ISTEC INSTITUTO SUPERIOR DE TECNOLOGIAS AVANÇADAS

VIRTUALIZAÇÃO DE SERVIDORES

CONCEÇÃO E DESENVOLVIMENTO DE BIBLIOTECA VIRTUAL

Licenciatura Informática – ISTEC Lisboa

2015/2016

Ricardo Ribeiro Torcato Nunes

Numero 1929

Coordenador: Prof. Doutor Pedro Brandão

Lisboa 2016

Esta página foi intencionalmente deixada em branco

Índice

Índice II

Dedicatória IV

Agradecimentos V

Resumo VI

Abstract VII

Lista de Abreviaturas VIII

Lista de Figuras IX

1. Introdução 1

2. Estado da Arte 3

2.1. Virtualização 3
2.2. Hypervisor 4
2.3. Arquitetura Hyper-V 6
2.4. SQL 7
2.5. ADO.NET 8

3. Contextualização 9

4. Desenvolvimento 11

4.1. Introdução 11
4.2. Arquitetura de Máquinas Virtuais 11
4.2.1. Máquinas Virtuais 11
4.2.2. Controlador de Domínio 12
4.2.3. Servidor de Base de Dados 13
4.2.1. Servidor SCVMM 14
4.3. Base de Dados 18
4.4. Preparação do Laboratório 21
4.4.1. Servidor DHCP 21
4.4.2. Configuração da Active Directory 21
4.4.1. Group Policy (GPO) 23

4.1. Desenvolvimento 27
4.1.1. Login no domínio 27
4.1.2. Ligação à Base de Dados 30
4.2. Interface Gráfico 32
4.2.1. Listar Artigos 32
4.2.2. Adicionar ou Editar Artigo 42
4.2.3. Editar Autores / Editoras / Tipos de Artigo 51
4.1. Instalador da Aplicação 59
4.1.1. InstallShield 2015 Limited Edition 59
5. Conclusão 63
Bibliografia 64

Dedicatória

Dedico este trabalho à minha mulher e ao meu filho por todo o apoio, carinho e amizade.

Agradecimentos

A elaboração deste trabalho não teria sido possível sem o apoio de algumas pessoas, às quais manifesto o meu agradecimento.

À minha família, especialmente à minha Sogra pelo apoio durante toda a Licenciatura e em especial durante os meses de execução deste trabalho.

Ao meu colega de Licenciatura, Nuno Mendes, pela ajuda fundamental no desenvolvimento da aplicação. Sem ela este trabalho não teria sido possível.

Aos meus colegas de Licenciatura, Duarte Bizarra, Ricardo Morais, Nuno Azevedo, Gonçalo Abreu e Tiago Soveral. Nos momentos mais complicados servimos de incentivo uns aos outros.

Aos professores Prof. Doutor Pedro Brandão e Eng. Pedro Crispim, pelo apoio no desenho e construção do laboratório.

Resumo

A Virtualização de servidores consiste na possibilidade de partilhar os recursos de *hardware*, na medida que o mesmo sistema pode executar vários sistemas operativos e aplicações isoladas em simultâneo.

O objetivo deste trabalho consiste em demonstrar a capacidade da virtualização, disponibilizando máquinas virtuais através de um servidor de *Fabric* e construindo uma rede domínio com um programa e Base de Dados baseados em servidores virtuais.

Este trabalho apresenta a implementação de uma Biblioteca Virtual com capacidade de consultar e descarregar livros em formato pdf.

Palavras-chave: Máquinas Virtuais, ADO, SQL, HYPER-V, SCVMM

Abstract

Virtualization of servers is the ability to share hardware resources, as the same system can run multiple operating systems and isolated applications simultaneously.

The objective of this study is to demonstrate the capability of virtualization, providing virtual machines through a Fabric server and building a domain network with a program and database based on Virtual Servers.

This work presents the implementation of a virtual library with the ability to browse and download books in pdf format.

Keywords: Virtual Machines, ADO, SQL, HYPER-V, SCVMM

Lista de Abreviaturas

SO - Sistema Operativo

API - Application Programming Interface

IOMMU - Input–output memory management unit

CPU - Central Processing Unit

ANSI - American National Standards Institute

ISO - Organização Internacional de Normalização

SCVMM - System Center Virtual Machine Manager

VMM - Virtual Machine Manager

AD - Active Directory

PDF - Portable Document Format

NAT - Network address translation

OU - Organizational Unit

GPO - Group Policy Object

PC - Personal Computer

BD – Data Base

Lista de Figuras

Figura 1 - Hypervisor do Tipo 1 (NCRIS, 2015) 5
Figura 2 - Hypervisor do Tipo 2 (NCRIS, 2015) 5
Figura 3 - Informação básica do controlador de domínio 12
Figura 4 - Informação básica do servidor de Bases de Dados 13
Figura 5 - Abertura de portas na Firewall 14
Figura 6 - VMM, VM Templates 14
Figura 7 - VMM, Create VM from Template 15
Figura 8 - VMM, VM Template (OS Configuration) 15
Figura 9 - VMM, VMs 16
Figura 10 - AD, "ComputadoresCliente" 16
Figura 11 - VM 8.1 Cliente, Primeiro login 17
Figura 12 - Win 8.1 Desktop 17
Figura 13 - Base de Dados, Tabela Autores 18
Figura 14 - Base de Dados, Tabela Editora 18
Figura 15 - Base de Dados, Tabela TipoDocumento 19
Figura 16 - Base de Dados, Foreing Key Relationships 19
Figura 17 - Base de Dados, Tabela Document 20
Figura 18 - Base de Dados, Diagrama 20
Figura 19 - DHCP Server 21
Figura 20 - Unidade Organizacional “Programa” e Grupos de segurança. 22
Figura 21 - Unidade Organizacional “System Center” e Grupos de segurança. 23
Figura 22 - GPO, Instalar a Aplicação 24
Figura 23 - Menu Iniciar, Atalho para o Programa 24
Figura 24 - Atalho Ambiente Trabalho 24
Figura 25 - GPO, Instalar o Adobe Reader XI 25
Figura 26 - GPO, Atalho para o OneDrive 25

Figura 27 - GPO, organização OU 26
Figura 28 - PowerShell, redircmp 26
Figura 29 - Arranque da aplicação 27
Figura 30 - Init Window, login 27
Figura 31 - Init Window,cs 27
Figura 32 - SecurityService.cs, validação de utilizador 28
Figura 33 - SecurityService.cs, validação de perfil e utilizador 28
Figura 34 - ConfigHelper.cs, validação do nome dos Grupos 29
Figura 35 - App.config, valores dos grupos de segurança 29
Figura 36 - InitWindow, sucesso ou erro 30
Figura 37 - InitWindow, Ligação à Base de Dados 30
Figura 38 - DBManger, construtor 31
Figura 39 - App.config, Connection String da AD 31
Figura 40 - MainWindow 32
Figura 41 - MainWindow, Search bar 33
Figura 42 - TextBox de Pesquisa, XAML 33
Figura 43 - Implementação da interface INotifyPropertyChanged 34
Figura 44 - PreviewMouseWhell (scroll do rato) 35
Figura 45 - Estilo de TextWrap 35
Figura 46 - Conversor de bits em Imagem, XAML 35
Figura 47 - Classe BinaryImageConverter 36
Figura 48 - StackPanel Botões 36
Figura 49 - Refresca Botões 37
Figura 50 - Retira botões (perfil de leitura) 37
Figura 51 - MouseDoubleClick 38
Figura 52 - Botão Adicionar Artigos 38
Figura 53 - Botão Editar Artigos 39

Figura 54 - Botão Eliminar Artigos 39
Figura 55 - DbServices, DeleteDoc 39
Figura 56 - Botão Download 40
Figura 57 – SaveBytesToFile 40
Figura 58 - MainWindow, StatusBar xaml 41
Figura 59 - MainWindow, StatusBar .cs 41
Figura 60 - Editar Artigos 42
Figura 61 - Binding, ValidationRules 43
Figura 62 - Class ValidationRule 43
Figura 63 - Exemplo gráfico campo preenchimento obrigatório 43
Figura 64 - ComboBox 44
Figura 65 - Exemplo ComboBox XAML 44
Figura 66 - Preenchimento das ComboBox, CS 44
Figura 67 - Preenchimento das ComboBox, relação com IDs 45
Figura 68 - Exemplo de Botão Editar em XAML 45
Figura 69 - GerirTipoDocumento_Click 46
Figura 70 - Botão upload pdf, XAML 46
Figura 71 - Evento Upload PDF 47
Figura 72 - Janela Windows, selecione um PDF 47
Figura 73 - Funçoes GetCover e GetContent para convert pdf 48
Figura 74 - Validação do botão Gravar Artigo, XAML 49
Figura 75 - Exemplo gráfico, botão Gravar Indisponível 49
Figura 76 - Evento de Gravação de Artigos 49
Figura 77 - DBService, Adicionar Artigo 50
Figura 78 - DBService, Editar Artigo 50
Figura 79 - Evento Botão Cancelar 51
Figura 80 - gridTipoEditora_SelectedCellsChanged 52

Figura 81 - Editar Editoras 52
Figura 82 - Editar Editoras, StackPanel Botões 53
Figura 83 - RefreshBtnsState, Ocultação de Botões 53
Figura 84 - RefreshState, Ocultação de Botões II 53
Figura 85 - Editar editoras, BtnCancel_Click 54
Figura 86 - Mensagens de erro, validação de campos 54
Figura 87 - Adicionar Editora, Validação de campos 55
Figura 88 - Inserir Editora 55
Figura 89 - DBService, AddEditora 55
Figura 90 - Identifica ID de nova editora e fecha janela 56
Figura 91 - Editar Editora 56
Figura 92 - DBService, EditEditora 57
Figura 93 - Confirmação Eliminar e validação 57
Figura 94 - Confirmação eliminar Editora 58
Figura 95 - Erro. A editora encontra-se associada 58
Figura 96 - EditoraEmUso 58
Figura 97 - DeleteEditora 58
Figura 98 - InstallShield 2015 Limited Edition 59
Figura 99 - Atalho da aplicação 59
Figura 100 - InstallShield, shortcut configuration 60
Figura 101 - Instalador da aplicação parte 1 60
Figura 102 - Instalador da aplicação parte 2 61
Figura 103 - Instalador da aplicação parte 2 61

Esta página foi intencionalmente deixada em branco

1. Introdução

Durante alguns anos os sistemas informáticos basearam-se em três componentes: *hardware*, sistema operativo e aplicações. Este fato levou a que se revelassem alguns constrangimentos, uma vez que, as aplicações só podem ser executadas sobre um sistema operativo para o qual foram desenvolvidas. Neste sentido era necessário ter vários equipamentos com os respetivos sistemas operativos e aplicações.

A Virtualização de computadores permite que o *hardware* execute vários sistemas operativos em simultâneo e de forma isolada.

A virtualização proporciona novos recursos, como a Alta Disponibilidade, permitindo que um sistema virtualizado mude de hospedeiro sem comprometer os sistemas em produção. De igual forma simplifica o sistema de *Backup* e *Restore* permitindo restaurar máquinas com tempos de reposição muito baixos bem como a rápida disponibilização de novas máquinas virtuais através de modelos pré-definidos.

Ao longo deste trabalho pretende-se demonstrar as potencialidades da tecnologia de Virtualização em ambientes empresariais através de uma aplicação prática da mesma.

Este documento compreende quatro capítulos para além do presente, organizando-se da seguinte forma:

No segundo capítulo é descrito o Estado da Arte, no qual é apresentado o conceito de Vitualização, Hypervisor, Hyper-V, SQL e ADO.NET;

No terceiro capítulo é apresentada uma Contextualização do trabalho e uma lista de pressupostos e pré-requisitos;

No quarto capítulo é descrito todo o desenvolvimento necessário para a implementação da solução e explicado o desenvolvimento da mesma;

Por fim é apresentada uma conclusão com as considerações finais sobre o trabalho realizado.

No que concerne aos objetivos do trabalho, pretende-se comprovar o conceito de virtualização através da disponibilização de vários sistemas operativos com diferentes aplicações num único servidor físico.

Esta página foi intencionalmente deixada em branco

2. Estado da Arte

2.1. Virtualização

O termo Máquina Virtual foi descrito pela IBM, nos anos 60, como um conceito de sistema operativo, uma abstração de *software*[1] com aparência de *hardware*[2] de um sistema de computador. (Rosenblum, 2004).

A máquina virtual fornece uma abstração de *hardware* compatível para que todo o *software* escrito para ele seja executado na mesma. Por exemplo, uma máquina virtual de nível de *hardware* irá executar todo o *software*, sistemas operativos e aplicações escritas para o *hardware*, da mesma forma que uma máquina virtual em nível de sistema operativo irá executar aplicações para esse sistema em particular, e uma máquina virtual de alto nível irá executar programas escritos na linguagem de alto nível (Rosenblum, 2004).

A virtualização é uma tecnologia de *software* que torna possível executar vários sistemas operativos e aplicações no mesmo servidor ao mesmo tempo. A tecnologia está a transformar o panorama das Tecnologias de Informação e fundamentalmente a forma como as pessoas utilizam a mesma. Trata-se de uma tecnologia que abstrai os recursos de processador, memória, armazenamento e rede em várias máquinas virtuais que, por sua vez, podem executar um sistema operativo e aplicações não modificadas (VMWare, 2007).

Segundo a Gartner cerca de 75% das cargas de trabalho de servidores x86 estão virtualizadas. As tecnologias de virtualização estão-se a tornar mais leves, suportando mais cargas de trabalho e um ágil desenvolvimento. O preço, interoperabilidade, nuvem e casos de uso específicos estão a levar as empresas a implantar várias tecnologias de virtualização (Gartner, 2015).

Em 1998, a VMware descobriu como virtualizar a plataforma x86[3], o que se pensava ser impossível, e criou o mercado de virtualização x86. A solução foi uma combinação de

---

[1] O termo "***software***" foi criado na década de 1940, *Software* seria tudo o que faz o computador funcionar excetuando-se a parte física dele.

[2] O termo "***hardware***" é usado para fazer referência a detalhes específicos de uma dada máquina, aplica-se entre outros à unidade central de processamento, à memória e aos dispositivos de entrada e saída

[3] **x86** ou 80x86 é o nome genérico dada à família (arquitetura) de processadores baseados no Intel 8086, da Intel Corporation.

tradução *binário*[4] e execução direta no processador que permitiu que vários sistemas operativos convidados pudessem ser executados em isolamento completo sobre o mesmo computador.

As poupanças que dezenas de milhares de empresas têm gerado a partir da implantação dessa tecnologia impulsionou ainda mais a rápida adoção da computação virtualizada desde o *desktop* até ao centro de processamento de dados (VMWare, 2007).

A primeira versão do Hyper-V foi lançada pela Microsoft em Junho de 2008 juntamente com o "*Windows Server 2008*"[5] e desde então está presente em todos os Sistemas Operativos lançados pela Microsoft. Em Outubro de 2008 foi lançada a primeira versão do "*Microsoft Hyper-V Server*" que permite aos clientes consolidar cargas de trabalho num único servidor físico (Microsoft, 2013).

Em Outubro de 2013 a Microsoft lançou o "Windows Server 2012 R2" que eficazmente fechou a maior parte do gap de funcionalidades que existia com a VMWare em termos de infraestrutura de virtualização de servidores x86 (Gartner, 2015).

2.2. Hypervisor

Em 1974 Gerald Popek e Roberto Goldbert classificaram o hypervisor em dois tipos:

1. Tipo 1, nativo ou *bare-metal*:

O *hypervisor* é executado diretamente no *hardware* do *host*[6] para controlar o hardware e para gerenciar sistemas operativos convidados. Por esta razão, por vezes são chamados *hypervisors bare metal*. (Popek & Goldberg, 1974);

O acesso e comunicação com o *hardware* são diretos, disponibilizados pelo próprio *software* de virtualização, permitindo dessa forma a partilha de recursos entre múltiplas máquinas virtuais de um servidor de alta disponibilidade (Meier, 2008).

---

[4] **Código Binário** - Formado por 0 e 1, consiste de uma sequência de números que significam uma sequência de instruções a serem executadas

[5] **Windows Server 2008** é um sistema operacional de servidores da Microsoft, desenvolvido como sucessor do Windows Server 2003

[6] **Host** ou *hospedeiro*, é qualquer máquina ou computador conectado a uma rede, podendo oferecer informações, recursos, serviços e aplicações aos usuários ou outros nós na rede.

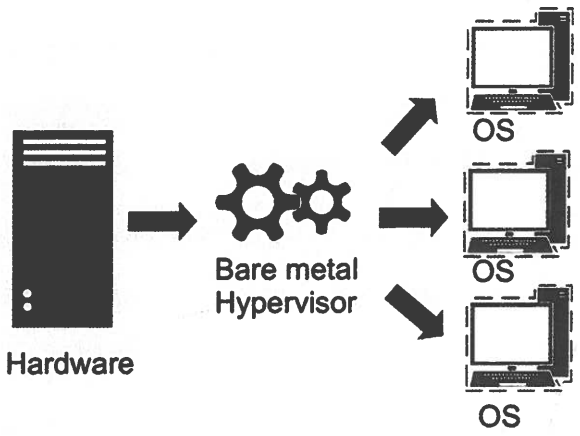

Figura 1 - Hypervisor do Tipo 1 (NCRIS, 2015)

2. Tipo 2, *Hypervisors* hospedados:

Estes *hypervisors* são executados através de um sistema operativo, assim como outros programas de computador. O sistema operativo corre sobre o *hardware* e por sua vez a camada do *hypervisor* corre em cima deste SO, sendo esta um *software hosted* (Popek & Goldberg, 1974);

O acesso e comunicação com o *hardware* é indireto por intermédio do sistema operativo hospedeiro. A máquina virtual é executada através de um *software* de virtualização (Meier, 2008).

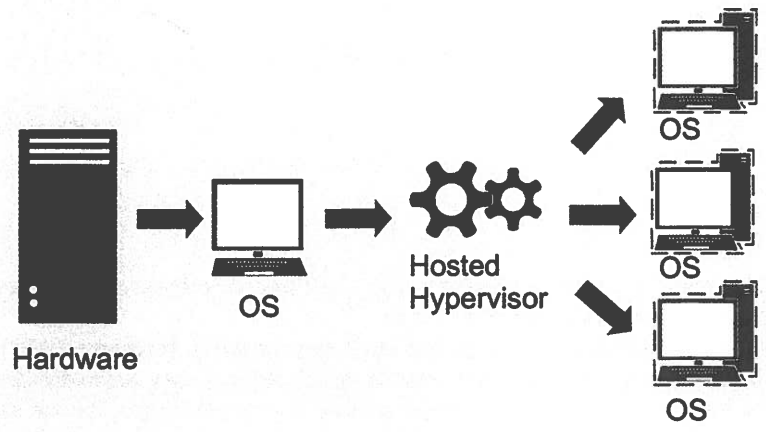

Figura 2 - Hypervisor do Tipo 2 (NCRIS, 2015)

2.3. Arquitetura Hyper-V

A virtualização de servidores tem evoluído ao longo dos últimos anos. Nesse processo, as empresas de todas as formas e tamanhos começaram a tirar partido da tecnologia para atender a mudanças nas necessidades de negócio. (Microsoft, 2013).

O Hyper-V é uma tecnologia de virtualização baseada em *hypervisor* do tipo 1 para as versões x64[7] do Windows Server 2008 ou superior, permitindo que vários sistemas operativos isolados possam compartilhar a mesma plataforma de *hardware* (Microsoft, 2013).

O Hyper-V suporta isolamento em termos de uma partição. Uma partição é uma unidade lógica de isolamento, suportada pelo *hypervisor*, na qual os sistemas operativos são executados. O *hypervisor* da Microsoft deve ter pelo menos um pai ou raiz. A pilha de virtualização é executada na partição pai e tem acesso direto aos dispositivos de *hardware*. A partição raiz cria as partições filhas que hospedam os sistemas operativos virtuais. Uma partição raiz cria partições filhas usando a Interface de Programação de Aplicações (API)[8] (Microsoft, 2013).

As Partições não têm acesso ao processador físico, nem tão pouco lidam com as interrupções do processador. Em vez disso, têm uma visão virtual do processador, executam numa zona de endereço de memória virtual que é particular para cada partição convidado. O *hypervisor* lida com as interrupções para o processador e redireciona-as para a respetiva partição. O Hyper-V também pode acelerar a tradução de endereço entre vários espaços de endereço virtual convidados, usando uma Unidade de Gerenciamento de Memória Input Output (IOMMU) que opera independente do *hardware* de gerenciamento de memória usado pelo CPU[9]. A IOMMU é usada para remapear endereços de memória física para os endereços que são usados pelas partições filho (Microsoft, 2013).

---

[7] **x64, AMD64** ou **x86-64**, é o nome genérico dado à família (arquitetura) de processadores baseados na tecnologia de 64 bits, utilizada pelos processadores da AMD e da Intel.

[8] **Interface de Programação de Aplicações ou Application Programming Interface (API),** é um conjunto de rotinas e padrões estabelecidos por um *software* para a utilização das suas funcionalidades por aplicações que não pretendem envolver-se em detalhes da implementação do *software*, mas apenas usar seus serviços.

[9] O **CPU** (*Central Processing Unit*), também conhecido como **processador**, é a parte de um sistema computacional, que realiza as instruções de um programa de computador, para executar a aritmética básica, lógica, e a entrada e saída de dados.

