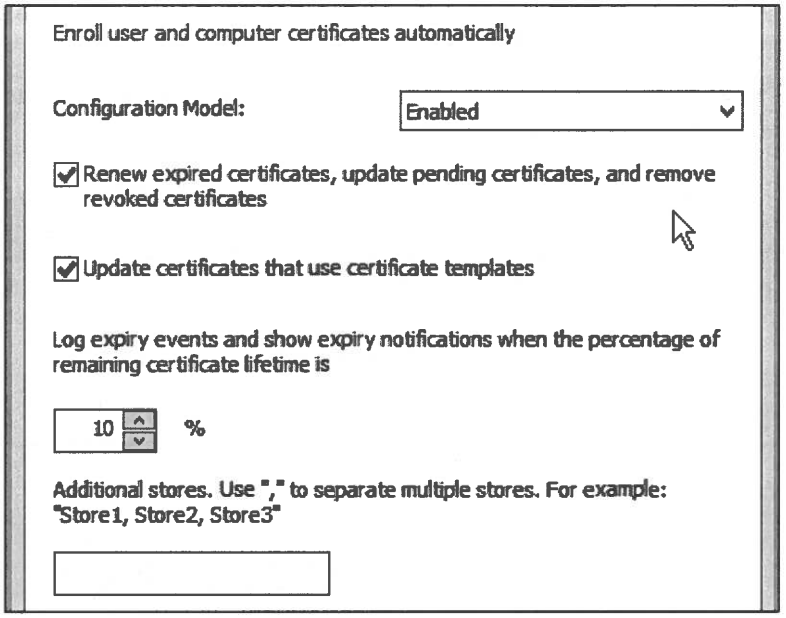


Figura 32 - Auto-Enrollment

3.1.4 DC2

Esta máquina virtual foi configurada como controlador de domínio secundário, para propósitos de replicação.

Replicação é o processo através do qual mudanças feitas num controlador de domínio, são sincronizadas sistematicamente com todos os outros controladores de domínio no domínio ou floresta, que armazena cópias da mesma informação.("Active Directory Replication," n.d.)

1. Instalar a *role* Active Directory Domain Services e iniciar o *wizard* de promoção a controlador de domínio.
2. Desta vez, sendo que já existe uma floresta com um controlador de domínio, selecionar a primeira opção, para adicionar este DC a um domínio existente. Inserir o nome do domínio e as credenciais de administrador.
3. Deixar as capacidades DNS e GC ativas, escolher o *site* do AD em que este DC se irá inserir (neste caso no Site2) e especificar as credenciais DSRM.
4. Nas "Additional Options" definir para replicar do DC1.

3.1.4.1 Replicação manual

Para iniciar o processo de replicação manualmente, tem de ser estabelecida uma ligação entre os DC's.

1. No Active Directory Sites and Services expandir o "Site2 – Servers – DC2". *Right-click* em "NTDS Settings – All tasks – Check replication topology". Fazendo *refresh*, o objeto de ligação ao DC1 já existe na lista.
2. Expandir o "Site1 – Servers – DC1". *Right-click em* "NTDS Settings – All tasks – Replicate configuration to the selected DC". Fazer também "check replication topology".
3. De volta ao DC2, *right-click* no objeto de ligação e selecionar "Replicate now" para iniciar o processo de replicação a partir do DC1.

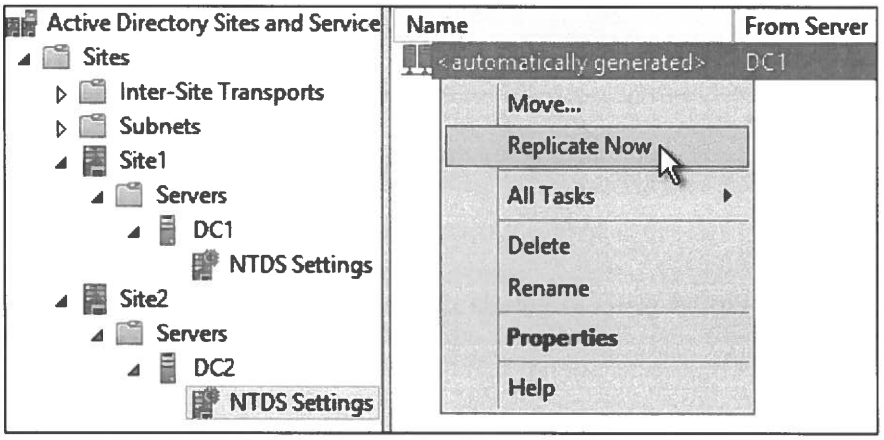


Figura 33 - Replicate

3.1.5 ROUTING

Esta máquina virtual foi configurada para servir como router, servidor de acesso remoto, *storage* de utilizadores e exchange. É o único servidor no laboratório que tem uma *NIC* com acesso direto ao exterior, através da qual é fornecido o acesso à Internet às outras máquinas na rede. É também responsável pela gestão do servidor interno de e-mail.

3.1.5.1 Remote Access

Os serviços desta role permitem configurar o acesso remoto à rede por *VPN* ou *dial-up*, *routing*(*LAN* e *NAT*).

1. No Server Manager, selecionar para instalar a *role* "Remote Access".
2. Nos serviços a instalar selecionar as opções "Routing", e "DirectAccess and VPN(RAS)". O *role* de *Web Server(IIS)* também tem de ser instalado, apesar de não ser usado neste laboratório.
3. Após a instalação aceder a "Tools – Routing and Remote Access".
4. *Right-click* no servidor – "Configure and Enable Routing and Remote Access".

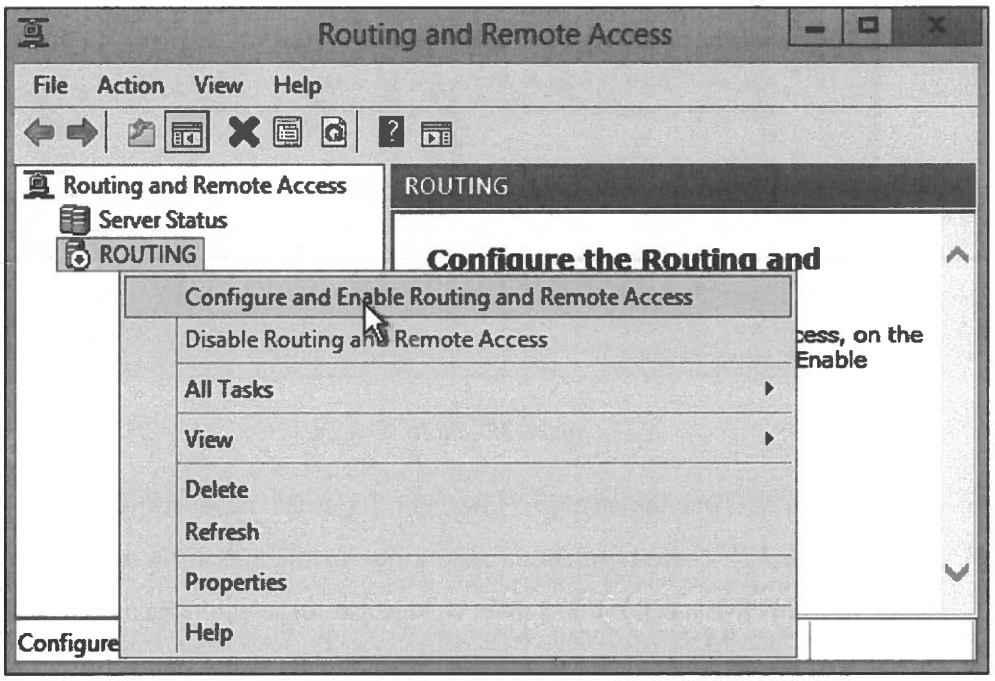


Figura 34 - Configure RRAS

5. Selecionar a opção de VPN e NAT (Network Address Translation). O NAT permite que várias máquinas numa rede, tenham acesso à Internet através de um único endereço IP público.

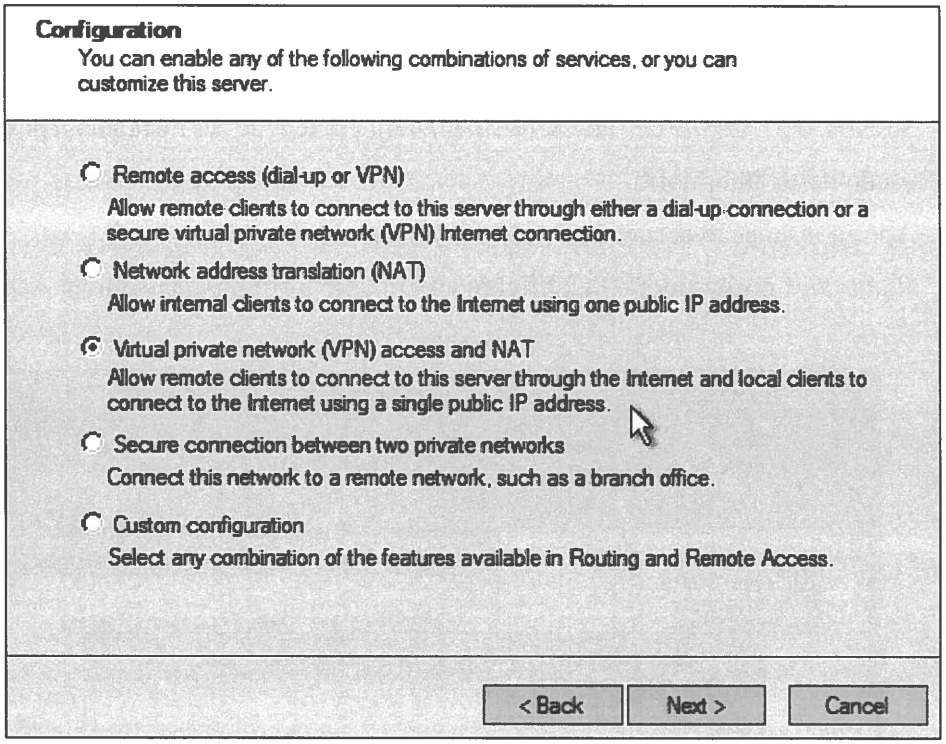


Figura 35 - RRAS Services

6. Selecionar a *NIC* que tem acesso à Internet ("External" neste caso).
7. Selecionar a *NIC* correspondente à rede a que os clientes *VPN* serão atribuídos para propósitos de DNS, acesso *dial-in*,etc. Neste caso foi selecionada a *NIC* "Internal (.0)".
8. Selecionar a segunda opção, para que sejam atribuídos endereços IP aos clientes remotos a partir de uma gama de endereços fixos e definir essa gama.

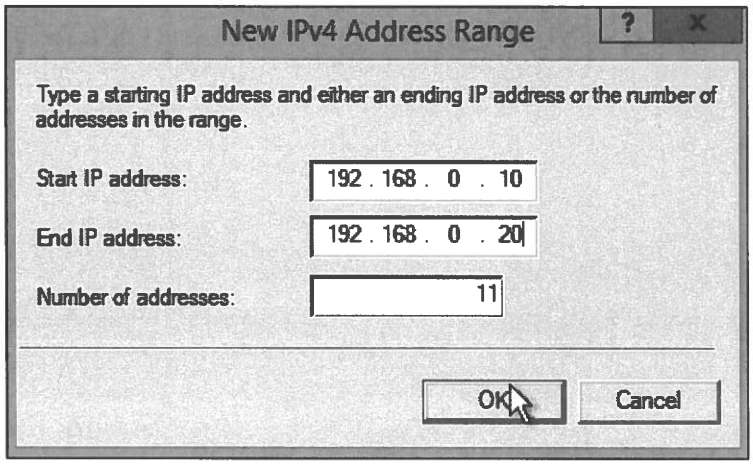


Figura 36 - VPN IPv4 Address Range

9. Selecionar a *interface* interna que irá, através do *NAT* receber acesso à Internet. Neste caso pretende-se que ambas as *interfaces* internas tenham acesso à Internet e neste wizard só pode ser selecionada uma, mas isso pode ser alterado após a configuração inicial.
10. Os pedidos de ligação à rede podem ser autenticados localmente ou endereçados a um servidor *RADIUS(Remote Authentication Dial-In User Service)*. Nesta altura, ainda não foi configurado o servidor *NPS,* que terá esse papel, por isso selecionar a primeira opção.
11. De volta à janela do *RRAS*, expandir a opção "IPv4", *right-click* em "General – New Routing Protocol" e selecionar o protocolo *NAT*. No *NAT,* adicionar uma nova *interface* (a que tem ligação à Internet, neste caso a "External"), definir o seu tipo como "public interface connected to the Internet", e ativar a opção de habilitar o *NAT* nessa *interface*. A partir deste momento todas as máquinas virtuais em ambos os *sites* do AD terão acesso à Internet.

3.1.5.2 Exchange Server 2013

O Exchange fornece a infraestrutura basilar à execução de um sistema de mensagens. Inclui uma base de dados para armazenar dados de e-mails, a infraestrutura para os mover entre vários locais, e os pontos de acesso para que sejam acedidos através de diferentes clientes. Quando utilizado com clientes como o Outlook ou o Outlook Web App, a caixa de correio eletrónico torna-se num ponto de armazenamento e gestão de informações pessoais tais como calendário, contactos, listas de tarefas, e outros ficheiros. Essas informações podem ser partilhadas com outros utilizadores, de forma a colaborarem entre si.

(Elfassy, 2013)

A funcionalidade do Exchange Server é dividida por *roles*. Este conceito, introduzido no Exchange Server 2007, foi criado para que a funcionalidade pudesse ser repartida por vários servidores, de forma a facilitar a escalabilidade.

Nas versões anteriores existiam mais *roles*, mas no caso do Exchange Server 2013, foram consolidados em apenas 2.

1. Mailbox Server Role
2. Client Access Server Role

O primeiro, gere todas as *mailboxes*, pastas públicas, e replicação da base de dados. Para além disso, e ao contrário das versões anteriores do Exchange, inclui também as funcionalidades de *Transport service* e *Unified Messaging*.

O segundo, age como um *thin server* que essencialmente redireciona os pedidos de clientes para servidores de *mailbox*.

Neste cenário, o Exchange foi instalado no servidor virtual "ROUTING" e foram utilizadas apenas 2 *mailboxes* para propósitos de demonstração, que podem comunicar entre si internamente no domínio.

(Elfassy, 2013)

3.1.5.2.1 Instalação

O Exchange Server 2013 pode ser instalado por linha de comandos ou GUI. Neste cenário foi utilizada a GUI.

1. Executar o instalador do Exchange Server 2013 SP1. Na etapa de "server role selection", selecionar ambos os *roles mailbox* e *client access*.

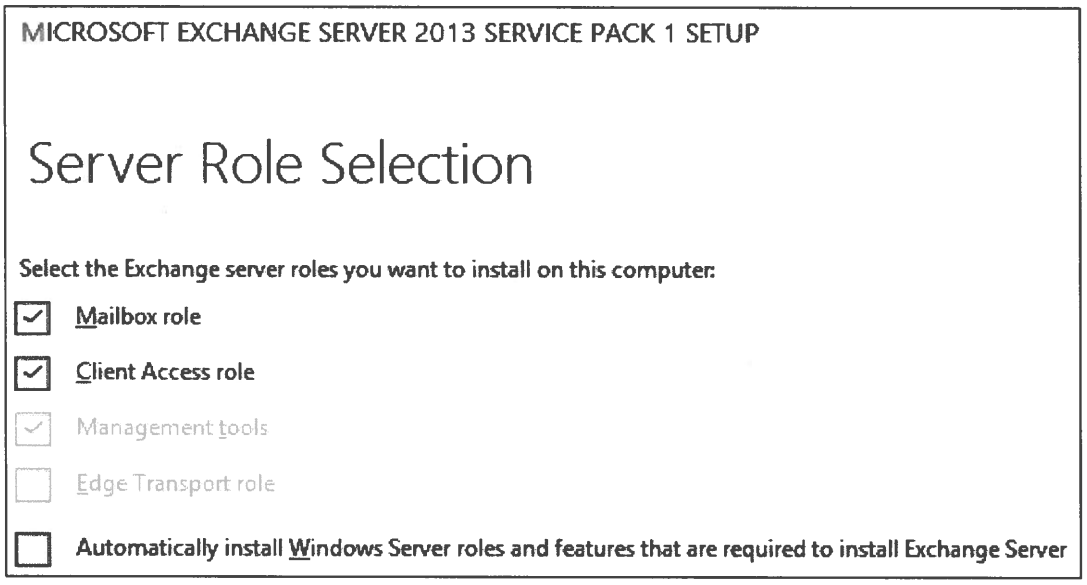


Figura 37 - Exchange Role Selection

Aqui pode ser definido para obter automaticamente os pré-requisitos necessários à instalação do Exchange (*roles* e funcionalidades do Windows Server), mas isso não é recomendado. Em vez disso, esses pré-requisitos devem ser instalados manualmente.

2. Numa linha de comandos Powershell, executar o seguinte comando:

"Install-WindowsFeature AS-HTTP-Activation, Desktop-Experience, NET-Framework-45-Features, RPC-over-HTTP-proxy, RSAT-Clustering, RSAT-Clustering-CmdInterface,RSAT-Clustering-Mgmt, RSAT-Clustering-PowerShell, Web-Mgmt-Console, WASProcess-Model, Web-Asp-Net45, Web-Basic-Auth, Web-Client-Auth, Web-Digest-Auth, Web-Dir-Browsing, Web-Dyn-Compression, Web-Http-Errors, Web-Http-Logging,Web-Http-Redirect, Web-Http-Tracing, Web-ISAPI-Ext, Web-ISAPI-Filter, Web-Lgcy-Mgmt-Console, Web-Metabase, Web-Mgmt-Console, Web-Mgmt-Service, Web-Net-Ext45,Web-Request-Monitor, Web-Server, Web-Stat-Compression, Web-Static-Content, Web-Windows-Auth, Web-WMI, Windows-Identity-Foundation"

Para além disso, deve ser instalado também o Microsoft Unified Communications Managed API 4.0 64-bit, que pode ser obtido no site da Microsoft.

3. Dar um nome à organização, e deixar a proteção contra Malware ativa no passo seguinte.

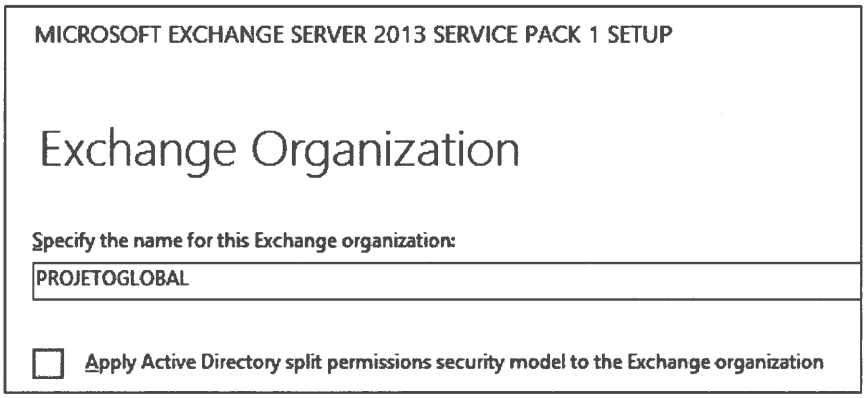


Figura 38 - Exchange Organization Name

4. Na análise de pré-requisitos, verificar que não existem erros, mas é normal aparecerem 2 avisos, a indicar que não foram detetados *roles* de Exchange de versões anteriores na topologia.

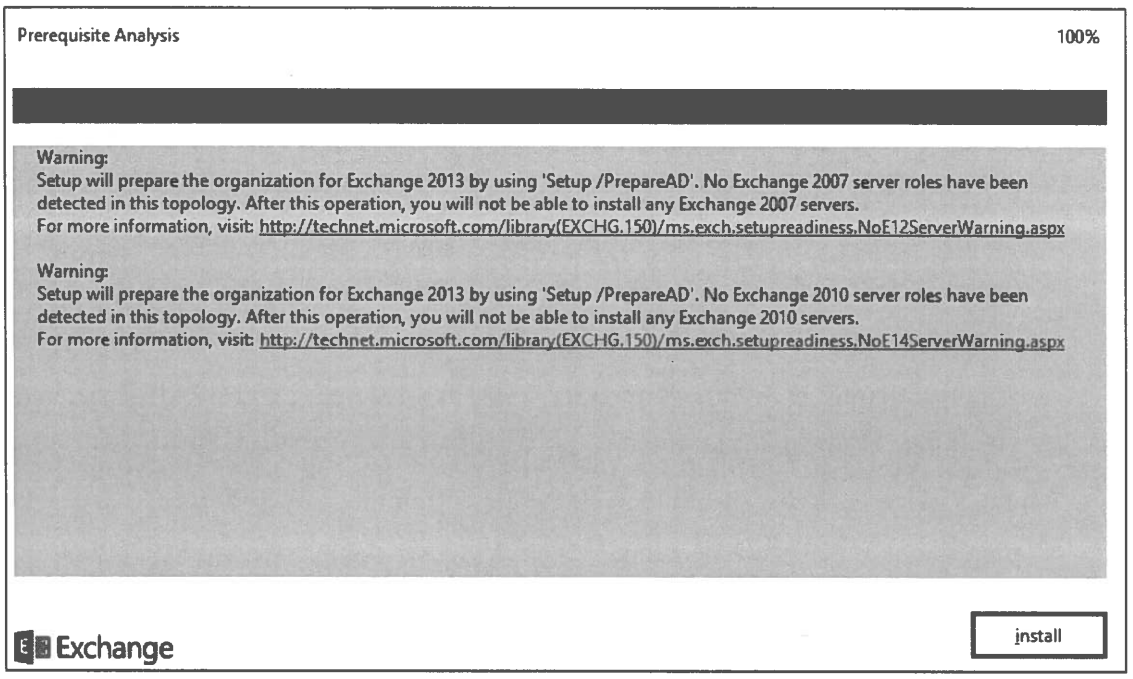


Figura 39 - Exchange Prerequisite Analysis

3.1.5.2.2 Centro de Administração

O Exchange pode ser gerido por linha de comandos com o Exchange Management Shell ou através de uma interface gráfica *web-based*. Neste cenário as configurações foram feitas por GUI, no seu centro de administração, que pode ser acedido através do seguinte link no browser: "https://localhost/ecp".

Figura 40 - Exchange Centro de Administração

Neste *site* podem ser geridas todas as funcionalidades e serviços do Exchange 2013.

Configurações por aba:

Destinatários: Por defeito, é criada uma *mailbox* para o administrador do Exchange. Será criada uma outra para propósitos de teste.

1. No menu superior, aceder a "caixas de correio" e clicar na cruz e adicionar uma nova caixa de correio de utilizador. Neste caso pretende-se adicionar um utilizador já existente no Active Directory por isso selecionar a opção "utilizador existente" e em procurar seleccioná-lo. Caso não existisse, poderia aqui ser criado e adicionado automaticamente ao Active Directory.

Fluxo de correio: Nesta secção, podem ser feitas configurações relativas a todo o fluxo de mensagens que ocorre através da *transport pipeline.* Esta *pipeline*, é uma coleção de serviços, conexões e componentes que trabalham em conjunto para rotear mensagens. ("Mail flow: Exchange 2013 Help," 2013)

Os conetores de receção são responsáveis pelo fluxo *inbound*, enquanto que os conetores de envio são responsáveis pelo fluxo *outbound* de correio eletrónico.

Para comunicações internas (entre servidores Exchange dentro da mesma floresta do AD), não é necessário criar qualquer conetor. Os conetores de receção necessários são automaticamente criados com a instalação das *roles* de *cliente access* ou *mailbox*, e os conetores de envio necessários fazem parte dos serviços de transporte. Estes últimos, chamados de conetores implícitos ou *intra-organization*, são automaticamente disponibilizados e não requerem qualquer gestão.

("Receive connectors," n.d.)

("Send connectors," n.d.)

Para comunicações com o exterior(Internet), é necessário criar conetores, e em particular para receber e-mail do exterior, é necessário configurar num servidor DNS público/externo, um registo que aponte para o endereço IP público do servidor Exchange no domínio. ("Configure DNS records," n.d.)

Servidores: Na aba "servidores" estão listados os servidores que desempenham *roles* do Exchange (neste caso apenas existe 1, que inclui ambas as *roles mailbox* e *client access*).

1. Editar o servidor existente, e em "pesquisas de DNS", definir manualmente o endereço IP do servidor DNS do domínio.

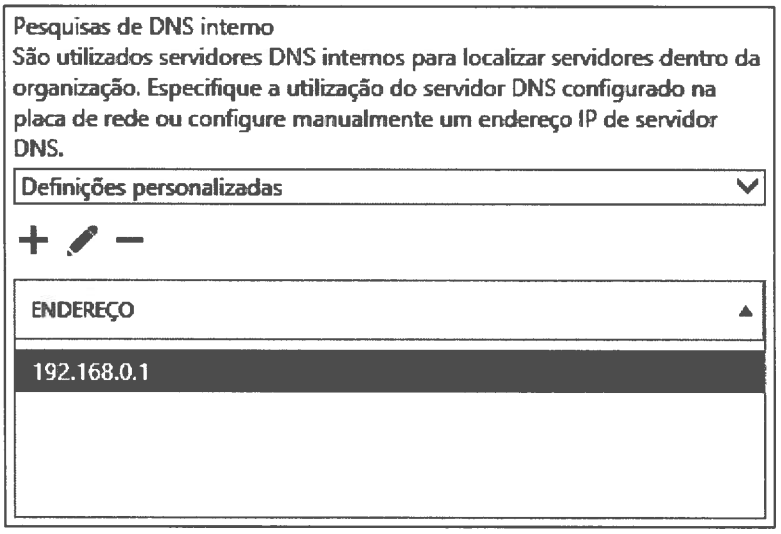


Figura 41 - Exchange Pesquisas de DNS

Na aba "certificados" podem ser geridos os certificados que estão associados ao Exchange. Por defeito, na instalação da *cliente access role*, são criados *self-signed certificates*. Este tipo de certificado deve apenas ser usado temporariamente, pois tem certas limitações comparativamente a um certificado emitido por uma autoridade certificadora. Algumas limitações:

- Apenas válido por 1 ano a partir da data de criação;

- Não pode ser usado com o Outlook Anywhere;

- Não pode ser usado para encriptar comunicações entre dispositivos ActiveSync e o servidor Exchange;

- São mostrados aos clientes do Microsoft Outlook Web Access avisos acerca do certificado não ser de fonte confiável.

("Understanding the Self-Signed Certificate in Exchange," n.d.)

Anteriormente, no DC1 foi instalada a *role* de *Certificate Authority*. Esta será agora usada para emitir um certificado de *web server* para o Exchange.

1. "New – Create a request for a Certificate from the Certification Authority".
2. Dar um nome ao certificado e especificar em que servidor o *request* será armazenado.
3. Especificar os domínios a serem incluídos no certificado, que correspondem ao acesso às várias páginas do Exchange. Neste caso foi adicionado o domínio

"mail.projetoglobal.local". Caso não tenha sido já feito, tem de ser criada uma nova entrada no DNS, com esse nome, a apontar para o servidor do Exchange(ROUTING).

4. Preencher o formulário com informações relativas à organização.
5. Guardar o *request* numa partilha de rede ou no disco local.

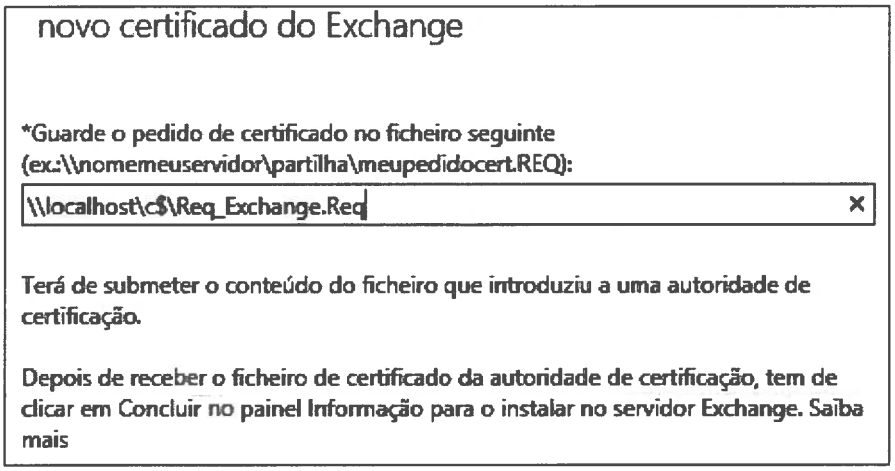


Figura 42 - Exchange New Certificate Request

6. Aceder ao DC1, e instalar o serviço do AD:CA "Certification Authority Web Enrollment" para que possam ser feitos pedidos e renovações de certificados através de uma interface web.
7. De volta ao ROUTING, aceder à diretoria virtual criada anteriormente pelo serviço instalado no DC1 através do link: "http://dc1/CertSrv".
8. "Request a certificate – Advanced certificate request – submit a certificate request by using a base 64-encoded CMC or PKCS file, or submit a rewewal request by using a base-64-encoded PKCS file".
9. Abrir o ficheiro de *request* criado anteriormente e copiar todo o texto para o *request* nesta página, selecionando também o *template web server*.

Figura 43 - Submit Web Server Certificate Request

10. Escolher "base 64 encoded" e fazer o *download* do certificado. Este certificado terá de ser instalado em todas as máquinas que seja pretendido que o utilizem.
11. No Exchange, na aba de certificados, no certificado pendente, clicar em *Complete* e apontar para o ficheiro ".cer", emitido agora pela autoridade certificadora do domínio.
12. Editar esse certificado e selecionar os serviços pretendidos para ele, neste caso SMTP, IMAP, POP e IIS. Caso surja um aviso acerca de substituir um certificado já existente selecionar a opção *yes*.

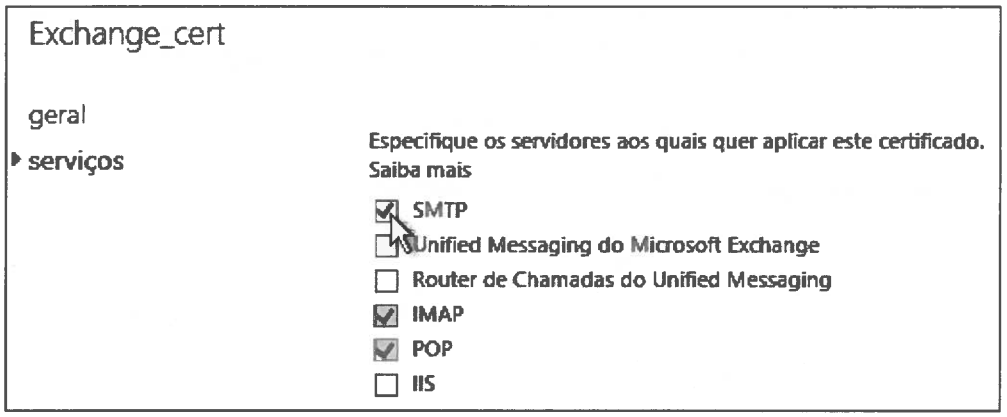


Figura 44 - Exchange Certificate Services

13. No IIS e no "Default Web Site – Edit Bindings", adicionar a binding para "mail.projetoglobal.local" do tipo "https", e associar-lhe o certificado emitido anteriormente.

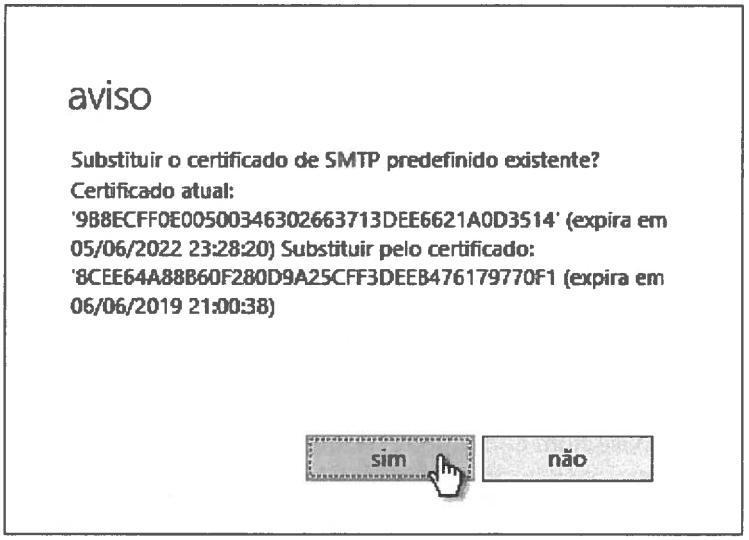


Figura 45 - Exchange Replace Certificate

No lado do cliente, basta instalar esse certificado, na *certificate store* "Trusted Root Certification Authorities". Acedendo agora ao link "https://mail.projetoglobal.local" será aberta a página do Outlook Web App sem quaisquer avisos de certificado.

A conta de *mail* também pode ser acedida através de um *mail client*. Neste caso, na máquina cliente foi instalado o "eM Client".

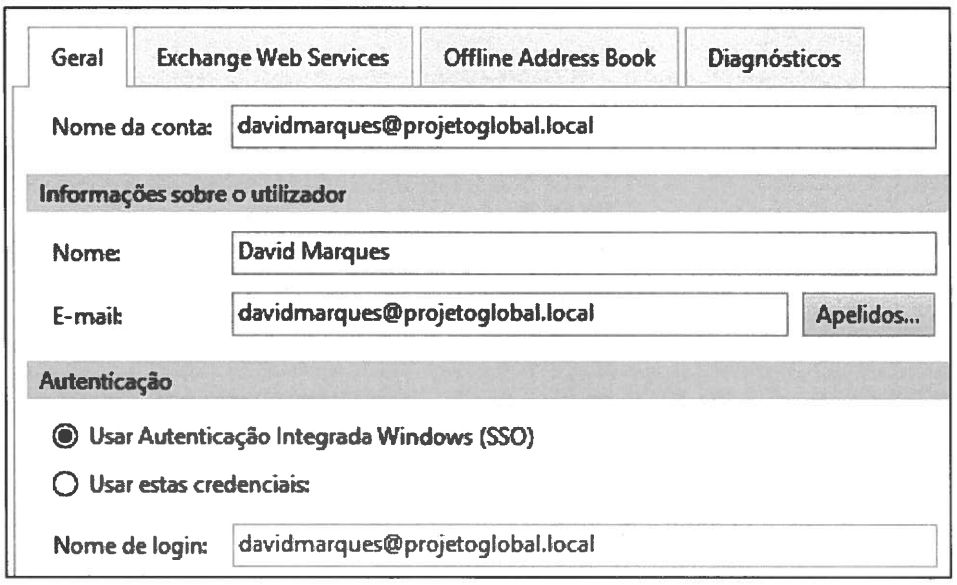


Figura 46 - eM Client

3.1.6 NPS

O NPS é a implementação da Microsoft do padrão RADIUS. Permite a gestão centralizada de contas, e autenticação e autorização para diferentes tipos de acesso, como por exemplo *wireless, dial-up* e *VPN*. Num domínio, o NPS acede à base de dados do Active Directory para autenticar os utilizadores na rede. O NPS pode ser utilizado como RADIUS *server, proxy*, ou ambos.
("Network Policy Server (NPS) | Microsoft Docs," 2017)

Esta máquina virtual foi configurada como RADIUS *server*, para controlar os acessos remotos à rede, com base num conjunto de políticas.