2.4. SQL

O SQL é uma linguagem de comunicações de dados. Consultas SQL são usadas para executar tarefas como selecionar, eliminar, classificar ou atualizar dados. Existem vários sistemas que usam linguagem SQL (MySQL, Microsoft SQL Server, Oracle, Sybase etc). (Ramakrishnan & Gehrke, 2003)

O SQL foi desenvolvido originalmente no início dos anos 70 pela IBM, que tinha por objetivo demonstrar a viabilidade da implementação do modelo relacional[10] proposto por E. F. Codd. O nome original da linguagem era *SEQUEL*, acrónimo para *"Structured English Query Language"* (Linguagem de Consulta Estruturada) (Chamberlin , et al., 1981)

O acrónimo SEQUEL foi mudado mais tarde para SQL porque "SEQUEL" era uma marca registrada da empresa aeronaves Hawker Siddeley sediada no Reino Unido (Oppel, 2011)

O SQL tornou-se a linguagem padrão primeiro pela ANSI (American National Standards Institute) em 1986, seguido da ISO (Organização Internacional de Normalização) em 1987. Desde então, a norma foi revista para incluir um conjunto maior de recursos. Apesar da existência de tais normas, a maioria do código SQL não é totalmente portátil entre diferentes sistemas de base de dados, tendo o mesmo de ser ajustado. (ISO/IEC 9075-1, 2011)

A última revisão da ISO ao padrão do SQL foi em 2011 ficando denominada de SQL:2011, esta revisão adicionou novas funcionalidades, sendo uma das principais novidades o melhor suporte para bancos de dados temporais (Zemke, 2012)

Atualmente o SQL é a linguagem de pesquisa declarativa padrão para bases de dados relacionais. Muitas das características originais do SQL foram inspiradas na álgebra relacional. (ISO/IEC 9075-1, 2011)

---

[10] O **modelo relacional** é um modelo de dados representativos (ou de implementação), adequado a ser o modelo subjacente de um Sistema Gerenciador de Banco de Dados (SGBD), que se baseia no princípio em que todos os dados estão guardados em tabelas (ou, matematicamente falando, relações). Toda sua definição é teórica e baseada na lógica de predicados e na teoria dos conjuntos.

2.5. ADO.NET

ActiveX Data Object.NET (ADO.NET) é uma biblioteca de software no Framework[11] .NET que consiste em componentes de software que prestam serviços de acesso a dados. O ADO.NET foi projetado para permitir que os programadores escrevam código gerenciado para a obtenção de acesso desconectado a fontes de dados, que pode ser relacional ou não-relacional (Thai & Lam, 2003)

O *.NET Data Provider* proporciona o acesso a bases de dados e a capacidade de executar comandos contra a fonte de dados. Possui vários tipos de *Data Providers* de forma a comunicar com diferentes sistemas de gestão de dados (ex. MS SQL[12] ou o ODBC[13]). Incluem as classes *Connection, Command, DataReader, Transaction, ParameterCollection* e *Parameter* (Hamilton & MacDonald, 2003).

O *DataSet* é constituído por *DataTables* e *DataRelations*, funciona desconectado da base de dados. Permite manipular os dados descarregados para a memória e depois reconciliar os mesmos com a fonte de dados. As classes desconectadas fornecem uma maneira comum de trabalhar com dados desconectados, independentemente da fonte de dados subjacente, incluem *DataSet, DataTable, DataColum, DataRow, Constraint, DataRelatinship* e *DataView* (Hamilton & MacDonald, 2003).

A classe *DataAdapter* constrói uma ponte entre a fonte de dados e as classes desconectadas. O *DataAdapter* é uma abstração das classes ligadas que preenche o *DataSet* e o *DataTable* com dados recolhidos na fonte de dados, atualizando também a fonte de dados para refletir as alterações feitas aos dados desconectados. Permite ainda efetuar consultas, inserir, atualizar e eliminar informação na Base de dados original. (Hamilton & MacDonald, 2003).

---

[11] **Framework**, é uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica.

[12] **MS SQL** (Microsoft SQL Server) é um sistema gerenciador de base de dados relacional desenvolvido pela Microsoft

[13] **ODBC** (Open Database Connecivity) é um padrão para acesso a sistemas gerenciadores de bancos de dados.

3. Contextualização

Uma biblioteca necessita de desenvolver uma aplicação que permita aos utilizadores credenciados terem acesso à base de dados principal da mesma.

Os funcionários da biblioteca necessitam dum perfil que permita adicionar e editar artigos, carregar os respetivos conteúdos bem como gerir listas independentes de Autores, Editores, e Tipos de Artigos.

O perfil de utilizador terá a funcionalidade de pesquisar por livros, autores, editoras, consultar a capa do documento e descarregar o material que vão investigando.

Os utilizadores terão acesso a um computador virtual com Windows 8.1 previamente criado e disponibilizado pelos administradores de sistema da biblioteca através dum servidor de SCVMM. Adicionalmente os utilizadores e funcionários deterão contas de acesso únicas, criadas na AD[14] da biblioteca. Os sistemas operativos clientes estarão ligados ao domínio sendo que a autenticação dos utilizadores na aplicação será automática através do utilizador do Windows. O perfil de acesso será validado através de grupos criados na AD para o efeito.

A aplicação terá de apresentar a capa do livro ou, na ausência da mesma, a primeira página do documento. Todos os artigos disponíveis na biblioteca virtual terão de ser passíveis de descarregar em formato pdf[15].

Todas as Máquinas Virtuais pertencentes ao domínio terão a aplicação instalada com um atalho no ambiente de trabalho. Adicionalmente também serão criados atalhos no ambiente de trabalho para o OneDrive e Abobe Acrobat Reader de forma aos utilizadores poderem visualizar os seus artigos e guardar os mesmos numa nuvem pública[16].

---

[14] O **Active Directory (AD)** é uma implementação de serviço de diretório no protocolo LDAP que armazena informações sobre objetos em rede de computadores e disponibiliza essas informações a utilizadores e administradores dessa rede. É um software da Microsoft utilizado em ambientes Windows.

[15] O **PDF (Portable Document Format)** é um formato de arquivo, desenvolvido pela Adobe Systems em 1993, para representar documentos de maneira independente. Um arquivo PDF pode descrever documentos que contenham texto, gráficos e imagens num formato independente de dispositivo e resolução.

[16] Uma nuvem é chamada de "**nuvem pública**" quando os serviços são apresentados por meio de uma rede que é aberta para uso público.

A biblioteca tem um servidor físico com um *hypervisor* onde estarão os controladores de domínio, servidores de Bases de Dados, Servidores de SCVMM e todas as máquinas criadas para os utilizadores.

4. Desenvolvimento

4.1. Introdução

O projeto prático em si divide-se em duas partes. A criação de uma infraestrutura virtual composta por um controlador de domínio, um servidor de base de dados e um servidor SCVMM que por sua vez irá disponibilizar máquinas virtuais com o Windows 8.1 para os utilizadores da biblioteca, e a criação de uma aplicação com ligação a uma Base de Dados instalada em SQL Server 2012 e integração no domínio.

A aplicação foi desenvolvida em WPF[17], utilizando a .NET Framework 4.5 e fazendo ligações à base de dados através de ADO.NET.

4.2. Arquitetura de Máquinas Virtuais

4.2.1. Máquinas Virtuais

As Máquinas Virtuais necessárias à execução deste trabalho foram configuradas no VMWare Workstation versão 12.0.1.

Foram criadas as seguintes máquinas virtuais com as respetivas funções:

- Servidor Windows 2012 R2 Standard – Função de Controlador de Domínio;
- Servidor Windows 2012 R2 Datacenter – Onde foi instalado o SQL 2012 Enterprise;
- Servidor Windows 2012 R2 Datacenter – Onde foi instalado o SC Virtual Machine Manager;
    - Máquinas Virtuais Windows 8.1 disponibilizadas através de um template Fabric no VMM (Hyper-V).

[17] O **Windows Presentation Foundation** (ou **WPF**), é um subsistema gráfico no .NET Framework 3.0, que usa uma linguagem de marcação, conhecida como XAML para desenvolvimento de GUIs ricas.

A nomenclatura utilizada nas máquinas é, **PG** (Projeto Global) **V** (virtual), **DC/SQL/SCVMM/CLIENT** (De acordo com a função da máquina), **XX** (Número da máquina por tipo). A título de exemplo o nome do controlador de domínio será **PGVDC01**.

4.2.2. Controlador de Domínio

O Servidor usado para Controlador de Domínio possui duas placas de rede, a primeira para a Rede Interna configurada no WMware como "Host-only" e uma segunda placa para acesso à internet configurada como "NAT". A placa para a rede interna foi configurada com o IP 10.1.1.1.

Foram instaladas as roles "*Active Directory Domain Services*", "*DNS Server*" e "*DHCP Server*" no servidor PGVDC01 e promovido o mesmo a controlador de domínio criando para o efeito uma nova floresta com o nome **PG1929.local** sendo **PG** referente a Projeto Global e **1929** o número de aluno do autor.

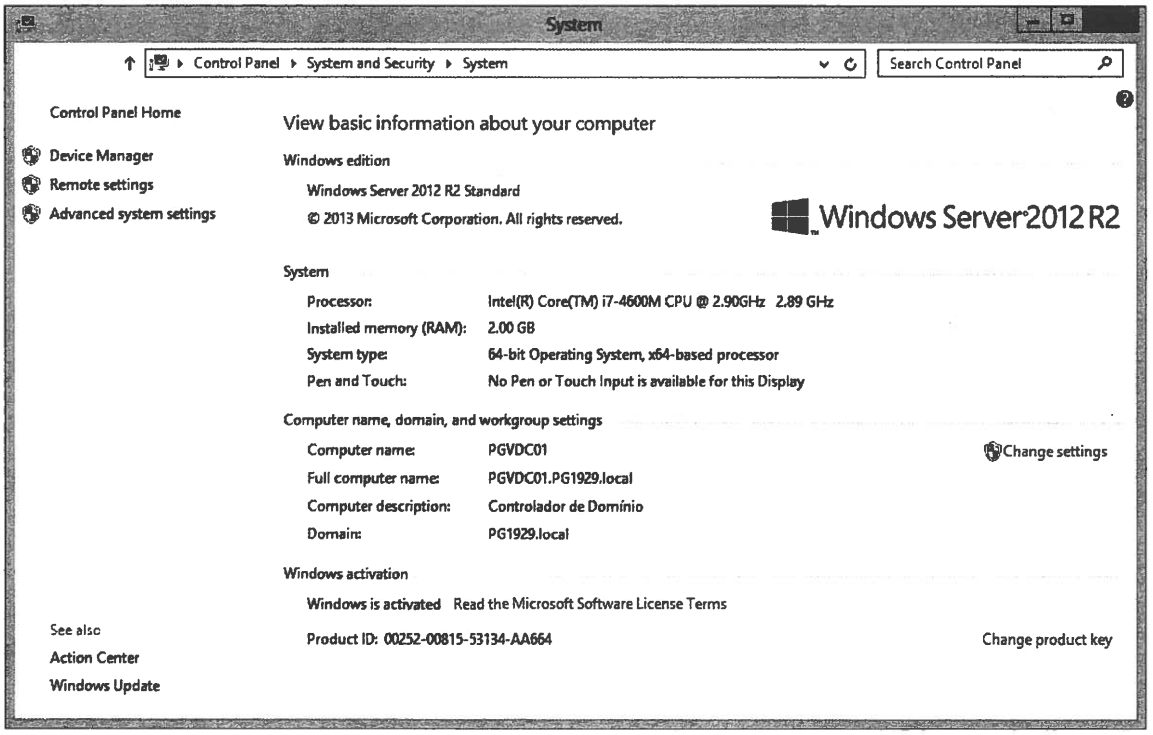

Figura 3 - Informação básica do controlador de domínio

4.2.3. Servidor de Base de Dados

O Servidor utilizado para as Bases de Dados possui duas placas de rede, a primeira para a Rede Interna configurada no WMware como "Host-only" e uma segunda placa para acesso à internet configurada como "NAT". A placa para a rede interna foi configurada com o IP 10.1.1.2, e adicionou-se o servidor ao domínio.

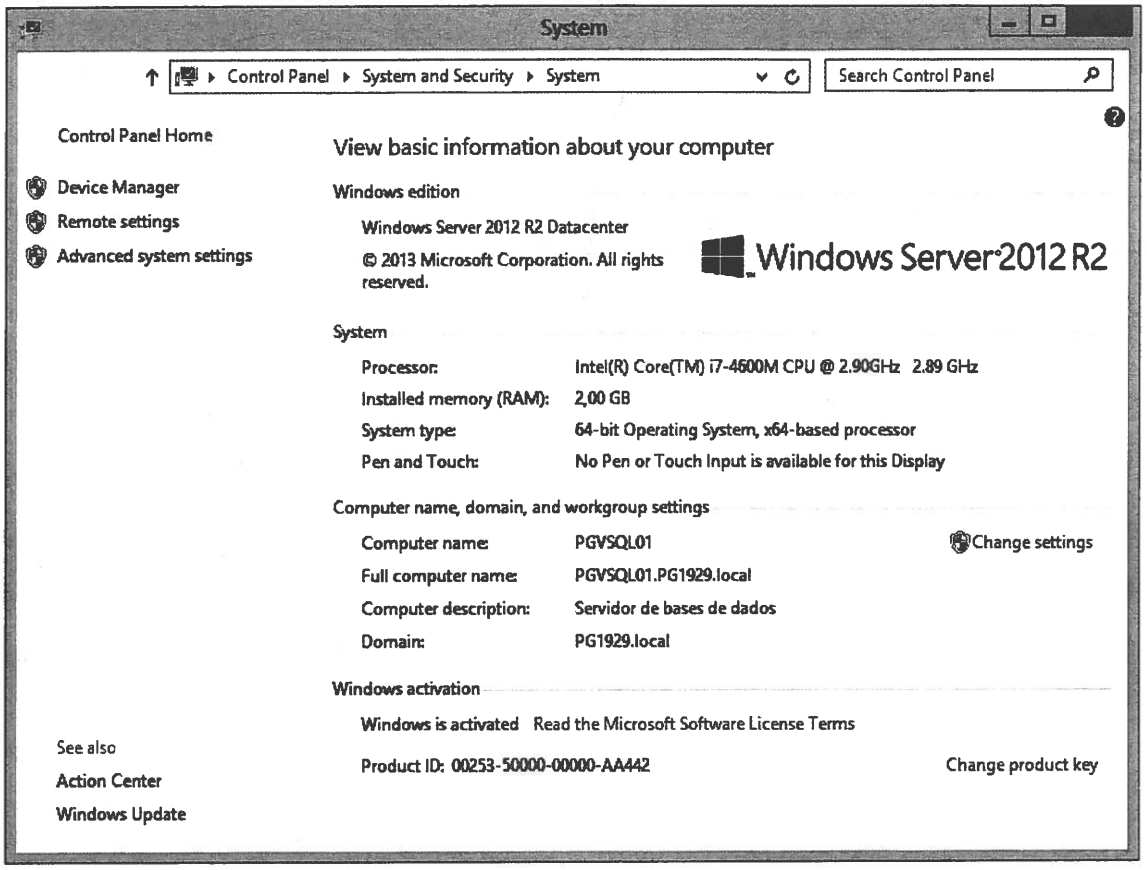

Figura 4 - Informação básica do servidor de Bases de Dados

De seguida foi instalado o SQL Server 2012 adicionando as Features "Database Engine Services" e "Management Tools – Complete". Não foi alterado o nome da Instancia tendo ficado com o nome default "**MSSQLSERVER**".

Por motivos de segurança foi apenas configurada a autenticação do SQL via Windows e adicionados os grupos de segurança **PGAPPuser** e **PGAPPsuper** criados anteriormente bem como o user **scvmmsql**.

Para a base de dados ser acessível a partir dos PCs cliente, foram abertas as portas TCP 1433 e UDP 1434 na Firewall

Inbound Rules

Name	Group	Profile	Enabled	Action	Override	Program	Local Address	Remote Address	Protocol	Local Port
SQL Server UDP		All	Yes	Allow	No	Any	Any	Any	UDP	1434
SQL Server		All	Yes	Allow	No	Any	Any	Any	TCP	1433

Figura 5 - Abertura de portas na Firewall

4.2.1. Servidor SCVMM

De forma aos administradores de sistemas poderem criar máquinas virtuais para os utilizadores da Biblioteca, foi instalado o SCVMM e criado um modelo de um Windows 8.1.

A segurança é uma preocupação da biblioteca, como tal, foi adicionado o antivírus "System Center Endpoint Protection" ao modelo.

Para criarem novos computadores virtuais, os administradores de sistemas da Biblioteca devem abrir o Virtual Machine Manager (VMM) disponível no servidor PGVSCVMM e selecionar "Library – Templates – VM Templates".

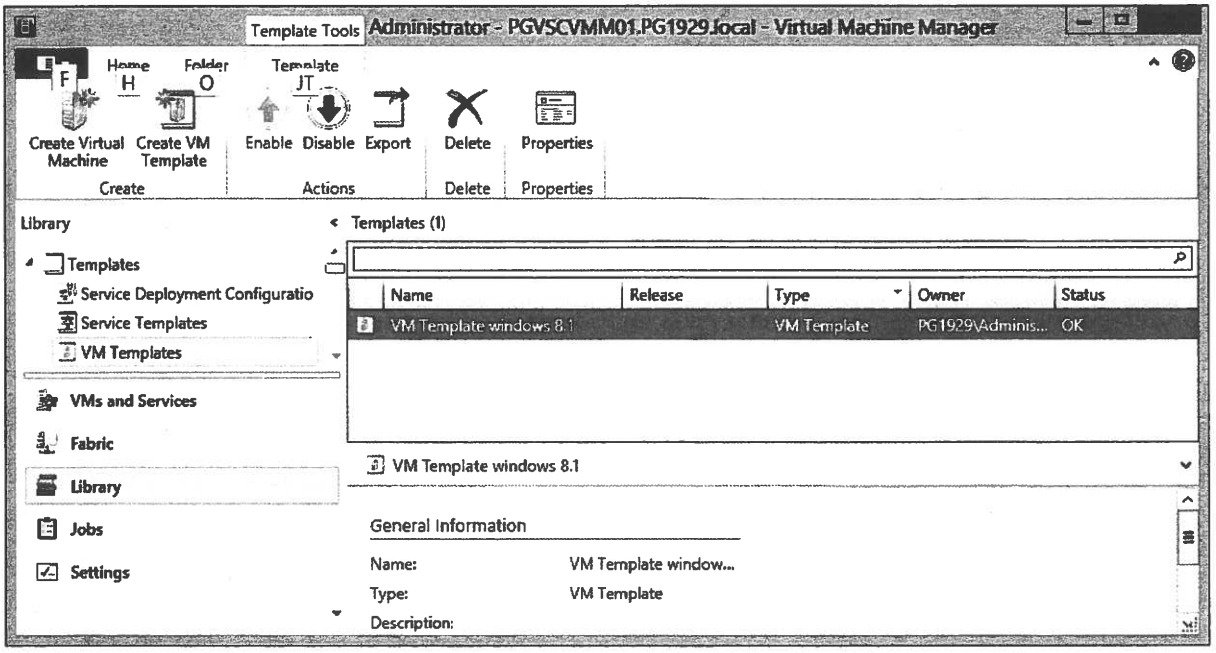

Figura 6 - VMM, VM Templates

Pressionar com a tecla da direita em "VM Template Windows 8.1" e escolher a opção "Create Virtual Machine"

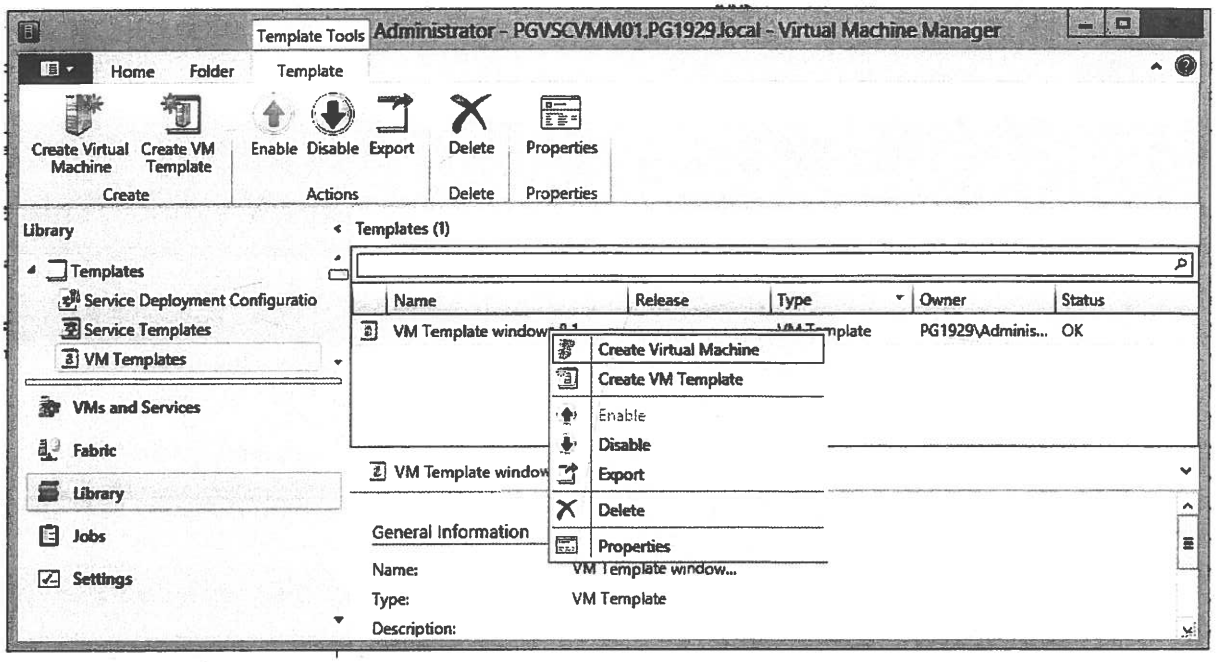

Figura 7 - VMM, Create VM from Template

Todas as configurações foram previamente preparadas pelo que só terá de pressionar em "Next" até concluir o assistente.