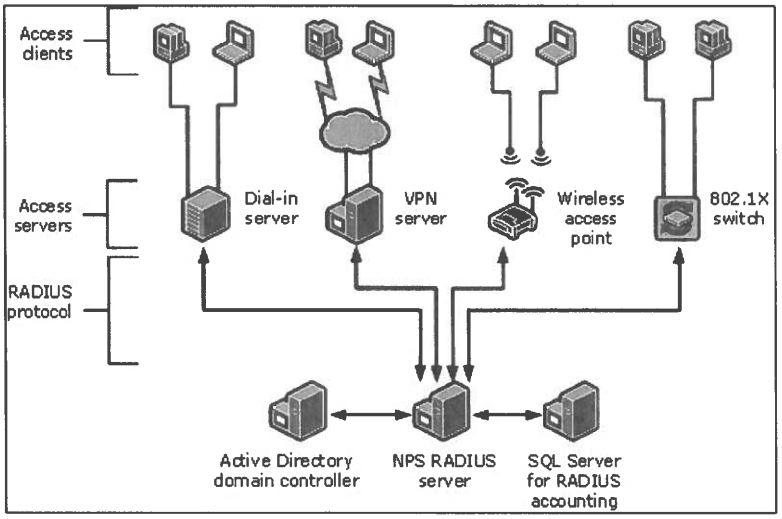


Figura 47 – Exemplo de RADIUS Server ("Network Policy Server (NPS) | Microsoft Docs," 2017)

3.1.6.1 Network Policy and Access Services

O NPAS permite definir e impor políticas para o acesso à rede, autenticação, autorização e saúde de clientes através dos serviços Network Policy Server(NPS), Health Registration Authority(HRA) e o protocolo Host Credential Authorization (HCAP). ("Network Policy and Access Services Overview," n.d.)

1. Abrir o MMC, adicionar o *snap-in* "Certificates" da máquina local, e requisitar um novo certificado do tipo "Domain Server Authentication". Antes de ser emitido tem de ser aprovado na máquina DC1, no *role* AD:CS em "Pending Requests".
2. Instalar o *role* Network Policy and Access Services através do Server Manager, selecionando apenas o serviço NPS.
3. Abrir o NPS em "Tools – Network Policy Server".
4. Registar o NPS no Active Directory fazendo *right-click* no seu nome e clicando em "register server in Active Directory". Este procedimento é necessário para que o servidor possa ler as propriedades de *dial-in* dos utilizadores no AD, de forma a poder autenticá-los.
5. Em "RADIUS Clients and Servers", clicar em "Configure RADIUS Clients".

Para utilizar um RADIUS *server*, tem de ser configurado um RADIUS *client*, (servidor de acesso à rede) que neste caso é o servidor Routing.

6. *Right-click* em "RADIUS Clients – New".

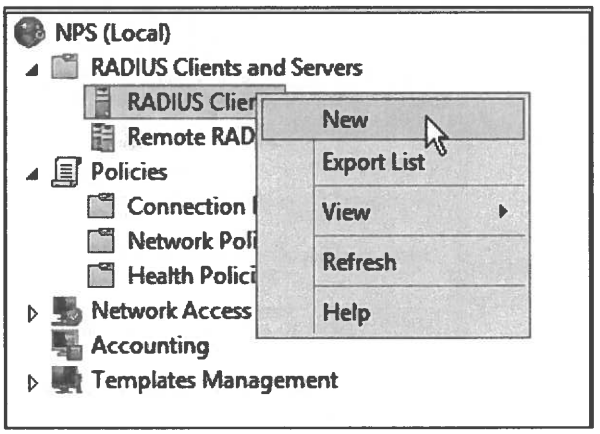


Figura 48 - RADIUS clients-1

7. Dar um nome amigável ao cliente, neste caso foi dado o nome "Routing-Site1" e definir o nome ou endereço IP do servidor.

8. Em "Shared Secret" tem de ser introduzida uma *password* ou gerada uma aleatoriamente por questões de segurança.

   Este "shared secret" é usado para verificar que as mensagens RADIUS vêm de uma fonte legítima, e que essa mensagem não foi adulterada em trânsito. É também usado para encriptar alguns dos atributos RADIUS, tais como palavras-chave de utilizadores e *tunnel-passwords*. O cliente e o servidor RADIUS têm de partilhar o mesmo "shared secret". ("Shared Secrets for NPS and RADIUS Clients," n.d.)

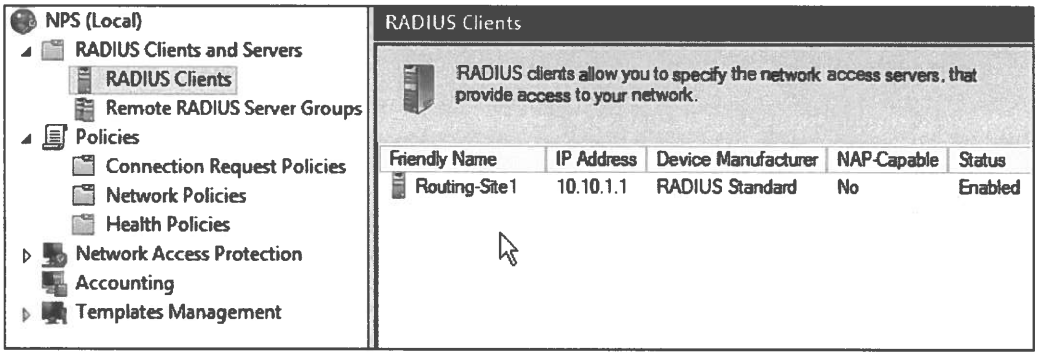


Figura 49 - RADIUS clients-2

9. Selecionar o "NPS (Local)" e no *drop-down* em "Standard Configuration", escolher a opção "RADIUS server for Dial-Up or VPN Connections". Clicar em "Configure VPN or Dial-Up".

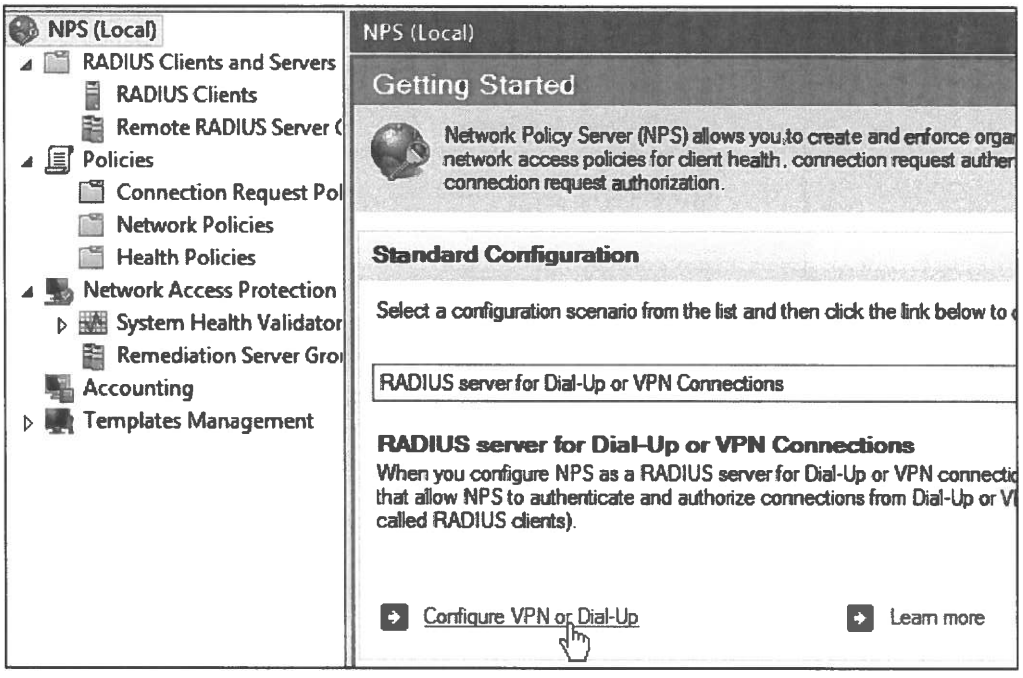


Figura 50 - New VPN policy

10. Escolher VPN e dar um nome à política (VPN-Site1).
11. Selecionar o RADIUS *client* criado anteriormente (Routing-Site1).
12. Deixar os métodos de autenticação e todas as etapas subsequentes com os valores padrão.

Todas estas configurações podem ser alteradas após a criação da política.

13. Após a sua criação, foram feitas algumas configurações adicionais. Em "Policies – Network Policies", *right-click* no nome da política criada e aceder às suas propriedades.

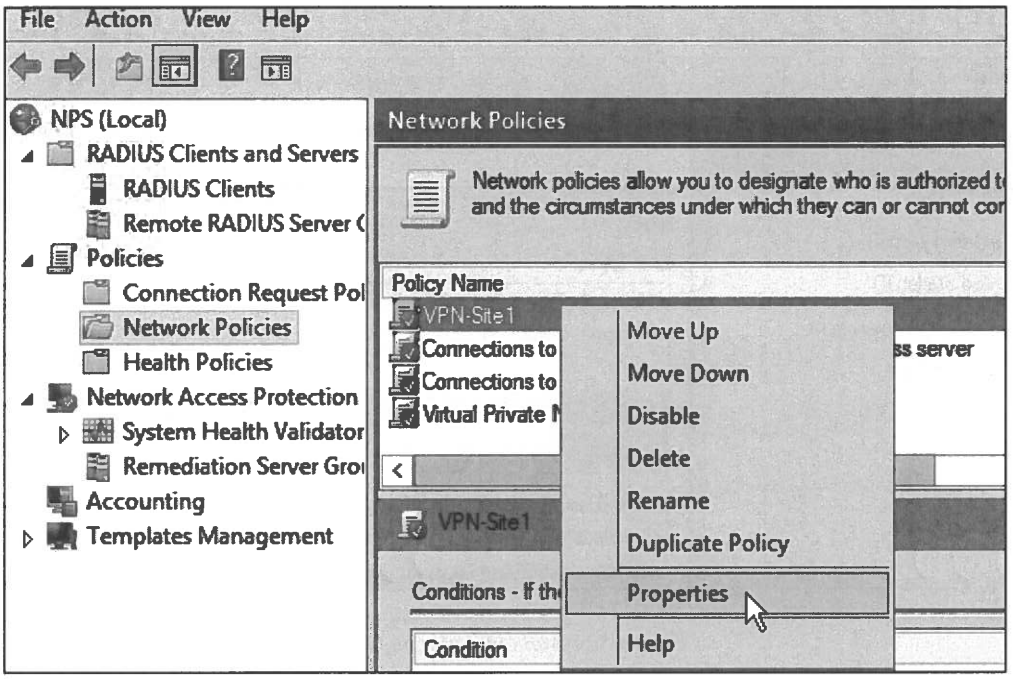


Figura 51 - Policy properties

Network Policies são conjuntos de condições, restrições e definições que permitem definir quem é autorizado a ligar-se à rede e em que circunstâncias pode ou não fazê-lo. ("Network Policies," n.d.)

14. Configurações realizadas por aba:

Overview : Desabilitar a opção "ignore user account dial-in properties", para que as configurações de *dial-in* dos utilizadores sejam levadas em conta pelo NPS.

Conditions : Aqui podem ser definidas as condições necessárias para que os pedidos de ligação sejam aceites por esta política. É obrigatório que exista pelo menos 1 condição.

Neste caso foi adicionado o grupo "VPN Users". Isto assegura que apenas utilizadores pertencentes a esse grupo no AD, terão acesso à rede por VPN.

Constraints : Parâmetros opcionais da política. Diferem das condições no sentido em que, quando uma condição não é cumprida, o NPS continua a avaliar outras políticas de rede na tentativa de autorizar a ligação. No caso das *constraints*, se alguma não corresponder ao pedido de ligação, essa ligação é imediatamente recusada.

Neste caso foi alterada a *constraint* "Authentication Methods", onde foram definidos 2 métodos distintos de autenticação.

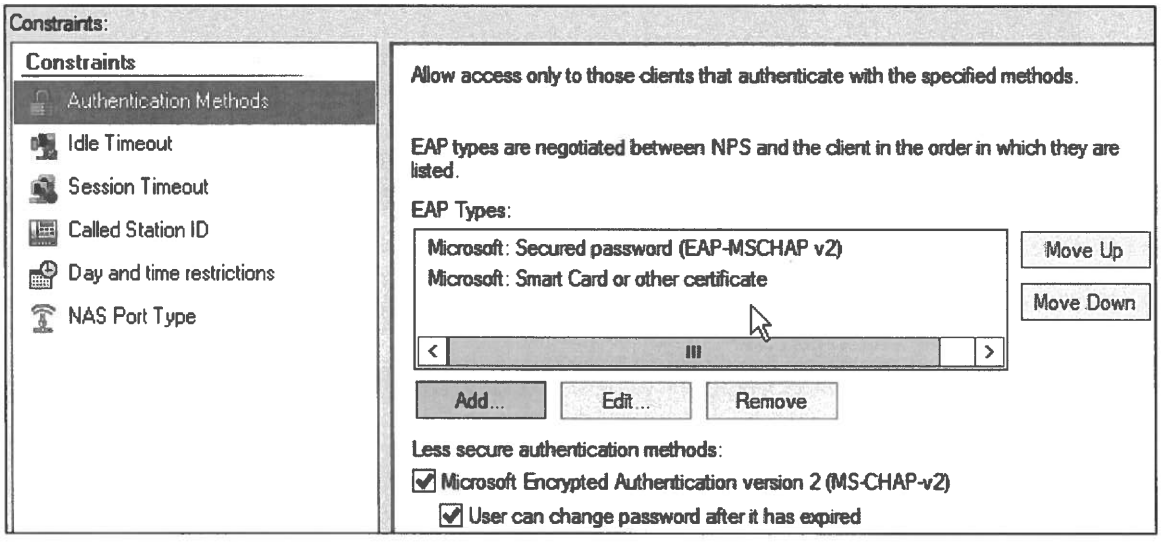


Figura 52 - Authentication methods

15. No método "Microsoft: Smart Card or other certificate", clicar em "Edit" e selecionar o certificado que foi instalado na máquina (Domain Server Authentication). Isto permite que este servidor possa provar a sua identidade ao cliente que se tenta ligar.

A autenticação com utilização de certificados digitais é um dos métodos que fornece maior segurança, eliminando a necessidade de utilização de *passwords*. As ligações VPN com o protocolo L2TP requerem a utilização de certificados, enquanto que com PPTP não.

("Certificates and NPS," n.d.)

3.1.7 SQL

Esta máquina virtual foi configurada como servidor de bases de dados. Para isso, foi instalado o SGBD SQL Server 2014.

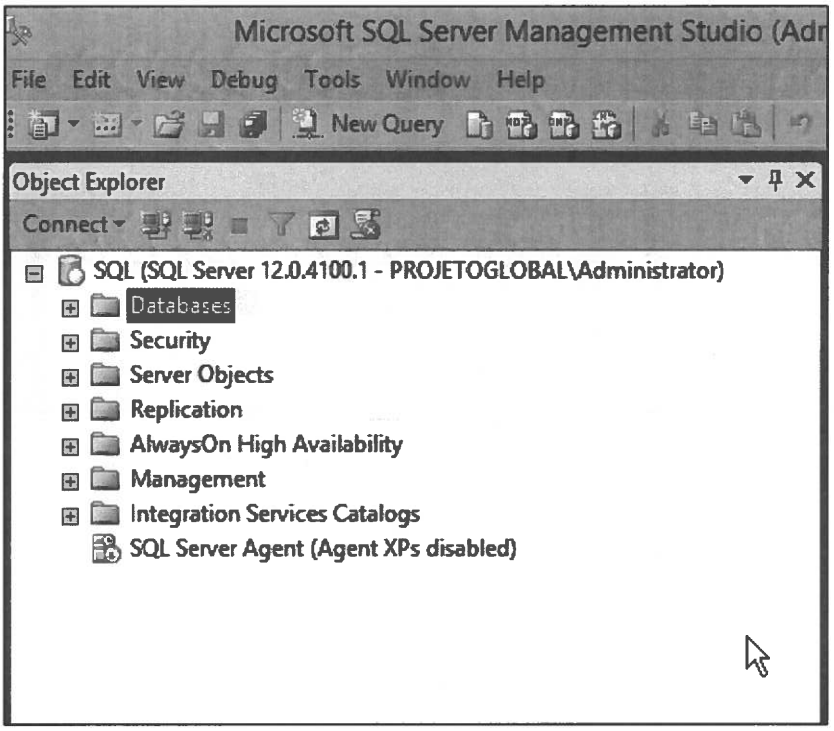


Figura 53 - SQL Management Studio

Para que as bases de dados sejam acessíveis pelo utilizador que acede remotamente por VPN, foi necessário proceder à criação de um novo *Login*.

1. Expandir a pasta "Security". *Right-click em* "Logins – New Login".

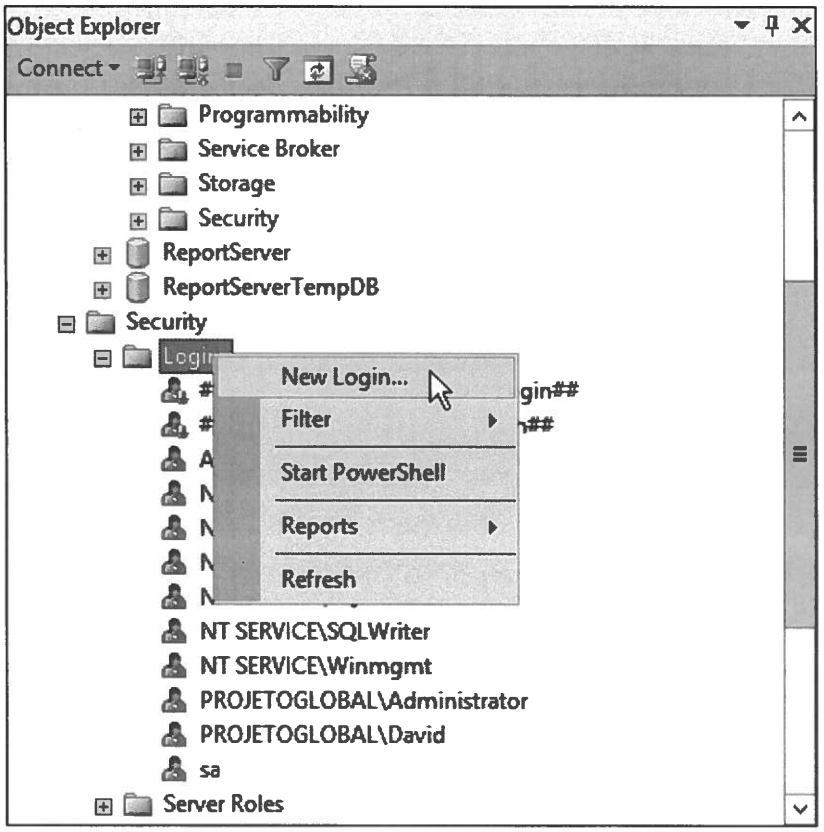


Figura 54 - New login

2. Na tab General - Manter o tipo de autenticação como "Windows Authentication". No "Login name" clicar em "Search…" e especificar o utilizador "David", na diretoria do domínio.

Este utilizador é um desenvolvedor de aplicações .NET e precisa de ter privilégios de administração no servidor SQL.

3. Na tab Server Roles – habilitar o conjunto de permissões "sysadmin".

É também necessário criar uma nova regra de *Inbound* na firewall para que sejam aceites as ligações ao *Database Engine*. A porta a abrir neste caso, sendo a *default instance*, é a 1433 – TCP.

3.1.8 W8.1

Esta máquina virtual foi criada com o propósito de servir como cliente de acesso remoto ao domínio. O utilizador desta máquina (David) desempenha funções de programação .NET, necessitando de aceder ao servidor SQL para a criação e gestão de bases de dados. Este utilizador tem um *roaming profile,* e tem acesso à sua *home folder* na rede do domínio.

3.1.8.1 Ligação VPN

Uma VPN é uma ligação ponto a ponto segura, feita através de uma rede privada ou pública (Internet). Um cliente VPN utiliza protocolos baseados em TCP/IP ou UDP chamados *tunneling protocols*, para estabelecer uma ligação encriptada com o servidor VPN através de determinada porta virtual. ("VPN connection types (Windows 10) | Microsoft Docs," 2017)

Foram criadas 2 ligações VPN para propósitos de demonstração.

A primeira utiliza PPTP (Point to Point Tunneling Protocol) com autenticação EAP e *password.*

A segunda utiliza L2TP (Layer 2 Tunneling Protocol with IPSec) com autenticação EAP e certificado digital emitido para o utilizador.

Antes de ser possível estabelecer a ligação por L2TP com certificado, têm de ser instalados os certificados necessários para que a máquina e o utilizador sejam autenticados na rede.

1. Numa conta de administrador, abrir o MMC e adicionar 2 *snap-ins* de certificate (user account e computer account).
2. Fazer *request* dos certificados "Domain User" e "Domain Computer Authentication", respetivamente.

Criar as ligações

1. No "Network and Sharing Center", criar uma nova ligação a um *workplace* por VPN.
2. Especificar o endereço IP público do servidor VPN (Routing) e um nome para a ligação. Ativar a opção "Allow other people to use this connection." Esta opção permite que seja possível iniciar sessão diretamente com a conta de domínio.

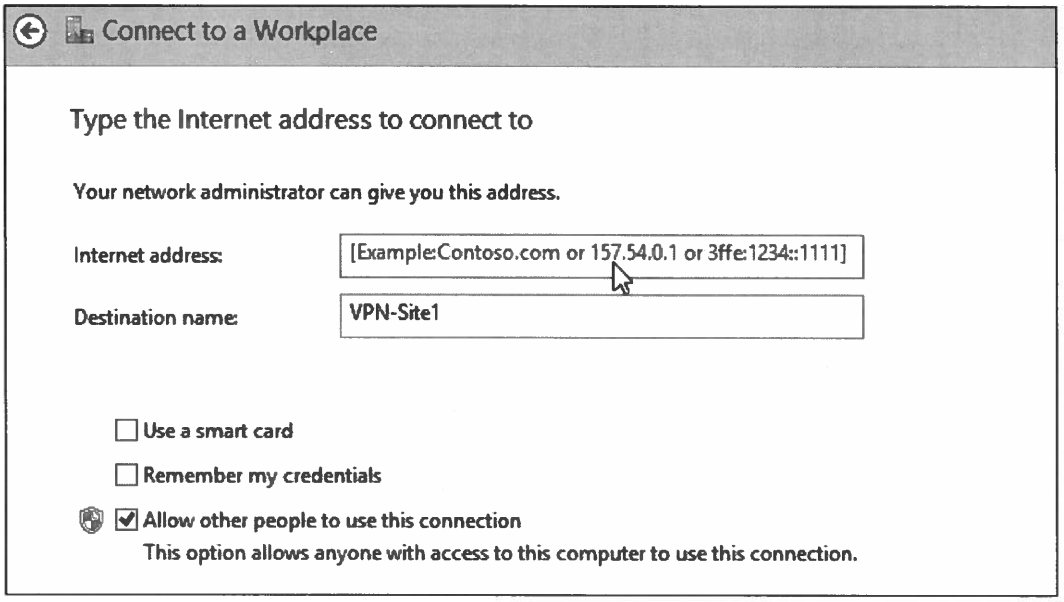


Figura 55 - Connect to VPN

3. Alterar as propriedades de segurança de ambas as ligações.

PPTP

Nome: VPN-Site1(PPTP)

Type of VPN: PPTP

Data encryption: Maximum strength encryption

Authentication: EAP - Microsoft Secured password (EAP-MSCHAP v2)

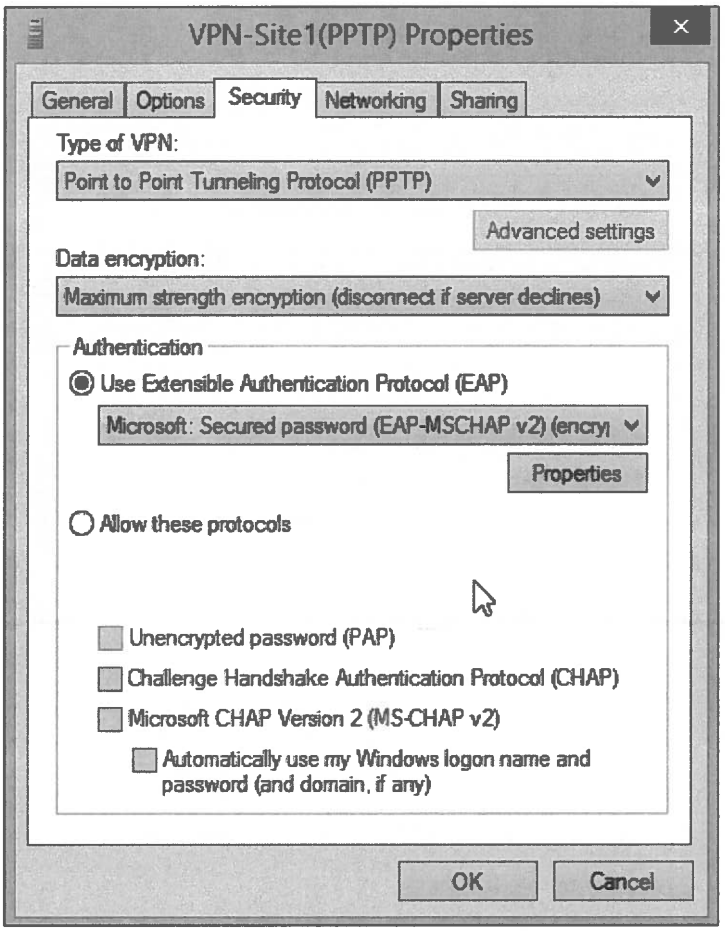


Figura 56 - PPTP

L2TP

Nome: VPN-Site1(Certificado)

Type of VPN: Layer 2 Tunneling Protocol with IPsec

Data encryption: Maximum strength encryption

Authentication: EAP – Microsoft: Smart Card or other certificate

Em "Advanced Settings" tem de ser especificado que a ligação utiliza um certificado, bem como a autoridade certificadora que o emitiu (PROJETOGLOBAL-CA).

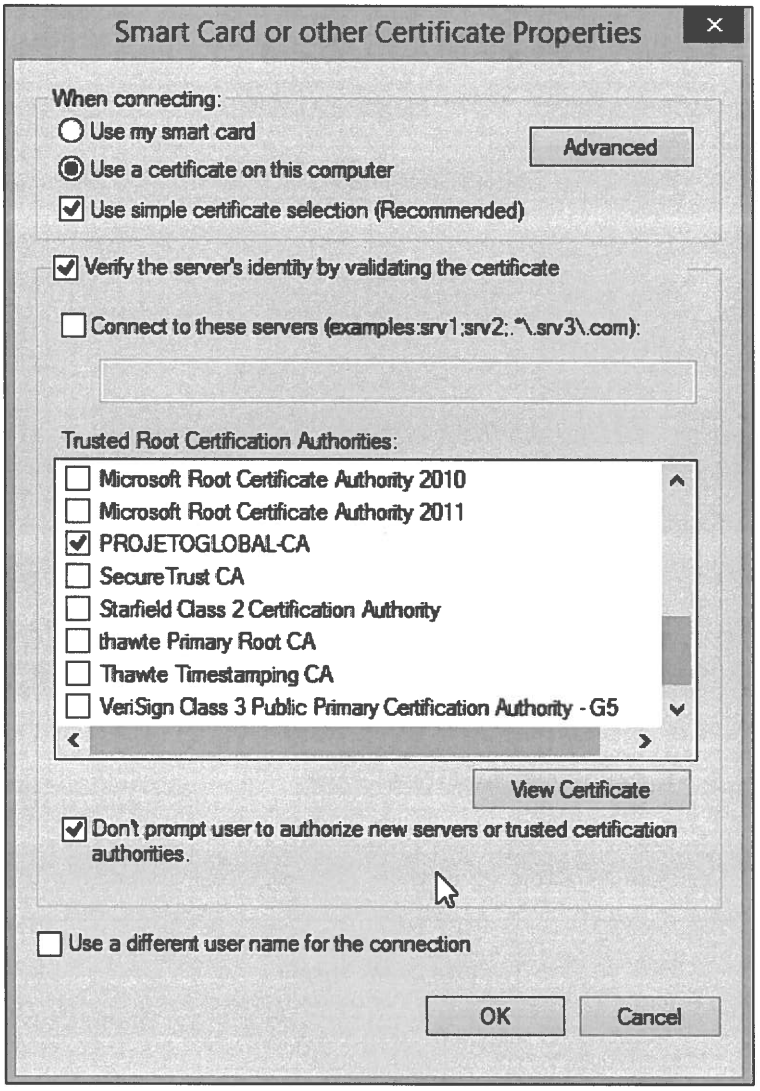


Figura 57 - L2TP

3.2 Programação

A aplicação foi desenvolvida em C# com interface WPF. Trata-se de uma aplicação de gestão de produtos informáticos de uma empresa fictícia (PC Express). Esses produtos são armazenados numa base de dados e acedidos pela aplicação através de objetos ADO.NET.

O software utilizado no desenvolvimento foi o seguinte:

- Visual Studio Community 2017

- SQL Server 2014

Funcionalidades:

- Registo e autenticação de utilizadores, com encriptação de *passwords*;
- Adicionar produtos com campos de nome, categoria, marca, stock, preço e imagem;
- Listar informação dos produtos, alterar e remover;
- Filtrar produtos por categoria ou marca;
- Pesquisar produtos por nome.

Credenciais de acesso à aplicação:

Administrador(nível 1 de acesso):

Utilizador: Administrator

Palavra-chave: P@ssw0rd

Cliente(nível 2 de acesso):

Utilizador: David

Palavra-chave: istec

A base de dados é composta por 6 tabelas:

Produtos: Produtos disponíveis;

Categorias: Categorias de produtos(CPU,GPU,etc);

Marcas: Marcas dos produtos(MSI,Asus,etc);

Utilizadores: Utilizadores registados, com propósitos de autenticação na base de dados;

Encomendas: Encomendas colocadas pelos clientes.

Produtos_encomendas: Tabela de ligação entre os produtos e as encomendas.

Esquema da base de dados

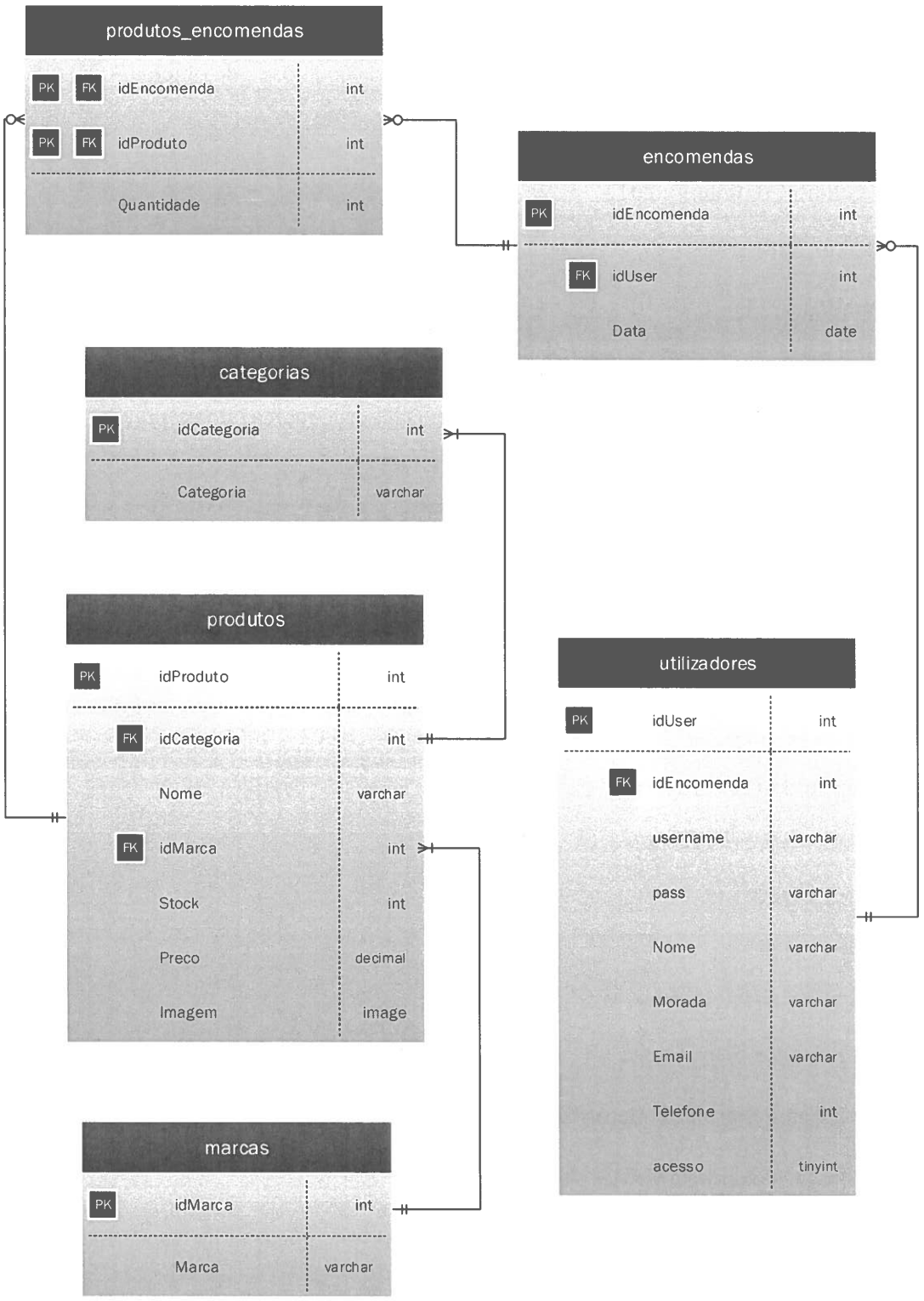


Figura 58 - Esquema da BD

3.2.1 Classes

Login

Classe com elementos de interface. Permite ao utilizador autenticar-se, introduzindo os seus dados em *textboxes*. O *layout* tem 2 botões, 2 caixas de texto e 1 label com informações sobre o desenvolvedor.

```
1.  public partial class Login : MetroWindow
2.  {
3.  //instancia a classe Utilizador
4.  Utilizador util = new Utilizador();
5.  //query que recebe os valores de parâmetros
6.  string query_login = "SELECT * from Utilizadores where username= @username and pass
    = @password";
7.  //connection string, definida no app.config
8.  string CS = ConfigurationManager.ConnectionStrings["CS"].ConnectionString;
9.    public Login()
10.   {
11.   InitializeComponent();
12.   }
13.     //botão de login
14.     private void btn_login_Click(object sender, RoutedEventArgs e)
15.
16.     {
17.       try
18.       {
19.     //instancia um novo objeto de ligação. o using permite que o objeto criado seja
    closed, flushed
20.     //e disposed quando os recursos para a sua utilização já não forem necessários.

21.         using (SqlConnection con = new SqlConnection(CS))
22.         {
23.         //abre a ligação à BD
24.         con.Open();
25.         //chama o método "encrypt" da classe "Utilizador" para encriptar a password
    inserida
26.         string cipher_pass = util.encrypt(txb_pass.Password);
27.           //cria uma nova query parametrizada
28.           using (SqlCommand cmd = new SqlCommand(query_login, con))
29.           {
30.           //cria uma query parametrizada que recebe os valores do utilizador e pass
    word
```

```
31.          cmd.Parameters.Add("@username", System.Data.SqlDbType.VarChar, 25).Value
   = txb_user.Text;
32.          cmd.Parameters.Add("@password", System.Data.SqlDbType.VarChar, 25).Value
   = cipher_pass;
33.            using (SqlDataReader dr = cmd.ExecuteReader())
34.            {
35.              //se a query devolver algum resultado, procede com o login
36.              if (dr.Read())
37.              {
38.              int access_type = (Convert.ToInt32(dr["acesso"]));
39.              //passa o username para a classe Cliente_orders
40.              Cliente_orders.username = txb_user.Text;
41.              txb_user.Text = "";
42.              txb_pass.Password = "";
43.              //se o acesso for 1 abre o formulário "AdminProdutos"
44.                if (access_type == 1)
45.                {
46.                AdminProdutos adm = new AdminProdutos();
47.                this.Hide();
48.                adm.ShowDialog();
49.                }
50.                //se o acesso for 2 abre o formulário "Cliente_orders"
51.                else if (access_type == 2)
52.                {
53.                Cliente_orders clo = new Cliente_orders();
54.                this.Hide();
55.                clo.ShowDialog();
56.                }
57.              } //read
58.                else
59.                {
60.                MessageBox.Show("Dados incorretos");
61.                }
62.                   } //sqldatareader
63.                } //try
64.          } //connection
65.        } //try
66.        catch (Exception ex)
67.        {
68.            MessageBox.Show(ex.Message);
69.        } //catch
70.    } //Login
71.
72.    //botão de registo
```

```
73.     private void btn_register_Click(object sender, RoutedEventArgs e)
74.     {
75.       //instancia a classe Register e abre o formulário "Register"
76.       Register reg = new Register();
77.       this.Hide();
78.       reg.ShowDialog();
79.       }
80.     private void Window_Closed(object sender, EventArgs e)
81.     {
82.       //termina a aplicação após fechar o formulário
83.       Application.Current.Shutdown();
84.
85.     }
86.     private void lbl_créditos_MouseDown(object sender, MouseButtonEventArgs e)
87.     {
88.         MessageBox.Show("Desenvolvido por: David Marques nº2029");
89.     }
90. }
```