O modelo adiciona imediatamente as máquinas ao Domínio com o nome PGVCLIENT##, sendo ## um número incrementado de forma automática.

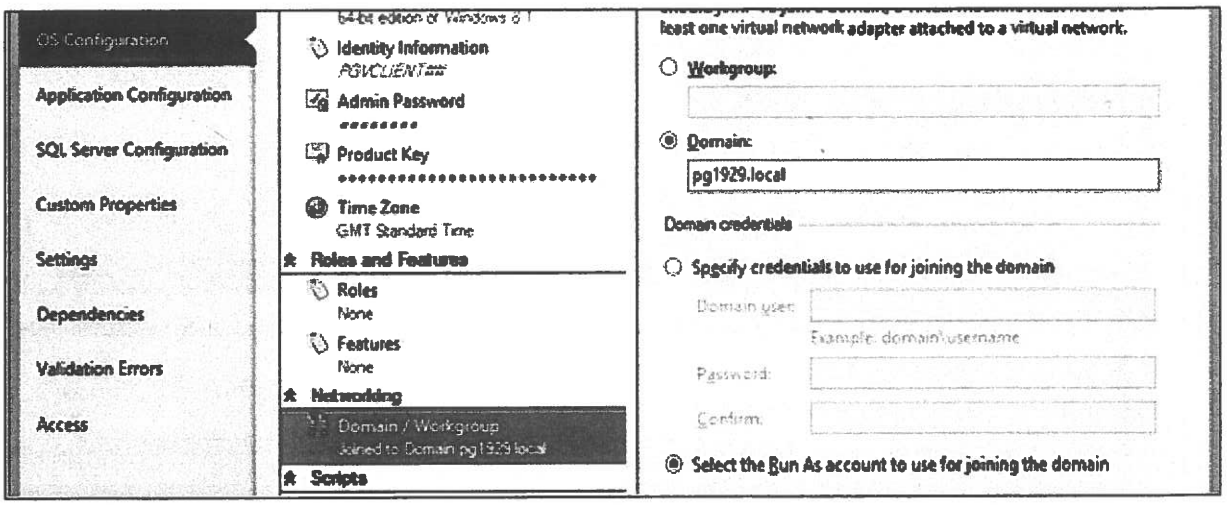

Figura 8 - VMM, VM Template (OS Configuration)

Uma vez criada, a máquina virtual encontra-se ligada, adicionada no domínio e pronta a ser utilizada por um utilizador.

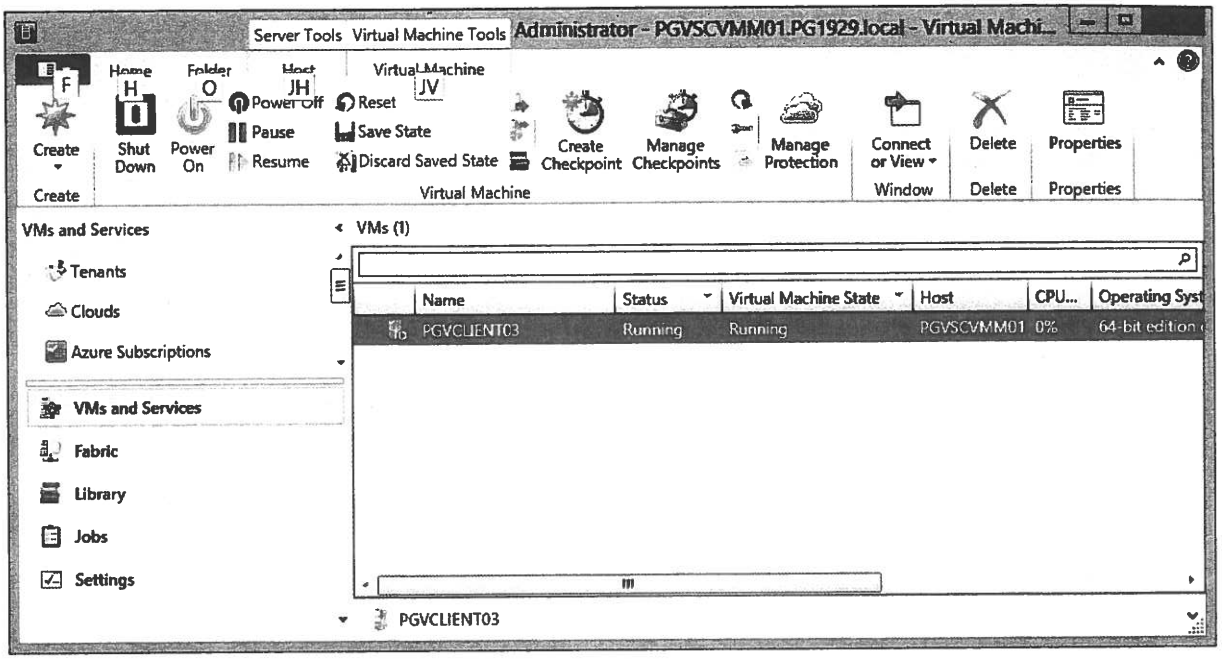

Figura 9 - VMM, VMs

A máquina foi automaticamente colocada na OU "ComputadoresCliente" de forma a serem aplicadas as GPO criadas para este grupo de clientes.

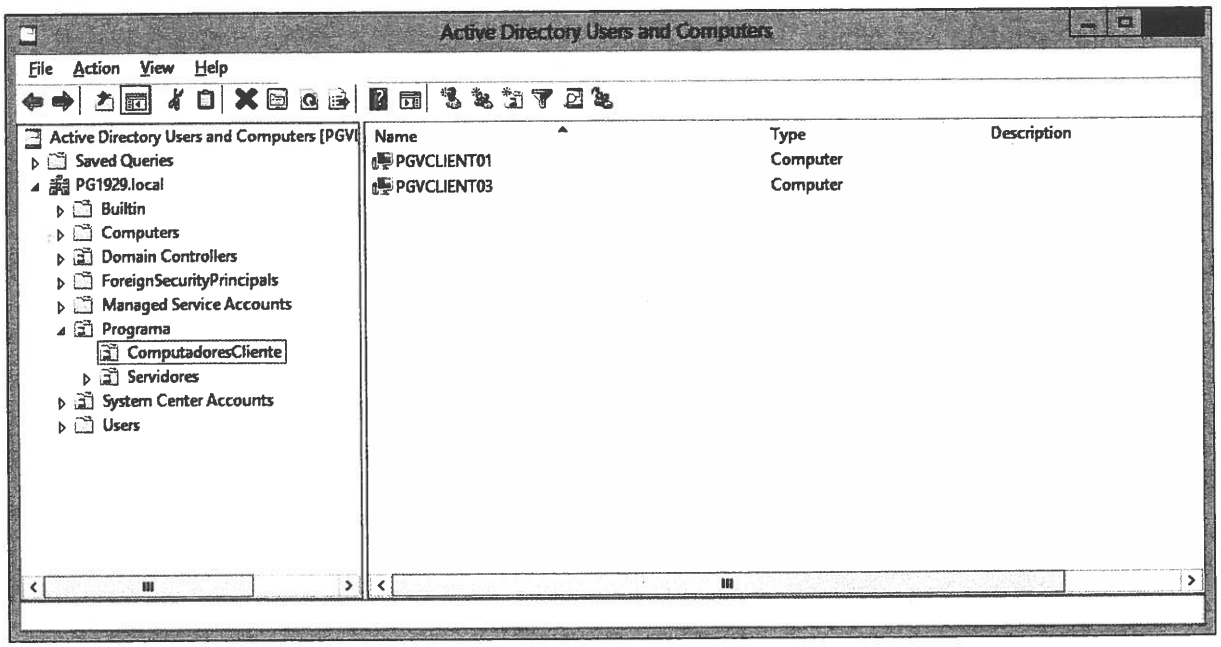

Figura 10 - AD, "ComputadoresCliente"

Uma vez iniciada não é necessária nenhuma configuração estando a mesma pronta para efetuar o primeiro login

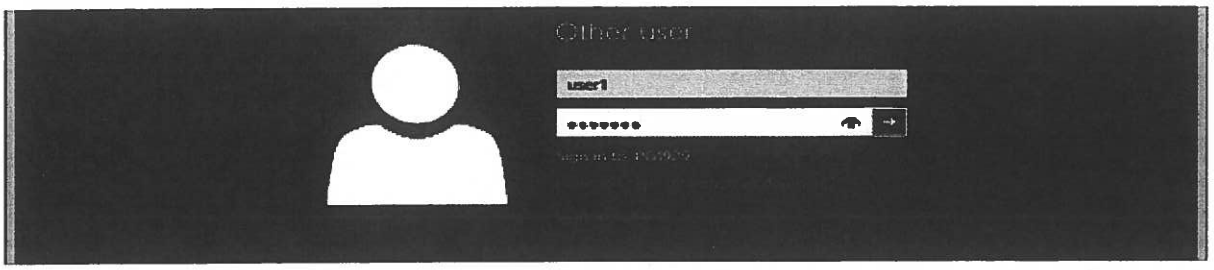

Figura 11 - VM 8.1 Cliente, Primeiro login

Após login, podemos constatar que os atalhos para a Aplicação, Adobe Reader e OneDrive se encontram no ambiente de trabalho e que o antivírus se encontra instalado.

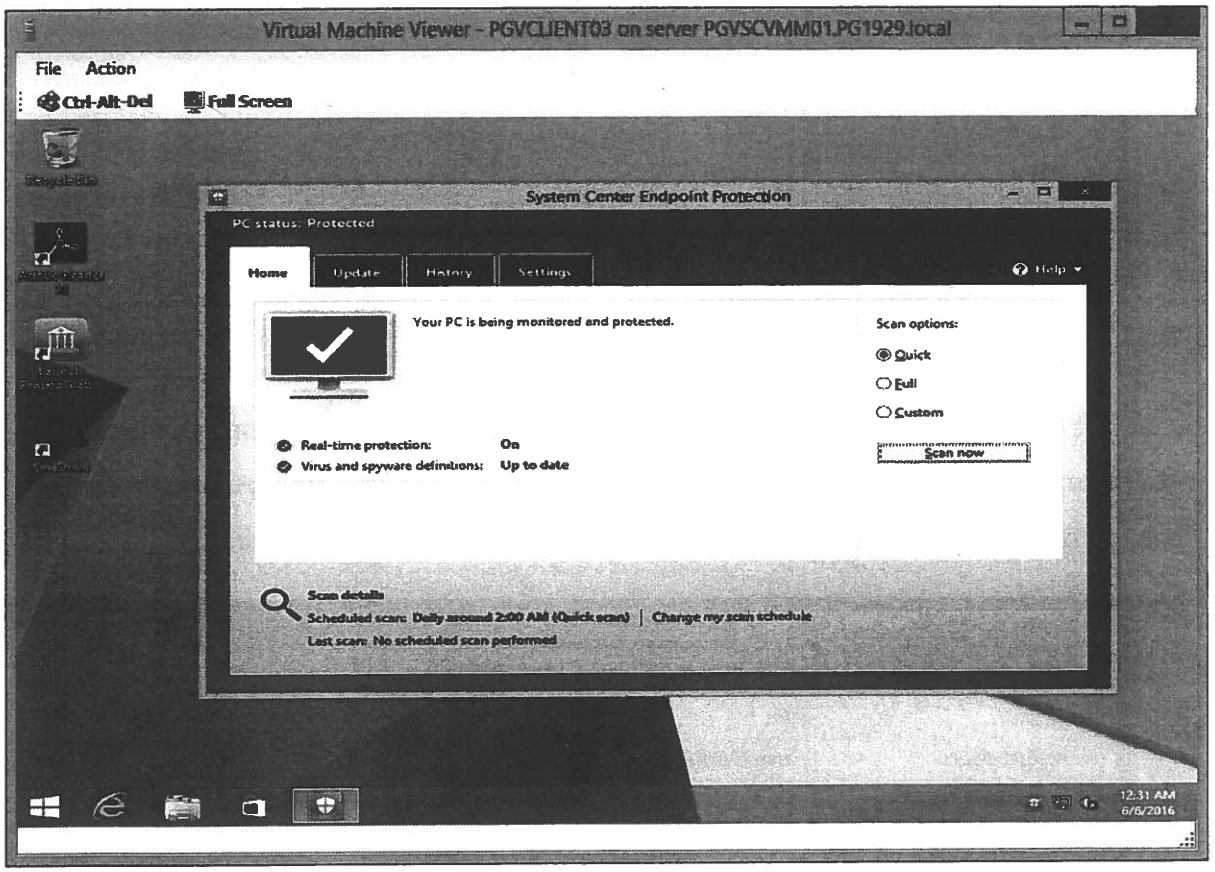

Figura 12 - Win 8.1 Desktop

4.3. Base de Dados

Para guardar toda a informação utilizada pela Aplicação, foi criada uma Base de Dados relacional com o nome **PQ1929DB** no SQL Server 2012 instalado no servidor PGVSQL01.

A Base de dados tem 4 tabelas relacionadas de forma a permitir gerir Documentos, Autores, Tipos de Documento e Editoras de forma distinta, evitando desta forma a repetição de dados. A título de exemplo, caso a biblioteca tenha vários livros do mesmo autor, o mesmo só será criado na base de dados uma vez.

A tabela **Autores** foi criada com os campos **Id** (Primary Key, Identity), **Nome** e **Nacionalidade,** sendo que o campo **Nacionalidade** não é de preenchimento obrigatório. Esta tabela servirá para guardar toda a informação dos Autores.

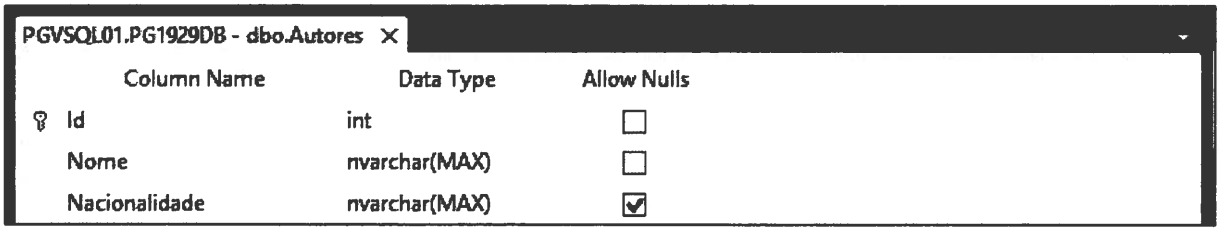

Figura 13 - Base de Dados, Tabela Autores

A tabela **Editora** foi criada com os campos **Id** (Primary Key, Identity), **NomeEditora, Localidade**, **Email** e **Contacto,** sendo que apenas o campo **NomeEditora** é de preenchimento obrigatório. Esta tabela servirá para guardar toda a informação das Editoras.

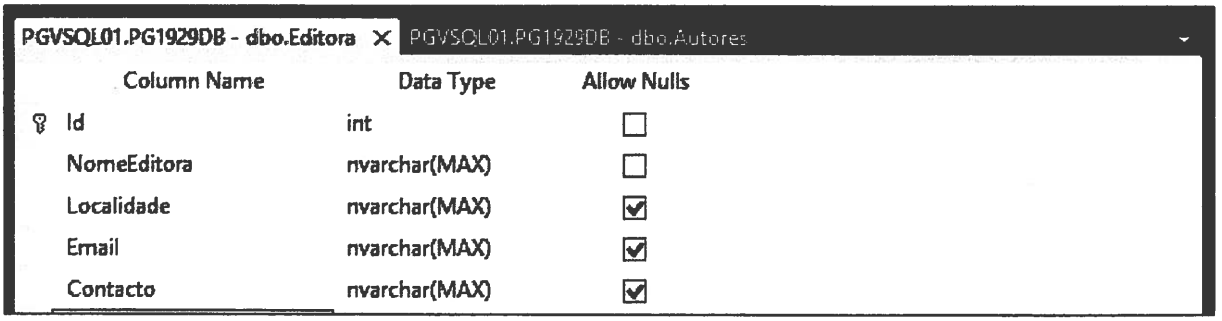

Figura 14 - Base de Dados, Tabela Editora

A tabela **TipoDocumento** foi criada com os campos **Id** (Primary Key, Identity) e Descricao. O campo descrição servirá para guardar os diferentes tipos de documentos que serão gravados na aplicação.

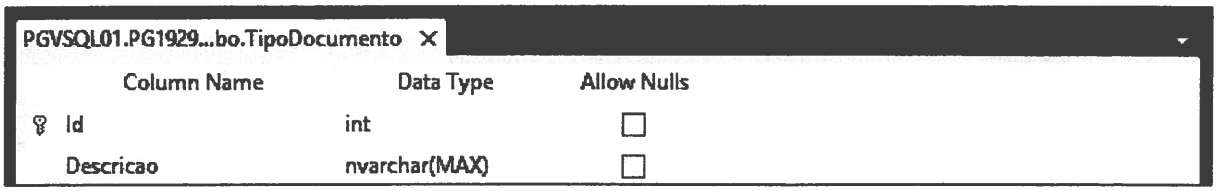

Figura 15 - Base de Dados, Tabela TipoDocumento

Por último, foi criada a tabela **Document** que relacionará todas a tabelas. Esta tabela foi criada com os campos **Id** (Primary Key, Identity). Os campos **AutorId**, **TipoDocumentoId**, e **EditoraId** são campos obrigatórios para relação com as respetivas tabelas e foram criados com as respetivas relações Foreing Keys:

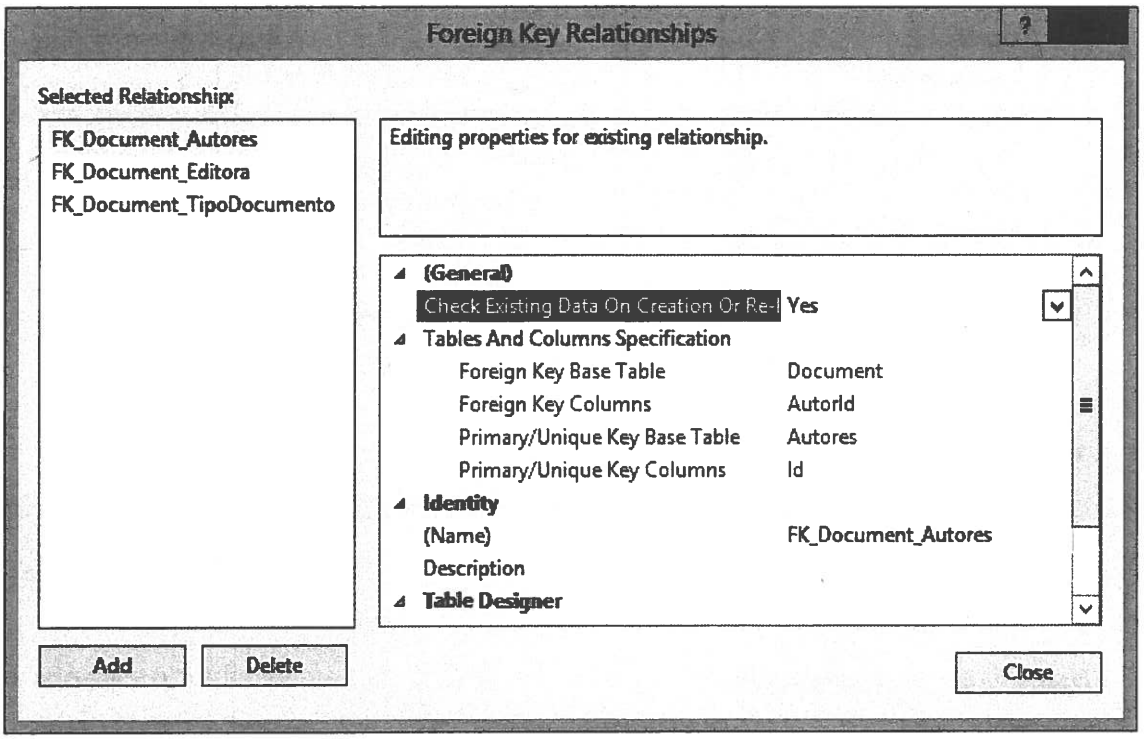

Figura 16 - Base de Dados, Foreing Key Relationships

O campo **Descricao** será usado pela aplicação para guardar o resumo do artigo, a capa e o respetivo conteúdo do artigo. Estes elementos serão convertidos em bytes e guardados nos campos **CoverBytes** e **ContentBytes**.