Register

Classe com elementos de interface. Permite ao utilizador efetuar o registo de um utilizador na base de dados, introduzindo um conjunto de dados em caixas de texto (*username*, *password*, nome, morada, telemóvel, e-mail). A *pass*word introduzida é encriptada com AES, um algoritmo de chave simétrica.

```
1.  public partial class Register : MetroWindow
2.  {
3.      string CS = ConfigurationManager.ConnectionStrings["CS"].ConnectionString;
4.      string query_selectuser = "SELECT username from utilizadores where username = @
    username";
5.      string query_registar = "INSERT INTO utilizadores (username, pass, nome, morada
    , telemovel, email, acesso) VALUES (@username, @password, @nome, @morada, @telemove
    l, @email, 1)";
6.      Utilizador util = new Utilizador();
7.      Login log = new Login();
8.      public Register()
9.      {
10.         InitializeComponent();
11.     }
12.     //botão para efetuar o registo
13.     private void btn_Registar_Click(object sender, RoutedEventArgs e)
14.     {
```

```
15.       try
16.       {
17.         //verifica se o utilizador preencheu os campos obrigatórios
18.         if (txb_username.Text == "" || txb_password.Password == "" || txb_email.Tex
    t == "")
19.           {
20.             MessageBox.Show("Preencha os campos obrigatórios");
21.           }
22.         else
23.           {
24.             using (SqlConnection con = new SqlConnection(CS))
25.               {
26.                 con.Open();
27.                 //cria e executa uma query parametrizada com o username introduzido

28.                 SqlCommand cmd = new SqlCommand(query_selectuser, con);
29.                 cmd.Parameters.Add("@username", System.Data.SqlDbType.VarChar, 25).
    Value = txb_username.Text;
30.                 SqlDataReader dr = cmd.ExecuteReader();
31.                 //verifica se o user inserido já existe na base de dados
32.                 dr.Read();
33.                   if (dr.HasRows)
34.                   {
35.                     MessageBox.Show("Esse utilizador já existe.");
36.                   }
37.                   else
38.                   {
39.                     //chama o método "encrypt" da classe Utilizador, que encripta a
     password introduzida com AES e chave de 256bit
40.                     string cipher_pass = util.encrypt(txb_password.Password);
41.                     //cria e executa uma query que recebe como parâmetros os dados
    introduzidos e a password encriptada
42.                     SqlCommand cmd2 = new SqlCommand(query_registar, con);
43.                     cmd2.Parameters.Add("@username", System.Data.SqlDbType.VarChar,
     25).Value = txb_username.Text;
44.                     cmd2.Parameters.Add("@password", System.Data.SqlDbType.VarChar,
     25).Value = cipher_pass;
45.                     cmd2.Parameters.Add("@nome", System.Data.SqlDbType.VarChar, 50)
    .Value = txb_nome.Text;
46.                     cmd2.Parameters.Add("@morada", System.Data.SqlDbType.VarChar, 5
    0).Value = txb_morada.Text;
47.                     cmd2.Parameters.Add("@telemovel", System.Data.SqlDbType.VarChar
    , 40).Value = txb_tele.Text;
```

```
48.                     cmd2.Parameters.Add("@email", System.Data.SqlDbType.VarChar, 40
    ).Value = txb_email.Text;
49.                     SqlDataReader dr2 = cmd2.ExecuteReader();
50.                     MessageBox.Show("Utilizador registado.");
51.                     this.Hide();
52.                     log.Show();
53.                   }
54.                 }
55.               }
56.         } //try
57.         catch (Exception ex)
58.         {
59.             MessageBox.Show(ex.Message);
60.         } //catch
61.     }
62.     //quando é fechada, volta à janela de login
63.     private void Window_Closed(object sender, EventArgs e)
64.     {
65.       Login log = new Login();
66.       log.ShowDialog();
67.       this.Close();
68.     }
69.     private void btn_cancelar_Click(object sender, RoutedEventArgs e)
70.     {
71.       this.Close();
72.     }
73.     //botão que limpa o texto de todas as textboxes na grid
74.     private void btn_clear_txt_Click(object sender, RoutedEventArgs e)
75.     {
76.       foreach (UIElement txB in Grid.Children)
77.       {
78.         TextBox textBox = txB as TextBox;
79.         if (textBox == null)
80.         continue;
81.         textBox.Text = null;
82.         }
83.     }
84. }
```

Utilizador

Esta classe é instanciada pela classe Register e tem como propósito encriptar a *password* introduzida no ato do registo.

```
1.  class Utilizador
2.      {
3.          //classe necessária à encriptação simétrica AES
4.          AesCryptoServiceProvider crypt_provider;
5.          public Utilizador()
6.          {
7.            crypt_provider = new AesCryptoServiceProvider();
8.            //tamanho do bloco
9.            crypt_provider.BlockSize = 128;
10.           //tamanho da chave
11.           crypt_provider.KeySize = 256;
12.           //chave do algoritmo. Também poderia ser gerada uma aleatoriamente
13.           crypt_provider.Key = Convert.FromBase64String("C53wafJw3QmImGBN8Se9EnIJgi
    Qq7LyoWHzUEFQI/B0=");
14.           //modo que encripta cada bloco individualmente
15.           crypt_provider.Mode = CipherMode.ECB;
16.         }
17.         public String encrypt(String clear_text)
18.         {
19.           ICryptoTransform transform = crypt_provider.CreateEncryptor();
20.           byte[] encrypted_bytes = transform.TransformFinalBlock(ASCIIEncoding.ASCI
    I.GetBytes(clear_text), 0, clear_text.Length);
21.           string str = Convert.ToBase64String(encrypted_bytes);
22.           return str;
23.         }
24.     }
```

AdminProdutos

Classe com elementos de interface. Contém a funcionalidade da aplicação, para utilizadores com acesso de administrador. Permite listar, criar, alterar, eliminar, filtrar e pesquisar produtos na base de dados.

```
1.  public partial class AdminProdutos : MetroWindow
2.    {
3.        string imgPath_dlg;
4.        //connection string, definida no app.config
5.        string CS = ConfigurationManager.ConnectionStrings["CS"].ConnectionString;
```

```
6.        Produto prod = new Produto();
7.
8.      public AdminProdutos()
9.        {
10.           InitializeComponent();
11.       }
12.       //evento executado quando a grid da janela é carregada
13.       private void MainGrid_Loaded(object sender, RoutedEventArgs e)
14.       {
15.       prod.dg = this.dg_produtos;
16.       prod.imgBox = this.img_Prod;
17.       prod.WindowGrid = this.MainGrid;
18.       //carrega a datagrid
19.       prod.Load_Produtos();
20.       //limpa todos os campos
21.       prod.Limpar();
22.       //esconde colunas específicas
23.       prod.HideCol();
24.       }
25.       private void Button_Admin_Click(object sender, RoutedEventArgs e)
26.         {
27.   //instancia um objeto Button para ser usado como sender para um switch de botões

28.       Button bt = (Button)sender;
29.       //queries
30.       string query_Del = "DELETE FROM produtos WHERE idProduto = '" + this.txb_prod
    utosId.Text + "'; ";
31.         string query_Updt_noImg = "UPDATE produtos SET Nome = @Nome, Categoria = @C
    ategoria, Marca = @Marca, Stock = @Stock, Preco = @Preco WHERE idProduto = '" + thi
    s.txb_produtosId.Text + "';";
32.           try
33.           {
34.             using (SqlConnection con = new SqlConnection(CS)) //o using permite que
     o objeto criado seja closed, flushed e disposed quando os recursos para a sua util
    ização já não forem necessários
35.             {
36.             //Estabelece ligação com a BD
37.             con.Open();
38.               switch (bt.Name)
39.               {
40.                 //Botão para atualizar a datagrid
41.                 case "btn_load_dt":
42.                 prod.dg = this.dg_produtos;
43.                 prod.imgBox = this.img_Prod;
```

```
44.                 prod.WindowGrid = this.MainGrid;
45.                 prod.Load_Produtos();
46.                 prod.Limpar();
47.                 prod.HideCol();
48.                 break;
49.
50.                 //Botão para adicionar produto
51.                 case "btn_add":
52.                   if (txb_produtosCat.Text == "" || txb_produtosNome.Text == "" ||
    txb_produtosMarca.Text == "" || txb_produtosStock.Text == "" || txb_produtosPreco.T
    ext == "")
53.                   {
54.                   MessageBox.Show("Preencha todos os campos");
55.                   }
56.                   else if (txb_produtosPreco.Text.Contains("."))
57.                   {
58.                   MessageBox.Show("Utilize virgulas no preço");
59.                   }
60.                   else
61.                   {
62.                   //executa uma stored procedure com os parâmetros recebidos
63.                   SqlCommand addProd = new SqlCommand("spInsProd", con);
64.                   addProd.CommandType = CommandType.StoredProcedure;
65.                   addProd.Parameters.Add("@Nome", System.Data.SqlDbType.VarChar, 50
    ).Value = txb_produtosNome.Text;
66.                   addProd.Parameters.Add("@Categoria", System.Data.SqlDbType.VarCha
    r, 30).Value = txb_produtosCat.Text;
67.                   addProd.Parameters.Add("@Marca", System.Data.SqlDbType.VarChar, 3
    0).Value = txb_produtosMarca.Text;
68.                   addProd.Parameters.Add("@Stock", System.Data.SqlDbType.Int).Value
     = txb_produtosStock.Text;
69.                   addProd.Parameters.Add("@Preco", System.Data.SqlDbType.Decimal).V
    alue = txb_produtosPreco.Text;
70.                   addProd.ExecuteNonQuery();
71.                   MessageBox.Show("Registo adicionado");
72.                   prod.Load_Produtos();
73.                   prod.Limpar();
74.                   prod.HideCol();
75.                   }
76.                   break;
77.
78.                   //Botão para alterar produto
79.                   case "btn_alter":
80.                   //verifica se um item está seleccionado
```

```
81.                     if (Convert.ToInt32(dg_produtos.SelectedItems.Count) == 0)
82.                       {
83.                       MessageBox.Show("Seleccione um item da tabela");
84.                       }
85.                       else if (txb_produtosPreco.Text.Contains("."))
86.                       {
87.                       MessageBox.Show("Utilize virgulas no preço");
88.                       }
89.                       else
90.                       {
91.                     //verifica se a variável que contém o Path da imagem(seleccionado
    no dialog) não é nula
92.                       if (imgPath_dlg != null)
93.                         {
94.                       SqlCommand update = new SqlCommand("spUpdtProd", con); //query
   de update completa, incluindo imagem
95.                         update.CommandType = CommandType.StoredProcedure;
96.                         byte[] img = null;
97.                        //objetos usado para escrever para o array de bytes, a imagem
   escolhida
98.                         FileStream fs = new FileStream(imgPath_dlg, FileMode.Open, Fi
   leAccess.Read);
99.                         BinaryReader br = new BinaryReader(fs);
100.                              img = br.ReadBytes((int)fs.Length);
101.                              update.Parameters.Add("@idProduto", System.Data.SqlDbT
   ype.Int).Value = txb_produtosId.Text;
102.                              update.Parameters.Add("@Nome", System.Data.SqlDbType.V
   arChar, 35).Value = txb_produtosNome.Text;
103.                              update.Parameters.Add("@Categoria", System.Data.SqlDbT
   ype.VarChar, 30).Value = txb_produtosCat.Text;
104.                              update.Parameters.Add("@Marca", System.Data.SqlDbType.
   VarChar, 30).Value = txb_produtosMarca.Text;
105.                              update.Parameters.Add("@Stock", System.Data.SqlDbType.
   Int).Value = txb_produtosStock.Text;
106.                              update.Parameters.Add("@Preco", System.Data.SqlDbType.
   Decimal).Value = txb_produtosPreco.Text;
107.                              update.Parameters.Add("@img", System.Data.SqlDbType.Im
   age).Value = img;
108.                              update.ExecuteNonQuery();
109.                     //aponta o img path para nulo, para que nos subsequentes updates
    não sejam atualizados produtos com a imagem previamente seleccionada
110.                              imgPath_dlg = null;
111.                              }
112.                              else
```

```
113.                             {
114.                             //query de update parcial, usada quando não se pretende
    alterar a imagem de um registo.
115.                             SqlCommand update2 = new SqlCommand("spUpdtProd_noImg"
    , con);
116.                             update2.CommandType = CommandType.StoredProcedure;
117.                             update2.Parameters.Add("@idProduto", System.Data.SqlDb
    Type.Int).Value = txb_produtosId.Text;
118.                             update2.Parameters.Add("@Nome", System.Data.SqlDbType.
    VarChar, 35).Value = txb_produtosNome.Text;
119.                             update2.Parameters.Add("@Categoria", System.Data.SqlDb
    Type.VarChar, 30).Value = txb_produtosCat.Text;
120.                             update2.Parameters.Add("@Marca", System.Data.SqlDbType
    .VarChar, 30).Value = txb_produtosMarca.Text;
121.                             update2.Parameters.Add("@Stock", System.Data.SqlDbType
    .SmallInt).Value = txb_produtosStock.Text;
122.                             update2.Parameters.Add("@Preco", System.Data.SqlDbType
    .Decimal).Value = txb_produtosPreco.Text;
123.                             update2.ExecuteNonQuery();
124.                             }
125.                             MessageBox.Show("Registo alterado");
126.                             prod.dg = this.dg_produtos;
127.                             prod.imgBox = this.img_Prod;
128.                             prod.WindowGrid = this.MainGrid;
129.                             prod.Load_Produtos();
130.                             prod.Limpar();
131.                             prod.HideCol();
132.                           }//else
133.                           break;
134.
135.                     //Botão para eliminar produtos
136.                     case "btn_delete":
137.                       if (Convert.ToInt32(dg_produtos.SelectedItems.Count) == 0)

138.                       {
139.                       MessageBox.Show("Seleccione um item da tabela");
140.                       }
141.                       else
142.                       {
143.                       SqlCommand delete = new SqlCommand(query_Del, con);
144.                       delete.ExecuteNonQuery();
145.                       prod.Load_Produtos();
146.                       prod.Limpar();
147.                       prod.HideCol();
```

```
148.                         MessageBox.Show("Registo eliminado");
149.                         }
150.                         break;
151.
152.                         //Botão para limpar todas as textboxes e comboboxes
153.                         case "btn_clear_txt":
154.                         prod.Limpar();
155.                         break;
156.                         //Botão fechar janela
157.                         case "btn_close_admForm":
158.                         Login log = new Login();
159.                         this.Close();
160.                         log.ShowDialog();
161.                         break;
162.                         //Botão para abrir o dialog de selecção de imagem
163.                         case "btn_browse":
164.                         //verifica se um item está seleccionado
165.                         if (Convert.ToInt32(dg_produtos.SelectedItems.Count) == 0)

166.                           {
167.                           MessageBox.Show("Seleccione um item da tabela");
168.                           }
169.                           else
170.                           {
171.                           OpenFileDialog dlg = new OpenFileDialog();
172.                           dlg.Filter = "All|*.jpg;*.png;*.bmp|JPG Files (*.jpg)|*.j
    pg|PNG Files (*.png)|*.png|BMP Files (*.bmp)|*.bmp";
173.                           dlg.Title = "Carregar imagem de Produto";
174.                           if (dlg.ShowDialog() == true)
175.                             {
176.                             imgPath_dlg = dlg.FileName;
177.                             img_Prod.Source = new BitmapImage(new Uri(dlg.FileName)
    );
178.                             }
179.                           }//else
180.                         break;
181.
182.                         //Botão para procurar produtos
183.                         case "btn_search":
184.                         SqlCommand search_cmd = new SqlCommand("spSearch", con);
185.                         search_cmd.Parameters.Add("@Search", System.Data.SqlDbType.
    VarChar, 35).Value = txb_procurar.Text;
186.                         search_cmd.CommandType = CommandType.StoredProcedure;
```

```
187.                         SqlDataAdapter dataAdapter5 = new SqlDataAdapter(search_cmd
    );
188.                         DataTable dt5 = new DataTable();
189.                         dataAdapter5.Fill(dt5);
190.                         dg_produtos.ItemsSource = dt5.DefaultView;
191.                         prod.HideCol();
192.                         break;
193.                          }//switch
194.                        }//connection
195.                   }
196.                   catch (Exception ex)
197.                   {
198.                   MessageBox.Show(ex.Message);
199.                   }
200.
201.              }
202.            //executa cada vez que uma seleção na datagrid muda
203.            private void dg_produtos_SelectionChanged(object sender, SelectionChange
    dEventArgs e)
204.              {
205.               try
206.               {
207.               SqlConnection con = new SqlConnection(CS);
208.               con.Open();
209.               //verifica se está seleccionado um item na datagrid
210.               if (dg_produtos.SelectedItems.Count > 0)
211.                 {
212.          //percorre o item seleccionado e guarda cada um dos registos em variáveis,
    para que possam ser posteriormente utilizados
213.                 for (int i = 0; i < dg_produtos.SelectedItems.Count; i++)
214.                  {
215.              DataRowView selectedItem = (DataRowView)dg_produtos.SelectedItems[i];

216.                   string idProduto = Convert.ToString(selectedItem.Row[0]);
217.                   string Nome = Convert.ToString(selectedItem.Row[1]);
218.                   string Categoria = Convert.ToString(selectedItem.Row[2]);
219.                   string Marca = Convert.ToString(selectedItem.Row[3]);
220.                   string Stock = Convert.ToString(selectedItem.Row[4]);
221.                   decimal Preco = Convert.ToDecimal(selectedItem.Row[5]);
222.                   string Imagem = Convert.ToString(selectedItem.Row[6]);
223.                   txb_produtosId.Text = idProduto;
224.                   txb_produtosNome.Text = Nome;
225.                   txb_produtosCat.Text = Categoria;
226.                   txb_produtosMarca.Text = Marca;
```

```
227.                 txb_produtosStock.Text = Stock;
228.                 txb_produtosPreco.Text = Convert.ToString(Preco);
229.                 string query_img = "SELECT Imagem from produtos WHERE idProduto='
    " + txb_produtosId.Text + "';";
230.                   using (SqlCommand cmd = new SqlCommand(query_img, con))
231.                   {
232.                     using (SqlDataReader reader = cmd.ExecuteReader())
233.                     {
234.                     reader.Read();
235.                 if (reader.HasRows && reader["Imagem"] != DBNull.Value) //verifica
     se a query devolve alguma row e se o valor dessa row na coluna "Imagens" não é nul
    o.
236.                       {
237.                       byte[] img = (byte[])(reader[0]);
238.                       MemoryStream ms = new MemoryStream(img);
239.                       img_Prod.Source = BitmapFrame.Create(ms, BitmapCreateOption
    s.None, BitmapCacheOption.OnLoad);
240.                       }
241.                       else
242.                       {
243.                       img_Prod.Source = null;
244.                       }
245.                       reader.Close();
246.                       con.Close();
247.                     }//reader
248.                   }//command
249.                }//for
250.              }//if
251.            }
252.                catch (Exception ex)
253.             {
254.             MessageBox.Show(ex.Message);
255.             }
256.          }
257.          //executa no load da combobox marca
258.          private void cBox_Marca_Loaded(object sender, RoutedEventArgs e)
259.            {
260.            //adiciona o item
261.            cBox_filterMarca.Items.Add("Marca");
262.            cBox_filterMarca.SelectedIndex = 0;
263.            using (SqlConnection con = new SqlConnection(CS))
264.            {
265.              try
266.              {
```

```
267.              con.Open();
268.              //devolve as rows nas colunas categoria e marca, sem duplicados
269.              string query_cBoxLoad = "SELECT distinct Marca from Marcas;";
270.                using (SqlCommand cmd = new SqlCommand(query_cBoxLoad, con))
271.                {
272.                SqlDataReader dr = cmd.ExecuteReader();
273.                  while (dr.Read())
274.                  {
275.                  string marca = dr.GetString(0);
276.                  cBox_filterMarca.Items.Add(marca);
277.                  }
278.                }
279.              }//try
280.              catch (Exception ex)
281.              {
282.              MessageBox.Show(ex.Message);
283.              }
284.            }//connection
285.            }
286.            //executa no load da combobox categoria
287.            private void cBox_Categoria_Loaded(object sender, RoutedEventArgs e)
288.            {
289.                cBox_filterCat.Items.Add("Categoria");
290.                cBox_filterCat.SelectedIndex = 0;
291.                using (SqlConnection con = new SqlConnection(CS))
292.                {
293.                try
294.                {
295.                con.Open();
296.              string query_cBoxLoad = "SELECT distinct Categoria from Categorias;";
    //devolve as rows nas colunas categoria e marca, sem duplicados
297.                  using (SqlCommand cmd = new SqlCommand(query_cBoxLoad, con))
298.                  {
299.                    using (SqlDataReader dr = cmd.ExecuteReader())
300.                    {
301.                      while (dr.Read())
302.                      {
303.                      string categoria = dr.GetString(0);
304.                      cBox_filterCat.Items.Add(categoria);
305.                      }//while
306.                    }//datareader
307.                  }//command
308.                }
309.                catch (Exception ex)
```

```
310.                 {
311.                 MessageBox.Show(ex.Message);
312.                 }
313.               }//connection
314.             }
315.             //executa na mudança de selecção na combobox marca
316.             private void cBox_filterMarca_SelectionChanged(object sender, Selectio
    nChangedEventArgs e)
317.             {
318.               try
319.                 {
320.                   using (SqlConnection con = new SqlConnection(CS))
321.                   {
322.                     if (cBox_filterMarca.SelectedIndex > 0)
323.                     {
324.                     //inicia a cbox com o item de indice 0 seleccionado
325.                     cBox_filterCat.SelectedIndex = 0;
326.                     con.Open();
327.                     using (SqlCommand cmd = new SqlCommand("spFilterMarca", con))

328.                     {
329.                     cmd.Parameters.Add("@Marca", System.Data.SqlDbType.VarChar, 30
    ).Value = cBox_filterMarca.SelectedValue;
330.                     cmd.CommandType = CommandType.StoredProcedure;
331.                       using (SqlDataReader dr = cmd.ExecuteReader())
332.                       {
333.                     //se existirem produtos cuja marca é a seleccionada, instancia
    uma nova datatable e lista-os
334.                         if (dr.HasRows)
335.                         {
336.                         DataTable dt = new DataTable();
337.                         dt.Load(dr);
338.                         dg_produtos.ItemsSource = dt.DefaultView;
339.                         prod.HideCol();
340.                         } //if has rows
341.                       } //datareader
342.                     } //command
343.                     } //if selectedindex
344.                   } //sql connection
345.                 } //try
346.                 catch (Exception ex)
347.                 {
348.                 MessageBox.Show(ex.Message);
349.                 } //catch
```

```
350.             }
351.             //executa na mudança de selecção na combobox categoria
352.             private void cBox_filterCat_SelectionChanged(object sender, SelectionC
    hangedEventArgs e)
353.             {
354.                 try
355.                 {
356.                 using (SqlConnection con = new SqlConnection(CS))
357.                 {
358.                   if (cBox_filterCat.SelectedIndex > 0)
359.                   {
360.                   cBox_filterMarca.SelectedIndex = 0;
361.                   con.Open();
362.                     using (SqlCommand cmd = new SqlCommand("spFilterCategoria", co
    n))
363.                     {
364.                     cmd.Parameters.Add("@Categoria", System.Data.SqlDbType.VarChar
    , 30).Value = cBox_filterCat.SelectedValue;
365.                     cmd.CommandType = CommandType.StoredProcedure;
366.                       using (SqlDataReader dr = cmd.ExecuteReader())
367.                       {
368.                         if (dr.HasRows)
369.                         {
370.                         DataTable dt = new DataTable();
371.                         dt.Load(dr);
372.                         dg_produtos.ItemsSource = dt.DefaultView;
373.                         prod.HideCol();
374.                         } //if has rows
375.                       } //datareader
376.                     } //command
377.                   } //if selectedindex
378.                 } //sql connection
379.               } //try
380.               catch (Exception ex)
381.               {
382.               MessageBox.Show(ex.Message);
383.               } //catch
384.             }
385.
386.             private void Window_Closed(object sender, EventArgs e)
387.             {
388.             Login log = new Login();
389.             log.ShowDialog();
390.             this.Close();
```

```
391.            }
392.  } //AdminProdutos class
```