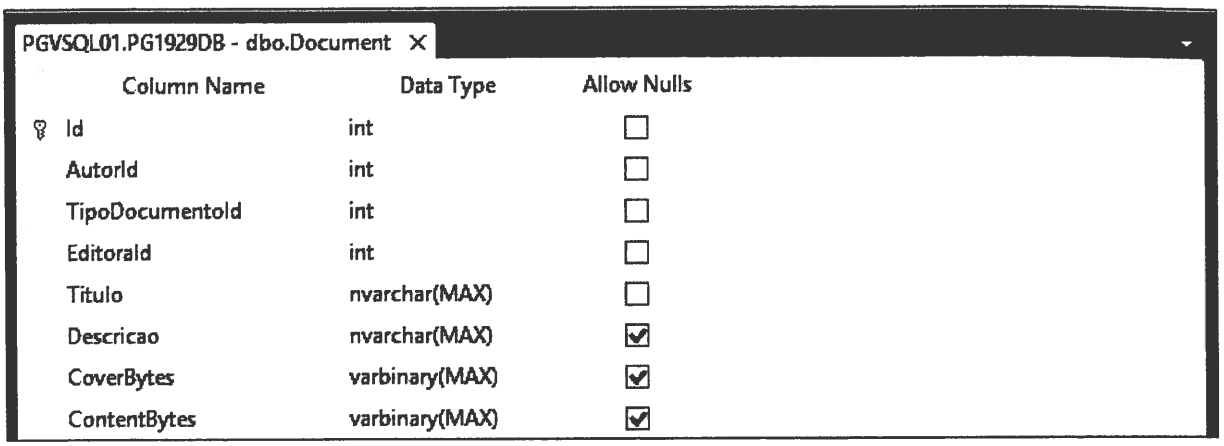

Figura 17 - Base de Dados, Tabela Document

Todas as tabelas se encontram relacionadas de acordo com o Diagrama abaixo:

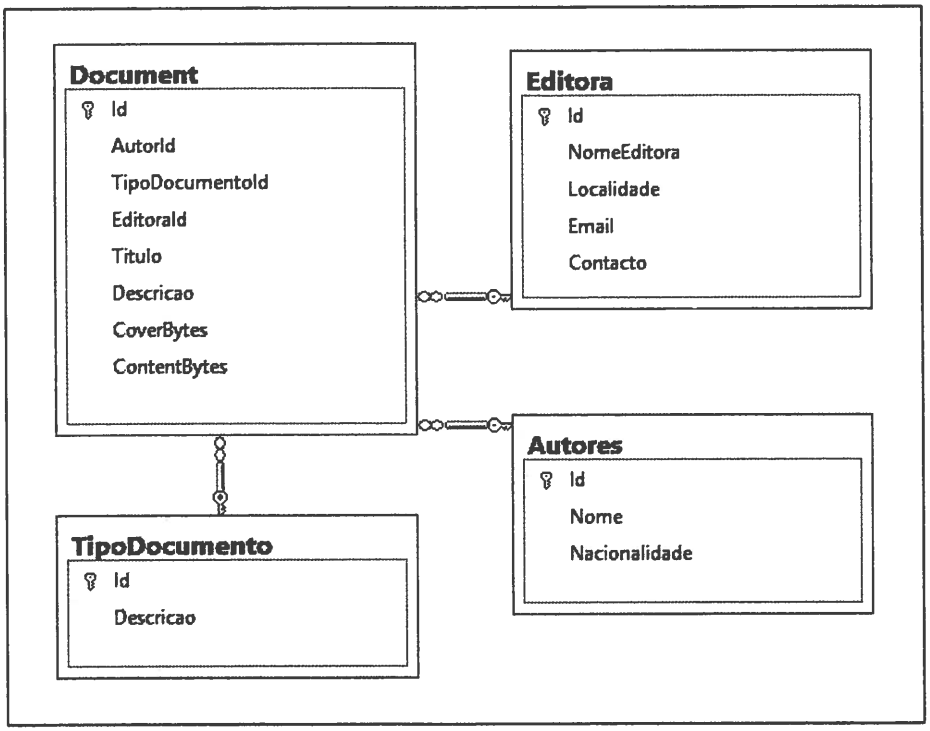

Figura 18 - Base de Dados, Diagrama

Os campos criados nas tabelas servem como prova de conceito da estrutura implementada, sendo que as tabelas deverão conter todos os campos que a biblioteca considere relevantes.

4.4. Preparação do Laboratório

4.4.1. Servidor DHCP

Foi configurado o servidor DHCP para atribuir endereços aos PCs que forem criados para os utilizadores da biblioteca. O DHCP foi configurado para atribuir endereços IP entre o 10.1.1.10 e o 10.1.1.200.

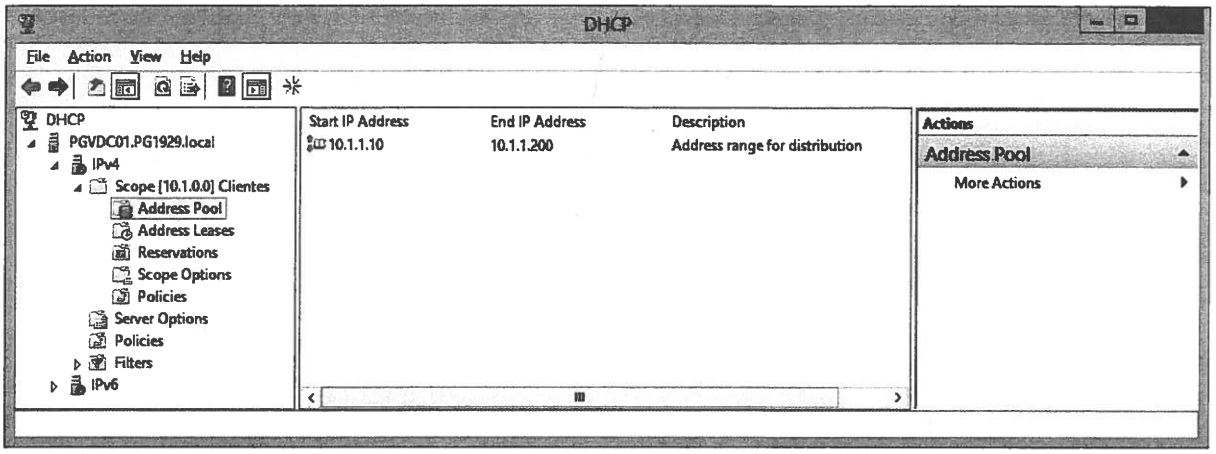

Figura 19 - DHCP Server

4.4.2. Configuração da Active Directory

A aplicação autentica os utilizadores através de dois Grupos criados para o efeito na AD. De forma a simplificar a organização foi criada uma Unidade Organizacional[18] com o nome "Programa". Dentro dessa OU foram criados dois grupos, o grupo "**PGAPPsuper**", que servirá para identificar utilizadores com privilégio de edição, e o Grupo "**PGAPPuser**", que será utilizado para os utilizadores com privilégios de leitura na aplicação.

[18] As **unidades organizacionais** são contentores do *Active Directory* nos quais se podem colocar utilizadores, grupos, computadores e outras unidades organizacionais.

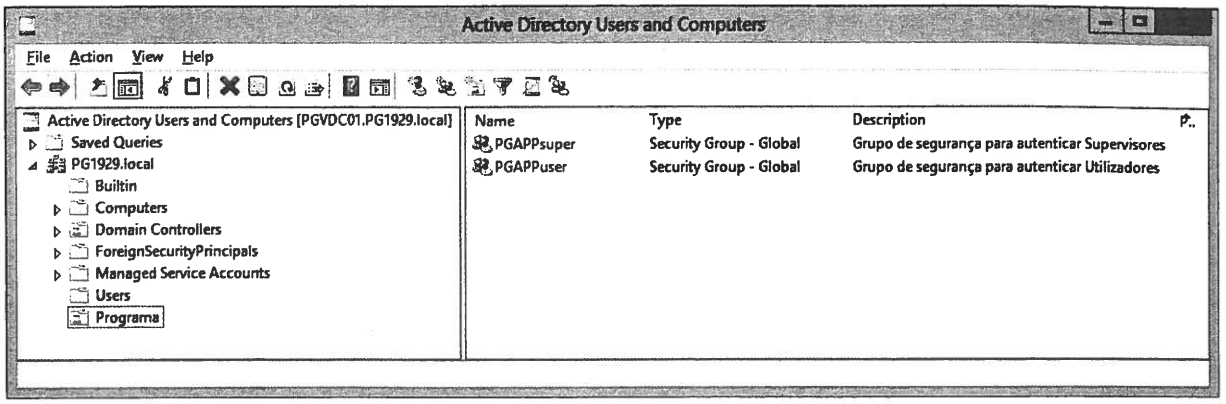

Figura 20 - Unidade Organizacional "Programa" e Grupos de segurança.

Lista e utilizadores criados e respetivas funções:

- user1, membro do grupo "PGAPPuser (privilégios de leitura na aplicação);
- super1, membro do grupo "PGAPPsuper – (privilégios de edição na aplicação).

Também foi criada uma OU para as contas do SCVMM. Nessa OU foram criados os seguintes utilizadores:

- admin_scvmm, conta de Administrador para o SCVMM;
- runas_scvmm, conta de RunAs para o SCVMM;
- svc_scvmm conta de Serviço para o SCVMM;
- svc_sql conta para o SCVMM se ligar ao SQL.

Adicionalmente foi criado o Grupo de Segurança SCVMMAdmins e adicionadas as contas ao mesmo.

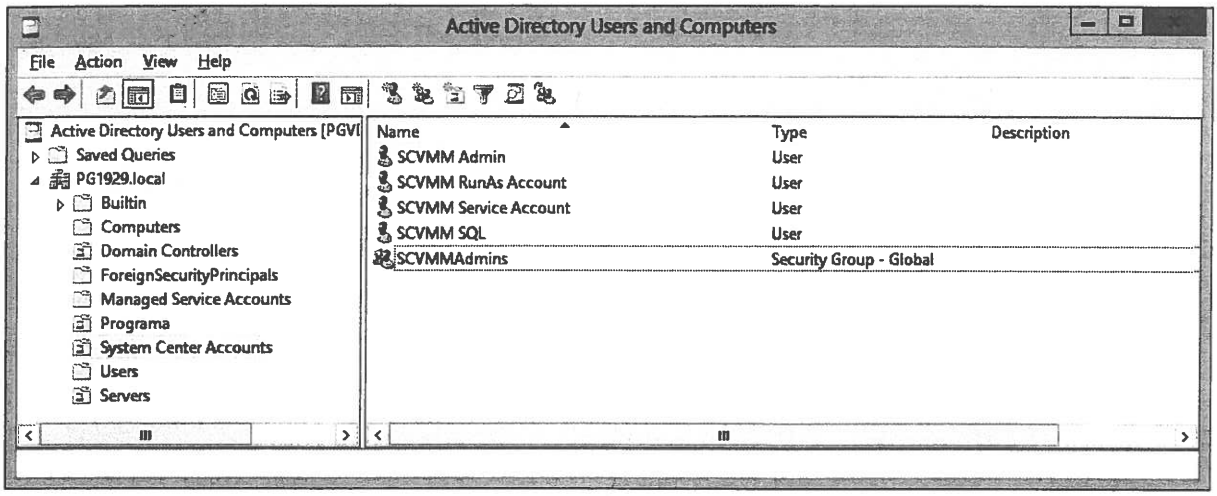

Figura 21 - Unidade Organizacional "System Center" e Grupos de segurança.

Todos os utilizadores foram criados com a palavra chave "*P@ssw0rd*".

4.4.1. Group Policy (GPO)

Um dos pressupostos é a aplicação estar disponível a partir do ambiente de trabalho dos utilizadores da biblioteca. Para o efeito foi criada uma GPO[19] para instalar a Aplicação nos computadores dos utilizadores.

Esta utilização da GPO garante que os clientes têm sempre a versão mais recente da Aplicação.

[19] **Group Policy (GPO)**, é uma funcionalidade da família de sistemas operativos Microsoft NT. É um conjunto de regras aplicadas nas contas de utilizador e computadores. Fornece a gestão e configuração centralizada de sistemas operativos, aplicações e configuração de utilizadores num ambiente *Active Directory*.

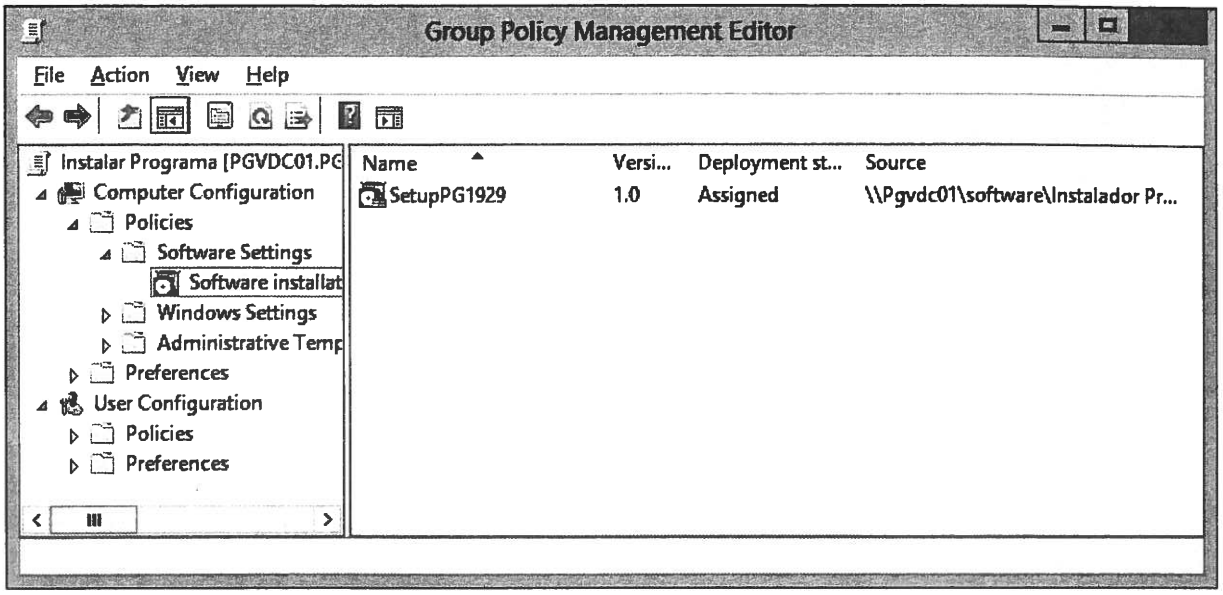

Figura 22 - GPO, Instalar a Aplicação

A aplicação é instalada e é adicionada uma entrada no menu iniciar bem como um atalho no Ambiente de Trabalho.

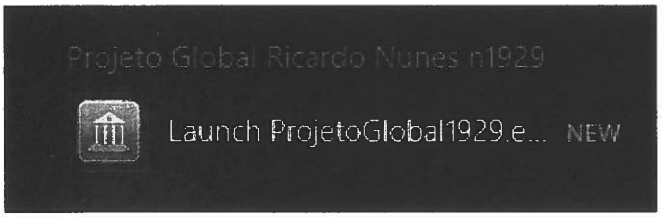

Figura 23 - Menu Iniciar, Atalho para o Programa

Figura 24 - Atalho Ambiente Trabalho

Foi adicionada da mesma forma uma GPO para a instalação do Acrobat Reader XI.

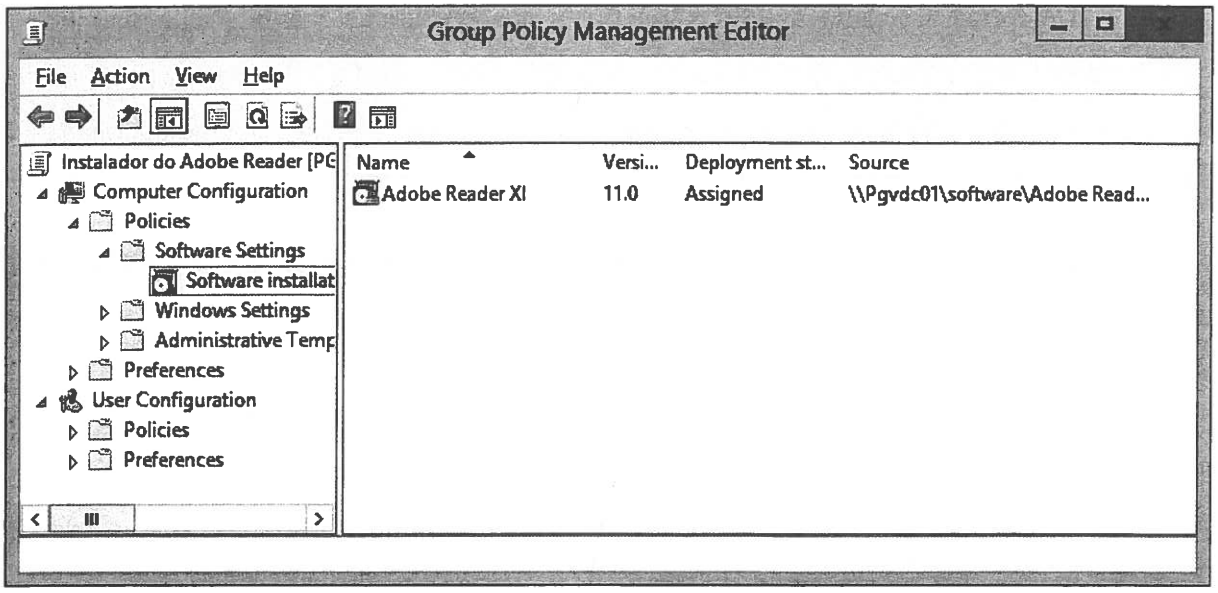

Figura 25 - GPO, Instalar o Adobe Reader XI

Outro pressuposto do projeto é os utilizadores terem acesso a uma Cloud Publica para poderem guardar os seus trabalhos. O Windows 8.1 já vem com o OneDrive integrado pelo que foi adicionado um atalho no ambiente de trabalho de forma aos utilizadores terem um acesso rápido à Cloud Publica. Caso o utilizador não tenha conta no OneDrive poderá criar no momento.

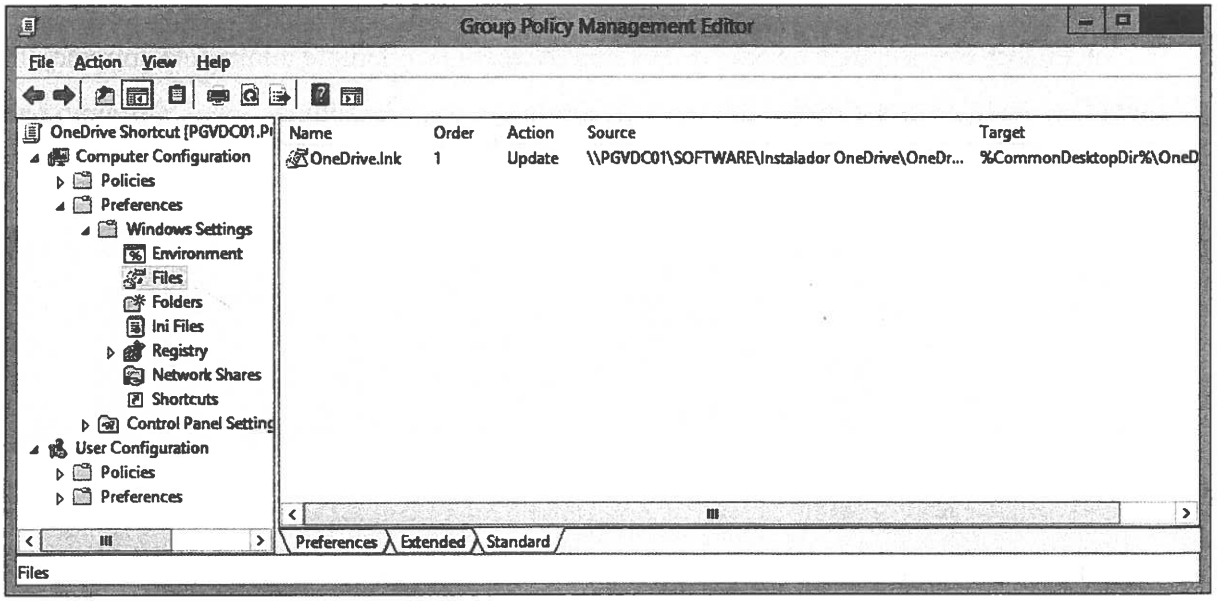

Figura 26 - GPO, Atalho para o OneDrive

De forma a separar as GPOs anteriores dos servidores e outras maquinas que não necessitem destas políticas, foram criadas duas OU dentro da OU Programa, uma para os Servidores e outra para os Clientes. As políticas estão apenas a afetar os computadores da OU "ComputadoresCliente".