Produto

Esta classe é instanciada pela classe AdminProdutos, e contém vários métodos relativos aos produtos mostrados e aos elementos de interface.

```
1.   public class Produto : MetroWindow
2.   {
3.     string CS = ConfigurationManager.ConnectionStrings["CS"].ConnectionString;
4.     public DataGrid dg;
5.     public TextBox txb;
6.     public Grid WindowGrid;
7.     public Image imgBox;
8.     public DataTable table;
9.   public void Load_Produtos()
10.  {
11.    //abre a ligação, executa um stored procedure para consultar todos os produtos
     e lista
12.    using (SqlConnection con = new SqlConnection(CS))
13.    {
14.    con.Open();
15.      using (SqlCommand load_dt = new SqlCommand("spAll", con))
16.      {
17.      load_dt.ExecuteNonQuery();
18.      SqlDataAdapter dataAdapter_load = new SqlDataAdapter(load_dt);
19.      table = new DataTable();
20.      dataAdapter_load.Fill(table);
21.      dg.ItemsSource = table.DefaultView;
22.      } //command
23.    } //sql connection
24.  } //load
25.
26.  //limpa todas as textboxes e a imagem
27.  public void Limpar()
28.  {
29.    foreach (UIElement txB in WindowGrid.Children) //limpa o texto de todas as text
     boxes dentro da grid principal
30.    {
31.    TextBox textBox = txB as TextBox;
32.    if (textBox == null)
33.    continue;
34.    textBox.Text = null;
```

```
35.     }
36.     imgBox.Source = null;
37.     dg.SelectedValue = null;
38.   }
39.
40.   //esconde colunas específicas da interface
41.   public void HideCol()
42.   {
43.     dg.Columns[0].Visibility = Visibility.Hidden;
44.     dg.Columns[5].Header = "Preço";
45.     dg.Columns[6].Visibility = Visibility.Hidden;
46.   }
47. }
```

Cliente_orders

Classe com elementos de interface. Contém a funcionalidade da aplicação, o utilizador com permissões de cliente. O objetivo era permitir ao cliente ver os produtos disponíveis, selecioná-los e submeter encomendas. Neste momento apenas permite listar os produtos e adicioná-los a uma segunda datagrid.

```
1.  public partial class Cliente_orders : MetroWindow
2.  {
3.    //connection string, definida no app.config
4.    string CS = ConfigurationManager.ConnectionStrings["CS"].ConnectionString;
5.    Produto prod = new Produto();
6.    public static string username;
7.    SqlDataAdapter da = new SqlDataAdapter();
8.    DataTable dt_encomendas = new DataTable();
9.  public Cliente_orders()
10. {
11.   InitializeComponent();
12. }
13. //carrega a datagrid dos produtos
14. private void Grid_Produtos(object sender, RoutedEventArgs e)
15. {
16.   prod.dg = this.dg_produtos;
17.   prod.Load_Produtos();
18.   prod.HideCol();
19. }
20. //instancia um objeto datarowview com o produto seleccionado, e guarda os valores d
    e cada registo em variáveis
21. private void btn_adicionarItem_Click(object sender, RoutedEventArgs e)
```

```
22. {
23.   DataRow dr;
24.   DataRowView v = (DataRowView)dg_produtos.Items[dg_produtos.SelectedIndex];
25.   int id = (int)v.Row["idProduto"];
26.   string nome = (string)v.Row["Nome"];
27.   string categoria = (string)v.Row["Categoria"];
28.   string marca = (string)v.Row["Marca"];
29.   int stock = (int)v.Row["Stock"];
30.   decimal preco = (decimal)v.Row["Preco"];
31.   byte[] imagem = (byte[])v.Row["Imagem"];
32.     //se ainda não existirem registos na datagrid de encomendas
33.     if (dg_encomenda.Items.Count == 0)
34.     {
35.       //cria uma row na dt de encomendas
36.       dr = dt_encomendas.NewRow();
37.       dr[0] = id;
38.       dr[1] = nome;
39.       dr[2] = categoria;
40.       dr[3] = marca;
41.       dr[4] = stock;
42.       dr[5] = preco;
43.       dr[6] = imagem;
44.       //adiciona a row à dt de encomendas
45.       dt_encomendas.Rows.Add(dr);
46.     }
47.     else
48.     {
49.       //percorre cada row na datagrid de encomendas
50.       foreach (DataRowView drow in dg_encomenda.ItemsSource)
51.       {
52.         //se já existir dá uma mensagem de erro
53.         if (drow[0].ToString() == v.Row["idProduto"].ToString())
54.         {
55.           MessageBox.Show("Produto já existe");
56.           break;
57.         }
58.         else
59.         {
60.           dr = dt_encomendas.NewRow();
61.           dr[0] = id;
62.           dr[1] = nome;
63.           dr[2] = categoria;
64.           dr[3] = marca;
65.           dr[4] = stock;
```

```
66.             dr[5] = preco;
67.             dr[6] = imagem;
68.             dt_encomendas.Rows.Add(dr);
69.             dg_encomenda.Columns[0].Visibility = Visibility.Hidden;
70.             dg_encomenda.Columns[4].Visibility = Visibility.Hidden;
71.             dg_encomenda.Columns[5].Header = "Preço";
72.             dg_encomenda.Columns[6].Visibility = Visibility.Hidden;
73.             break;
74.           } //else
75.         } //foreach
76.       } //else
77.       dg_encomenda.ItemsSource = dt_encomendas.DefaultView;
78.     }
79. //clona a estrutura(colunas) da datatable de produtos
80. private void dg_encomenda_Loaded(object sender, RoutedEventArgs e)
81. {
82.   dt_encomendas = prod.table.Clone();
83. }
84. //executa cada vez que uma seleção na datagrid muda
85. private void dg_produtos_SelectionChanged(object sender, SelectionChangedEventArgs
    e)
86. {
87.   try
88.   {
89.     SqlConnection con = new SqlConnection(CS);
90.     con.Open();
91.   //verifica se está seleccionado um item na datagrid
92.   if (dg_produtos.SelectedItems.Count > 0)
93.   {
94.     //percorre o item seleccionado e guarda cada um dos registos em variáveis, para
     que
95.     //possam ser posteriormente utilizados
96.     for (int i = 0; i < dg_produtos.SelectedItems.Count; i++)
97.     {
98.      DataRowView selectedItem = (DataRowView)dg_produtos.SelectedItems[i];
99.      string idProduto = Convert.ToString(selectedItem.Row[0]);
100.           string Nome = Convert.ToString(selectedItem.Row[1]);
101.           string Categoria = Convert.ToString(selectedItem.Row[2]);
102.           string Marca = Convert.ToString(selectedItem.Row[3]);
103.           decimal Preco = Convert.ToDecimal(selectedItem.Row[4]);
104.           string Imagem = Convert.ToString(selectedItem.Row[5]);
105.           txb_produtosId.Text = idProduto;
106.           txb_produtosNome.Text = Nome;
107.           txb_produtosCat.Text = Categoria;
```

```
108.            txb_produtoMarca.Text = Marca;
109.            txb_produtoPreco.Text = Convert.ToString(Preco);
110.            string query_img = "SELECT Imagem from produtos WHERE idProduto='" + tx
    b_produtosId.Text + "';";
111.              using (SqlCommand cmd = new SqlCommand(query_img, con))
112.              {
113.                using (SqlDataReader reader = cmd.ExecuteReader())
114.                {
115.                  reader.Read();
116.                  if (reader.HasRows && reader["Imagem"] != DBNull.Value) //verific
    a se a query devolve alguma row e se o valor dessa row na coluna "Imagens" não é nu
    lo.
117.                  {
118.                    byte[] img = (byte[])(reader[0]);
119.                    MemoryStream ms = new MemoryStream(img);
120.                    img_Prod.Source = BitmapFrame.Create(ms, BitmapCreateOptions.No
    ne, BitmapCacheOption.OnLoad);
121.                  }
122.                  else
123.                  {
124.                    img_Prod.Source = null;
125.                  }
126.                  reader.Close();
127.              con.Close();
128.                  }//reader
129.              }//command
130.          }//for
131.        }//if
132.
133.          catch (Exception ex)
134.          {
135.            MessageBox.Show(ex.Message);
136.          }
137.        }
138.
139.        private void btn_sair_Click(object sender, RoutedEventArgs e)
140.        {
141.          //Botão fechar janela
142.          Login log = new Login();
143.          this.Close();
144.          log.ShowDialog();
145.        }
146.        private void Window_Closed(object sender, EventArgs e)
147.        {
```

```
148.          Login log = new Login();
149.          log.ShowDialog();
150.          this.Close();
151.      }
152.      } //class
```

3.2.2 Stored Procedures

spAll

```
1.  BEGIN
2.  SELECT p.idProduto, p.Nome, c.Categoria, m.Marca, p.Stock, p.Preco, p.Imagem
3.  FROM produtos AS p
4.  JOIN categorias AS c ON p.idCategoria = c.idCategoria
5.  JOIN marcas as m ON p.idMarca = m.idMarca;
6.  END
```

spFilterCategoria

```
1.  @Categoria varchar(30);
2.  BEGIN
3.  SELECT p.idProduto, p.Nome, c.Categoria, m.Marca, p.Stock, p.Preco, p.Imagem
4.  FROM produtos AS p, categorias c, marcas m
5.  WHERE p.idCategoria = c.idCategoria
6.  AND m.idMarca = p.idMarca
7.  AND c.Categoria = @Categoria;
8.  END
```

spFilterMarca

```
1.  @Marca varchar(30);
2.  BEGIN
3.  SELECT p.idProduto, p.Nome, c.Categoria, m.Marca, p.Stock, p.Preco, p.Imagem
4.  FROM produtos AS p, categorias c, marcas m
5.  WHERE p.idMarca = m.idMarca
6.  AND c.idCategoria = p.idCategoria
7.  AND m.Marca = @Marca;
8.  END
```

spInsertCategoria

```
1.  DECLARE @temp int;
2.  BEGIN
3.  IF NOT EXISTS (SELECT * FROM categorias WHERE Categoria = 'Corsair')
```

```
4.  BEGIN
5.    INSERT INTO categorias(Categoria) VALUES ('Corsair');
6.    SELECT @Temp = idCategoria from categorias where Categoria='Corsair';
7.    INSERT INTO Produtos(idCategoria) VALUES (@Temp);
8.  END
9.  ELSE
10.   BEGIN
11.   SELECT @Temp = idCategoria from categorias where Categoria='Corsair';
12.   INSERT INTO Produtos(idCategoria) VALUES (@Temp);
13.   END
14. END
```

spInsertMarca

```
1.  DECLARE @temp int;
2.  BEGIN
3.  IF NOT EXISTS (SELECT * FROM marcas WHERE Marca = 'Asus')
4.    BEGIN
5.    INSERT INTO marcas(Marca) VALUES ('Asus');
6.    SELECT @Temp = idMarca from marcas where Marca='Asus';
7.    INSERT INTO Produtos(idMarca) VALUES (@Temp);
8.    END
9.  ELSE
10.   BEGIN
11.   SELECT @Temp = idMarca from marcas where Marca='Asus';
12.   INSERT INTO Produtos(idMarca) VALUES (@Temp);
13.   END
14. END
```

spInsProd

```
1.  @Nome varchar(50),
2.  @Categoria varchar(30),
3.  @Marca varchar(30),
4.  @Stock int,
5.  @Preco decimal(6, 2)
6.  AS
7.  DECLARE @Marc_temp int,
8.  @Cat_temp int;
9.  BEGIN
10. IF NOT EXISTS (SELECT * FROM marcas WHERE Marca = @Marca)
11. BEGIN
12.   INSERT INTO marcas(Marca) VALUES (@Marca);
13.   END
```

```
14.   IF NOT EXISTS (SELECT * from categorias WHERE Categoria = @Categoria)
15. BEGIN
16.   INSERT INTO categorias(Categoria) VALUES (@Categoria);
17.   END
18. BEGIN
19.   SELECT @Marc_temp = idMarca from marcas where Marca= @Marca
20.   SELECT @Cat_temp = idCategoria from categorias where Categoria = @Categoria;
21.   INSERT INTO Produtos(Nome, idCategoria, idMarca, Stock, Preco) VALUES (@Nome,
    @Cat_temp, @Marc_temp, @Stock, @Preco);
22. END
23. END
```

spSearch

```
1.  @Search varchar(35)
2.  AS
3.  BEGIN
4.    SELECT p.idProduto, p.Nome, c.Categoria, m.Marca, p.Stock, p.Preco, p.Imagem
5.    FROM produtos AS p, categorias c, marcas m
6.    WHERE p.Nome LIKE '%' + @Search + '%'
7.    AND m.idMarca = p.idMarca
8.    AND c.idCategoria = p.idCategoria;
9.  END
```

spUpdtProd

```
1.  @idProduto int,
2.  @Nome varchar(50),
3.  @Categoria varchar(30),
4.  @Marca varchar(30),
5.  @Stock int,
6.  @Preco decimal(6, 2),
7.  @img image
8.  AS
9.  DECLARE @idMarc_temp int,
10. @idCat_temp int;
11. BEGIN
12.   SELECT @idMarc_temp = idMarca from produtos WHERE idProduto = @idProduto;
13.   SELECT @idCat_temp = idCategoria from produtos WHERE idProduto = @idProduto;
14.   UPDATE marcas SET Marca = @Marca WHERE idMarca = @idMarc_temp;
15.   UPDATE categorias SET Categoria = @Categoria WHERE idCategoria = @idCat_temp;
16.   UPDATE Produtos SET Nome = @Nome, Stock = @Stock, Preco = @Preco, Imagem = @img
    WHERE idProduto = @idProduto;
17. END
```

SpUpdtProd_noImg

```
@idProduto int,
@Nome varchar(50),
@Categoria varchar(30),
@Marca varchar(30),
@Stock int,
@Preco decimal(6, 2)
AS
DECLARE @idMarc_temp int,
@idCat_temp int;
BEGIN
  SELECT @idMarc_temp = idMarca from produtos WHERE idProduto = @idProduto;
  SELECT @idCat_temp = idCategoria from produtos WHERE idProduto = @idProduto;
  UPDATE marcas SET Marca = @Marca WHERE idMarca = @idMarc_temp;
  UPDATE categorias SET Categoria = @Categoria WHERE idCategoria = @idCat_temp;
  UPDATE Produtos SET Nome = @Nome, Stock = @Stock, Preco = @Preco WHERE idProduto
= @idProduto;
END
```

3.2.3 Scripts de criação de tabelas

Utilizadores

```
CREATE TABLE [dbo].[utilizadores](
    [idUser] [int] IDENTITY(1,1) NOT NULL,
    [username] [varchar](25) NOT NULL,
    [pass] [varchar](25) NOT NULL,
    [Nome] [varchar](50) NULL,
    [Morada] [varchar](50) NULL,
    [Telemovel] [int] NULL,
    [Email] [varchar](50) NULL,
    [acesso] [tinyint] NOT NULL,
 CONSTRAINT [PK_utilizadores] PRIMARY KEY CLUSTERED
```

Categorias

```
CREATE TABLE [dbo].[categorias](
    [idCategoria] [int] IDENTITY(1,1) NOT NULL,
    [Categoria] [varchar](30) NOT NULL,
 CONSTRAINT [PK_Categorias] PRIMARY KEY CLUSTERED
```

Encomendas

```
1.  CREATE TABLE [dbo].[encomendas](
2.      [idEncomenda] [smallint] IDENTITY(1,1) NOT NULL,
3.      [idUser] [int] NOT NULL,
4.      [Data] [date] NULL,
5.   CONSTRAINT [PK_encomendas] PRIMARY KEY CLUSTERED
6.  (
7.      [idEncomenda] ASC
8.  )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_R
    OW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
9.  ) ON [PRIMARY]
10.
11. GO
12.
13. ALTER TABLE [dbo].[encomendas]  WITH CHECK ADD  CONSTRAINT [FK_encomendas_utilizado
    res] FOREIGN KEY([idUser])
14. REFERENCES [dbo].[utilizadores] ([idUser])
15. GO
16.
17. ALTER TABLE [dbo].[encomendas] CHECK CONSTRAINT [FK_encomendas_utilizadores]
18. GO
```

Marcas

```
1.  CREATE TABLE [dbo].[marcas](
2.      [idMarca] [int] IDENTITY(1,1) NOT NULL,
3.      [Marca] [varchar](30) NOT NULL,
4.   CONSTRAINT [PK_Marcas] PRIMARY KEY CLUSTERED
```

Produtos

```
1.  CREATE TABLE [dbo].[produtos](
2.      [idProduto] [int] IDENTITY(1,1) NOT NULL,
3.      [Nome] [varchar](50) NOT NULL,
4.      [idCategoria] [int] NOT NULL,
5.      [idMarca] [int] NOT NULL,
6.      [Stock] [int] NOT NULL,
7.      [Preco] [decimal](6, 2) NOT NULL,
8.      [Imagem] [image] NULL,
9.   CONSTRAINT [PK_artigos] PRIMARY KEY CLUSTERED
10. (
11.     [idProduto] ASC
```

```
12. )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_R
    OW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
13. ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
14.
15. GO
16.
17. SET ANSI_PADDING OFF
18. GO
19.
20. ALTER TABLE [dbo].[produtos]  WITH CHECK ADD  CONSTRAINT [FK_produtos_Categorias] F
    OREIGN KEY([idCategoria])
21. REFERENCES [dbo].[categorias] ([idCategoria])
22. GO
23.
24. ALTER TABLE [dbo].[produtos] CHECK CONSTRAINT [FK_produtos_Categorias]
25. GO
26.
27. ALTER TABLE [dbo].[produtos]  WITH CHECK ADD  CONSTRAINT [FK_produtos_marcas] FOREI
    GN KEY([idMarca])
28. REFERENCES [dbo].[marcas] ([idMarca])
29. GO
30.
31. ALTER TABLE [dbo].[produtos] CHECK CONSTRAINT [FK_produtos_marcas]
32. GO
```

Produtos_encomendas

```
1.  CREATE TABLE [dbo].[produtos_encomendas](
2.      [idProduto] [int] NOT NULL,
3.      [idEncomenda] [smallint] NOT NULL,
4.      [Quantidade] [nchar](10) NULL,
5.   CONSTRAINT [PK_produtos_encomendas] PRIMARY KEY CLUSTERED
6.  (
7.      [idProduto] ASC,
8.      [idEncomenda] ASC
9.  )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_R
    OW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
10. ) ON [PRIMARY]
11.
12. GO
13.
```

```
14. ALTER TABLE [dbo].[produtos_encomendas]  WITH CHECK ADD  CONSTRAINT [FK_produtos_en
    comendas_encomendas1] FOREIGN KEY([idEncomenda])
15. REFERENCES [dbo].[encomendas] ([idEncomenda])
16. GO
17.
18. ALTER TABLE [dbo].[produtos_encomendas] CHECK CONSTRAINT [FK_produtos_encomendas_en
    comendas1]
19. GO
20.
21. ALTER TABLE [dbo].[produtos_encomendas]  WITH CHECK ADD  CONSTRAINT [FK_produtos_en
    comendas_produtos] FOREIGN KEY([idProduto])
22. REFERENCES [dbo].[produtos] ([idProduto])
23. GO
24.
25. ALTER TABLE [dbo].[produtos_encomendas] CHECK CONSTRAINT [FK_produtos_encomendas_pr
    odutos]
26. GO
```

3.2.4 Interface

Na interface, foi utilizado o "Mahapps.Metro", um package obtido através do NuGet Package Manager. Este package permite a criação de *interfaces* mais modernas e apelativas, com um conjunto de elementos customizados.

Passos para começar a utilizar:

No XAML

1. No xaml da *window*, dentro da tag *Window*, adicionar o seguinte atributo:

```
"xmlns:Controls="clr-namespace:MahApps.Metro.Controls;assembly=MahApps.Metro"
```

2. Alterar a tag "<Window ... para "`<Controls:MetroWindow ...`

No CodeBehind

Adicionar o namespace "MahApps.Metro.Controls" e alterar a classe base de "Window", para "MetroWindow":

```
"public partial class MainWindow : MetroWindow"
```

Todos os recursos do "MahApps.metro" estão contidos em dicionários separados de recursos. Para que a maioria dos controlos adote esses recursos, têm de ser adicionados esses dicionários ao ficheiro "App.xaml" do projeto:

```
  <Application.Resources>
    <ResourceDictionary>
      <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Controls.xaml" />
        <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Fonts.xaml" />
        <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Colors.xaml" />
        <!-- Accent and AppTheme setting -->
        <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/Blue.xaml" />
        <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/BaseLight.xaml" />
      </ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
  </Application.Resources>
</Application>
```

("MahApps.Metro - Quick Start," n.d.)

Login.xaml

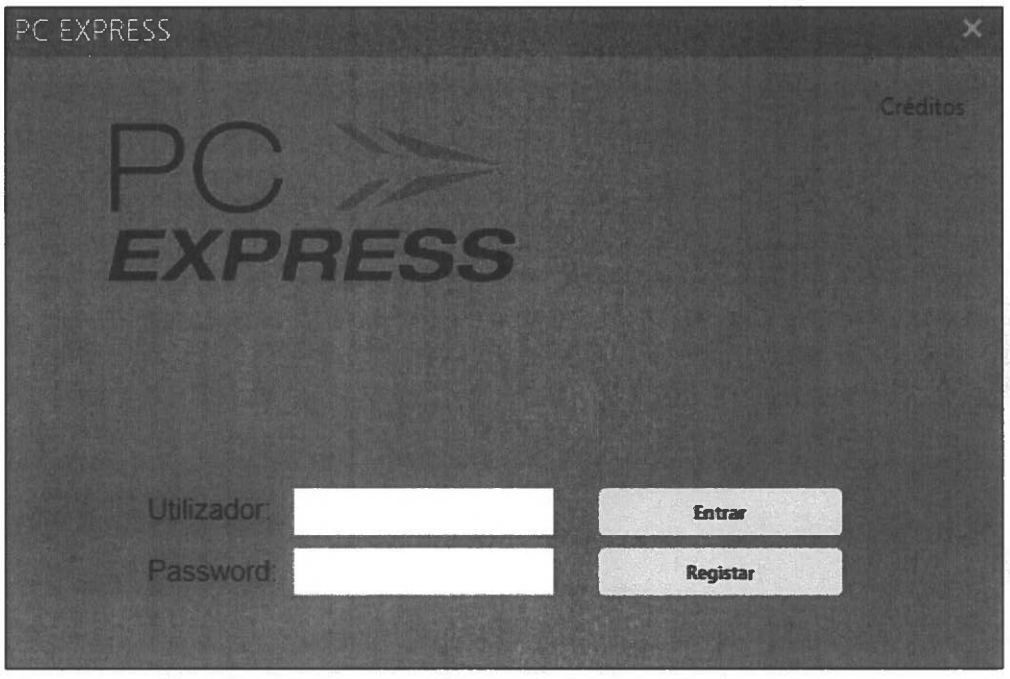


Figura 59 - Login.xaml

Register.xaml

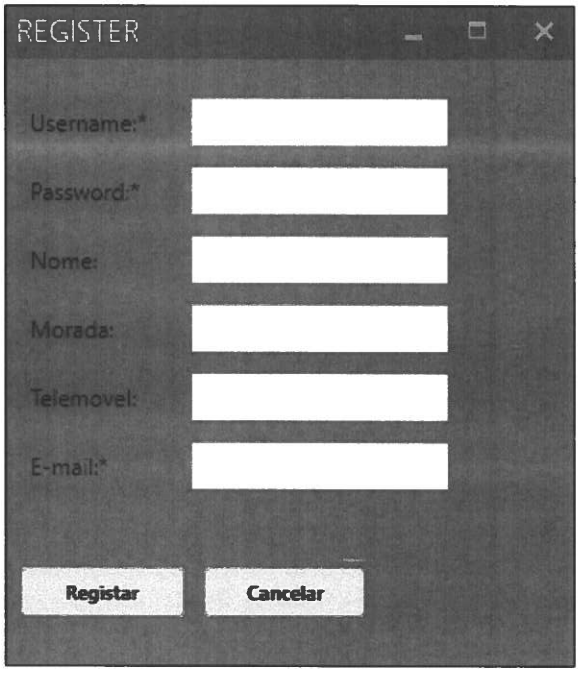


Figura 60 - Register.xaml

AdminProdutos.xaml

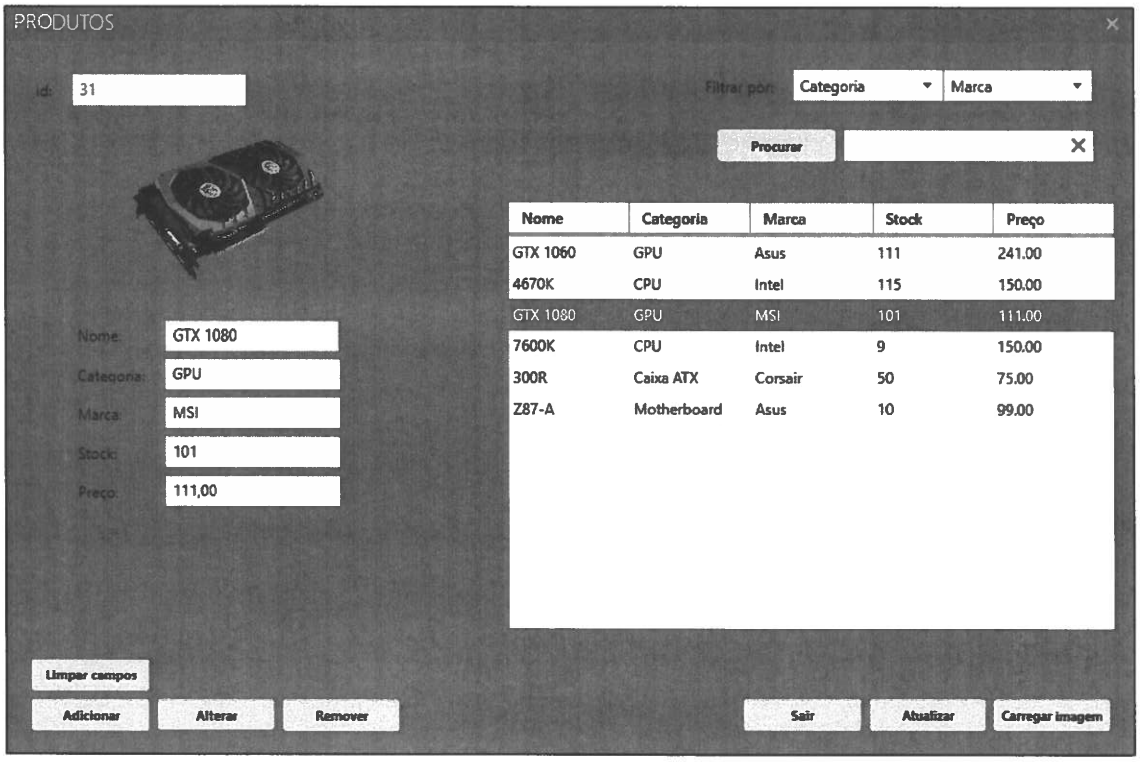


Figura 61 - AdminProdutos.xaml

Cliente_orders.xaml

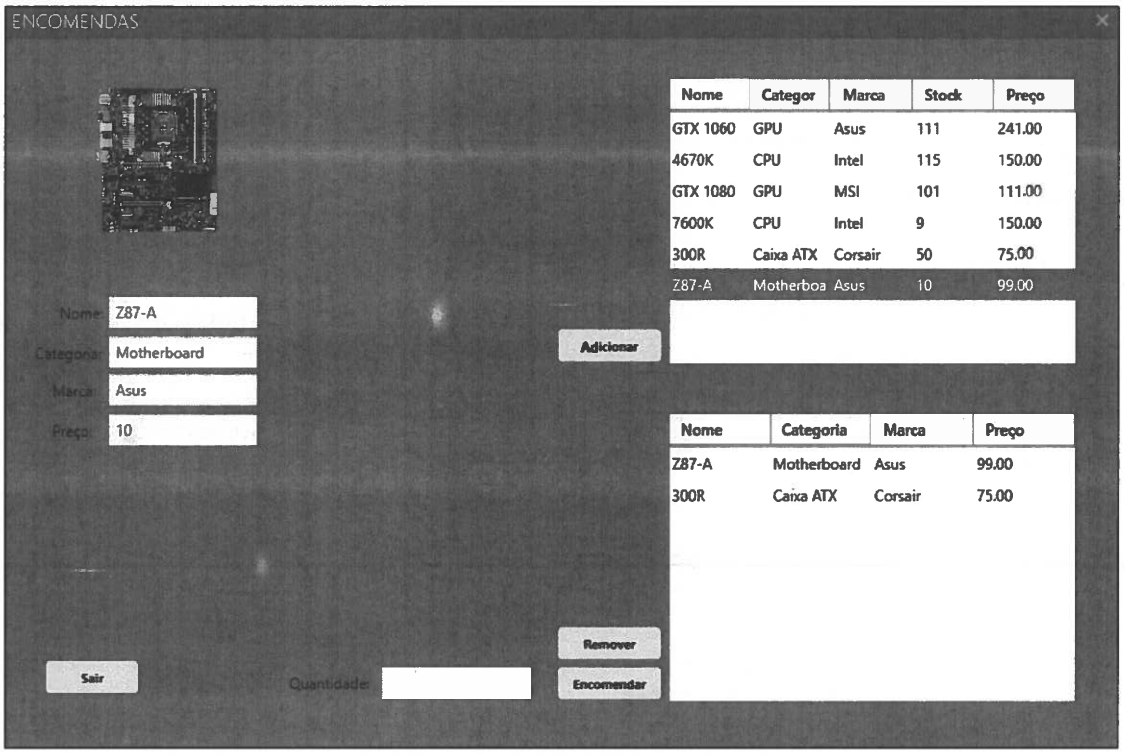


Figura 62 - Cliente_orders.xaml

3.2.5 Deployment

O *deployment* da aplicação é feito através de um instalador .msi, criado no Visual Studio com um projeto do tipo "Visual Studio Installer".