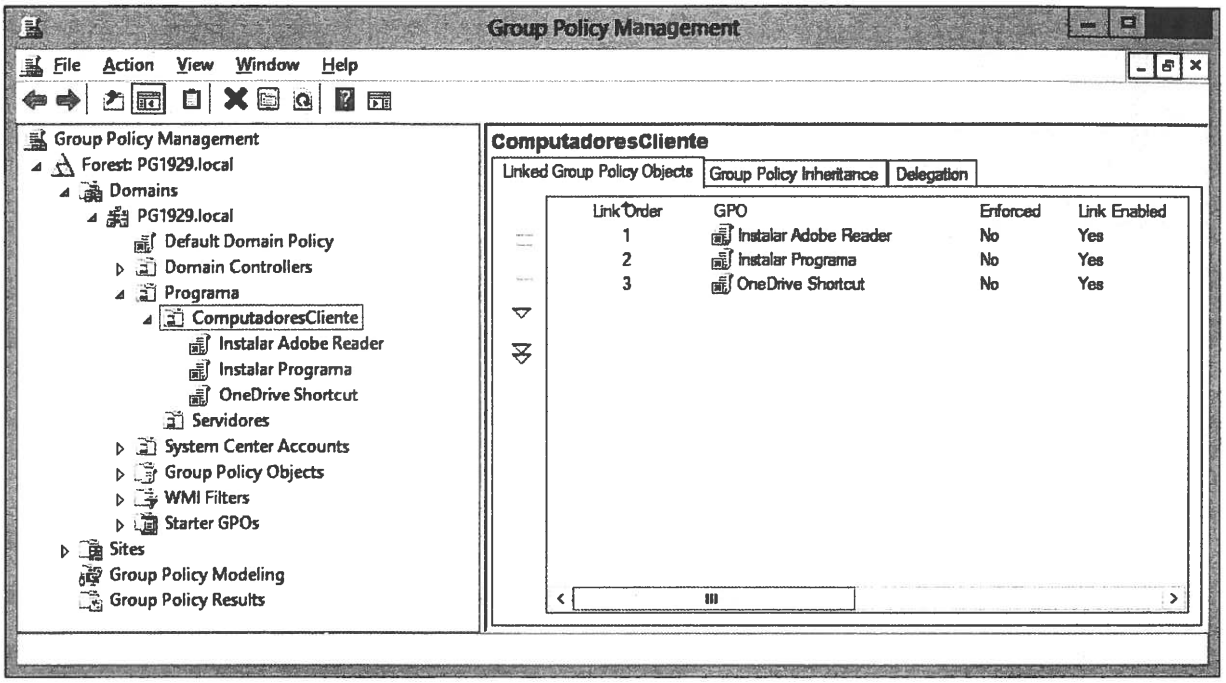

Figura 27 - GPO, organização OU

Para tornar o processo de adição de sistemas operativos cliente totalmente automático foi alterada a OU pré-definida de adição de novas máquinas ao Domínio para "ComputadoresCliente"

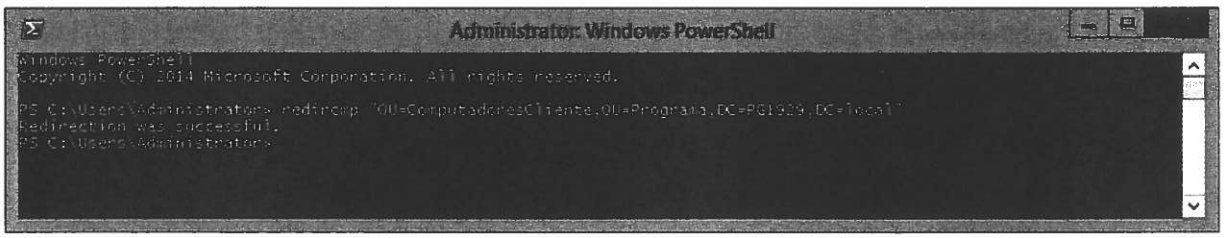

Figura 28 - PowerShell, redircmp

4.1. Desenvolvimento

4.1.1. Login no domínio

A aplicação inicia o programa com o ficheiro App.xaml, que por sua vez tem configurada a InitWindow.xaml como sendo a janela de arranque da aplicação.

```
<Application x:Class="ProjetoGlobal1929.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:ProjetoGlobal1929"
             StartupUri="InitWindow.xaml">
```

Figura 29 - Arranque da aplicação

Esta janela é apenas composta por uma pequena interface gráfica que providencia informação sobre o estado da ligação enquanto invoca a classe que faz a validação de login - a SecutityService.cs.

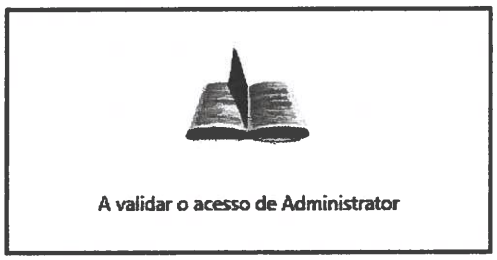

Figura 30 - Init Window, login

```
ProjetoGlobal1929 | ProjetoGlobal1929.InitWindow | OnContentRendered(EventArgs e)

1 reference | 0 changes | 0 authors, 0 changes
41  protected override void OnContentRendered(EventArgs e)
42  {
43      base.OnContentRendered(e);
44
45      Task.Run(() =>
46      {
47          TxtStatus.Dispatcher.Invoke(new Action(() => { TxtStatus.Text = "A validar o acesso de " + Environment.UserName ; }));
48          User = SecurityService.Instance.GetUserInContext(); // Valida user
49          TxtStatus.Dispatcher.Invoke(new Action(() => { TxtStatus.Text = "A carregar dados..."; }));
50          DbService.Instance.InitializeReferenceData();
51          Docs = DbService.Instance.GetAllDocs();
52
53      }).ContinueWith((t) =>
54      {
```

Figura 31 - Init Window,cs

A utilização da referência "System.DirectoryServices.AccountManagement" permite acesso uniforme, manipulação de utilizadores, computadores ou grupos de segurança criados na AD.

Para a autenticação através do utilizador da AD, foi criada a classe SecutityService e adicionado o uso de "System.DirectoryServices.AccountManagement".

O código desenvolvido começa por procurar o utilizador no contexto domínio.

```
ProjetoGlobal1929 | ProjetoGlobal1929.Services.SecurityService
1   using ProjetoGlobal1929.Models;
2   using ProjetoGlobal1929.Utilities;
3   using System;
4   using System.Collections.Generic;
5   using System.DirectoryServices.AccountManagement; //Para autenticar no dominio
6   using System.Linq;
7   using System.Text;
8
9   namespace ProjetoGlobal1929.Services
10  {
        2 references | 0 changes | 0 authors, 0 changes
11      public class SecurityService : GenericSingleton<SecurityService>
12      {
            1 reference | 0 changes | 0 authors, 0 changes
13          public AppUser GetUserInContext()
14          {
15              AppUser result = new AppUser() { Username = Environment.UserName };
16
17              PrincipalContext ctx = null;
18
19              ctx = new PrincipalContext(ContextType.Domain); //contexto
```

Figura 32 - SecurityService.cs, validação de utilizador

De seguida procura os Grupos de Segurança na AD e valida se o utilizador é membro de algum desses grupos.

```
ProjetoGlobal1929 | ProjetoGlobal1929.Services.SecurityService | GetUserInContext()
21              using (ctx)
22              {
23                  result.DomainName = ctx.ConnectedServer;
24
25                  // Procura o utilizador
26                  UserPrincipal user = UserPrincipal.FindByIdentity(ctx, result.Username);
27
28                  // Procura Grupos de Segurança
29                  GroupPrincipal adminGroup = GroupPrincipal.FindByIdentity(ctx, ConfigHelper.Instance.GroupNameAdmin);
30                  if (adminGroup == null)
31                      throw new Exception("Não foi encontrado o grupo de acesso " + ConfigHelper.Instance.GroupNameAdmin);
32                  GroupPrincipal normalGroup = GroupPrincipal.FindByIdentity(ctx, ConfigHelper.Instance.GroupNameNormal);
33                  if (normalGroup == null)
34                      throw new Exception("Não foi encontrado o grupo de acesso " + ConfigHelper.Instance.GroupNameNormal);
35                  if (user != null)
36                  {
37                      // Valida se o utilizador é membro de um grupo
38                      if (user.IsMemberOf(adminGroup))
39                          result.CurrentAccessMode = AccessMode.Admin;
40                      else if (user.IsMemberOf(normalGroup))
41                          result.CurrentAccessMode = AccessMode.Normal;
42                  }
43              }
```

Figura 33 - SecurityService.cs, validação de perfil e utilizador

De forma a tornar a aplicação configurável, foi criada uma classe ConfigHelper.cs onde foram criados os métodos para ler as APPSettings no App.config, no sentido de obter os nomes dos grupos de utilizadores bem como a connection string de acesso à BD.

```
ProjetoGlobal1929                                   ProjetoGlobal1929.Utilities.ConfigHelper
                    3 references | 0 changes | 0 authors, 0 changes
10                  public class ConfigHelper : GenericSingleton<ConfigHelper>
11                  {
                        2 references | 0 changes | 0 authors, 0 changes
12                      public string GroupNameAdmin
13                      {
14                          get
15                          {
16                              return ConfigurationManager.AppSettings["GroupNameAdmin"] ?? string.Empty;
17                          }
18                      }
19
                        2 references | 0 changes | 0 authors, 0 changes
20                      public string GroupNameNormal
21                      {
22                          get
23                          {
24                              return ConfigurationManager.AppSettings["GroupNameNormal"] ?? string.Empty;
25                          }
26                      }
```

Figura 34 - ConfigHelper.cs, validação do nome dos Grupos

Os valores dos grupos de segurança criados na AD estão registados no ficheiro App.config.

```
<appSettings>
  <add key="GroupNameAdmin" value="PGAPPsuper"/>
  <add key="GroupNameNormal" value="PGAPPuser"/>
</appSettings>
```

Figura 35 - App.config, valores dos grupos de segurança

A classe SecutityService devolve o perfil de acesso do utilizador ou um erro caso o utilizador não seja encontrado nos grupos respetivos.

Retornando ao InitWindow, em caso de sucesso avança para a MainWindow.xaml, em caso de insucesso devolve uma mensagem de erro.

```
ProjetoGlobal1929                    ProjetoGlobal1929.InitWindow          OnContentRendered(EventArg
67      {
68          if (User.CurrentAccessMode == AccessMode.Unknown) // Se não encontrou uma relação user/grupo
69          {
70              MessageBox.Show("O seu utilizador não pertence a nenhum grupo com acesso à aplicação.", "Acesso negado",
71              this.Close(); // Fecha a aplicação em caso de falha na autenticação.
72          }
73
74          MainWindow main = new MainWindow(User, Docs);
75          main.Show(); // Abre a janela EditarLivros em caso de sucesso
76          this.Close();
```

Figura 36 - InitWindow, sucesso ou erro

4.1.2. Ligação à Base de Dados

Na InitWindow é chamada a classe DBService para carregar em memória as listas de valores de referência, nomeadamente, autores, tipos documento e editoras.

Em seguida no GetAllDocs são obtidos todos os documentos que se encontram na base dados, para serem passados à MainWindow.

```
ProjetoGlobal1929                    ProjetoGlobal1929.InitWindow          OnContentRendered(EventArgs e)
45          Task.Run(() =>
46          {
47              TxtStatus.Dispatcher.Invoke(new Action(() => { TxtStatus.Text = "A validar o acesso de " + Environment.UserName ; }));
48              User = SecurityService.Instance.GetUserInContext(); // Valida user
49              TxtStatus.Dispatcher.Invoke(new Action(() => { TxtStatus.Text = "A carregar dados..."; }));
50              DbService.Instance.InitializeReferenceData(); // Base de dados
51              Docs = DbService.Instance.GetAllDocs();
52
53          }).ContinueWith((t) =>
54          {
```

Figura 37 - InitWindow, Ligação à Base de Dados

A classe DBManager abre uma SQL Connection e cria um DataAdapter. Também aqui são utilizados o ConfigHelper e o App.Config de forma a ser mais simples editar as ligações.

```
ProjetoGlobal1929                    ProjetoGlobal1929.DataAccess.DbManager    ExecuteC
 9  namespace ProjetoGlobal1929.DataAccess
10  {
        54 references | 0 changes | 0 authors, 0 changes
11      public class DbManager : GenericSingleton<DbManager>
12      {
            21 references | 0 changes | 0 authors, 0 changes
13          public DataTable GetDataTable(string selectStatement)
14          {
15              DataTable dt = new DataTable();
16
17              using (SqlConnection con = new SqlConnection(ConfigHelper.Instance.MyConnectionString))
18              using (SqlDataAdapter adp = new SqlDataAdapter(selectStatement, con))
19              {
20                  adp.Fill(dt);
21
22              }
23
24              return dt;
25          }
```

Figura 38 - DBManger, construtor

A Connection String contém os parâmetros "Data Source" que identifica o nome do servidor onde está a base de dados, "Initial Catalog" que identifica o nome da Base de Dados e o "Trusted_Connection" que identifica o acesso à base de dados através de utilizadores ou grupos de segurança criados na AD.

```
 9  <connectionStrings>
10    <add name="MyConnectionString" connectionString="Data Source=PGVSQL01;Initial Catalog=PG1929DB;Trusted_Connection=yes;"
11      providerName="System.Data.SqlClient" />
12  </connectionStrings>
```

Figura 39 - App.config, Connection String da AD

Depois de criada a ligação à Base de dados é criado um novo SqlDataAdapter, utilizando a connection string criada anteriormente.

4.2. Interface Gráfico

4.2.1. Listar Artigos

4.2.1.1. Design Gráfico

Uma vez autenticado, o utilizador é automaticamente encaminhado para a MainWindow.

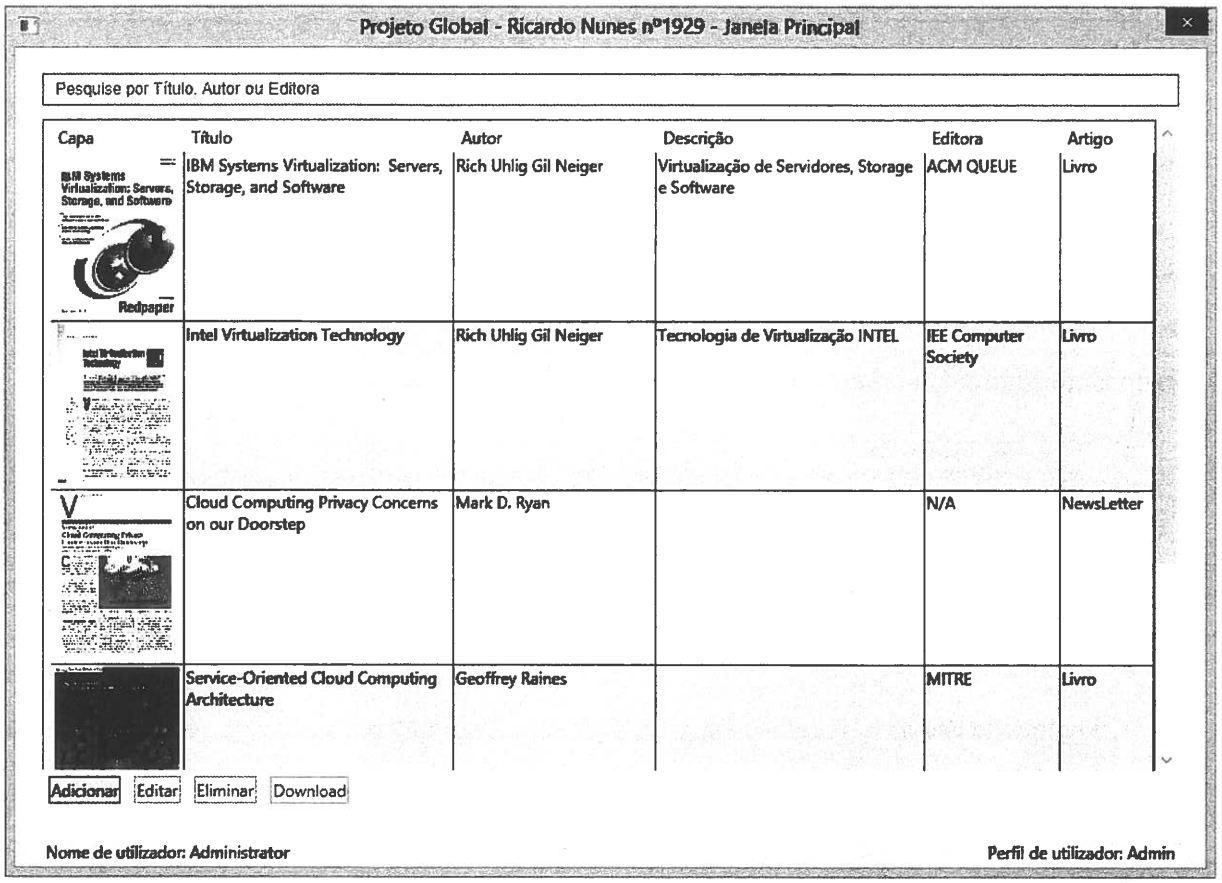

Figura 40 - MainWindow

A aplicação foi criada em WPF sendo que a base da janela é composta por um DockPanel. Por sua vez o DockPanel é composto por uma Grid e uma StatusBar.

4.2.1.2. Caixa de pesquisa

No topo da Grid criada dentro DockPanel foi adicionada uma TextBox que permite pesquisar por Títulos de artigos, Autores, Tipos de Documento ou Editoras.

```
44          <TextBox Grid.Column="0" Name="TxtSearch"
45                   Text="{Binding Path=SearchQuery, Mode=TwoWay, Delay=500,
46                   UpdateSourceTrigger=PropertyChanged}" Tag="Pesquise por Título, Autor ou Editora"
47                   Style="{StaticResource MyWaterMarkStyle}" />
48
```

Figura 41 - MainWindow, Search bar

De forma a obter uma pesquisa a cada alteração do texto na caixa de pesquisa, foi necessário que a Window implementasse a Interface INotifyPropertyChanged, de forma a serem notificadas as partes interessadas de que determinada propriedade da Window foi alterada no seu "Set", uma vez que esta interface obriga à implementação do método NotifyPropertyChanged.

É também necessário configurar no XAML, na propriedade Text da caixa de pesquisa o Binding apropriado, nomeadamente

- Path - o caminho da fonte dados (neste caso será a propriedade SearchQuery da Window que implementa o NotifyPropertyChanged);
- Mode - como sendo "TwoWay" para ser atualizado nos dois sentidos, tanto do XAML para o CS, como do CS para o XAML;
- UpdateSourceTrigger - como PropertyChanged para forçar a atualização a cada alteração da propriedade SearchQuery;
- Delay de 500 milisegundos para só fazer a pesquisa ao final desse tempo, salvaguardando assim pesquisas desnecessárias que degradam a performance da aplicação.

```
<TextBox Grid.Column="0" Name="TxtSearch"
         Text="{Binding Path=SearchQuery, Mode=TwoWay, Delay=500,
         UpdateSourceTrigger=PropertyChanged}" Tag="Pesquise por Título, Autor ou Editora"
         Style="{StaticResource MyWaterMarkStyle}" />
```

Figura 42 - TextBox de Pesquisa, XAML

ProjetoGlobal1929 ▾ ProjetoGlobal1929.MainWindow

```
                4 references | 0 changes | 0 authors, 0 changes
25              public partial class MainWindow : Window, INotifyPropertyChanged
26              {
27                  //Caixa de pesquisa
28                  #region INotify
29                  public event PropertyChangedEventHandler PropertyChanged;
                    1 reference | 0 changes | 0 authors, 0 changes
30                  protected void NotifyPropertyChanged(string propertyName)
31                  {
32                      if (PropertyChanged != null)
33                          PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
34                  }
35                  #endregion
36
37                  private string searchQuery;
                    2 references | 0 changes | 0 authors, 0 changes
38                  public string SearchQuery
39                  {
40                      get
41                      {
42                          return searchQuery;
43                      }
44
45                      set
46                      {
47                          searchQuery = value;
48                          NotifyPropertyChanged("SearchQuery");
49                          Refresh();
50                      }
51                  }
52
```

Figura 43 - Implementação da interface INotifyPropertyChanged

4.2.1.3. Lista de Artigos

De forma a criar uma barra de scroll para a listagem de artigos, utilizou-se um ScrollViewer com uma DataGrid inserida. Para a utilização do scroll do rato foi criado o evento `PreviewMouseWheel="ScrollViewer_PreviewMouseWheel"` e adicionado o seguinte código.

```
219         private void ScrollViewer_PreviewMouseWheel(object sender, MouseWheelEventArgs e) //Scroll do rato.
220         {
221             ScrollViewer scrollviewer = sender as ScrollViewer;
222             if (e.Delta > 0)
223             {
224                 scrollviewer.LineUp();
225             }
226             else
227             {
228                 scrollviewer.LineDown();
229             }
230             e.Handled = true;
231         }
```

Figura 44 - PreviewMouseWhell (scroll do rato)

A DataGrid contem as colunas Capa, Título, Autor, Descrição, Editora e Artigo com largura fixa. Foi adicionado um estilo de *Wrap* para permitir que o texto mude de linha.

```
<Style>
    <Setter Property="TextBlock.TextWrapping" Value="Wrap" />
</Style>
```

Figura 45 - Estilo de TextWrap

A capa encontra-se guardada num Array de bytes na Base de Dados. De forma a apresentar a mesma dentro da DataGrid foi criada a classe BinaryImageConverter que é chamada no xaml:

```
<DataGridTemplateColumn Width="100" Header="Capa">
    <DataGridTemplateColumn.CellTemplate>
        <DataTemplate>
            <Image Width="95" Source="{Binding CoverBytes, Converter={StaticResource imgConverter}}" />
        </DataTemplate>
    </DataGridTemplateColumn.CellTemplate>
</DataGridTemplateColumn>
```

Figura 46 - Conversor de bits em Imagem, XAML

A classe BinaryImageConverter converte o Array de bits do campo Capa em imagem de forma à mesma ser apresentada na tabela:

```
public class BinaryImageConverter : IValueConverter
{
    0 references | 0 changes | 0 authors, 0 changes
    object IValueConverter.Convert(object value,
        Type targetType,
        object parameter,
        System.Globalization.CultureInfo culture)
    {
        if (value != null && value is byte[])
        {
            byte[] bytes = value as byte[];

            MemoryStream stream = new MemoryStream(bytes);

            BitmapImage image = new BitmapImage();
            image.BeginInit();
            image.StreamSource = stream;
            image.EndInit();

            return image;
        }

        return null;
    }

    0 references | 0 changes | 0 authors, 0 changes
    object IValueConverter.ConvertBack(object value,
        Type targetType,
        object parameter,
        System.Globalization.CultureInfo culture)
    {
        return null;
    }
}
```

Figura 47 - Classe BinaryImageConverter

4.2.1.4. Botões

Os botões de "Adicionar", "Editar", "Eliminar" e "Download" de artigos encontram-se imediatamente abaixo da grelha de livros. Os botões foram adicionados dentro de um StackPanel.

```
<StackPanel Grid.Row="3" Name="PnlButtons" Orientation="Horizontal" HorizontalAlignment="Left">
    <Button Name="BtnAdd" Content="Adicionar" Click="BtnAdd_Click" Margin="5" />
    <Button Name="BtnEdit" Content="Editar" Click="BtnEdit_Click" Margin="5" />
    <Button Name="BtnDel" Content="Eliminar" Click="BtnDel_Click" Margin="5" />
    <Button Name="BtnDownload" Content="Download" Click="BtnDownload_Click" Margin="5" />
</StackPanel>
```

Figura 48 - StackPanel Botões

Caso não esteja nenhum artigo selecionado, o único botão disponível é o "Adicionar". Ao escolher um artigo os botões "Editar" e "Eliminar" ficam disponíveis, o botão "Download" só fica disponível quando o artigo selecionado tem conteúdo possível de descarregar. Para cumprir o funcionamento de disponibilidade dos botões foi criado um método para refrescar o estado dos mesmos.

```
3 references | 0 changes | 0 authors, 0 changes
private void RefresBtnsState()
{
    BtnEdit.IsEnabled = gridDocs.SelectedIndex > -1;
    BtnDel.IsEnabled = gridDocs.SelectedIndex > -1;
    BtnDownload.IsEnabled = gridDocs.SelectedIndex > -1 && ((Document)gridDocs.SelectedItem).ContentBytes != null;
}
```

Figura 49 - Refresca Botões

Os botões "Adicionar", "Editar" e "Eliminar" são escondidos caso o utilizador não tenha privilégios de editar.

```
if(user.CurrentAccessMode != AccessMode.Admin)
{
    BtnAdd.Visibility = Visibility.Collapsed;
    BtnEdit.Visibility = Visibility.Collapsed;
    BtnDel.Visibility = Visibility.Collapsed;
    LblProfile.Text = "Perfil de utilizador: Normal";
}
```

Figura 50 - Retira botões (perfil de leitura)

Sempre que o utilizador altera o artigo selecionado na tabela, é iniciado o evento SelectedCellsChanged="gridDocs_SelectedCellsChanged" e refrescado o estado dos botões.

Ao pressionar o rato duas vezes em cima de um artigo, é iniciado o evento MouseDoubleClick="gridDocs_MouseDoubleClick" que por sua vez valida o perfil do utilizador. Caso o utilizador tenha privilégios de edição este evento irá atuar da mesma forma que o botão "Editar", caso o utilizador só tenha privilégios de leitura, este evento irá descarregar o pdf do artigo.

```
private void gridDocs_MouseDoubleClick(object sender, MouseButtonEventArgs e)
{
    if (User.CurrentAccessMode != AccessMode.Admin && ((Document)gridDocs.SelectedItem).ContentBytes != null)
    {
        try
        {
            Document document = (Document)gridDocs.SelectedItem;
            System.Windows.Forms.FolderBrowserDialog dialog = new System.Windows.Forms.FolderBrowserDialog();
            if (dialog.ShowDialog() == System.Windows.Forms.DialogResult.OK )
            {
                string filename = System.IO.Path.Combine(dialog.SelectedPath, document.Titulo + ".pdf");
                PdfHelper.Instance.SaveBytesToFile(filename, document.ContentBytes);
                System.Diagnostics.Process.Start(filename);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Erro", MessageBoxButton.OK, MessageBoxImage.Error);

        }
    }
    else if(User.CurrentAccessMode != AccessMode.Admin && ((Document)gridDocs.SelectedItem).ContentBytes == null)
    {
        return;
    }
    else
    {

        Document doc = (Document)gridDocs.SelectedItem;
        EditarLivros window = new EditarLivros(EditionMode.Edit, doc);
        bool? result = window.ShowDialog();
        if (result.HasValue && result.Value)
            Refresh();
    }
}
```

Figura 51 - MouseDoubleClick

Ao pressionar o botão "Adicionar" é aberta a janela EditarLivros em modo de Adicionar. Se for adicionado um novo artigo, a lista de artigos é refrescada no retorno.

```
private void BtnAdd_Click(object sender, RoutedEventArgs e)
{
    try
    {
        EditarLivros window = new EditarLivros(EditionMode.Add); //Modo de edição Adicionar
        bool? result = window.ShowDialog();
        if (result.HasValue && result.Value) //Refesca lista de livros no retorno
            Refresh();
    }
    catch(Exception ex)
    {
        MessageBox.Show(ex.Message, "Erro", MessageBoxButton.OK, MessageBoxImage.Error);
    }

}
```

Figura 52 - Botão Adicionar Artigos

Ao pressionar o botão "Editar" é aberta a janela EditarLivros em modo de Editar. Neste caso também é enviado o Id do artigo selecionado de forma ao mesmo poder ser editado.

```
private void BtnEdit_Click(object sender, RoutedEventArgs e)
{
    try
    {
        Document doc = (Document)gridDocs.SelectedItem; //Identifica o artigo selecionado
        EditarLivros window = new EditarLivros(EditionMode.Edit, doc);
        bool? result = window.ShowDialog();
        if (result.HasValue && result.Value) // Refresca a lista de artigos no retorno
            Refresh();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Erro", MessageBoxButton.OK, MessageBoxImage.Error);
    }

}
```

Figura 53 - Botão Editar Artigos

Ao pressionar o botão "Eliminar" é apresentada uma caixa de confirmação para validar a remoção do artigo. Caso o utilizador confirme, é chamado o método "DeleteDoc" enviando o Id do Artigo a ser eliminado.

```
private void BtnDel_Click(object sender, RoutedEventArgs e)
{
    try
    {
        Document doc = (Document)gridDocs.SelectedItem;
        if (MessageBox.Show("Tem a certeza que pretende eliminar o item selecionado?", "Confirmação",
        {
            DbService.Instance.DeleteDoc(doc.Id);
            Refresh();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Erro", MessageBoxButton.OK, MessageBoxImage.Error);
    }

}
```

Figura 54 - Botão Eliminar Artigos

Por sua vez o método DeleteDoc apaga o artigo da base de dados.

```
1 reference | 0 changes | 0 authors, 0 changes
public bool DeleteDoc(int id)
{
    DbManager.Instance.ExecuteCommand("delete from Document where Id = " + id);
    return true;
}
```

Figura 55 - DbServices, DeleteDoc

Ao pressionar o botão "Download", é descarregado o pdf do artigo selecionado. O pdf foi anteriormente gravado na base de dados no formato de bytes. Para gerir o upload e download dos pdf foi criada a classe PdfHelper.cs que contém as funções de carregar e descarregar o pdf bem como o método de criação da capa. Para descarregar o ficheiro o botão executa ao método SaveBytesToFile indicando o título do artigo que será o nome do ficheiro pdf.

```
private void BtnDownload_Click(object sender, RoutedEventArgs e)
{
    try
    {
        Document doc = (Document)gridDocs.SelectedItem;
        System.Windows.Forms.FolderBrowserDialog dialog = new System.Windows.Forms.FolderBrowserDialog();
        if (dialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)
        {
            string filename = System.IO.Path.Combine(dialog.SelectedPath, doc.Titulo + ".pdf");
            PdfHelper.Instance.SaveBytesToFile(filename, doc.ContentBytes); // Descarrega artigo
            System.Diagnostics.Process.Start(filename);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Erro", MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
```

Figura 56 - Botão Download

```
3 references | 0 changes | 0 authors, 0 changes
public void SaveBytesToFile(string filename, byte[] bytesToWrite)
{
    if (filename != null && filename.Length > 0 && bytesToWrite != null)
    {
        if (!Directory.Exists(Path.GetDirectoryName(filename)))
            Directory.CreateDirectory(Path.GetDirectoryName(filename));

        FileStream file = File.Create(filename);

        file.Write(bytesToWrite, 0, bytesToWrite.Length);

        file.Close();
    }
}
```

Figura 57 – SaveBytesToFile

4.2.1.5. Barra de estado

No fundo do DockPanel foi adicionada uma StatusBar. Por sua vez a StatusBar contém uma Grid onde mostra o perfil e nome do utilizador logado.

```
TextBlock                                                    Name
13    <DockPanel>
14        <StatusBar DockPanel.Dock="Bottom">
15            <StatusBar.ItemsPanel>
16                <ItemsPanelTemplate>
17                    <Grid>
18                        <Grid.RowDefinitions>
19                            <RowDefinition Height="*"/>
20                        </Grid.RowDefinitions>
21                        <Grid.ColumnDefinitions>
22                            <ColumnDefinition Width="*"/>
23                            <ColumnDefinition Width="*"/>
24                        </Grid.ColumnDefinitions>
25                    </Grid>
26                </ItemsPanelTemplate>
27            </StatusBar.ItemsPanel>
28            <StatusBarItem>
29                <TextBlock Name="LblUser" Margin="20 0 0 0">User</TextBlock>
30            </StatusBarItem>
31            <StatusBarItem Grid.Column="1" HorizontalContentAlignment="Right">
32                <TextBlock Name="LblProfile" Margin="0 0 20 0">Profile</TextBlock>
33            </StatusBarItem>
34        </StatusBar>
35
```

Figura 58 - MainWindow, StatusBar xaml

Os blocos de texto LblUser e LblProfile são preenchidos no arranque da janela através da seguinte validação.

```
55        public MainWindow(AppUser user, List<Document> docs)
56        {
57            InitializeComponent();
58            this.User = user;
59            this.Docs = docs;
60            gridDocs.ItemsSource = Docs;
61            RefresBtnsState();
62            if(user.CurrentAccessMode != AccessMode.Admin)
63            {
64                BtnAdd.Visibility = Visibility.Collapsed;
65                BtnEdit.Visibility = Visibility.Collapsed;
66                BtnDel.Visibility = Visibility.Collapsed;
67                LblProfile.Text = "Perfil de utilizador: Normal";
68            }
69            else
70            {
71                LblProfile.Text = "Perfil de utilizador: Admin";
72            }
73            LblUser.Text = "Nome de utilizador: " + User.Username;
74            this.DataContext = this;
75
```

Figura 59 - MainWindow, StatusBar .cs

4.2.2. Adicionar ou Editar Artigo

4.2.2.1. Design Gráfico

Após um utilizador com perfil de edição efetuar um duplo clique num artigo da MainWindow ou pressionar o botão "Editar" abre-se a janela de Editar Artigos trazendo o contexto do Artigo selecionado.

Caso o utilizador selecione a opção "Adicionar" é aberta a mesma Janela não trazendo contexto de nenhum Artigo, permitindo dessa forma adicionar novos artigos.

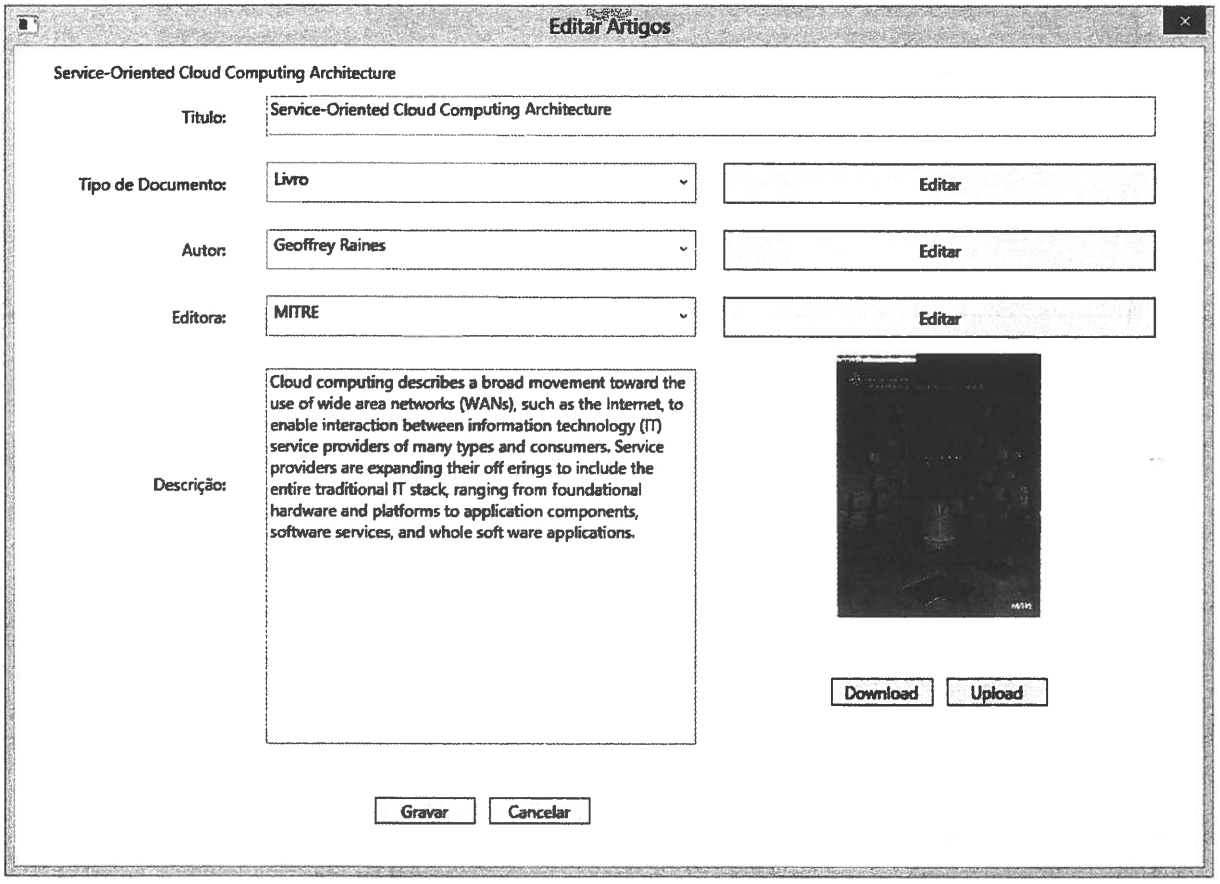

Figura 60 - Editar Artigos

O design foi criado em XAML sendo que a base da janela é composta por uma Grid.

Foi adicionada uma regra de validação para todos os campos da tabela, de forma a os tornar campos obrigatórios. A capa é criada de forma automática no momento do "*Upload*" de um documento pdf. O "*Upload*" de documentos não é obrigatório.

Exemplo de adição de validações de campo obrigatório no XAML:

```
<TextBox Name="TxtTitle" Grid.Column="1" Grid.ColumnSpan="2" Grid.Row="0" Height="30" Margin="10">
    <TextBox.Text>
        <Binding Path="Doc.Titulo" Mode="TwoWay" UpdateSourceTrigger="PropertyChanged">
            <Binding.ValidationRules>
                <validators:RequiredField ErrorMessage="Título é obrigatório." />
            </Binding.ValidationRules>
        </Binding>
    </TextBox.Text>
</TextBox>
```

Figura 61 - Binding, ValidationRules

Para o Binding anterior funcionar, foi adicionada a classe RequiredField, que valida se o campo identificado no XAML está vazio e retorna um aviso de erro caso o campo não esteja preenchido.

```
0 references | 0 changes | 0 authors, 0 changes
public class RequiredField : ValidationRule
{
    private String _errorMessage = String.Empty;
    1 reference | 0 changes | 0 authors, 0 changes
    public string ErrorMessage
    {
        get { return _errorMessage; }
        set { _errorMessage = value; }
    }

    0 references | 0 changes | 0 authors, 0 changes
    public override ValidationResult Validate(object value, CultureInfo cultureInfo)
    {
        var str = value as string;

        if (String.IsNullOrEmpty(str))
        {
            return new ValidationResult(false, this.ErrorMessage);
        }

        return new ValidationResult(true, null);
    }
}
```

Figura 62 - Class ValidationRule

Exemplo gráfico da validação:

Figura 63 - Exemplo gráfico campo preenchimento obrigatório

4.2.2.1. ComboBox

As listas de Autores, Editores e Tipos de Documentos foram criadas nas respetivas tabelas da Base de Dados, de forma a relacionar as mesmas com os artigos, otimizando a BD sem repetição de dados.

Nesta janela foram adicionadas três *ComboBox,* uma para cada lista de forma a simplificar a adição e edição quando o respetivo nome já existe na Base de Dados. O funcionamento das ComboBox é semelhante.

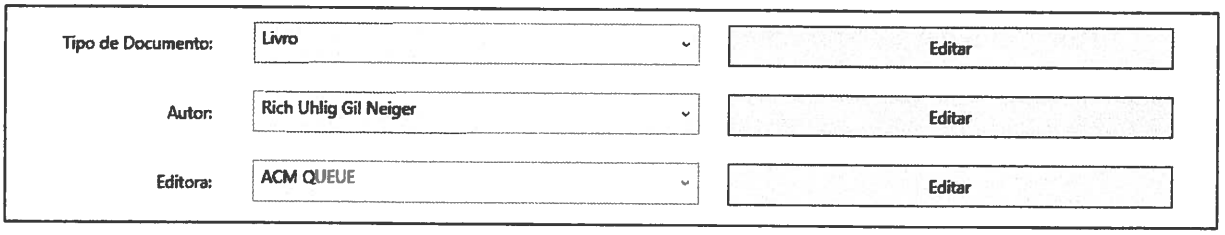

Figura 64 - ComboBox

O conteúdo da ComboBox é apresentado através dos atributos ItemsSource e SelectedValue:

```
<ComboBox Grid.Column="1" Grid.Row="1" Name="CmbDocType" Height="30" Margin="10"
ItemsSource="{Binding TipoDocumento, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" DisplayMemberPath="Descricao"
SelectedValuePath="Id" SelectedValue="{Binding Path=Doc.TipoDocumentoId}" MaxDropDownHeight="300" />
```

Figura 65 - Exemplo ComboBox XAML

De forma a preencher a lista de valores das ComboBox as listas são instanciadas no arranque da Janela.

```
public EditarLivros(EditionMode mode, Document doc = null)
{
    this.Mode = mode;
    this.Doc = doc;
    this.Autores = DbService.Instance.Autores;
    this.TipoDocumento = DbService.Instance.TipoDocumento;
    this.Editora = DbService.Instance.Editora;
```

Figura 66 - Preenchimento das ComboBox, CS

Se estiver em modo de edição, as ComboBox são automaticamente preenchidas com os dados do artigo selecionado.

```
case EditionMode.Edit:
    this.Title = "Editar Documento";
    if (this.Doc == null)
    {
        this.Doc = new Document();
        this.Doc.AutorId = this.Autores.FirstOrDefault().Id;
        this.Doc.EditoraId = this.Editora.FirstOrDefault().Id;
        this.Doc.TipoDocumentoId = this.TipoDocumento.FirstOrDefault().Id;
    }
    break;
```

Figura 67 - Preenchimento das ComboBox, relação com IDs

Caso o nome pretendido não exista na base de dados deve ser pressionado o respetivo botão editar, de forma a seguir para a Janela de edição avançada descrita no capitulo 4.2.3.

4.2.2.2. Botões. Editar listagens

Os Botões "Editar" funcionam todos de forma semelhante. Em primeiro lugar foram adicionados na Grid através de XAML e adicionado o "EventHandler" do tipo Botão, "Click"

```
<Button Grid.Column="2" Grid.Row="1" Name="GerirTipoDocumento" Margin="10 10"
        Content="Editar" Click="GerirTipoDocumento_Click" ></Button>
```

Figura 68 - Exemplo de Botão Editar em XAML

O evento Click do Botão instancia o Tipo de Documento com o respetivo ID e abre a janela descrita no capítulo 4.2.3.

No retorno refresca a ComboBox com a informação atualizada.

```
private void GerirTipoDocumento_Click(object sender, RoutedEventArgs e)
{
    EditarTipoDocumento etd = new EditarTipoDocumento();
    bool? result = etd.ShowDialog();

        this.TipoDocumento = DbService.Instance.TipoDocumento;
        CmbDocType.ItemsSource = this.TipoDocumento; //refresh à combo na volta
        CmbDocType.GetBindingExpression(ComboBox.ItemsSourceProperty).UpdateSource();

    if (etd.Selected != null) // Se não estiver nenhum selecionado procura o ID do novo
    {
        this.Doc.AutorId = etd.Selected.Id;
        CmbDocType.SelectedValue = this.Doc.AutorId;
    }

}
```

Figura 69 - GerirTipoDocumento_Click

4.2.2.3. Inserir pdf

Os utilizadores com privilégios de edição podem carregar os conteúdos em formato pdf através do botão "Upload".