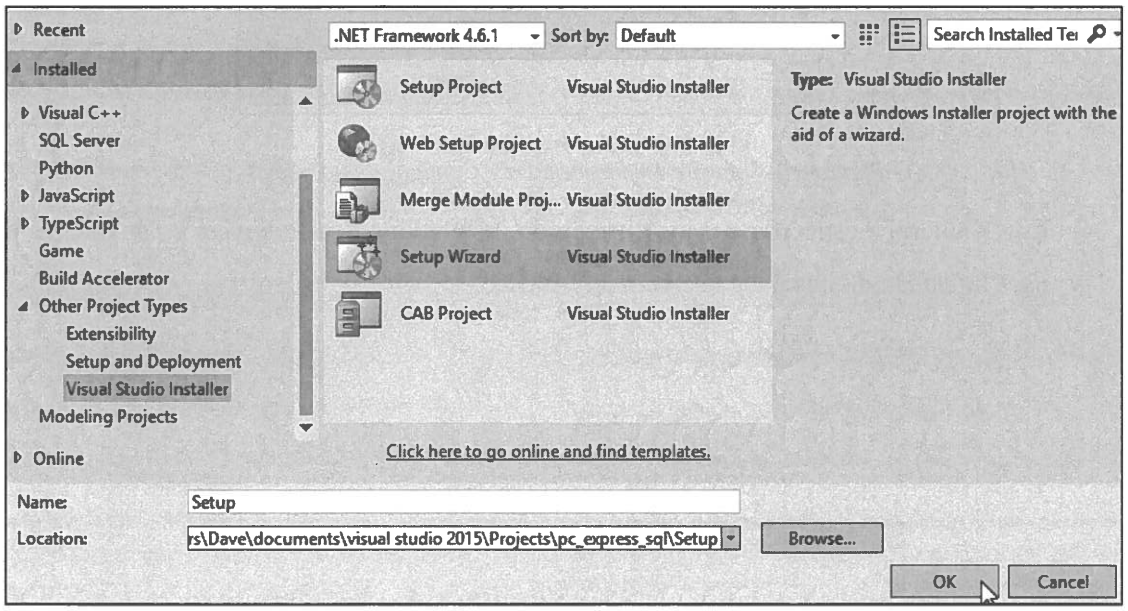


Figura 63 - Visual Studio - New Installer Project

Neste projeto foi adicionado apenas o *primary output* do projeto principal, e foram adicionadas algumas informações tais como o nome da aplicação, ícone, atalhos, nome do autor, fabricante, entre outras.

Este instalador foi inserido numa política de *Group Policy*, para instalação automática nas máquinas cliente.

1. Colocar o instalador numa pasta partilhada e acessível às máquinas do domínio.
2. No "DC1 – Group Policy Management", criar um novo GPO para a unidade organizacional "VPN Users" e abrir o seu editor.
3. Em "User Configuration – Software Settings – Right-click em Software Installation - Properties".

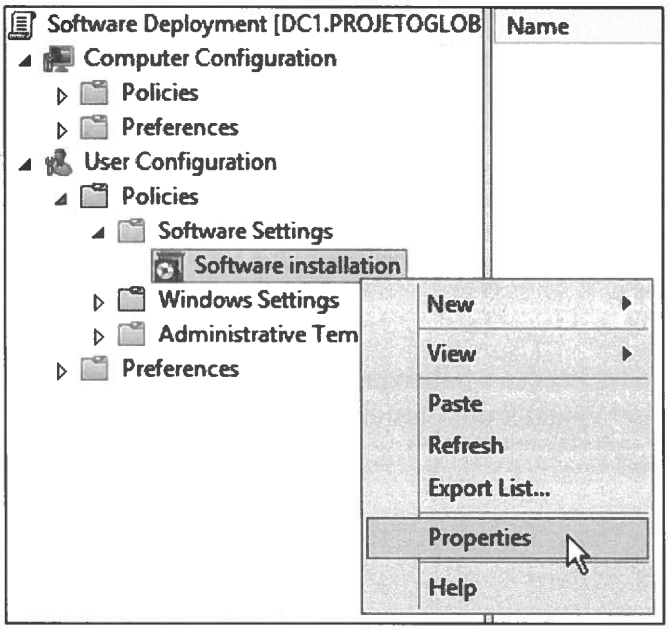


Figura 64 - Software Deployment User Configuration Policies

4. Em "General", especificar a localização do instalador a ser *deployed*, e nas opções de *user interface* da instalação selecionar "Basic".

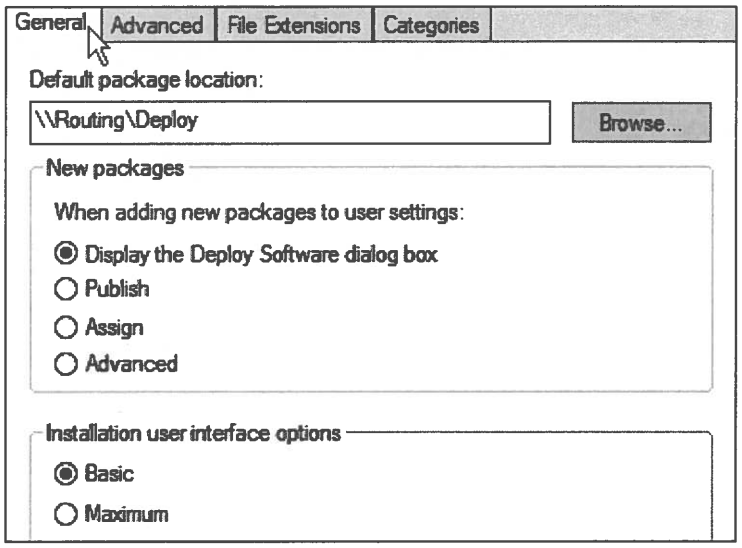


Figura 65 - Software Installation Properties

5. Em "Advanced", e habilitar a opção "Uninstall the applications when they fall out of the scope of management".

Existem 2 formas para fazer o *deployment* de uma aplicação no Active Directory. Pode ser *published* ou *assigned*. Se for *published*, só pode sê-lo feito para utilizadores, enquanto que se for *assigned* pode sê-lo feito para utilizadores ou computadores.

Published – Pode ser published apenas para utilizadores, e a instalação tem de ser feita pelo próprio utilizador, ficando esta disponibilizada no *Add/Remove Programs*.

Assigned – Pode ser *assigned* a utilizadores ou computadores. Neste tipo de *deployment*, parte da instalação da aplicação é forçada, e é feita automaticamente no momento do *login* do utilizador. Essa instalação é finalizada no momento em que o utilizador iniciar a aplicação pela primeira vez.

(Posey, n.d.)

6. No espaço em branco, "New – Package", selecionar o pacote de instalação com o método de *deployment* "Assigned".

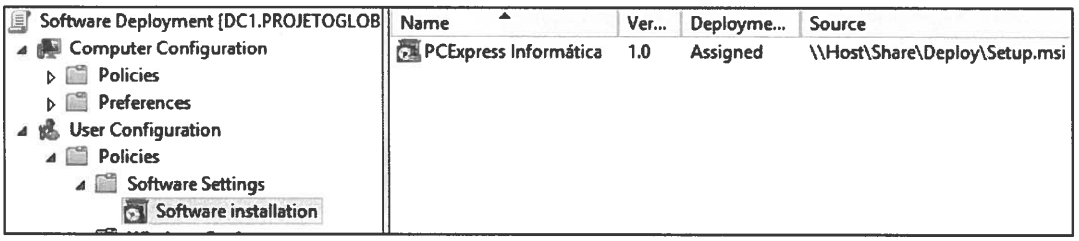


Figura 66 - New Software Package

7. Aceder às propriedades do *package*, e habilitar a opção "Install this application at logon".

Settings do GPO:

Software Deployment

Data collected on: 12/06/2017 00:15:27 — show all

Computer Configuration (Enabled) — hide

No settings defined.

User Configuration (Enabled) — hide

Policies — hide

Software Settings — hide

Assigned Applications — hide

PCExpress Informática — hide

Product Information — show

Deployment Information — hide

General	Setting
Deployment type	Assigned
Deployment source	\\Routing\Deploy\Setup.msi
Installation user interface options	Basic
Uninstall this application when it falls out of the scope of management	Enabled
Do not display this package in the Add/Remove Programs control panel	Disabled
Install this application at logon	Enabled

Advanced Deployment Options	Setting
Ignore language when deploying this package	Disabled
Make this 32-bit X86 application available to Win64 computers	Enabled
Include OLE class and product information	Enabled

Diagnostic Information	Setting
Product code	{a1cc354d-96dd-4fe4-b615-b115ace8234c}
Deployment Count	0

Figura 67 - Software Deployment GPO Settings

Conclusão

Este projeto teve como objetivos principais o desenvolvimento de um laboratório de rede virtual e uma aplicação em C# com ligação a base de dados. Neste sentido, surgiram vários obstáculos, que na sua maioria foram ultrapassados ao longo do processo de investigação. Aliado a esse processo, o conhecimento adquirido no decorrer da Licenciatura em Informática foi essencial para as várias fases em direção ao alcance dos objetivos definidos, especialmente nas unidades curriculares de *networking*, criptografia, programação, e nas várias unidades lecionadas no âmbito da pós-graduação em Virtualização e Cloud Computing.

Como foi demonstrado, a tecnologia de virtualização foi revolucionária, sendo cada vez mais utilizada no mundo empresarial, pois oferece soluções para vários problemas, tais como o excesso de ocupação de espaço físico com *hardware*, os consumos energéticos excessivos com equipamento, e o mau aproveitamento de recursos. Para além disso, serve como base para a tecnologia de *Cloud computing*, que tem sido um grande fator no aumento de produtividade empresarial. No decorrer do desenvolvimento do laboratório, os benefícios da virtualização tornaram-se evidentes, pois apesar das limitações de *hardware* existentes, não deixou de ser possível criar um ambiente virtualizado composto por 6 máquinas, configuradas de forma a que pudessem satisfazer as necessidades de uma pequena empresa.

Relativamente à aplicação desenvolvida, apesar de cumprir com os requisitos propostos para o projeto, alguns objetivos não foram alcançados, devido a alguns obstáculos que não foram ultrapassados. Alguns dos passos seguintes no seu desenvolvimento seriam a implementação de funcionalidades que permitissem ao utilizador efetuar encomendas de produtos, com gestão de *stock* na base de dados, e otimização de código para um melhor desempenho.

A realização deste projeto foi muito valiosa para mim, potencializando o meu conhecimento na área e a minha capacidade de investigação e resolução de problemas.

Referências

Brandão, P. (2016, June) Green Data Center, Exame Informática,22.

1.1.1. Brief History of Virtualization. (n.d.). Retrieved June 24, 2017, from https://docs.oracle.com/cd/E26996_01/E18549/html/VMUSG1010.html

Active Directory Certificate Services Overview. (2013). Retrieved May 23, 2017, from https://technet.microsoft.com/en-us/library/hh831740(v=ws.11).aspx

Active Directory Domain Services Overview. (2013). Retrieved May 23, 2017, from https://technet.microsoft.com/en-us/library/hh831484(v=ws.11).aspx

Active Directory Replication. (n.d.). Retrieved May 26, 2017, from https://technet.microsoft.com/en-us/library/cc961788.aspx

Barnier, R., Brown, C., & Dittmann, C. (2008). Virtualization and Its Benefits. *AITP – Research and Strategy Advisory Group*, 1–40. Retrieved from http://barniergroup.com/docs/wp/Virtualization and Its Benefits.pdf

Brian, O., Brunschwiler, T., Dill, H., Christ, H., Falsafi, B., Fischer, M., … Kaiserswerth, M. (2009). Cloud Computing ArChiteCture. *Computing*, *40*(June), 29–53. https://doi.org/10.1145/1556154.1556175

Broderick, K., & Jenkins, D. (2015). The Efficient Data Center.

Certificates and NPS. (n.d.). Retrieved May 25, 2017, from https://technet.microsoft.com/en-us/library/cc772401(v=ws.10).aspx

Chappell, D., Microsoft, C., & All, C. (2013). *Introducing Microsoft System Center. Management*.

Configure DNS records. (n.d.). Retrieved June 7, 2017, from https://technet.microsoft.com/en-us/library/mt473798(v=exchg.150).aspx

Create a New Certificate Template. (n.d.). Retrieved May 23, 2017, from https://technet.microsoft.com/en-us/library/cc753370(v=ws.11).aspx

Desai, A., Oza, R., Sharma, P., & Patel, B. (2013). Hypervisor : A Survey on Concepts and Taxonomy, (3), 222–225.

Does Exchange play well with Hyper-V Dynamic Memory? (So many questions. So little time.

Part 21.) | Full of I.T. (2012). Retrieved June 5, 2017, from https://blogs.technet.microsoft.com/kevinremde/2012/03/06/does-exchange-play-well-with-hyper-v-dynamic-memory-so-many-questions-so-little-time-part-21/

Elfassy, D. (2013). *Mastering Exchange Server 2013*. Sybex. https://doi.org/10.1017/CBO9781107415324.004

Fallis, A. . (2013). *SQL For Dummies 5th Edition*. *Journal of Chemical Information and Modeling* (Vol. 53). https://doi.org/10.1017/CBO9781107415324.004

Introduction to Configuration and Management. (n.d.). Retrieved May 27, 2017, from https://technet.microsoft.com/en-us/library/cc938748.aspx

MacDonald, M., Freeman, A., & Szpuszta, M. (2010). *Pro ASP.NET 4 in C# 2010*. https://doi.org/10.1007/978-1-4302-2530-0

MahApps.Metro - Quick Start. (n.d.). Retrieved June 11, 2017, from http://mahapps.com/guides/quick-start.html#installation

Mail flow: Exchange 2013 Help. (2013). Retrieved June 7, 2017, from https://technet.microsoft.com/en-us/library/aa996349(v=exchg.150).aspx

Microsoft. (2016). Hybrid and Hyperscale Cloud with SQL Server 2016, (April).

Minasi, M., Greene, K., Booth, C., Butler, R., McCabe, J., Panek, R., … Roth, S. (2014). *Mastering Windows Server 2012 R2*. Sybex.

Mistry, R., & Misner, S. (2014). *Introducing Microsoft SQL Server 2014 Technical Overview*.

Network Policies. (n.d.). Retrieved May 25, 2017, from https://technet.microsoft.com/en-us/library/cc754107(v=ws.10).aspx

Network Policy and Access Services Overview. (n.d.). Retrieved May 24, 2017, from https://technet.microsoft.com/en-us/library/cc731321(v=ws.11).aspx

Network Policy Server (NPS) | Microsoft Docs. (2017). Retrieved May 24, 2017, from https://docs.microsoft.com/en-us/windows-server/networking/technologies/nps/nps-top

Oppel, A., & Sheldon, R. (2009). *SQL A Beginner ' s Guide Third Edition*. *Fortune* (3rd ed.). https://doi.org/10.1036/0071548645

Overview of Active Directory Sites and Services. (2012). Retrieved May 23, 2017, from

https://technet.microsoft.com/en-us/library/cc731907(v=ws.11).aspx

Pfaff, B., Pettit, J., Koponen, T., Amidon, K., Casado, M., & Shenker, S. (2009). Extending Networking into the Virtualization Layer. *8th ACM Workshop on Hot Topics inNetworks, VIII*, 6. Retrieved from http://www.icsi.berkeley.edu/pubs/networking/extendingnetworking09.pdf

Pope, M. (2012). Introducing ASP . NET Web Pages 2.

Posey, B. (n.d.). Using Group Policy to Deploy Applications - TechGenix. Retrieved June 11, 2017, from http://techgenix.com/Group-Policy-Deploy-Applications/

Receive connectors. (n.d.). Retrieved June 7, 2017, from https://technet.microsoft.com/en-us/library/aa996395(v=exchg.150).aspx

Roles, Role Services, and Features. (n.d.). Retrieved May 23, 2017, from https://technet.microsoft.com/en-us/library/cc754923(v=ws.11).aspx

Savill, J. (2014). *Mastering Hyper-V 2012 R2 with System Center and Windows Azure*. Sybex.

Schaefer, K., Cochran, J., Forsyth, S., Glenderning, D., & Perkins, B. (2013). Professional Microsoft IIS 8.

Send connectors. (n.d.). Retrieved June 7, 2017, from https://msdn.microsoft.com/en-us/library/aa998662(v=exchg.160).aspx

Shared Secrets for NPS and RADIUS Clients. (n.d.). Retrieved May 24, 2017, from https://technet.microsoft.com/en-us/library/cc771660(v=ws.10).aspx

Talaber, R., Brey, T., & Lamers, L. (2009). Using virtualisation to improve data center efficiency. *The Green Grid, Tech. Rep*, 1–22.

The Cloud OS Becomes Reality. (2012). Retrieved June 10, 2017, from https://blogs.technet.microsoft.com/serverandtools/2012/09/04/the-cloud-os-becomes-reality-windows-server-2012-now-available/

Troelsen, A., & Japikse, P. (2016). *Apress.C.7th.Edition.May.2016.ISBN.1484213335* (7º).

Tulloch, M., & Team, W. S. (2013). *Introducing Windows Server 2012 R2* (Vol. 15). Retrieved from https://books.google.com/books?id=IlxuAwAAQBAJ&pgis=1

Types of Certification Authorities. (n.d.). Retrieved May 23, 2017, from

https://technet.microsoft.com/en-us/library/cc732368(v=ws.11).aspx

Understanding Sites, Subnets, and Site Links. (2012). Retrieved May 22, 2017, from https://technet.microsoft.com/en-us/library/cc754697(v=ws.11).aspx

Understanding the Self-Signed Certificate in Exchange. (n.d.). Retrieved June 9, 2017, from https://technet.microsoft.com/en-us/library/bb851554(v=exchg.80).aspx

VPN connection types (Windows 10) | Microsoft Docs. (2017). Retrieved May 27, 2017, from https://docs.microsoft.com/en-us/windows/access-protection/vpn/vpn-connection-type

Ziembicki, D., & Tulloch, M. (2014). *Microsoft System Certer integrated Cloud Platform.*