Ao pressionar o botão é chamado o evento "BtnUpload_Click" que por sua vez abre uma Janela do tipo "OpenFileDialog" do Windows para carregar os conteúdos.

```
<Button x:Name="BtnUpload" Content="Upload" Height="20" Width="75" Margin="5" Click="BtnUpload_Click"/>
```

Figura 70 - Botão upload pdf, XAML

```
private void BtnUpload_Click(object sender, RoutedEventArgs e)
{
    OpenFileDialog op = new OpenFileDialog();
    op.Title = "Selecione um PDF";
    op.Filter = "Ficheiros PDF|*.pdf";
    if (op.ShowDialog() == true)
    {

        Doc.CoverBytes = PdfHelper.Instance.GetCover(op.FileName);
        Doc.ContentBytes = PdfHelper.Instance.GetContent(op.FileName);

        MemoryStream stream = new MemoryStream(Doc.CoverBytes);
        BitmapImage image = new BitmapImage();
        image.BeginInit();
        image.StreamSource = stream;
        image.EndInit();
        ImgCover.Source = image;

        RefreshBtns();

    }
}
```

Figura 71 - Evento Upload PDF

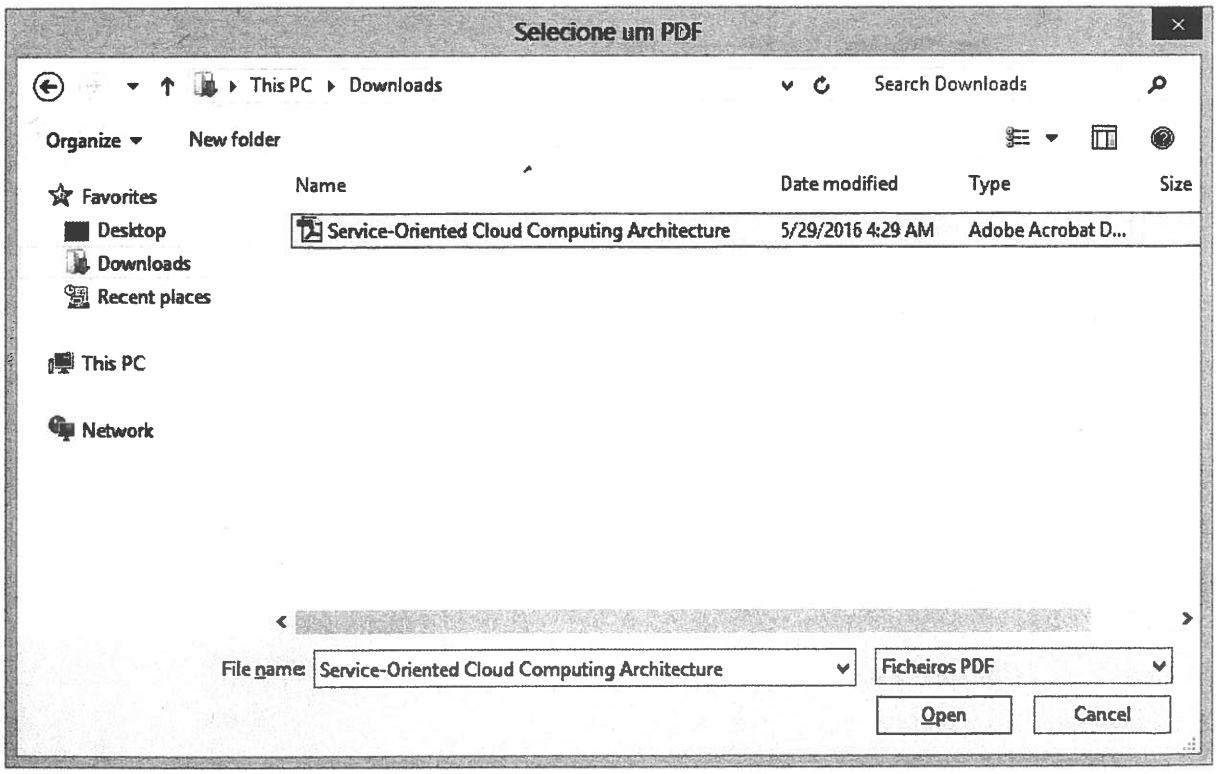

Figura 72 - Janela Windows, selecione um PDF

De forma a gravar a imagem na Base de Dados, foi criada a class PdfHelper que converte o ficheiro pdf num array de bytes.

Dentro da class PdfHelper foram criados dois métodos, o GetCover e GetContent. O primeiro gera uma imagem da primeira página do pdf (utilizando a API Ghostscript), que é transformada num array de bytes de forma a ser guardada na BD. O GetContent, por sua vez, converte todo o conteúdo do pdf num array de bytes para também ser guardado na BD.

No processo de gravação do pdf são instanciados ambos os métodos de forma a gravar os conteúdos nas respetivas tabelas da Base de Dados.

```
public byte[] GetCover(string pdfFilePath)
{
    byte[] result = null;

    string outputPath = Path.GetTempFileName();

    GhostscriptWrapper.GeneratePageThumb(pdfFilePath, outputPath, 1, 40, 60);

    using (Image img = Image.FromFile(outputPath))
    using (MemoryStream ms = new MemoryStream())
    {
        img.Save(ms, ImageFormat.Png);
        result = ms.ToArray();
    }

    File.Delete(outputPath);

    return result;
}

1 reference | 0 changes | 0 authors, 0 changes
public byte[] GetContent(string pdfFilePath)
{
    return File.ReadAllBytes(pdfFilePath);
}
```

Figura 73 - Funçoes GetCover e GetContent para convert pdf

4.2.2.4. Botões Gravar e Cancelar

O botão "Gravar" utiliza o Binding de validação criado anteriormente de forma a só ficar disponível quando todos os campos obrigatórios estiverem preenchidos.

```
<Button x:Name="BtnSave" Content="Gravar" Height="20" Width="75" Margin="5" Click="BtnSave_Click">
    <Button.Style>
        <Style TargetType="Button">
            <Setter Property="IsEnabled" Value="False"/>
            <Style.Triggers>
                <MultiDataTrigger>
                    <MultiDataTrigger.Conditions>
                        <Condition Binding="{Binding Path=(Validation.HasError), ElementName=TxtTitle}" Value="False"/>
                        <Condition Binding="{Binding Path=(Validation.HasError), ElementName=CmbAutor}" Value="False"/>
                        <Condition Binding="{Binding Path=(Validation.HasError), ElementName=CmbDocType}" Value="False"/>
                        <Condition Binding="{Binding Path=(Validation.HasError), ElementName=CmbPublisher}" Value="False"/>
                        <Condition Binding="{Binding Path=(Validation.HasError), ElementName=TxtDescricao}" Value="False"/>
                    </MultiDataTrigger.Conditions>
                    <Setter Property="IsEnabled" Value="True"/>
                </MultiDataTrigger>
            </Style.Triggers>
        </Style>
    </Button.Style>
</Button>
```

Figura 74 - Validação do botão Gravar Artigo, XAML

Exemplo gráfico do botão indisponível:

Figura 75 - Exemplo gráfico, botão Gravar Indisponível

Após pressionar o botão de Gravar é despoletado o evento "BtnSave_Click". O evento valida se a Janela está em modo de edição ou em modo de Adição e invoca a classe DBService modo respetivo.

```
private void BtnSave_Click(object sender, RoutedEventArgs e)
{
    switch (Mode)
    {
        case EditionMode.Add:
            DbService.Instance.AddDoc(Doc);
            this.DialogResult = true;
            break;
        case EditionMode.Edit:
            DbService.Instance.UpdateDoc(Doc);
            this.DialogResult = true;
            break;
    }

}
```

Figura 76 - Evento de Gravação de Artigos

A classe DBService contém todos os métodos para acesso à Base de Dados. Para este botão foram criados os métodos AddDoc e UpdateDoc

```
public bool AddDoc(Document doc)
{
    SqlCommand cmd = new SqlCommand("insert into Document (AutorId,EditoraId,TipoDocumentoId,Titulo,Descricao,CoverBytes,ContentBytes) " +
                                    "values(@AutorId,@EditoraId,@TipoDocumentoId,@Titulo,@Descricao,@CoverBytes,@ContentBytes)");
    cmd.Parameters.Add(new SqlParameter("@AutorId", doc.AutorId));
    cmd.Parameters.Add(new SqlParameter("@EditoraId", doc.EditoraId));
    cmd.Parameters.Add(new SqlParameter("@TipoDocumentoId", doc.TipoDocumentoId));
    cmd.Parameters.Add(new SqlParameter("@Titulo", doc.Titulo));
    cmd.Parameters.Add(new SqlParameter("@Descricao", doc.Descricao));
    SqlParameter coverParam = new SqlParameter("@CoverBytes", SqlDbType.VarBinary);
    if (doc.CoverBytes == null)
        coverParam.Value = DBNull.Value;
    else
        coverParam.Value = doc.CoverBytes;
    cmd.Parameters.Add(coverParam);
    SqlParameter contentParam = new SqlParameter("@ContentBytes", SqlDbType.VarBinary);
    if (doc.ContentBytes == null)
        contentParam.Value = DBNull.Value;
    else
        contentParam.Value = doc.ContentBytes;
    cmd.Parameters.Add(contentParam);

    DbManager.Instance.ExecuteCommand(cmd);
    return true;
}
```

Figura 77 - DBService, Adicionar Artigo

```
public bool UpdateDoc(Document doc)
{
    SqlCommand cmd = new SqlCommand("update Document set " +
                                    "AutorId = @AutorId, " +
                                    "TipoDocumentoId = @TipoDocumentoId, " +
                                    "EditoraId = @EditoraId, " +
                                    "Titulo = @Titulo, " +
                                     "Descricao = @Descricao, " +
                                    "CoverBytes = @CoverBytes, " +
                                    "ContentBytes = @ContentBytes " +
                                "where Id = @Id");

    cmd.Parameters.Add(new SqlParameter("@Id", doc.Id));
    cmd.Parameters.Add(new SqlParameter("@AutorId", doc.AutorId));
    cmd.Parameters.Add(new SqlParameter("@EditoraId", doc.EditoraId));
    cmd.Parameters.Add(new SqlParameter("@Descricao", doc.Descricao));
    cmd.Parameters.Add(new SqlParameter("@TipoDocumentoId", doc.TipoDocumentoId));
    cmd.Parameters.Add(new SqlParameter("@Titulo", doc.Titulo));
    SqlParameter coverParam = new SqlParameter("@CoverBytes", SqlDbType.VarBinary);
    if (doc.CoverBytes == null)
        coverParam.Value = DBNull.Value;
    else
        coverParam.Value = doc.CoverBytes;
    cmd.Parameters.Add(coverParam);
    SqlParameter contentParam = new SqlParameter("@ContentBytes", SqlDbType.VarBinary);
    if (doc.ContentBytes == null)
        contentParam.Value = DBNull.Value;
    else
        contentParam.Value = doc.ContentBytes;
    cmd.Parameters.Add(contentParam);

    DbManager.Instance.ExecuteCommand(cmd);
    return true;
}
```

Figura 78 - DBService, Editar Artigo

Após pressionar no botão "Cancelar", é executado o evento "BtnCancel_Click". Este evento apenas fecha a Janela de forma a voltar para a MainWindow.

```
private void BtnCancel_Click(object sender, RoutedEventArgs e)
{
    this.DialogResult = false;
}
```

Figura 79 - Evento Botão Cancelar

4.2.3. Editar Autores / Editoras / Tipos de Artigo

As Janelas para editar Autores, Editoras e Tipo de Artigo são evocadas a partir da Janela Editar Artigos descrita no capítulo 4.2.2.

O funcionamento e respetivo código destas janelas são idênticos pelo que neste capitulo apenas será detalhado o funcionamento da tabela de Editar Editoras.

4.2.3.1. Design Gráfico

Após pressionar o botão de editar editoras na Janela Editar Artigo é aberta a janela Editar Editoras.

O design foi criado em XAML sendo que a base da janela é composta por uma Grid.

No topo da Grid foi adicionada uma caixa de pesquisa cujo funcionamento é idêntico ao descrito no capítulo 4.2.1.2.

Por baixo da caixa de pesquisa foi adicionada uma ScrollViewer com uma DataGrid utilizando o procedimento descrito no capítulo 4.2.1.3

De seguida foram adicionadas caixas de texto que são automaticamente preenchidas caso se selecione uma linha da tabela. Para isso foi adicionado no evento o seguinte código "gridTipoEditora_SelectedCellsChanged" que deteta alterações na seleção da tabela.

```
private void gridTipoEditora_SelectedCellsChanged(object sender, SelectedCellsChangedEventArgs e)
{
    if (gridTipoEditora.SelectedIndex > -1)
    {
        Editora edit = (Editora)gridTipoEditora.SelectedItem;
        TxtContacto.Text = edit.Contacto;
        TxtEmail.Text = edit.Email;
        TxtLocalidade.Text = edit.Localidade;
        TxtNomeEditora.Text = edit.NomeEditora;
        RefresBtnsState();
    }
}
```

Figura 80 - gridTipoEditora_SelectedCellsChanged

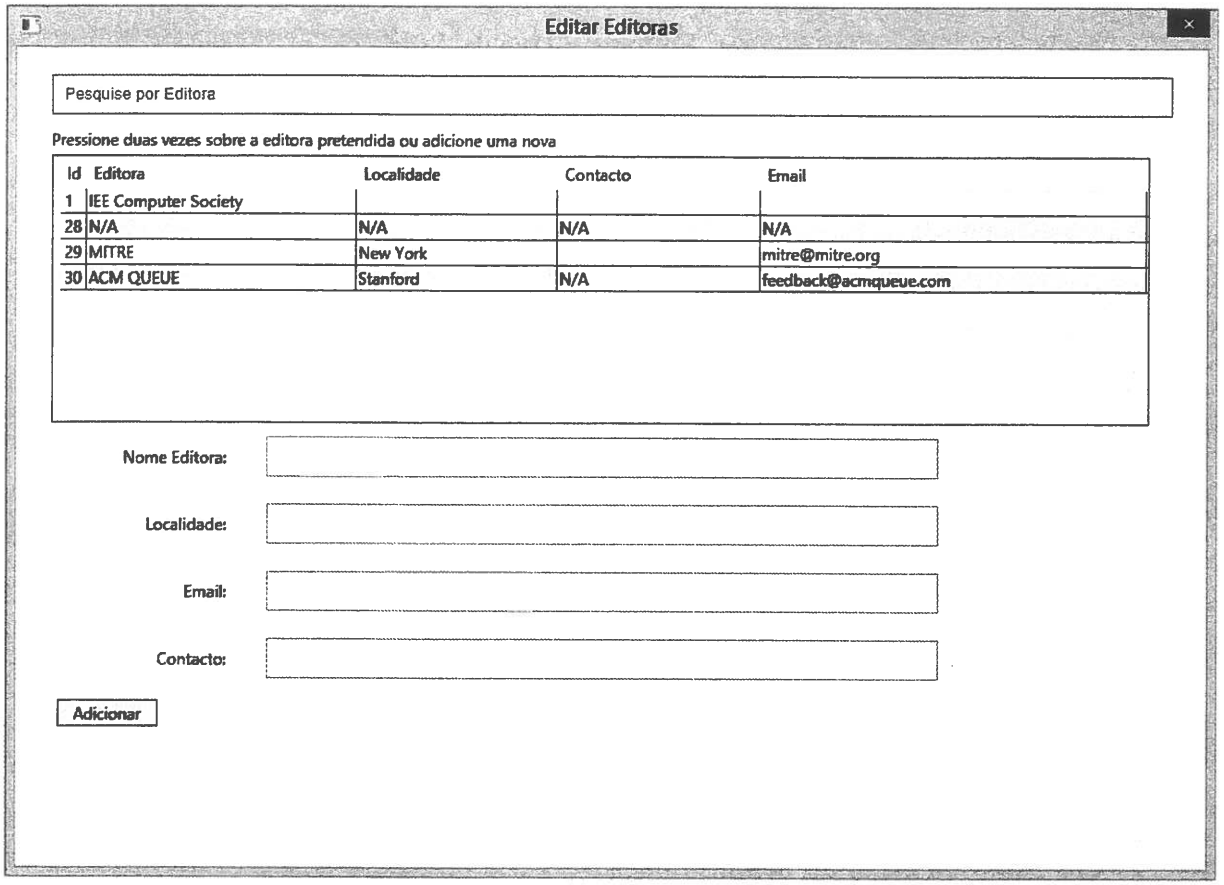

Figura 81 - Editar Editoras

Por último foi criado um StackPanel para receber quatro botões que aparecem conforme o estado da Janela.

```
<StackPanel Orientation="Horizontal" Grid.Row="4" >
    <Button x:Name="BtnAdd" Content="Adicionar" Width="75" Margin="5,6,5,5" Click="BtnAdd_Click"  />
    <Button x:Name="BtnEdit" Content="Gravar" Height="20" Width="75" Margin="5" Click="BtnEdit_Click"  />
    <Button x:Name="BtnCancel" Content="Cancelar" Height="20" Width="75" Margin="5" Click="BtnCancel_Click"  />
    <Button x:Name="BtnDel" Content="Eliminar" Height="20" Width="75" Margin="5" Click="BtnDel_Click"  />
</StackPanel>
```

Figura 82 - Editar Editoras, StackPanel Botões

Caso não esteja nenhuma editora selecionada na DataGrid os botões de "Editar", "Eliminar" e "Cancelar" são ocultados através do método "RefresBtnState" ficando disponível apenas o botão "Adicionar".

```
private void RefresBtnsState()
{
    if (gridTipoEditora.SelectedIndex == -1)
    {
        BtnEdit.Visibility = Visibility.Collapsed;
        BtnDel.Visibility = Visibility.Collapsed;
        BtnAdd.Visibility = Visibility.Visible;
        BtnCancel.Visibility = Visibility.Collapsed;
    }
```

Figura 83 - RefreshBtnsState, Ocultação de Botões

Quando é selecionada uma editora da DataGrid, o comportamento dos botões inverte sendo ocultado o botão "Adicionar"

```
else
{
    BtnAdd.Visibility = Visibility.Collapsed;
    BtnEdit.Visibility = Visibility.Visible;
    BtnDel.Visibility = Visibility.Visible;
    BtnCancel.Visibility = Visibility.Visible;
}
```

Figura 84 - RefreshState, Ocultação de Botões II

O botão "Cancelar" desmarca a seleção da DataGrid e limpa o conteúdo das caixas de texto, de forma a voltar ao modo de "Adicionar".

```
private void BtnCancel_Click(object sender, RoutedEventArgs e)
{
    gridTipoEditora.SelectedIndex = -1;
    TxtNomeEditora.Text = "";
    TxtLocalidade.Text = "";
    TxtEmail.Text = "";
    TxtContacto.Text = "";
    RefresBtnsState();

}
```

Figura 85 - Editar editoras, BtnCancel_Click

4.2.3.2. Adicionar Novo Registo

Para adicionar uma nova Editora, o utilizador necessita de preencher o campo "Nome Editora" e pressionar o botão Adicionar.

Os campos Localidade, Email e Contacto não são de preenchimento obrigatório, contudo o campo Email contem um método que valida o formato do mesmo através de uma expressão REGEX e não permite a gravação caso o formato esteja incorreto, devolvendo uma mensagem de erro.

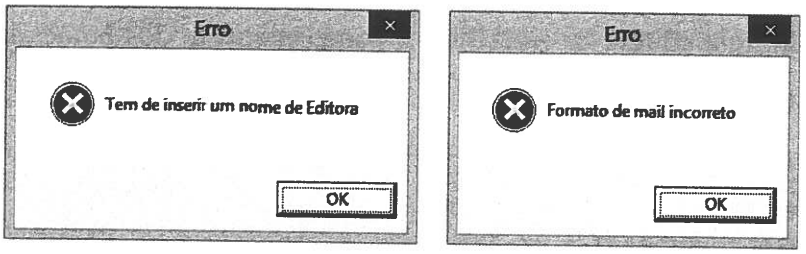

Figura 86 - Mensagens de erro, validação de campos

```
if (TxtNomeEditora.Text == "")
{
    MessageBox.Show("Tem de inserir um nome de Editora", "Erro", MessageBoxButton.OK, MessageBoxImage.Error);
    return;
}
Editora editora = new Models.Editora();
editora.NomeEditora = TxtNomeEditora.Text;
editora.Localidade = TxtLocalidade.Text;
if (TxtEmail.Text != "") //REGEX para validar o mail
{
    string validamail = TxtEmail.Text;
    Regex regex = new Regex(@"^(?(""")(""".+?(?<!\\)""@)|(([0-9a-z]((\.(?!\.))|[-!#\$%&'\*\+/=\?\^`\{\}\|~\w])*)(?<=[0-9a-z])@))" +
                        @"(?(\[)(\[(\d{1,3}\.){3}\d{1,3}\])|(([0-9a-z][-\w]*[0-9a-z]*\.)+[a-z0-9][\-a-z0-9]{0,22}[a-z0-9]))$");
    Match match = regex.Match(validamail);
    if (match.Success == false)
    {
        MessageBox.Show("Formato de mail incorreto", "Erro", MessageBoxButton.OK, MessageBoxImage.Error);
        return;
    }
}
```

Figura 87 - Adicionar Editora, Validação de campos

Caso não existam erros, o evento de adicionar registos irá guardar os dados das caixas de texto numa lista de editoras e invocar o método "AddEditora" da classe "DBService".

```
editora.Email = TxtEmail.Text;
editora.Contacto = TxtContacto.Text;
DbService.Instance.AddEditora(editora); //Gravar na Base de Dados
DbService.Instance.RefreshEditora(); // Refresca a Lista
this.Editora = DbService.Instance.Editora; // Seleciona na combobox o Id inserido
this.gridTipoEditora.ItemsSource = Editora;
```

Figura 88 - Inserir Editora

Por sua vez o método "AddEditora" irá gravar os dados na base de dados.

```
public bool AddEditora(Editora editora)
{
    SqlCommand cmd = new SqlCommand("insert into Editora (NomeEditora,Localidade,Email,Contacto) " +
                                    "values(@NomeEditora,@Localidade,@Email,@Contacto)");

    cmd.Parameters.Add(new SqlParameter("@NomeEditora", editora.NomeEditora));
    cmd.Parameters.Add(new SqlParameter("@Localidade", editora.Localidade));
    cmd.Parameters.Add(new SqlParameter("@Email", editora.Email));
    cmd.Parameters.Add(new SqlParameter("@Contacto", editora.Contacto));

    DbManager.Instance.ExecuteCommand(cmd);
    return true;
}
```

Figura 89 - DBService, AddEditora

De seguida a Janela é fechada e o novo editor é automaticamente preenchido na ComboBox respetiva da Janela Editar Artigo.

```
int id = DbService.Instance.GetLatestId("Editora");
this.Selected = DbService.Instance.GetEditora(id);
this.DialogResult = true;
```

Figura 90 - Identifica ID de nova editora e fecha janela

4.2.3.3. Editar Editoras

Para editar, o utilizador necessita de selecionar uma editora da GataGrid e pressionar o botão "Editar".

O método de editar editoras é muito semelhante ao de adição. Este método contém as mesmas validações de campo obrigatório e email do adicionar.

Após validação de erros o evento irá guardar os dados das caixas de texto numa lista de editoras e invocar o método "EditEditora" da classe "DBService" passando também o ID selecionado.

```
edit.Email = TxtEmail.Text;
edit.Contacto = TxtContacto.Text;
edit.Id = tedit.Id;
DbService.Instance.EditEditora(edit); //Gravar na Base de Dados
DbService.Instance.RefreshEditora(); // refresca a lista
this.Editora = DbService.Instance.Editora;
this.gridTipoEditora.ItemsSource = Editora;
```

Figura 91 - Editar Editora

Por sua vez o método "EditEditora" irá atualizar os dados na base de dados.

```
public bool EditEditora(Editora edit)
{
    SqlCommand cmd = new SqlCommand("update Editora set " +
                                    "NomeEditora = @NomeEditora, " +
                                    "Localidade = @Localidade, " +
                                    "Email = @Email, " +
                                    "Contacto = @Contacto " +
                                "where Id = @Id");

    cmd.Parameters.Add(new SqlParameter("@NomeEditora", edit.NomeEditora));
    cmd.Parameters.Add(new SqlParameter("@Localidade", edit.Localidade));
    cmd.Parameters.Add(new SqlParameter("@Email", edit.Email));
    cmd.Parameters.Add(new SqlParameter("@Contacto", edit.Contacto));
    cmd.Parameters.Add(new SqlParameter("@Id", edit.Id));

    DbManager.Instance.ExecuteCommand(cmd);
    return true;
}
```

Figura 92 - DBService, EditEditora

4.2.3.4. Eliminar Editoras

Para Eliminar, o utilizador necessita de selecionar uma editora da GataGrid e pressionar o botão "Eliminar".

Ao pressionar o botão "Eliminar", é apresentada uma caixa de confirmação para validar a remoção do artigo.

```
Editora aut = (Editora)gridTipoEditora.SelectedItem;
if (MessageBox.Show("Tem a certeza que pretende eliminar o item selecionado?", "Confirmação",
    MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)
{
    if (DbService.Instance.EditoraEmUso(aut.Id))
    {
        MessageBox.Show("A Editora encontra-se associada a um ou mais artigos", "Erro",
            MessageBoxButton.OK, MessageBoxImage.Error);
        return;
    }
```

Figura 93 - Confirmação Eliminar e validação

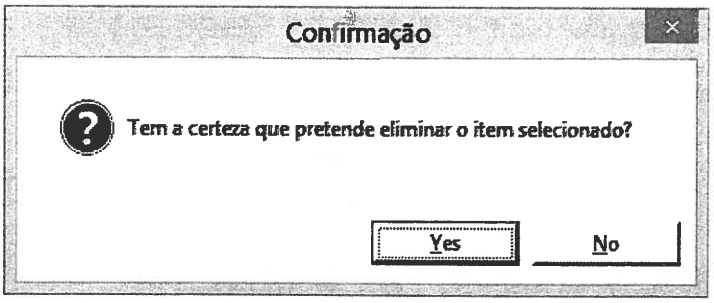

Figura 94 - Confirmação eliminar Editora

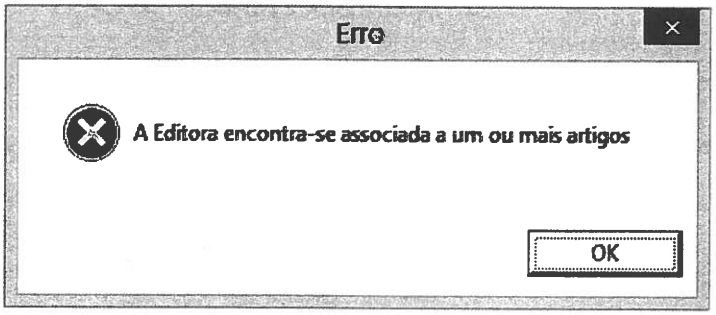

Figura 95 - Erro. A editora encontra-se associada

Para validar quando a editora se encontra em uso, foi criado o método "EditoraEmUso"

```
public bool EditoraEmUso(int Id)
{
    DataTable dt = DbManager.Instance.GetDataTable("select a.* from Editora" +
        "a inner join Document doc on a.Id = doc.EditoraId where a.Id = " + Id);
    return dt.Rows.Count > 0;
}
```

Figura 96 - EditoraEmUso

Caso o utilizador confirme e a editora não esteja a ser utilizada é executado o método "DeleteEditora" enviando o Id do Artigo a ser eliminado.

```
public bool DeleteEditora(int id)
{
    DbManager.Instance.ExecuteCommand("delete from Editora where Id = " + id);
    return true;
}
```

Figura 97 - DeleteEditora

4.1. Instalador da Aplicação

4.1.1. InstallShield 2015 Limited Edition

Por fim foi adicionado o InstallShield 2015 ao projeto no VisualStudio, de forma a criar um instalador .msi para a aplicação.

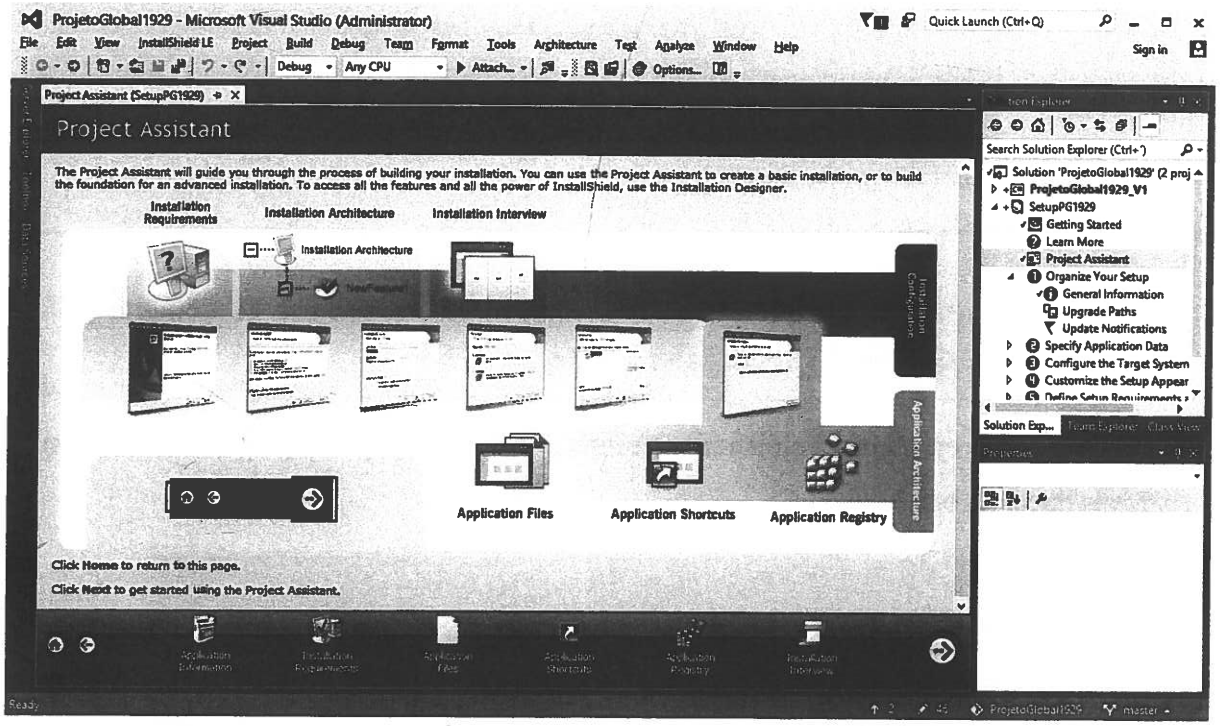

Figura 98 - InstallShield 2015 Limited Edition

O InstallShield permite, entre outras funções, definir um icon da aplicação. Para este projeto foi adicionado o seguinte icon.

Figura 99 - Atalho da aplicação

Adicionalmente o instalador foi configurado de forma a criar um atalho no ambiente de trabalho dos utilizadores.

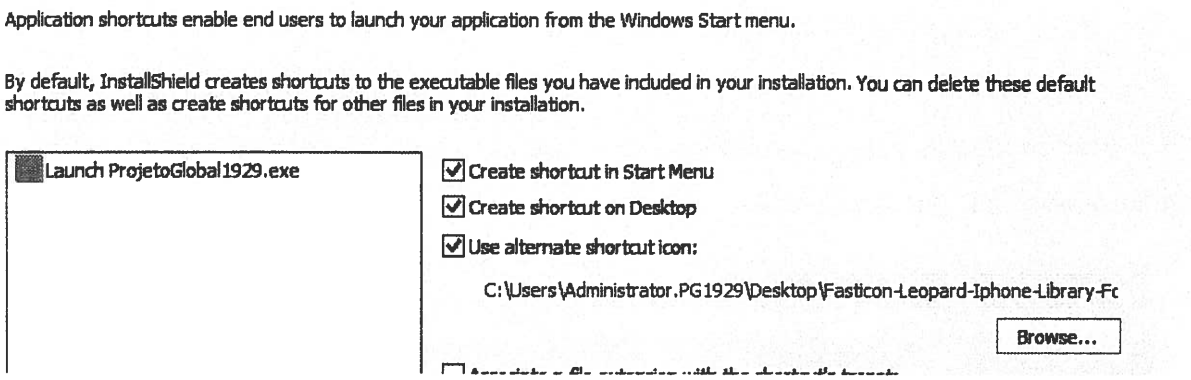

Figura 100 - InstallShield, shortcut configuration

A aplicação é instalada em todas as máquinas pertencentes ao domínio de forma automática através de uma GPO criada para o efeito.

Contudo caso um administrador necessite também é possível instalar de forma manual através do setup.exe

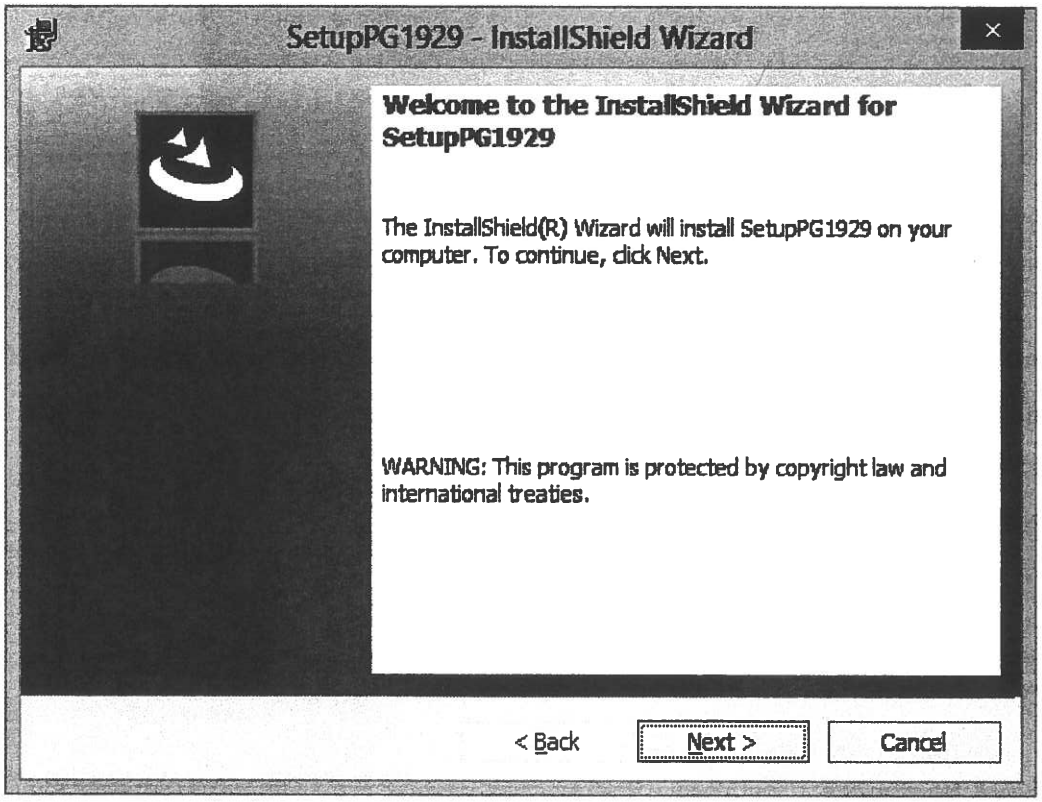

Figura 101 - Instalador da aplicação parte 1

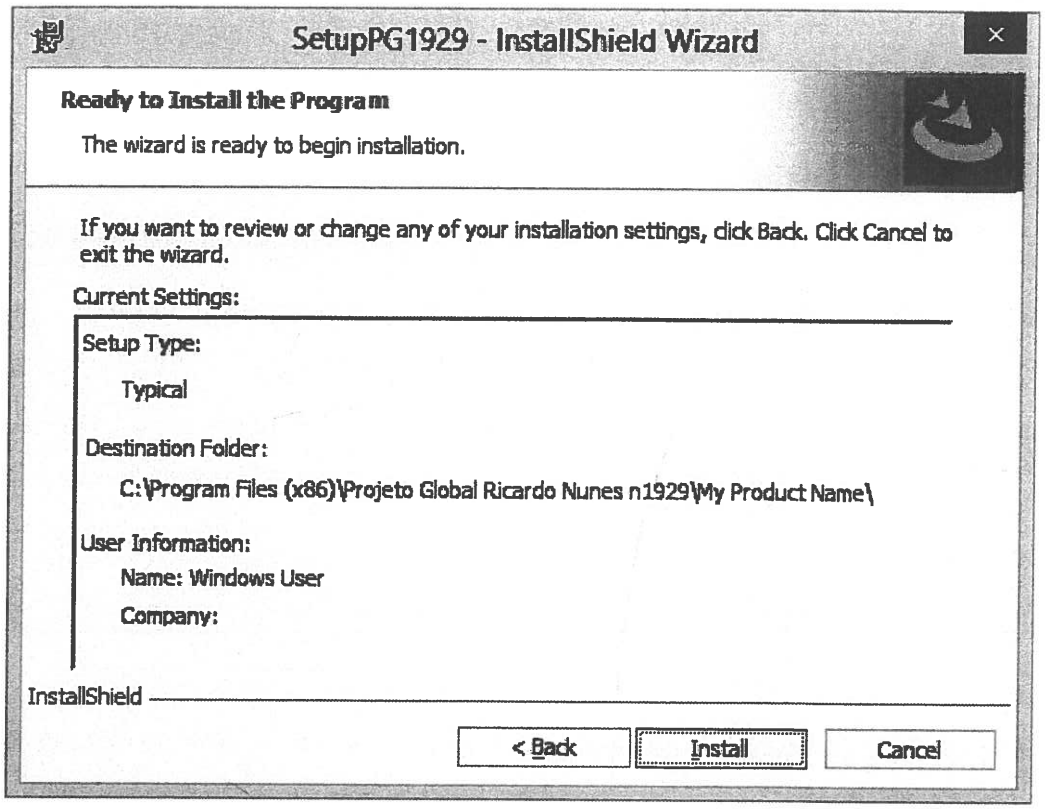

Figura 102 - Instalador da aplicação parte 2

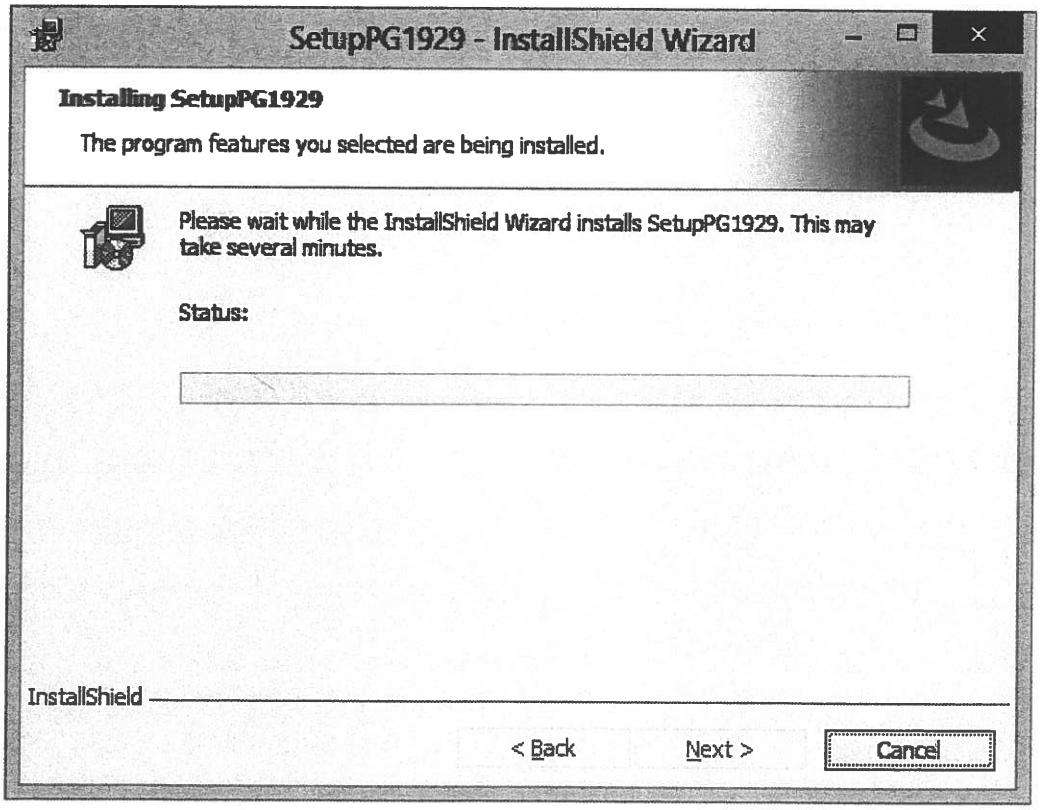

Figura 103 - Instalador da aplicação parte 2

Esta página foi intencionalmente deixada em branco

5. Conclusão

Ao longo deste trabalho, procurou-se demonstrar as potencialidades da tecnologia de Virtualização em ambientes empresariais através de uma aplicação prática da mesma.

No que concerne aos objetivos, esta pesquisa comprovou o conceito de virtualização através da disponibilização de vários sistemas operativos com diferentes aplicações num único servidor físico. Antes da existência desta tecnologia teriam sido necessários vários servidores físicos bem como componentes de rede para os interligar.

A Virtualização de computadores é uma tecnologia robusta e bastante eficiente, alem de que é bastante económica comparando com os tradicionais equipamentos físicos.

Todos os requisitos do cenário da biblioteca foram atingidos. Neste contexto, de acordo com a investigação efetuada, a solução de virtualização apresenta benefícios consideráveis nos diversos ramos computacionais, tanto em nuvens públicas como em privadas.

Bibliografia

Chamberlin , D., Astrahan , M., Blasgen , M., Gray , J., Frank King, W., Lindsay , B., . . . Yost , R. A. (1981). A History and Evaluation os System R. *Communications os the ACM*, 632-646.

Gartner. (2015, Julho 14). *Magic Quadrant for x86 Server Virtualization Infrastructure*. Retrieved from www.gartner.com: https://www.gartner.com/doc/reprints?id=1-2JFZ1KP&ct=150715&st=sb

Hamilton, B., & MacDonald, M. (2003). *ADO.NET in a Nutshell.* O'Reilly.

ISO/IEC 9075-1. (2011). ISO/IEC 9075-1.

Meier, S. (2008, April 2). *IBM Redbooks.* Retrieved from IBM Redbooks: ibm.com/redbooks

Microsoft. (2013, Outubre). *Why Hyper-V?* Retrieved from https://www.microsoft.com/en-us/server-cloud/solutions/virtualization.aspx

NCRIS - National Research Infrastructure for Australia. (2015). *nectarcloud.* Retrieved from nectarcloud: http://training.nectar.org.au/package01/sections/virtualization.html

Oppel, A. (2011). *Databases DeMYSTiFieD, 2nd Edition.* New York: The McGrow Hill Companies.

Popek, G., & Goldberg, R. (1974). Formal Requirements for Virtualizable Third Generation Architectures. *Communications of the ACM.*

Ramakrishnan, R., & Gehrke, J. (2003). *Database Management Systems, 3rd Edition.* New York: McGraw-Hill Higher Education.

Rosenblum, M. (2004, August). *The Reincarnation of Virtual Machines*. Retrieved from Standford University and VMWare: http://dl.acm.org/ft_gateway.cfm?id=1017000&ftid=274909&dwn=1

Safari Books Online. (2016). *www.safaribooksonline.com*. Retrieved from Safari Books Online: https://www.safaribooksonline.com/library/view/adonet-in-a/0596003617/ch01s02.html

Thai, T., & Lam, H. Q. (2003). *.NET Framework Essentials.* O'Reilly.

VMWare. (2007). Retrieved from https://www.vmware.com/files/pdf/VMware_paravirtualization.pdf

Zemke, F. (2012). What's new in SQL:2011. *SIGMOD Record.*